A New Family of Practical Non-Malleable Protocols

Andrew C. Yao* Yunlei Zhao†

Abstract

Nowadays, achieving cryptosystems secure in an asynchronous network like the Internet is demanded to be necessary, where concurrent non-malleable proof-of-knowledge and universal composability are among the most powerful and fundamental security properties. But, when achieving more and more complex cryptosystems secure in an open network like the Internet, it is often the case that generic solutions are either impossible or infeasible.

In this work, we investigate highly practical approaches for achieving non-malleable cryptosystems secure against concurrent man-in-the-middles. We start our study with the Diffie-Hellman key-exchange (DHKE) protocol, which is at the root of public-key cryptography and is one of the main pillars of both theory and practice of cryptography. We develop the mechanisms of non-malleable joint proof-of-knowledge (NMJPOK) and self-sealed joint proof-of-knowledge (SSJPOK), which are of independent values. In particular, using NMJPOK and SSJPOK as the key building tools, we present a new family of DHKE protocols, with remarkable performance among security, privacy, efficiency and easy deployment. Particularly important to applied crypto engineering, the newly developed DHKE protocols add novelties and values to a range of key industry standards for ensuring network security (e.g., IKE, (H)MQV, SSH, etc). Along the way, we also reinvestigate the security definition frameworks for DHKE, and clarify various subtleties surrounding the design and analysis of non-malleable DHKE protocols.

Then, motivated by the building tools, NMJPOK and SSJPOK, proposed and justified in this work, we formulate non-malleable extractable joint one-way function (NME-JOWF), and demonstrate general applications of NME-JOWF (including 3-round CNMZK and UCZK in the plain model). Then, we propose candidates of NME-JOWF based upon bilinear pairings, and show various concrete applications of the pairing-based NME-JOWF candidates.

*Institute for Theoretical Computer Science (ITCS), Tsinghua University, Beijing, China. andrewcyao@tsinghua.edu.cn

†Software School, Fudan University, Shanghai 200433, China. ylzhao@fudan.edu.cn

Contents

1 Introduction **5**
1.1 Contributions and Organization 6

2 Preliminaries **7**
2.1 The Knowledge-of-Exponent Assumption (KEA), Revisited 9

3 Three-Round Adaptive Tag-Based CNMZKAOK for DL **11**
3.1 Definition of Adaptive Tag-Based CNMZKAOK 11
3.2 Protocol Implementation and Analysis of CNMZKAOK for DL 13
3.3 Implication of Concurrent and Forward Deniable Authentication 19

4 Deniable Internet Key-Exchange **20**
4.1 DIKE Protocol Implementation 21
4.2 Formulating Tag-Based Robust Non-Malleability for DHKE Protocols 22
4.3 TBRNM Analysis of the DIKE Protocol 25
4.4 Analysis of DIKE in the Canetti-Krawczyk (CK) Post-Specified Peer Framework . . 34
4.4.1 Brief Description of the CK-framework 34
4.4.2 The DIKE Protocol is SK-Secure in the Post-Specified Peer Setting 36
4.5 Discussions on the Resistance of DIKE Against Some Concrete Attacks 41
4.6 Deniable Internet Key-Exchange: the Aggressive Model 42
4.7 Comparisons with SIGMA 43
4.8 Identity-Based Deniable IKE 44

5 Prove DH-Knowledge Jointly, Non-Malleably, and (Online) Efficiently **46**
5.1 Tag-Binding Self-Sealed Diffie-Hellman 47
5.2 Self-Sealed Joint Proof-of-Knowledge (SSJPOK) 48
5.2.1 Viewing SSJPOK as a New Assumption 53
5.3 Reinvestigations of Some DHKE Protocols, and the Exponent-Dependent Attacks . . 54
5.3.1 Re-Examination of the Burmester-Desmedet (BD) Group Diffie-Hellman . . . 54
5.3.2 Re-Examination of the Chen-Kudla (CK) Identity-Based KE Protocol 55
5.3.3 Re-Examination of (H)MQV 56

6 Towards (Online) Optimal Diffie-Hellman Key-Exchange **58**
6.1 The YZ Protocol Family 58
6.2 YZ Variants 60
6.2.1 Client Privacy-Preserving and Server DoS-Resistant YZ 62
6.3 Advantageous Features of YZ 65

7 Implications of YZ, and Analysis in the Canetti-Krawczyk Framework **67**
7.1 Challenge-Divided Fiat-Shamir Paradigm and Challenge-Divided Schnorr Signatures 67
7.2 Casting (s)YZ in Terms of *Online-Efficient* and *Strongly-Secure* HDR/HCR Signatures 75
7.2.1 (s)YZ-HCR Signatures 83
7.3 Analysis of (s)YZ with Offline Pre-computation in the CK-Framework 84

8 Security of YZ Beyond the CK-framework **88**
8.1 Arguments for Tag-Based Robust Non-Malleability Against Powerful Secrecy Exposure 88
8.1.1 High-level Discussion of the TBRNM Security 88
8.1.2 Non-Black-Box Argument of TBRNM with the JKEA Assumption 90
8.2 Security with Public Computations 91

8.3 Resistance to KCI, and Weak PFS . . . 92
8.4 Summary of the Security Comparisons Between YZ vs. (H)MQV . . . 94

9 Diffie-Hellman Key-Exchange in the Distributed Client-Server Setting 96
9.1 More Applications of SSJPOK . . . 96
9.1.1 SSJPOK-Based Secure Shell (SSH) . . . 96
9.1.2 Online Efficient Signcryption . . . 97
9.2 Password-Based TLS/SSL . . . 97
9.3 Wireless Group Key-Exchange for IEEE 802.11 . . . 100

10 Formulation of Non-Malleable Extractable (Joint) OWF (NME-JOWF) 101
10.1 Formulating Self-Non-Malleable Extractable Joint One-Way Function . . . 101
10.2 Formulating Resettable Self-Non-Malleable Extractable Joint OWF . . . 103
10.3 Formulating Collections of (Resettable) Non-Malleable Extractable JOWF . . . 103
10.4 Formulating (Resettable) Universal Non-Malleable Extractable JOWF . . . 104
10.5 Comparisons with Related Notions . . . 105
10.5.1 Brief Comparisons with Extractable Functions . . . 105
10.5.2 Brief Comparisons with Non-Malleable OWF . . . 106

11 Generic Applications of NME-OWF 107
11.1 3-Round (Statistical) CNMZKAOK for $\mathcal{NP}$ with Full Adaptive Security . . . 107
11.1.1 Formulating CNMZK with Full Adaptive Security . . . 107
11.1.2 Building Tools . . . 109
11.1.3 Three-Round CNMZKAOK for $\mathcal{NP}$ . . . 110
11.2 Non-Malleable Extractable Commitments, and Straight-Line CNMZK . . . 111
11.2.1 Two-Round Public-Coin Non-Malleable Extractable Commitments . . . 111
11.2.2 Three-Round CNMZK with Both Straight-Line Simulation and Extraction . 113
11.3 Constant-Round Concurrent Non-Malleable Coin-Tossing . . . 114
11.4 Applications to Complete Non-Malleable Encryption and Signature . . . 116

12 Candidates of (Resettable) NME-JOWF 116
12.1 General Candidates of (Resettable) NME-JOWF Based on Pairings . . . 117
12.2 Some Concrete Instantiations . . . 117

13 Applications of the Concrete Pairing-Based RNME-OWF Candidates 118
13.1 (Deterministic) Complete Non-Malleable Signatures of Knowledge . . . 118
13.2 Non-Malleable Batch (Two-Party or Triparty) DHKE . . . 120
13.3 Identity-Based (Certificateless) Non-Malleable Signature of Knowledge . . . 121
13.4 ID-Based (Certificateless) Non-Malleable DHKE . . . 122
13.5 Non-Malleable ID-Based Encryption . . . 124
13.6 (Universal) Non-Malleable Verifiable Random Functions . . . 125
13.7 Non-Malleable Set Membership and Range Proofs . . . 126
13.7.1 Motivation for *Non-Malleable* Set Membership and Range Proofs . . . 126
13.7.2 Analysis and Achieving NMR Proof Based on YHMSWY scheme . . . 127
13.7.3 Analysis and Achieving NMSM argument Based on CCS scheme . . . 130

14 Conclusion and Future Investigations 132

A The SIGMA Protocol 140

B Variants of (H)MQV 141

C More variants of YZ **141**

1 Introduction

Cryptographic protocols, namely distributed algorithms aimed at guaranteeing some "security properties" in face of adversarial behaviors, have become an integral part of our society and everyday lives [26]. Nowadays, achieving cryptosystems secure in an asynchronous network like the Internet is demanded to be necessary, where (concurrent) non-malleable proof-of-knowledge and universal composability are among the most powerful and fundamental security properties.

Among cryptographic protocols, the Diffie-Hellman key-exchange (DHKE) protocol [46] is one of the most fundamental and important, which is at the root of public-key cryptography (PKC), and is one of the main pillars of both theory and practice of cryptography. Generally, key-exchange plays a central role in modern cryptography, in the sense that it serves as the bridge between public-key cryptography and secret-key cryptography (SKC). Specifically, PKC is convenient for use but inefficient, while SKC is efficient but inconvenient for use. In common applied crypto applications, PKC is mainly used to generate and share a secret-key among users, and then SKC comes into force with the shared secret-key. Particularly important to applied crypto engineering, DHKE underlies a wide range of industrial standards for ensuring network security (e.g., SSL, SSH, IPsec, MQV/HMQV, etc).

Despite its seeming conceptual simplicity, designing "*sound*" and "*right*" DHKE protocols turns out to be extremely error prone and can be notoriously subtle. The reason for this is the high system complexity and enormous number of subtleties surrounding the design, definition and analysis of DHKE protocols. Telling apart secure from insecure protocols can hardly be done by immediate inspection or using simple intuition, and the analysis of even a simple cryptographic protocol in intricate adversarial settings like the Internet can be dauntingly difficult and complex [26].

Modern cryptography is based upon one-way function (OWF). But, when achieving more and more complex cryptosystems secure in an asynchronous network like the Internet, it is often the case that (black-box) solutions based upon OWF are either impossible or infeasible. In recent years, there is a trend to strengthen and/or abstract cryptographic primitives and assumptions, in order to get conceptually more simple cryptosystem design and analysis.

In this work, we present a new family of DHKE protocols, including deniable Internet key-exchange (referred to as DIKE) and non-malleable online optimal DHKE (referred to as YZ-KE), with remarkable performance among security, privacy, efficiency and easy deployment. Along the way, we also develop a list of building tools and clarify the various subtleties surrounding the design, definition, assumptions and analysis of the new family of DHKE protocols, which might be possibly of independent interest and value. Particularly important to applied crypto engineering, the newly developed DHKE protocols add novelties and values to a range of key industry standards for ensuring network security.

Then, motivated by the building tools, non-malleable joint proof-of-knowledge (NMJPOK) and self-sealed joint proof-of-knowledge (SSJPOK), introduced and employed in the DIKE and YZ-KE protocols, we formulate non-malleable extractable joint one-way function (NME-JOWF), and demonstrate general applications of NME-JOWF, including 3-round CNMZK and UCZK in the plain model. Then, we propose candidates of NME-JOWF based upon bilinear pairings, and show various concrete applications of the pairing-based NME-JOWF candidates (including non-malleable signature of knowledge, non-malleable (batch) DHKE, ID-based and certificateless non-malleable signature of knowledge, ID-based and certificateless (batch) non-malleable DHKE, ID-based encryption, non-malleable verifiable random function, and non-malleable set membership or range proofs, etc).

1.1 Contributions and Organization

- In Section 2, we present the preliminaries. In particular, we revisit the knowledge-of-exponent assumption (KEA) [41], demonstrate and clarify the subtleties and insufficiency of employing the KEA assumption to argue the security of DH-based *interactive* cryptographic protocols when they are run *concurrently in the public-key model* (as is the focus of this work). This motivates us to introduce a new extended KEA assumption, called concurrent KEA (CKEA) assumption. Interestingly, the CKEA assumption can be viewed as the non-black-box counterpart of the gap Diffie-Hellman (GDH) assumption [106], while the original KEA assumption is that of the computational Diffie-Hellman (CDH) assumption. As we shall show in this work, the CKEA assumption is powerful for achieving highly practical cryptographic protocols provably secure against concurrent man-in-the-middle (CMIM) adversaries in concurrent settings like the Internet. We suggest the CKEA-based approach for achieving concurrent non-malleability (CNM) can be a useful paradigm in DH-based cryptographic practice, with a reasonable trade-off between practical efficiency and provable security.

- In Section 3, we first define adaptive tag-based concurrent non-malleable (statistical) zero-knowledge arguments-of-knowledge (CNMZKAOK). Then, we show how to achieve 3-round adaptive tag-based (straight-line) CNMZKAOK for discrete logarithm (DL) under the CKEA assumption in the restricted random oracle model [126], which in particular implies 3-round concurrent forward deniable authentications.

- In Section 4, we develop a new family of *non-malleable* and *deniable* DHKE protocols, which are named deniable Internet key-exchange (DIKE). The newly developed DIKE protocols are of conceptual simplicity and clarity, provide much remarkable privacy protection to protocol participants, are of highly practical (online) efficiency, and of well compatibility with the current IKEv2 standard of IPsec and the SIGMA protocol (on which IKEv2 is based) as well as the widely standardized (H)MQV protocols. We also present ID-based variant of the DIKE protocol based upon bilinear pairings.

  For the security of the DIKE protocols, we formulate the notion of tag-based robust non-malleability (TBRNM) for DHKE protocols, which essentially says that a CMIM adversary can successfully finish a session of a DHKE protocol only if it does know both the secret-key and the DH-exponent corresponding to the public-key and the DH-component alleged and sent by the CMIM adversary for that session. The TBRNM formulation takes security and privacy in an integrity, which particularly implies concurrent forward deniability for both the protocol initiator and the protocol responder. We show that the TBRNM security and the SK-security formulated in the Canetti-Krawczyk framework (CK-framework) [33] are mutually complementary, and thus much desirable to have DHKE protocols that enjoy both of them simultaneously. We show our DIKE protocol is indeed both TBRNM secure and SK-secure, which particularly implies the resistance against most security vulnerabilities not necessarily captured by the SK-security in the CK-framework.

- In Section 5, motivated by the core authentication mechanism of the newly developed DIKE protocols, we investigate the essential properties and principles for highly practical mechanisms of proving multiple DH-knowledges jointly and non-malleably, with multiple shared DH-secrets involved (particularly, *multiplied*) for achieving more advanced and complex cryptographic systems. A series of careful investigations bring us to a tool named self-sealed joint proof-of-knowledge (SSJPOK, in short). Along the way, we also re-examine a list of well-known DHKE protocols, and develop a family of attacks named "exponent-dependent attacks (EDA)" together with remedy approaches.

We suggest the SSJPOK, as a practical paradigm of proving multiple DH-knowledges jointly and non-malleably, may be of independent interest and of wide applications (particularly to authentication and key-exchange). In particular, they can be applicable to any cryptographic applications using the sign-then-MAC authentication mechanism [79] or involving the multiplications of multiple DH-secrets.

- As one core application of the newly introduced SSJPOK, and motivated for achieving robust non-malleability against powerful secrecy exposure (in the spirit of achieving security as robust as possible while relying on secrecy as minimal as possible), we develop and analyze a new family of DHKE protocols, referred to as the YZ protocol family, in Section 6, 7 and 8. The YZ protocols are remarkably of conceptual simplicity and clarity, and as we shall analyze, enjoy remarkable performance among security, privacy, efficiency and easy deployment.

- In Section 9, we investigate more applications of the protocols and techniques developed in this work, which include SSH, SSL, wireless group key-exchange for IEEE 802.11, etc. These applications may also be possibly of independent value and interest, particularly, to applied cryptographic engineering.

- In Section 10 and 11, motivated by the building tool of SSJPOK introduced and employed in the YZ-KE protocols, we formulate non-malleable extractable joint one-way function (NME-JOWF), and demonstrate general applications of NME-JOWF, including 3-round CNMZK and UCZK in the plain model.

- In Section 12 and 13, we propose candidates of NME-JOWF based upon bilinear pairings, and then show various concrete applications of the pairing-based NME-JOWF candidates (including non-malleable signature of knowledge, non-malleable (batch) DHKE, ID-based and certificateless non-malleable signature of knowledge, ID-based and certificateless (batch) non-malleable DHKE, ID-based encryption, non-malleable verifiable random function, and non-malleable set membership or range proofs, etc).

- We conclude in Section 14, with suggestions for future research.

2 Preliminaries

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots ; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots ; r)$. If S is a finite set then $x \leftarrow S$, sometimes also written as $x \in_{\mathrm{R}} S$, is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. By $[R_1; \cdots ; R_n : v]$ we denote the set of values of v that a random variable can assume, due to the distribution determined by the sequence of random processes $R_1, R_2, \cdots, R_n$. By $\Pr[R_1; \cdots ; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$.

On a system parameter n (also written as 1^n), a function $\mu(\cdot)$ is negligible if for every polynomial $p(\cdot)$, there exists a value N such that for all $n > N$ it holds that $\mu(n) < 1/p(n)$. Let $X = \{X(n,z)\}_{n\in N, z\in\{0,1\}^*}$ and $Y = \{Y(n,z)\}_{n\in N, z\in\{0,1\}^*}$ be distribution ensembles. Then we say that X and Y are computationally (resp., statistically) indistinguishable, if for every probabilistic polynomial-time (resp., any power-unbounded) algorithm D, for all sufficiently large n's, and every $z \in \{0,1\}^*$, $|\Pr[D(n,z,X(n,z)) = 1] - \Pr[D(n,z,Y(n,z)) = 1]|$ is negligible in n.

Let G' be an abelian group of order N, and $G = \langle g \rangle$ be a unique subgroup of G', generated by the generator g, of prime order q which is ensured by requiring $gcd(t, q) = 1$ for $t = N/q$. Denote

$Z_q = \{0, 1, \cdots, q-1\}$ and $Z_q^* = \{1, 2, \cdots, q-1\}$, denote by 1_G the identity element of G' and by $G/1_G$ the set of elements of G except 1_G. In the specification of this paper, w.l.o.g., we assume G' is the multiplicative group Z_p^* of order $N = p-1$ for a large prime p, and G is the unique subgroup of order q for some prime divisor q of $N = p-1$. Typically, the length of p (i.e., the length of group element for a DL-based cryptographic system), denoted $|p| = n$, is treated as the *system* parameter, and the length of q, denoted $|q| = k$, is treated as the *security* parameter. The value $t = (p-1)/q$ is called the *cofactor*. The specification can be trivially applicable to the groups based on elliptic curves. In elliptic curve systems, G' is the group of points $E(L)$ on an elliptic curve E defined over a finite field L, and G is a subgroup of $E(L)$ of prime order q. For elliptic curve based groups, the cofactor t is typically very small.

Let $H, H_K : \{0,1\}^* \to \{0,1\}^{|q|}$ and $h : \{0,1\}^* \to \{0,1\}^l/\{0\} \subseteq Z_q^*$ be hash functions, which are modeled as random oracles in security analysis. Here, for presentation simplicity, we have assumed the hash functions H, H_K are of the same output length. In practice, they may be of different output lengths. Sometimes, we also take $l \leq |q| = k$ (rather than k) as the security parameter.

Definition 2.1 (interactive argument system) *A pair of interactive machines, $\langle \hat{A}, \hat{B} \rangle$ where $\hat{A}$ stands for the prover and $\hat{B}$ stands for the verifier, is called an interactive argument system for a language $\mathcal{L}$ if both are probabilistic polynomial-time (PPT) machines and the following conditions hold:*

- *Completeness. For every $x \in \mathcal{L}$, there exists a string w such that for every string $z \in \{0,1\}^*$, $\Pr[\langle \hat{A}(w), \hat{B}(z) \rangle(x) = 1] = 1$.*
- *Soundness. For every polynomial-time interactive machine $\hat{A}^*$, and for all sufficiently large n's and every $x \notin \mathcal{L}$ of length n and every w and z, $\Pr[\langle \hat{A}^*(w), \hat{B}(z) \rangle(x) = 1]$ is negligible in n.*

Definition 2.2 (Computational Diffie-Hellman (CDH) assumption) *Let G be a cyclic group of prime order q generated by an element g, for two elements $X = g^x$, $Y = g^y$ in G, where $x, y \in Z_q$, we denote by $CDH(X,Y) = g^{xy \mod q} \mod p$ (the mod operation is usually omitted for presentation simplicity). An algorithm is called a* CDH solver for G *if it takes as input pairs of elements (X,Y) (and also a generator g of G) and outputs the value of $CDH(X,Y)$. We say the CDH assumption holds in G if for any polynomial-time CDH solver for G, the probability that on a pair (X,Y), for $X, Y \leftarrow G$ (i.e., each of x and y is taken uniformly at random from Z_q), the solver computes the correct value $CDG(X,Y)$ is negligible. The probability is taken over the random coins of the solver, and the choice of X, Y uniformaly at random in G.*

The strong Diffie-Hellman (SDH) assumption, introduced in [1], essentially says that in the group G, computing $CDH(X,Y)$, for $X, Y \leftarrow G$, is strictly harder than deciding if $Z = CDH(X,V)$ for arbitrary $(V,Z) \in G^2$. (Note that the name strong DH assumption also sometimes refers to a different assumption defined over bilinear maps in more recent papers [22].)

Definition 2.3 (Strong Diffie-Hellman (SDH) assumption [1]) *Let G be a cyclic group of prime order q generated by an element g, and a decision predicate algorithm $\mathcal{O}_X$ for $X \leftarrow G$ be a* Decisional Diffie-Hellman (DDH) Oracle *(w.r.t. X taken uniformly at random from G) for the group G and generator g such that on input $(V,Z) \in G^2$, oracle $\mathcal{O}_X$ outputs 1 if and only if $Z = CDH(X,V)$. We say the SDH assumption holds in G if for any polynomial-time CDH solver for G, the probability that on a pair of random elements $(X,Y) \leftarrow G$ the solver computes the correct value $CDG(X,Y)$ is negligible, even when the algorithm is provided with a DDH-oracle $\mathcal{O}_X$ for G. The probability is taken over the random coins of the solver, and the choice of X, Y (each one of them is taken uniformly at random from G).*

The gap DH assumption (GDH) [106] essentially says that in the group G, computing $CDH(X, Y)$, for $X, Y \leftarrow G$, is strictly harder than deciding if $Z = CDH(U, V)$ for an arbitrary triple $(U, V, Z) \in G^3$. Clearly, the GDH assumption is a generalization of the SDH assumption.

Definition 2.4 (Gap Diffie-Hellman (GDH) assumption [106]) *Let G be a cyclic group generated by an element g, and a decision predicate algorithm $\mathcal{O}$ be a (*full*) Decisional Diffie-Hellman (DDH) Oracle for the group G and generator g such that on input (U, V, Z), for* arbitrary *$(U, V) \in G^2$, oracle $\mathcal{O}$ outputs 1 if and only if $Z = CDH(U, V)$. We say the GDH assumption holds in G if for any polynomial-time CDH solver for G, the probability that on a pair of random elements $(X, Y) \leftarrow G$ the solver computes the correct value $CDG(X, Y)$ is negligible, even when the algorithm is provided with the (full) DDH-oracle $\mathcal{O}$ for G. The probability is taken over the random coins of the solver, and the choice of X, Y (each one of them is taken uniformly at random in G).*

2.1 The Knowledge-of-Exponent Assumption (KEA), Revisited

Definition 2.5 (Knowledge-of-Exponent Assumption (KEA)) *Let G be a cyclic group of prime order q generated by an element g, and consider algorithms that on input a triple $(g, C = g^c, z)$ output a pair $(Y, Z) \in G^2$, where c is taken uniformly at random from Z_q^* and $z \in \{0,1\}^*$* is an arbitrary string that is generated independently of C. *Such an algorithm $\mathcal{A}$ is said to be a KEA algorithm if with non-negligible probability (over the choice of g, c and $\mathcal{A}$'s random coins) $\mathcal{A}(g, g^c, z)$ outputs $(Y, Z) \in G^2$ such that $Z = Y^c$. Here, $C = g^c$ is the random challenge to the KEA algorithm $\mathcal{A}$, and z captures the auxiliary input of $\mathcal{A}$ that is independent of the challenge C.*

We say that the KEA assumption holds over G, if for every efficient (probabilistic polynomial-time) KEA algorithm $\mathcal{A}$ for G there exists another efficient algorithm $\mathcal{K}$, referred to as the KEA-extractor, for which the following property holds except for a negligible probability: let (g, g^c, z) be an input to $\mathcal{A}$ and ρ a vector of random coins for $\mathcal{A}$ on which $\mathcal{A}$ outputs $(Y, Z = Y^c)$, then on the same inputs and random coins $\mathcal{K}(g, C, z, \rho)$ outputs the triple $(Y, Z = Y^c, y)$ where $Y = g^y$.

The KEA assumption is derived from the CDH assumption, and is a *non-black-box* assumption by nature [9]. The KEA assumption was introduced in [41], and has been used in many subsequent works (e.g., [10, 9, 44, 80, 42, 45], etc). In particular, the KEA assumption plays a critical role for provable deniability of authentication and key-exchange (e.g., [44, 80, 45]).

In this section, we re-investigate the subtleties of employing the KEA assumption for the provable security of cryptographic protocols in the public-key model and against concurrent man-in-the-middle adversaries (as is the focus of this work).

Subtleties of employing the KEA assumption in the public-key model. Note that the requirement of independence between the challenge C and the auxiliary input z plays a critical role in the KEA assumption. For example, when using KEA for provable security of cryptographic protocols running concurrently in the public-key model, in many cases the challenge C is actually the player's *public-key*. In this case, a valid answer $(A, B = A^c)$, with respect to the challenge C, could be just got by an adversary $\mathscr{A}$ from its auxiliary input that models arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from executions of other (possibly *different*) protocols *before the interaction of the protocol at hand takes place*. Note that, in this case, it is impossible to efficiently extract the value a from the internal state and auxiliary input of the adversary $\mathscr{A}$. This shows that for protocols with provable security based on the KEA assumption w.r.t. *public* challenges, the independence requirement between the auxiliary input z and the public challenge C (corresponding to player's public-key) can significantly limit the composability of the protocol in practice. In other words, in practice it is unrealistic to assume adversary's auxiliary input to be independence of player's public-keys, particularly for protocol running concurrently in the public-key model. To bypass this subtlety of KEA with public challenges and to render robust composability

to cryptographic protocols, in this work we insist using *ephemeral* fresh challenges in designing and analyzing protocols in the public-key model with the KEA assumption.

Note that, as clarified, adversary's auxiliary input captures any information collected over the network from executions of arbitrary (different) protocols before any interaction of the protocol at hand takes place. When using the KEA assumption for provable security of protocols running concurrently in the public-key model, the protocol should be designed to ensure that the transcript of the concurrent executions of the protocol at hand should also not render the adversary CDH solver oracles; Such issues are left in the security analysis of the protocol at hand, and should be the responsibility and care of the protocol designer.

Subtleties of employing the KEA assumption for *interactive* protocols in the *concurrent* setting, and the concurrent KEA (CKEA) assumption. The KEA assumption was originally introduced to argue the (non-malleability) security of public-key encryption (that is a non-interactive cryptographic primitive) [41]. But, when arguing the security of *interactive* protocols running concurrently against concurrent man-in-the-middle (CMIM) adversaries, we note that, in many scenarios (particularly for DH-based authentication and key-exchange as is the focus of this work), the KEA assumption is insufficient. The reason is that, in such settings, the CMIM adversary can potentially get access to a list of (polynomially many) DDH-oracles, with each being w.r.t. an element taken randomly and independently in G by an honest player instance.

For example, consider a two party protocol $\langle \hat{A}, \hat{B} \rangle$, where $\hat{A}$ generates and sends $X = g^x \in G$ and $\hat{B}$ generates and sends $Y = g^y \in G$; After (or during) the exchange of X and Y, each party uses the shared DH-secret g^{xy} to authenticate some values, and aborts in case the authenticated values from its peer are deemed to be invalid. Now, consider a CMIM adversary who, on a system parameter 1^n, simultaneously interacts with $s(n)$ instances of $\hat{A}$ (by playing the role of $\hat{B}$) and $s(n)$ instances of $\hat{B}$ (by playing the role of $\hat{A}$), where $s(\cdot)$ is a positive polynomial. On an arbitrary value $Z \in G$, a random element X_i generated by $\hat{A}$ (or $\hat{B}$), $1 \leq i \leq s(n)$, and another arbitrary element $Y_j \in G$ where Y_j may also be one of the random elements generated by $\hat{A}$ or $\hat{B}$, the CMIM adversary $\mathscr{A}$ can simply use Z (as the supposed DH-secret) to authenticate a value to the party who sends X_i: if the party aborts, $\mathscr{A}$ concludes $Z \neq CDH(X_i, Y_j)$, otherwise it concludes $Z = CDH(X_i, Y_j)$. This simple protocol example demonstrates that in the concurrent settings for (DH-based) interactive protocols, the CMIM adversary can actually get access to polynomially many DDH-oracles, with each one being w.r.t. an element in G generated randomly and independently by an honest player instance). This motivates us to introduce the following assumption, named concurrent knowledge-of-exponents assumption (in reminiscence of the motivation for arguing the *concurrent* security of interactive cryptographic schemes against concurrent man-in-the-middle adversaries).

Definition 2.6 (Concurrent knowledge-of-exponents assumption (CKEA)) *Let G be a cyclic group of prime order q generated by an element g, 1^n is the system parameter, $p(\cdot)$ and $q(\cdot)$ are positive polynomials, and a decision predicate algorithm $\mathcal{O}_{\mathcal{C}}$ for $\mathcal{C} = \{C_1 = g^{c_1}, \cdots, C_{p(n)} = g^{c_{p(n)}}\}$ (where c_i, $1 \leq i \leq p(n)$, is taken uniformly at random from Z_q^*) be a DDH-Oracle (w.r.t. the random challenge set $\mathcal{C}$) for the group G and generator g, such that on input (X, Y, Z), for arbitrary $(X, Y) \in G^2$, the oracle $\mathcal{O}_{\mathcal{C}}$ outputs 1 if and only if $X \in \mathcal{C}$ and $Z = CDH(X, Y)$. Consider algorithms that on input a triple $(g, \mathcal{C}, z)$, with oracle access to $\mathcal{O}_{\mathcal{C}}$, output a set of triples $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$, where $z \in \{0,1\}^*$ is an arbitrary string that is generated independently of $\mathcal{C}$.[1] Such an algorithm $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$ is said to be a CKEA algorithm if with non-negligible probability (over the choice of $g, c_1, \cdots, c_{p(n)}$ and $\mathcal{A}$'s random coins) $\mathcal{A}(g, \mathcal{C}, z)$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$ such that $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(n)$.*

[1] Specifically, we can consider an experiment where the DH-components in the set $\mathcal{C}$ are generated only after the auxiliary string z is fixed.

We say that the CKEA assumption holds over G, if for every PPT CKEA-algorithm $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ there exists another efficient PPT algorithm $\mathcal{K}$, referred to as the CKEA-extractor, such that for any $p(\cdot), q(\cdot)$ and sufficiently large n the following property holds except for a negligible probability: let $(g, \mathcal{C}, z)$ be the input to $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$, ρ a vector of random coins for $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ and ϖ a vector of answers given by $\mathcal{O}_\mathcal{C}$ on queries made by $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ on which $\mathcal{A}$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$ such that $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(n)$, then on the same inputs and random coins and oracle answers $\mathcal{K}(g, \mathcal{C}, z, \rho, \varpi)$ outputs $\{(X_1, Y_1, Z_1, y_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)}, y_{q(n)})\}$ where $Y_i = g^{y_i}$ for all i, $1 \leq i \leq q(n)$.

We remark that the CKEA assumption can be viewed as the non-black-box counterpart of the gap Diffie-Hellamn assumption, while the original KEA assumption is that of the CDH assumption. As we shall show in this work, the CKEA assumption is powerful for achieving highly practical cryptographic protocols provably secure against CMIM adversaries in concurrent settings like the Internet. We suggest the CKEA-based approach for achieving concurrent non-malleability can be a useful paradigm in DH-based cryptographic practice, with a reasonable trade-off between practical efficiency and provable security. We remark that the CKEA assumption is itself precise, and the CKEA-based security analysis can be rigorous.

3 Three-Round Adaptive Tag-Based CNMZKAOK for DL

In this section, we first define adaptive tag-based concurrent non-malleable (statistical) zero-knowledge arguments-of-knowledge (CNMZKAOK). Then, we show how to achieve 3-round adaptive tag-based (straight-line) CNMZKAOK for discrete logarithm (DL) under the CKEA assumption in the restricted random oracle model [126], which in particular implies 3-round concurrent (statistically) deniable authentications.

3.1 Definition of Adaptive Tag-Based CNMZKAOK

Let $\{\langle \hat{A}_{TAG}, \hat{B}_{TAG} \rangle(1^n)\}_{n \in N, TAG \in \{0,1\}^{p(n)}}$, where $p(\cdot)$ is a positive polynomial, be a family of argument systems for an $\mathcal{NP}$-language $\mathcal{L}$ specified by $\mathcal{NP}$-relation $\mathcal{R}_\mathcal{L}$. For each system parameter 1^n and $TAG \in \{0,1\}^{p(n)}$, $\langle \hat{A}_{TAG}, \hat{B}_{TAG} \rangle(1^n)$ is an instance of the protocol $\langle \hat{A}, \hat{B} \rangle$, where $\hat{A}$ is the prover and $\hat{B}$ is the verifier, which is indexed by TAG and works for inputs in $\mathcal{L} \cup \{0,1\}^n$.

We consider an experiment $\text{EXPE}(1^n, \overline{x}, \overline{TAG}, z)$, where 1^n is the system parameter, $s(\cdot)$ is a positive polynomial, $\overline{x} \in (\mathcal{L} \cup \{0,1\}^n)^{s(n)}$, $\overline{TAG} \in (\{0,1\}^{p(n)})^{s(n)}$ and $z \in \{0,1\}^*$. The input $(\overline{x}, \overline{TAG})$ captures the predetermined inputs and tags of the prover instances in the following left CMIM part, and the string $z \in \{0,1\}^*$ captures the auxiliary input to the following adaptive CMIM adversary $\mathscr{A}$. Note that the auxiliary string z is independent of the messages to be sent by the instances of the honest players $\hat{A}$ and $\hat{B}$. In the following specifications, $m \in \{0,1\}^*$ denotes a message sent by an honest player instance, $\overline{m}$ denotes some pre-determined system input, and $\tilde{m}$ denotes the message actually set and sent by the CMIM adversary adaptively based on its view. In the experiment $\text{EXPE}(1^n, \overline{x}, \overline{TAG}, z)$, on input $(1^n, \overline{x}, \overline{TAG}, z)$ the *adaptive* CMIM adversary $\mathscr{A}$ is simultaneously participating in two interaction parts:

The left CMIM part: in which $\mathscr{A}$ *concurrently* interacts with $s(n)$ prover instances: $\hat{A}_{\widetilde{TAG}_1^l}(\tilde{x}_1^l)$, $\hat{A}_{\widetilde{TAG}_2^l}(\tilde{x}_2^l)$, $\cdots$, $\hat{A}_{\widetilde{TAG}_{s(n)}^l}(\tilde{x}_{s(n)}^l)$, where $(\widetilde{TAG}_i^l, \tilde{x}_i^l)$, $1 \leq i \leq s(n)$, are set by $\mathscr{A}$ adaptively based on its view (in both left sessions and right sessions) satisfying: for each i, $1 \leq i \leq s(n)$, $\tilde{x}_i^l \in \{0,1\}^n$, $\widetilde{TAG}_i^l \in \{0,1\}^{p(n)}$, and the membership of $\tilde{x}_i^l \in \mathcal{L} \cup \{0,1\}^n$ can be efficiently

checked;[2] In case $\tilde{x}_i^l \in \mathcal{L} \cup \{0,1\}^n$ (that can be efficiently checked), then a witness $\tilde{w}_i^l$ such that $(\tilde{x}_i^l, \tilde{w}_i^l) \in \mathcal{R}_{\mathcal{L}}$ is given to the prover instance $\hat{A}_{\widetilde{TAG}_i^l}$. The interactions with the instance $\hat{A}_{\widetilde{TAG}_i^l}(\tilde{x}_i^l)$ are called the i-th left session, in which $\mathscr{A}$ plays the role of $\hat{B}_{\widetilde{TAG}_i^l}$. Here, for presentation simplicity, we assume that for each i, $1 \leq i \leq s(n)$, the input $\tilde{x}_i^l$ is set by the CMIM adversary $\mathscr{A}$ at the beginning of the i-th left session, but the session-tag $\widetilde{TAG}_i^l$ may not. Note that, $\mathscr{A}$ can just set $(\tilde{x}_i^l, \widetilde{TAG}_i^l)$ to be $(\overline{x}_i, \overline{TAG}_i)$, which captures the case of predetermined inputs and tags to the i-th left session.

The right CMIM part: in which $\mathscr{A}$ *concurrently* interacts with $s(n)$ verifier instances: $\hat{B}_{\widetilde{TAG}_1^r}(\tilde{x}_1^r)$, $\hat{B}_{\widetilde{TAG}_2^r}(\tilde{x}_2^r)$, $\cdots$, $\hat{B}_{\widetilde{TAG}_{s(n)}^r}(\tilde{x}_{s(n)}^r)$, where $(\widetilde{TAG}_i^r, \tilde{x}_i^r)$, $1 \leq i \leq s(n)$, are set by $\mathscr{A}$ adaptively based on its view (in both left sessions and right sessions) satisfying $\tilde{x}_i^r \in \{0,1\}^n$ and $\widetilde{TAG}_i^r \in \{0,1\}^{p(n)}$. The interactions with the instance $\hat{B}_{\widetilde{TAG}_i^r}(\tilde{x}_i^r)$ are called the i-th right session, in which $\mathscr{A}$ plays the role of $\hat{A}_{\widetilde{TAG}_i^r}$. Here, without loss of generality and for presentation simplicity, we assume the number of left sessions is identical to that of right sessions.

Denote by $view_{\mathscr{A}}(1^n, \overline{x}, \overline{TAG}, z)$ the random variable describing the view of $\mathscr{A}$ in the above experiment EXPE$(1^n, \overline{x}, \overline{TAG}, z)$, which includes the input $(1^n, \overline{x}, \overline{TAG}, z)$, its random tape, and all messages received in the $s(n)$ left sessions and the $s(n)$ right sessions. For protocols in the RO model, $\mathscr{A}$'s view also includes the RO (see [11] for more details).

Then, we say that the family of argument systems $\{\langle \hat{A}_{TAG}, \hat{B}_{TAG} \rangle(1^n)\}_{n \in N, TAG \in \{0,1\}^{p(n)}}$ is adaptive tag-based CNMZKAOK, if for any PPT adaptive CMIM adversary $\mathscr{A}$ defined above, there exists an (expected) polynomial-time algorithm S, such that for any sufficiently large n, any $\overline{x} \in (\mathcal{L} \cup \{0,1\}^n)^{s(n)}$ and $\overline{TAG} \in (\{0,1\}^{p(n)})^{s(n)}$, and any $z \in \{0,1\}^*$, the output of $S(1^n, \overline{x}, \overline{TAG}, z)$ consists of two parts (str, sta) such that the following hold, where we denote by $S_1(1^n, \overline{x}, \overline{TAG}, z)$ (the distribution of) its first output str.

- **Statistical simulatability.** The following ensembles are statistically indistinguishable: $\{view_{\mathscr{A}}(1^n, \overline{x}, \overline{TAG}, z)\}_{n \in N, \overline{x} \in (\mathcal{L} \cup \{0,1\}^n)^{s(n)}, \overline{TAG} \in (\{0,1\}^{p(n)})^{s(n)}, z \in \{0,1\}^*}$ and $\{S_1(1^n, \overline{x}, \overline{TAG}, z)\}_{n \in N, \overline{x} \in (\mathcal{L} \cup \{0,1\}^n)^{s(n)}, \overline{TAG} \in (\{0,1\}^{p(n)})^{s(n)}, z \in \{0,1\}^*}$
- **Knowledge extraction.** sta consists of a set of $s(n)$ strings, $\{\tilde{w}_1^r, \tilde{w}_2^r, \cdots, \tilde{w}_{s(n)}^r\}$, satisfying the following:
  - For any i, $1 \leq i \leq s(n)$, if the i-th right session in str is aborted or with a tag identical to that of one of the left sessions, then $\tilde{w}_i^r = \perp$;
  - Otherwise, i.e., the i-th right session in str is successful and $\widetilde{TAG}_i^r \neq \widetilde{TAG}_j^l$ for all j, $1 \leq j \leq s(n)$, then $(\tilde{x}_i^r, \tilde{w}_i^r) \in \mathcal{R}_{\mathcal{L}}$, where $\tilde{x}_i^r$ is the input to the i-th right session in str (set by the adaptive CMIM adversary $\mathscr{A}$).

Furthermore, if S works in strict (rather than expected) polynomial-time and never rewinds the underlying CMIM algorithm $\mathscr{A}$ (run actually as a subroutine by the S) in its simulation and knowledge extraction, we say that the family of argument systems $\{\langle \hat{A}_{TAG}, \hat{B}_{TAG} \rangle(1^n)\}_{n \in N, TAG \in \{0,1\}^{p(n)}}$

[2] We remark that within our definition experiment, it is necessary (as well as conventional) to require the membership of each statement $\tilde{x}^l$ chosen by $\mathscr{A}$ can be efficiently checked; otherwise, the experiment may render an $\mathcal{NP}$-membership oracle to $\mathscr{A}$.

is adaptive tag-based *straight-line* CNMZKAOK. Unfortunately, adaptive tag-based straight-line CNMZKAOK, which essentially implies universal composable ZK [25, 37], is impossible in the plain model [36, 90, 89].

On simulation with *restricted* random oracles (for interactive protocols). The simulation paradigm is fundamental and powerful for formulating and proving the security of cryptographic protocols, which essentially says that whatever can be done by an efficient adversary with arbitrary interactions in reality can also be done (simulated) by an efficient simulator alone [60, 58, 30]. The random oracle (RO) model, roughly speaking, assumes certain cryptographic hash functions behave like random oracles [12]. When employing the simulation paradigm for proving the security of cryptographic protocols in the RO model, the RO is usually programmed by the simulator (i.e., the simulator provides random answers to RO queries, provided that multiple identical RO-queries are answered with the same answer). But, recent explorations showed that "simulation with (fully) programmable RO" can be problematic [103, 108, 126]: the problems are due to the arbitrary ability of the simulator in programming RO.

To overcome the shortcomings of simulation with programmable RO, the works of [103, 108] proposed the unprogrammable RO model where all parties have access to an unprogrammable (fixed) RO. A further investigation, made in [126] (particularly for *interactive* protocols), showed that, in most cases (particularly for the subtleties observed in [103, 108, 126]), the problem lies in the ability of the simulator in defining (i.e., programming) the RO on queries (first) made by the simulator itself in order to simulate honest parties of private inputs. Specifically, the simulator runs the underlying adversary as a subroutine and mimics honest parties in its simulation. Typically, honest parties (e.g., honest ZK provers) possess some private inputs and get access to un unprogrammable RO in reality; The simulator (in its simulation) has to take the advantage of its ability in programming the RO (to be more precise, programming the RO on queries first made by the simulated honest parties) in order to successfully simulate messages generated by honest parties. But, the simulated honest-party messages may not necessarily be generated with the unprogrammable RO actually accessed by the honest parties in reality. This is precisely the reason for the problems, particularly the loss of deniability, observed in [103, 108, 126] for simulation with programmable RO. The work of [126] proposed the *restricted* RO model, where all parties (particularly, all honest parties and the simulator) *except the adversary* get access to an unprogrammable RO but the adversary (who is polynomial-time and possesses no private inputs) is still allowed to access a programmable RO. For presentation simplicity, we can simply view the restricted RO model is identical to the original RO model, except for that the simulator is confined to programming the RO only on queries first made by the adversary (run by the simulator as its subroutine). Clearly, the restricted RO model is a hybrid of the original programmable RO model [12] and the unprogrammable RO model [103, 108]. The restricted RO model allows efficient (interactive) protocol implementations, while still reasonably avoiding the loss of some properties (particularly, deniability) caused by simulation with fully programmable RO. To our knowledge, no (constant-round straight-line adaptive tag-based) CNMZKAOK schemes for non-trivial languages are known even in the restricted RO model.

3.2 Protocol Implementation and Analysis of CNMZKAOK for DL

Let $A = g^a \in G$ be the common input (where the group G is specified by the parameters (p, q, g)), $a \in Z_q$ be the private input of the prover $\hat{A}$, Tag be the session-tag, and H be a hash function that is assumed to be a (restricted) random oracle. The protocol of adaptive tag-based CNMZKAOK for DL is depicted in Figure 1 (page 14), where the DH-component X (resp., Y) is taken randomly and independently from $G/1_G$ by $\hat{A}$ (resp., $\hat{B}$) and each player checks the $G/1_G$ membership of its peer's DH-component.

Note that in the protocol specification, for presentation simplicity, the session-tag Tag is predetermined and known to both the prover $\hat{A}$ and the verifier $\hat{B}$ prior to the protocol run. In an actual adversarial setting, the session-tag may be set by the CMIM adversary adaptively during the protocol run based on its view in all the concurrent (left and right) sessions, and the following security analysis is w.r.t. this general adversarial setting.

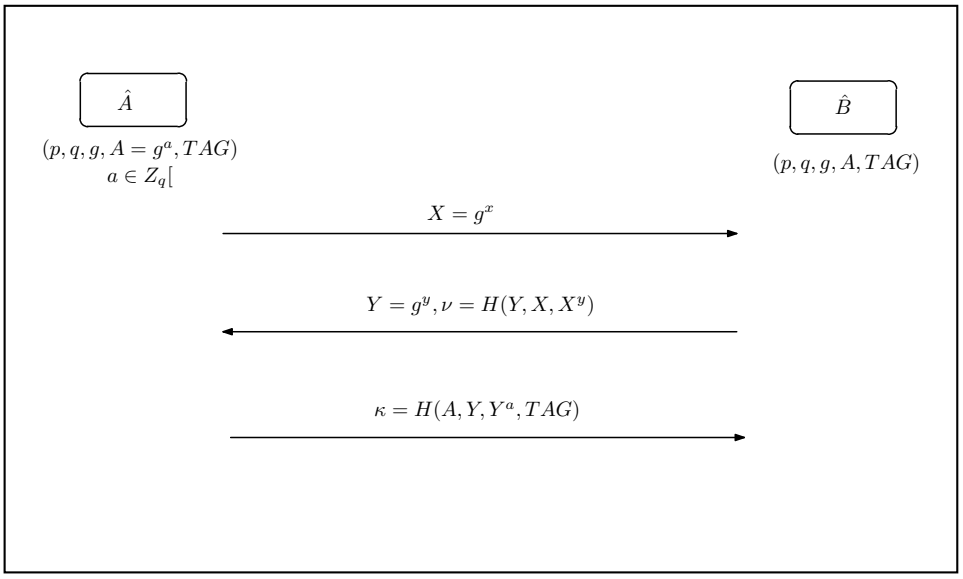


Figure 1: Adaptive tag-based straight-line CNMZKAOK for DL in the restricted RO model

Theorem 3.1 *The 3-round protocol $\langle \hat{A}, \hat{B} \rangle$ depicted in Figure 1 is an adaptive tag-based CNMZKAOK protocol for DL under the CKEA assumption in the restricted random oracle model.*

Proof (outline). On a system parameter 1^n (where $n = |p|$), denote by $\overline{x} = \{A_1, A_2, \cdots, A_{s(n)}\} \in G^{s(n)}$ and $\overline{TAG} = \{TAG_1, TAG_2, \cdots, TAG_{s(n)}\} \in (\{0,1\}^{p(n)})^{s(n)}$ the predetermined inputs and tags of the prover instances in the left CMIM part of the experiment $\mathrm{EXPE}(1^n, \overline{x}, \overline{TAG}, z)$ (described in the definition of adaptive tag-based CNMZKAOK), where $z \in \{0,1\}^*$ is the auxiliary input of the CMIM adversary $\mathscr{A}$ and (A_i, TAG_i) are for the i-th left session, $1 \leq i \leq s(n)$. Recall that the CMIM adversary $\mathscr{A}$ is allowed to *adaptively* set/replace both the input and the session-tag of each (left or right) session. For presentation simplicity, we assume the input of each session is set at the onset of that session, but the session-tag may be set during the session run. *In particular, we assume the CMIM adversary sets and sends the session-tag at the second-round of each left session and at the third-round of each right session.* In the following description, we use the notation $\tilde{m}$ to denote a message actually set and sent by the CMIM adversary adaptively based on its view.

Construction of the simulator/extractor S. On inputs $(1^n, \overline{x}, \overline{TAG}, z)$, the simulator/extractor S works as follows, by running the CMIM adversary $\mathscr{A}$ as a subroutine and with access to the *restricted* random oracle H where S can program H only on queries first made by the CMIM adversary (recall that simulation with restricted RO is crucial for preserving deniability):

Restricted random oracle H. For presentation simplicity, we use a single *restricted* RO H in protocol description and security analysis.[3] Specifically, S can only define the RO answers to queries first made by the CMIM adversary and (the rest part of) the RO is unprogrammable

[3] In the actual specification of the restricted RO methodology [126], two separate ROs are employed in the system: one programmable RO for the adversary and one unprogrammable RO for all other parties in the system.

for all other queries. S stores the list of all pairs of RO-query and RO-answer, and distinguishes each pair as un unprogrammable RO pair (if the RO-answer is got through the unprogrammable RO) or a programmable RO pair (if the RO answer is provided by the simulator itself).

DDH-test list $\mathcal{L}_{DDH}$. During the simulation and knowledge-extraction, S will perform a list of DDH-tests, i.e., checking whether $Z = CDH(X, Y)$ for $(X = g^x, Y = g^y, Z) \in G^3$ where S possesses x or y. For each DDH-test on (X, Y, Z), S stores a record (X, Y, Z, θ) in a list $\mathcal{L}_{DDH}$, where θ is a predicate such that $\theta = 1$ if and only if $Z = CDH(X, Y)$.

Left-session first-round simulation. For each i, $1 \leq i \leq s(n)$, at the first-round of the i-th left session, S acts just as an honest prover does. Specifically, it generates and sends $X_i^l = g^{x_i^l}$, where x_i^l is randomly taken from Z_q^*.

Right-session second-round simulation. For each j, $1 \leq j \leq s(n)$, at the second-round of the j-th right session, S acts as follows: After receiving the first-round message $\tilde{X}_j^r$ from the CMIM adversary $\mathscr{A}$, S checks $\tilde{X}_j^r \in G/1_G$ (and aborts the j-th right-session if $\tilde{X}_j^r \notin G/1_G$) and generates $Y_j^r = g^{y_j^r}$, where y_j^r is taken randomly from Z_q^*. Then, S checks whether the RO H has been defined on the point of $(Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$.[4] This is done by checking all oracle queries to H of the form $(Y_j^r, \tilde{X}_j^r, \tilde{\vartheta}_j^r)$ for some $\tilde{\vartheta}_j^r \in G$, and testing whether $\tilde{\vartheta}_j^r = CDH(Y_j^r, \tilde{X}_j^r) = (\tilde{X}_j^r)^{y_j^r}$ (recall that all DDH-test records will be stored into $\mathcal{L}_{DDH}$). If H has been defined on $(Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$, S just sets ν_j^r to be the already defined $H(Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$; otherwise, S sets ν_j^r to be a random value (taken uniformly at random) in $\{0,1\}^{|q|}$. Finally, S sends (Y_j^r, ν_j^r) to $\mathscr{A}$ as the second-round message of the j-th right session.

In case the value ν_j^r is set to be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary $\mathscr{A}$ queries the RO with a triple $\mathcal{Q} \in G^3$, S tests whether $\mathcal{Q} = (Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r)$ and $\tilde{Z}_j^r = CDH(Y_j^r, \tilde{X}_j^r) = (\tilde{X}_j^r)^{y_j^r}$ (recall that, as specified above, all DDH-test records will be stored in $\mathcal{L}_{DDH}$). If it is this case, S just defines the RO answer, i.e., $H(Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r)$, to be the already chosen ν_j^r (otherwise, S defines the RO answer to be a random value in $\{0,1\}^{|q|}$, provided that multiple identical queries are answered with the same answer). Note that if $\mathscr{A}$ never queries the RO with $(Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r = (\tilde{X}_j^r)^{y_j^r})$, the RO on this point remains undefined. *In particular, S never defines $H(Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$ on its own* (S defines it only when $\mathscr{A}$ makes such a query).

Left-session second-round knowledge-extraction and third-round simulation. For each i, $1 \leq i \leq s(n)$, after receiving the second-round message, denoted $(\widetilde{TAG}_i^l, \tilde{Y}_i^l, \tilde{\nu}_i^l)$, from the CMIM adversary $\mathscr{A}$ in the i-th left session, where $\widetilde{TAG}_i^l \in \{0,1\}^{p(n)}$ is the session-tag set adaptively by $\mathscr{A}$ for the i-th left-session, S first checks whether $\tilde{Y}_i^l \in G/1_G$. If $\tilde{Y}_i^l \notin G/1_G$ S aborts the i-th left-session, otherwise, S does the following:

Case-L1. $\tilde{Y}_i^l$ was not generated and sent by the instances of the honest $\hat{A}$ or $\hat{B}$ in any existing left or right session. In this case, S first checks whether $\mathscr{A}$ has queried the RO with $(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l)$ and $\tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l}$, which is done by checking all the RO-queries made by $\mathscr{A}$ one by one (and each DDH-test record $(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l, \theta_i^l)$ is stored into $\mathcal{L}_{DDH}$). If not,

[4] Actually, in this case, as the random DH-component Y_j^r is freshly generated, with overwhelming probability the RO H has not been defined over $(Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$.

S simply aborts the i-th left session; Otherwise, i.e., $\mathscr{A}$ queried H with $(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l})$, S further checks whether $\tilde{\nu}_i^l = H(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l)$, and aborts the i-th left-session if $\tilde{\nu}_i^l \neq H(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l)$. Finally, if $\mathscr{A}$ queried H with $(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l})$ and $\tilde{\nu}_i^l = H(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l)$, S runs the CKEA-extractor $\mathcal{K}$ on inputs of $(1^n, \overline{x}, \overline{TAG}, z)$, random coins of S (actually the partial random coins used up to $\mathscr{A}$ outputting $(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l})$) *except the coins used to generate the DH-components X_i^l's and Y_j^r's (particularly the DH-exponents x_i^l's and y_j^r's)*, the RO H (i.e, all the RO queries and RO answers up to now) and the DDH-test list $\mathcal{L}_{DDH}$, to try to extract the discrete logarithm, denoted $\tilde{y}_i^l$, of $\tilde{Y}_i^l = g^{\tilde{y}_i^l}$.

Case-L1.1. If the knowledge-extractor $\mathcal{K}$ fails in extracting $\tilde{y}_i^l$, S outputs "failure" and stops.

Case-L1.2. If $\mathcal{K}$ successfully outputs $\tilde{y}_i^l$, S computes $Z_i^l = (\tilde{A}_i^l)^{\tilde{y}_i^l}$, where $\tilde{A}_i^l$ is the common input set by $\mathscr{A}$ for the i-th left-session. Then, S checks whether the RO H has been defined on $(\tilde{A}_i^l, \tilde{Y}_i^l, Z_i^l, \widetilde{TAG}_i^l)$, and if not S makes the (fixed) RO-query $(\tilde{A}_i^l, \tilde{Y}_i^l, Z_i^l, \widetilde{TAG}_i^l)$ to H. Finally, S sends $\kappa_i^l = H(\tilde{A}_i^l, \tilde{Y}_i^l, Z_i^l, \widetilde{TAG}_i^l)$ to $\hat{A}$ as the third-round message of the i-th left session, thereby finishing the i-th left session.

Case-L2. $\tilde{Y}_i^l$ was generated and sent by the honest prover $\hat{A}$ (at the first-round) in one of the left-sessions. In this case, S does the same as in Case-L1.

Case-L3. $\tilde{Y}_i^l$ was generated and sent by the honest verifier $\hat{B}$ (at the second-round) in the j-th right session, i.e., $\tilde{Y}_i^l = Y_j^r$, for some j,$1 \leq j \leq s(n)$. We further distinguish several subcases:

Case-L3.1. $(X_i^l, \tilde{Y}_i^l) \neq (\tilde{X}_j^r, Y_j^r)$, i.e., $X_i^l \neq \tilde{X}_j^r$. In this case, S does the same as in Case-L1.

Case-L3.2. $(X_i^l, \tilde{Y}_i^l) = (\tilde{X}_j^r, Y_j^r)$. We further distinguish two subcases:

Case-L3.2.1. $\tilde{\nu}_i^l \neq \nu_j^r$. In this case, S simply aborts the i-th left session.

Case-L3.2.2. $\tilde{\nu}_i^l = \nu_j^r$, i.e, the transcript of the first two rounds of the i-th left-session is identical to that of the first two rounds of the j-th right session.
In this case, S checks whether the RO H has been defined on the point $(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, CDH(\tilde{A}_i^l, \tilde{Y}_i^l), \widetilde{TAG}_i^l)$, where $\tilde{A}_i^l$ is the common input set by $\mathscr{A}$ for the i-th left-session. This is done by checking all queries to H of the form $(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, \tilde{\vartheta}_i^l, \widetilde{TAG}_i^l)$ for some $\tilde{\vartheta}_i^l \in G$, and testing whether $\tilde{\vartheta}_i^l = CDH(\tilde{A}_i^l, \tilde{Y}_i^l) = CDH(\tilde{A}_i^l, Y_j^r) = (\tilde{A}_i^i)^{y_j^r}$ (again, all DDH-test records are stored into $\mathcal{L}_{DDH}$). If H has been defined on $(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, CDH(\tilde{A}_i^l, \tilde{Y}_i^l), \widetilde{TAG}_i^l)$, S just sets κ_i^l to be the already defined $H(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, CDH(\tilde{A}_i^l, \tilde{Y}_i^l), \widetilde{TAG}_i^l)$; otherwise, S sets κ_i^l to be a random value in $\{0,1\}^{|q|}$. Finally, S sends κ_i^l to $\mathscr{A}$ as the third-round message of the i-th left-session, thereby finishing the i-th left-session.
In case κ_i^l is set to be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary $\mathscr{A}$ queries the RO with a quaternion $(\mathcal{Q}, \widetilde{TAG}_i^l)$ where $\mathcal{Q} \in G^3$, S checks whether $\mathcal{Q} = (\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, \tilde{Z}_i^l)$ and $\tilde{Z}_i^l = CDH(\tilde{A}_i^l, Y_j^r) = (\tilde{A}_i^l)^{y_j^r}$ (again, all DDH-test records will be stored in $\mathcal{L}_{DDH}$). If it is this case, S just defines the RO answer, i.e., $H(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, \tilde{Z}_i^l, \widetilde{TAG}_i^l)$, to be (the already fixed) ν_i^l.
Note that if $\mathscr{A}$ never queries the RO with $(\tilde{A}_i^l, \tilde{Y}_i^l = Y_j^r, \tilde{Z}_i^l = (\tilde{A}_i^l)^{y_j^r}, \widetilde{TAG}_i^l)$, the RO on this point remains undefined. In particular, S never defines $H(\tilde{A}_i^l, \tilde{Y}_i^l =$

$Y_j^r, CDH(\tilde{A}_i^l, Y_j^r), \widetilde{TAG}_i^l)$ on its own (S defines it only when $\mathscr{A}$ makes such a query).

Right-session third-round knowledge-extraction. For each j, $1 \leq j \leq s(n)$, denote by $\tilde{A}_j^r$ the common input set adaptively by the CMIM adversary $\mathscr{A}$ for the j-th right-session. After receiving the third-round message of the j-th right session, denoted $(\tilde{\kappa}_j^r, \widetilde{TAG}_j^r)$, from $\mathscr{A}$, where $\widetilde{TAG}_j^r$ is the session-tag set by $\mathscr{A}$ for the j-th right session, S works as follows:

- **Case-R1.** $\widetilde{TAG}_j^r \neq \widetilde{TAG}_i^l$ for all i, $1 \leq i \leq s(n)$. In this case, S first checks whether $\mathscr{A}$ has made the (programmable) RO query $(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r, \widetilde{TAG}_j^r)$ for $\tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}$ (this is done by checking all RO-queries made by $\mathscr{A}$ one by one, and all DDH-test records are stored into $\mathcal{L}_{DDH}$). If not, S simply aborts the j-th right session; otherwise S further checks whether $\tilde{\kappa}_j^r = H(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}, \widetilde{TAG}_j^r)$ and aborts the j-th right session if the check fails. Finally (i.e., all checks are successful), S accepts this session, and then runs the CKEA-extractor $\mathcal{K}$ on inputs of $(1^n, \overline{x}, \overline{TAG}, z)$, random coins of S (actually the partial random coins used up to $\mathscr{A}$ outputting $(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}, \cdot)$ *at the first time* in one of its RO-queries) *except the coins used to generate the DH-components X_i^l's and Y_j^r's (particularly the DH-exponents x_i^l's and y_j^r's)*, the RO H and the DDH-test list $\mathcal{L}_{DDH}$, to try to extract the discrete logarithm, denoted $\tilde{a}_j^r$, of $\tilde{A}_j^r = g^{\tilde{a}_j^r}$. If the knowledge-extraction of $\tilde{a}_j^r$ is failed, S outputs "failure" and stops, otherwise S sets $\tilde{w}_j^r = \tilde{a}_j^r$.
- **Case-R2.** $\widetilde{TAG}_j^r = \widetilde{TAG}_i^l$ for some $i \in \{i_1, \cdots, i_k\} \subseteq \{1, \cdots s(n)\}$ where $1 \leq k \leq s(n)$. S just sets $\tilde{w}_j^r$ to be $\perp$ in this case, and further acts according to the following two subcases:
  - **Case-R.2.1.** There exists a ξ, $1 \leq \xi \leq k$, such that $(\tilde{A}_j^r, Y_j^r) = (\tilde{A}_{i_\xi}^l, \tilde{Y}_{i_\xi}^l)$ and the i_ξ-th left-session is successfully finished. In this case, if $\tilde{\kappa}_j^r \neq \kappa_{i_\xi}^l$ S aborts the j-th right session, otherwise (i.e., $\tilde{\kappa}_j^r = \kappa_{i_\xi}^l$) S accepts this session.
  - **Case-R.2.2.** In all other cases(i.e., for each $i \in \{i_1, \cdots, i_k\}$ $(\tilde{A}_j^r, Y_j^r) \neq (\tilde{A}_i^l, \tilde{Y}_i^l)$, or $(\tilde{A}_j^r, Y_j^r) = (\tilde{A}_i^l, \tilde{Y}_i^l)$ but the i-th left-session is aborted), S checks whether $\mathscr{A}$ has made the (programmable) RO query $(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r, \widetilde{TAG}_j^r)$ for $\tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}$ (again, the DDH-test records are stored into $\mathcal{L}_{DDH}$). If not, S aborts the j-th right session; otherwise S further checks whether $\tilde{\kappa}_j^r = H(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}, \widetilde{TAG}_j^r)$ and accepts this session if and only if the check is successful.

Output of S. During the simulation, whenever $\mathscr{A}$ aborts a session S also automatically aborts that session. In case S does not stop with the "failure" output (before $\mathscr{A}$ stops), S outputs (str, sta), where sta includes $\{\tilde{w}_1^r, \cdots, \tilde{w}_{s(n)}^r\}$, and str is set to be the set of the input $(1^n, \overline{x}, \overline{TAG}, z)$, the random coins of $\mathscr{A}$, all messages sent by S emulating the honest player instances (Round-1 and Round-3 messages of left-sessions and Round-2 messages of right-sessions), and the RO H.

Analysis of the simulator/extractor S. First, it is clear that S works in *strict* polynomial-time (particularly because the CMIM adversary and the CKEA-extractor work in strict polynomial-time), and never rewinds the CMIM adversary (run as its subroutine), and thus the simulation and knowledge-extraction of S are straight-line. Also, S only defines (programs) the RO H on queries first made by the CMIM adversary $\mathscr{A}$, while the rest part of the RO is unprogrammable, and thus S works in the *restricted* RO model.

The differences between the view of $\mathscr{A}$ in the simulation under the run of S and the view of $\mathscr{A}$ in its real CMIM interactions lie in that:

Bad session-abortion. S may abort a session, but the honest player instance in real interactions may not.

Knowledge-extraction failure. S may stop and outputs "failure" in simulation, due to knowledge-extraction failure.

We first show that, pending on S never fails in knowledge-extraction (in Case-L.3.2.2 and Case-R1), bad session-abortion appears with negligible probability. For Case-L1, Case-L2 and Case-L3.1, as no honest player instance (simulated by S) ever sent $H(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l})$ that the correct $\tilde{\nu}_i^l$ should be (recall that each honest player instance generates and sends a random DH-component in each session), the CMIM adversary $\mathscr{A}$ cannot collect this value from (honest player messages of) existing sessions. Thus, bad session-abortion appears in these cases only if the CMIM adversary successfully guesses the RO output $H(\tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = (\tilde{Y}_i^l)^{x_i^l})$ without making the corresponding RO query. But, this probability is negligible in the RO model. Similarly, bad session-abortion occurs in Case-R1 also with negligible probability. For Case-R.2.2 w.r.t. the j-th right-session, there are two subcases: for each $i \in \{i_1, \cdots, i_k\}$ such that $\widetilde{TAG}_j^r = \widetilde{TAG}_i^l$, $(\tilde{A}_j^r, Y_j^r) \neq (\tilde{A}_i^l, \tilde{Y}_i^l)$, or $(\tilde{A}_j^r, Y_j^r) = (\tilde{A}_i^l, \tilde{Y}_i^l)$ but the i-th left-session is aborted. For both of these two subcases of Case-R.2.2, also no honest player instance (simulated by S) ever sent $H(\tilde{A}_j^r, Y_j^r, \tilde{Z}_j^r = (\tilde{A}_j^r)^{y_j^r}, \widetilde{TAG}_j^r)$ that the correct $\tilde{\kappa}_j^r$ should be, and thus bad session-abortion occurs in Case-R.2.2 also with negligible probability. Finally, besides above cases, session-abortion in all other cases is never "bad", which perfectly mimics what happens in real interactions.

All left (to establish the theorem) is to show that the probability of knowledge-extraction failure in Case-L.3.2.2 and Case-R1 is negligible. Note that pending on the negligible probability of knowledge-extraction failure (and bad session-abortion), the simulated view of $\mathscr{A}$ output by S is *statistically* indistinguishable from the view of $\mathscr{A}$ in its real interactions. Specifically, except for a negligible probability, the distributions of the simulated view and the real view are identical.

The key observation (or the key trick to apply the CKEA assumption) here is that, according to the construction of the simulator/extractor S, the DH-exponents, x_i^l's and y_j^r's, $1 \leq i, j \leq s(n)$ (generated by S itself emulating honest player instances), are used by S *merely* for performing DDH-tests in its simulation. That is, what done by the simulator S can also be done by another efficient oracle machine $S^{\mathcal{O}_\mathcal{C}}$ on the same input $(1^n, \overline{x}, \overline{TAG}, z)$ and the random coins used by S *except the coins used to generate the DH-components X_i's and Y_j's*, $1 \leq i, j \leq s(n)$, where $\mathcal{O}_\mathcal{C}$ is a DDH-oracle for $\mathcal{C} = \{X_1^l, \cdots, X_{s(n)}^l, Y_1^r, \cdots, Y_{s(n)}^r\}$. Specifically, $S^{\mathcal{O}_\mathcal{C}}$ works just as S does, but with the following modifications:

- $S^{\mathcal{O}_\mathcal{C}}$ just sets the DH-component for the i-th left-session (resp., the j-th right-session) to be the value (X_i^l) (resp., Y_j^r), $1 \leq i, j \leq s(n)$, given in the set of $\mathcal{C}$, rather than being generated by itself as S does.
- Whenever S needs to perform a DDH-test w.r.t. a DH-component in $\mathcal{C}$, S queries the DDH-test to its oracle $\mathcal{O}_\mathcal{C}$ and stores the record of the DDH-test result into $\mathcal{L}_{DDH}$.

Then, by the CKEA assumption, whenever $\mathscr{A}$ outputs a triple (X, Y, Z) such that $X \in \mathcal{C}$ and $Z = CDH(X, Y)$ (that can be tested by $S^{\mathcal{O}_\mathcal{C}}$), on input of $(1^n, \overline{x}, \overline{TAG}, z)$, the random coins of $S^{\mathcal{O}_\mathcal{C}}$ (actually the partial coins used up to $\mathscr{A}$ outputting $(X \in \mathcal{C}, Y, Z = CDH(X, Y))$) *that just correspond to the coins of S except the coins used to generates the DDH-components X_i^l's and Y_j^r's*, the RO H and the $\mathcal{L}_{DDH}$ that consists of the vector of records of DDH-tests performed

by $\mathcal{O}_{\mathcal{C}}$, the CKEA-extractor $\mathcal{K}$ will successfully extract the corresponding discrete-logarithm of Y with overwhelming probability (except for a negligible probability of potential knowledge-extraction failure). Thus, except for a negligible probability, S never fails due to knowledge-extraction failure by the CKEA assumption. □

3.3 Implication of Concurrent and Forward Deniable Authentication

We note the 3-round adaptive tag-based CNMZKAOK protocol for DL, depicted in Figure 1, implies a 3-round concurrent and *forward* deniable authentication protocol, based on the CKEA assumption and the DL assumption.

Roughly speaking, an interactive protocol $\langle \hat{A}, \hat{B} \rangle$, where the authenticator $\hat{A}$ is of public-key A and the authentication verifier $\hat{B}$ does not necessarily possess public-key, is a concurrent and forward deniable authentication system, if (1) it is concurrent statistical zero-knowledge and (2) it is unforgeable against adaptive chosen message attacks. Here, unforgeability against adaptive chosen message attacks is with respect to any PPT many-left-one-right CMIM adversary $\mathscr{A}$ who tries to impersonate the player $\hat{A}$ to one instance of the $\hat{B}$ on the right while simultaneously interacting with polynomially many instances of $\hat{A}$ (with all instances of $\hat{A}$ being of the same public-key A) in the left CMIM-part by playing the role of $\hat{B}$. $\mathscr{A}$ can ask the instances of $\hat{A}$ to authenticate any messages chosen adaptively by $\mathscr{A}$, and the goal of $\mathscr{A}$ is to successfully authenticate a message m (in the name of $\hat{A}$) to $\hat{B}$ such that the message m was actually not authenticated by any of the instances of $\hat{A}$ in the left CMIM part.

It is easy to see that, by viewing messages to be authenticated as the session-tags, the many-right-one-left CMIM adversary $\mathscr{A}$ corresponds to a very restricted version of the CMIM adversary allowed in the experiment $\text{EXPE}(1^n, \overline{x}, \overline{TAG}, z)$ in the definition of adaptive tag-based CNMZKAOK. Specifically, in the game of defining unforgeability against adaptive chosen message attacks, the CMIM adversary $\mathscr{A}$ is only many-right-one-left, and more importantly, is restricted to only set session-tags adaptively (as the common inputs of all sessions are fixed to be the public-key of $\hat{A}$).

The 3-round adaptive tag-based CNMZKAOK protocol depicted in Figure 1 is trivially concurrent statistical ZK. The unforgeability against adaptive chosen message attacks is also satisfied under the CKEA assumption and the DL-assumption. Specifically, by viewing messages to be authenticated as the session-tags, if the many-left-one-right CMIM adversary can successfully authenticate a new massage m to $\hat{B}$ (i.e., finishing the one right-session with a session-tag different from those of all left-sessions), by the CKEA assumption the discrete-logarithm of A can be efficiently computed out, which violates the DL assumption.

On imposing *statistical* zero-knowledge for ensuring forward deniability. In the traditional definition of deniable authentication, sender Alice wants, in protecting her privacy, to prevent (possibly malicious) receiver Bob from proving to a third party that he has received a message from Alice. This is ensured by requiring that the protocol is actually a (computational) zero-knowledge protocol in the sense that Bob's view can be simulated.

However, as clarified in [44], there are scenarios in which deniability is actually a concern to the receiver's privacy. Consider the following example given in [44]: Alice and Bob are involved in some shady transaction, like drug-dealing or money laundering. Alice wants to make sure that her communications to Bob cannot be later linked to her, so she uses deniable authentication. Bob thinks that such communications are indeed deniable, stores all the messages in his hard disk. Later the operation is busted by the police and Alice and Bob end up in jail, and Bob's computer is seized. Alice is offered a sweet deal in exchange for her cooperation in linking Bob to the crime (Bob is claiming that the messages in his hard disk are not coming from Alice, that he never talked to her, actually does not even know her, all messages are just simulations!). Alice produces some piece of secret information (her secret-key for example) that indeed shows that the transcripts in

Bob's hard disk are actually authentic and not simulated. Bob ends up in jail, cursing himself for dropping out of crypto class in graduate school.

Another typical example where forward deniability is important is for electronic elections [44]. There, while it is important that both parties, the voting authority and the voter, authenticate each other (for the authority to know that the voter has the right to vote, for the voter to know that his or her vote will be counted), it is also mandatory to prevent either party from walking away with a non-repudiable proof of what the actual vote was. This application shows the importance of forward deniability: if the voter (sender) is authenticating his or her vote to the authority, not only the latter should not be able to prove to a third party how the voter voted, but even more importantly, the voter should not be able to do so either at a later stage, to prevent coercion and vote-selling.

The above examples show that deniability is not just a concern for the sender, but also for the receiver. What we would like to happen is that if the sender acts honestly during the protocol run, she should not be able at a later stage to claim the messages are authentic. This property is called *forward deniability*, as it has some affinity to the notion of forward secrecy. In particular, Di Raimondo and Gennaro have shown that computational ZK does not guarantee forward deniability [44], but statistical ZK does.

4 Deniable Internet Key-Exchange

The Internet Key-Exchange (IKE) protocols [66, 76], pioneered by Photuris [72], SKEME [78], Oakley [107] and SIGMA [79], are the *core* cryptographic protocols to ensure Internet security, which specifies key exchange mechanisms used to establish shared keys for use in the Internet Protocol Security (IPsec) standards [77]. The IPsec and IKE are intended to protect messages communicated *in the IP layer*, i.e., "layer 3" of ISO-OSI, which process the transmission of messages *using the network addresses possibly without knowing end-user peers' identities*. The IKE and IPsec can in turn be used to offer confidentiality, authentication and privacy for communication protocols in the higher layers of ISO-OSI.

The standard of IKE has gone through two generations (versions). The first generation IKEv1 [66] uses public-key encryption as the authentication mechanism (with SKEME serving as the basis in part), and IKEv2 [76] uses public-key signature as the authentication mechanism (with SIGMA serving as the basis in part). The SIGMA protocol is recalled in Appendix A.

In this section, we develop a new family of *non-malleable* and *deniable* DHKE protocols, which are named deniable Internet key-exchange (DIKE). The newly developed DIKE protocols are of conceptual simplicity and clarity, provide much remarkable privacy protection to protocol participants, are of highly practical (online) efficiency, and of well compatibility with the IKEv2 and SIGMA protocols as well as the widely standardized (H)MQV protocols [83, 80].

For the security of the DIKE protocols, we formulate the notion of *tag-based robust non-malleability* (TBRNM) for DHKE protocols, which essentially says that a CMIM adversary can successfully finish a session of a DHKE protocol only if it does know both the secret-key and the DH-exponent corresponding to the public-key and the DH-component alleged and sent by the CMIM adversary for that session. The TBRNM formulation takes security and privacy in an integrity, which particularly implies concurrent forward deniability for both the protocol initiator and the protocol responder. We show that the TBRNM security and the SK-security formulated in the Canetti-Krawczyk framework (CK-framework) [33] are mutually complementary, and thus much desirable to have DHKE protocols that enjoy both of them simultaneously. We show our DIKE protocol is indeed both TBRNM secure and SK-secure, which particularly implies the resistance against most security vulnerabilities not captured by the SK-security in the CK-framework.

On the desirability of (forward) deniability for IKE. For key-exchange protocols, both

security and privacy are desired, which would also have been being the major criterion underlying the evolution of a list of important industrial standards of DHKE. Among privacy concerns, deniability is an essential privacy property, and has always been a central concern in personal and business communications, with off-the-record communication serving as an essential social and political tool [44, 45, 96]. Given that many of these interactions now happen over digital media (email, instant messaging, web transactions, virtual private networks), it is of critical importance to provide these communications with "off-the-record" or deniability capability *for both the sender and the receiver*.[5]

In general, we can just run a forward deniable authentication protocol for each message to be sent when deniability of the message is desired. However, the beauty of using forward deniable key-exchange is that if the key-exchange protocol is deniable, then all the transactions (of public messages) using the session-key produced by the key-exchange protocol can be deniable (i.e., simulatable) for both the protocol participants. Moreover, for Internet key exchange (IKE) that is the *core* cryptographic protocol to ensure Internet security, offering deniability by IKE running at the IP layer within the IPsec standard [77] is much more desirable, because it enables various privacy services to be offered at the higher layers with uncompromised quality. Note that a privacy problem at the IP layer can cause irreparable privacy damage at the application layer. For example, an identity connected to an IP address, if not deniable, certainly nullifies an anonymous property offered by a fancy cryptographic protocol running at the application level. (If deniability is not desired, for some cases, then a non-repudiable proof can always be issued at the application level.)

4.1 DIKE Protocol Implementation

Let $(A = g^a, a)$ be the public-key and secret-key of the player $\hat{A}$, and $(X = g^x, x)$ be the DH-component and DH-exponent of $\hat{A}$; Let $(B = g^b, b)$ be the public-key and secret-key of the player $\hat{B}$, and $(Y = g^y, y)$ be the DH-component and DH-exponent of the $\hat{B}$, where a, b, x, y are taken randomly and independently from Z_q^*. The deniable Internet key-exchange protocol, for the main model, is depicted in Figure 2 (page 22). We assume the underlying public-key infrastructure (PKI) does not mandate players to prove to the certificate authority (CA) the possession or knowledge of their secret-keys during their key registration, but the CA will check the $G/1_G$ membership of each registered public-key. Also, each party checks the $G/1_G$ membership of the DH-component from its peer.

Note that, for the DIKE protocol described in Figure 2, the messages from one party do not bear the information of its peers's ID and public-key. Further, all authentic messages, $NMZK(\hat{B}, y)$ and $NMZK(b, y)$ (resp., $NMZK(a, x)$), from $\hat{B}$ (resp., $\hat{A}$) can be computed *merely* from its peer's DH-exponent x (resp., y).

On computational complexity of DIKE. Besides some hashing operations and the validation of peer's public-key certificate, the player $\hat{A}$ computes (Y^q, Y^a, Y^x) and (X, B^x), the player $\hat{B}$ computes (X^q, X^b, X^y) and (Y, A^y). Note that the computation of (Y^q, Y^a, Y^x) (resp., (X^q, X^b, X^y)) *in parallel* actually amounts to about 1.3 modular exponentiations. The DH-component X (resp., Y) can always be off-line pre-computed by $\hat{A}$ (resp., $\hat{B}$). Moreover, if the peer's identity is pre-specified, $\hat{A}$ (resp., $\hat{B}$) can further off-line pre-compute the value B^x (resp., A^y) with the pre-specified-peer $\hat{B}$ (resp., $\hat{A}$). That is, the total computational complexity at each player side is about 3.3 exponentiations, and the on-line computational complexity at each player side can remarkably be only 1.3 exponentiations!

[5] Needless to say, there are special applications where non-repudiable communication is essential, but this is not the case for most of our nowaday communications over Internet [44, 45] where deniable authentication is much more desirable than non-repudiable authentication.

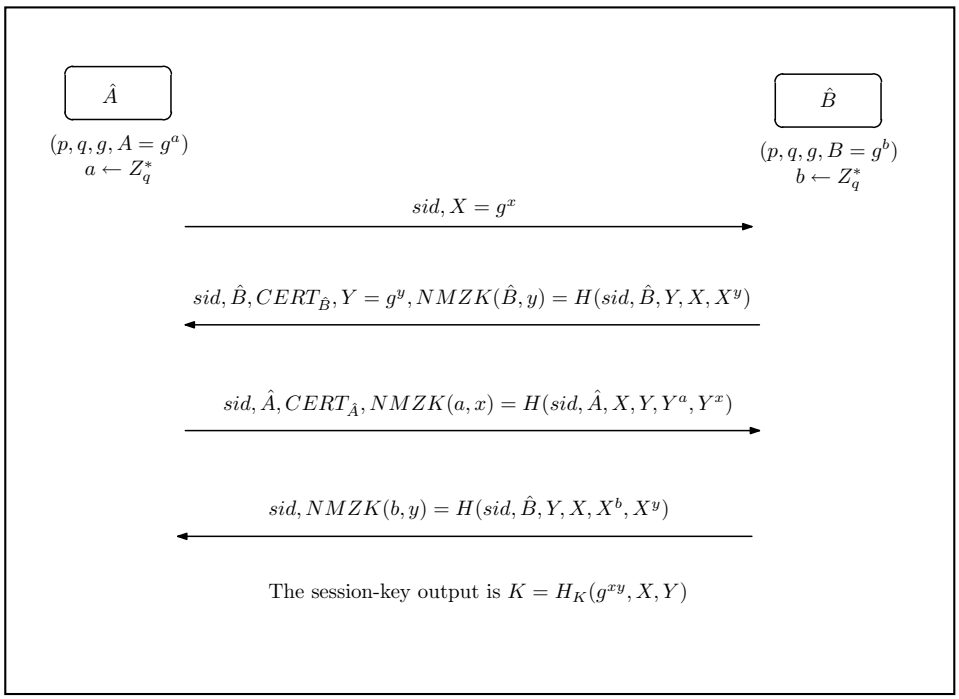


Figure 2: Deniable Internet key-exchange: the main model

4.2 Formulating Tag-Based Robust Non-Malleability for DHKE Protocols

Honest players and the underlying public-key infrastructure (PKI). For a DHKE protocol, there are polynomially many players running the protocol in the system. For presentation clarity, the execution instances of the DHKE protocol, between two players of identities $\hat{A}$ and $\hat{B}$, are denoted as the sessions of the protocol $\langle \hat{A}, \hat{B} \rangle$, in order to distinguish them from the executions of the same protocol among other players. Here, for presentation simplicity, in the following description we actually use the two identities $\hat{A}$ and $\hat{B}$ to stand for all uncorrupted players in the system. Also, for presentation simplicity, we assume the honest player $\hat{A}$ always plays the role of the session initiator, and the honest $\hat{B}$ always plays the role of the session responder. We remark that our security formulation and analysis apply to the general case, where each player can play both the role of the initiator and the role of the responder (in particular, the identity of the session-initiator and that of the session responder can be identical).

Let $(A = g^a, a)$ be the public-key and secret-key of the player $\hat{A}$, and $(X = g^x, x)$ be the DH-component and DH-exponent of $\hat{A}$; Let $(B = g^b, b)$ be the public-key and secret-key of the player $\hat{B}$, and $(Y = g^y, y)$ be the DH-component and DH-exponent of the $\hat{B}$, where a, b, x, y are taken randomly and independently from Z_q^*.

The underlying PKI does not mandate players to prove the possession or knowledge of their secret-keys during the key registration with the certificate authority, but the CA will check the order q and non-identity (i.e., $G/1_G$ membership) of registered public-keys. For presentation simplicity, we assume each party in the system has a unique identity, and the CA ensures each player identity corresponds to at most one valid registered public-key.

The concurrent man-in-the-middle adversary. We consider a general adversarial scenario, where polynomially many concurrent executing instances (sessions) of the protocol $\langle \hat{A}, \hat{B} \rangle$ take place in an asynchronous setting (appropriate for environments such as the Internet). In such a scenario, honest players are assumed oblivious of each other's existence, nor do they generally know the topology of the network, and thus cannot coordinate their executions. All communication channels, among all the concurrent sessions of the protocol $\langle \hat{A}, \hat{B} \rangle$, are assumed to be unauthenticated and

controlled by a probabilistic polynomial-time (PPT) CMIM adversary $\mathscr{A}$. This means that the honest player instances cannot directly communicate with each other, since all communication messages are done through the adversary. All honest player instances are working independently with independent random tapes in different sessions (but with the same public-key), and answer messages sent by $\mathscr{A}$ promptly. Once a session is finished, the honest players always erase the ephemeral (private) state information generated during the session, and only keep in privacy the session key output. Sessions can also be expired, and for expired sessions the session keys are also erased.

The CMIM adversary $\mathscr{A}$ (controlling all communication channels) can do whatever it wishes. In particular, $\mathscr{A}$ can interact with polynomial number of instances of $\hat{A}$ in the name of any player playing the role of the responder; such sessions are called the left-sessions. At the same time, $\mathscr{A}$ can interact with polynomial number of instances of $\hat{B}$ in the name of any player playing the role of the initiator; such sessions are called the right-sessions. For presentation simplicity, we assume the number of left-sessions is equal to that of right-sessions, which is $s(n)$ for some positive polynomial $s(\cdot)$ where n is the system parameter. The adversary $\mathscr{A}$ can decide to simply relay the messages of honest player instances. But, it can also decide to block, delay, divert, or modify messages arbitrarily at its wish.

The CMIM adversary $\mathscr{A}$ also takes some arbitrary auxiliary input $z \in \{0,1\}^*$, which captures arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from the executions of arbitrary (*possibly different*) protocols prior to its actual session interactions with the instances of $\hat{A}$ or $\hat{B}$. For example, the auxiliary input z may consist of a CDH triple (X, B, g^{xb}) that is collected over the Internet where $B = g^b$ is the public-key of the player $\hat{B}$, or just the secret-key b in case the CMIM attacker ever broke in the machine of $\hat{B}$. But, the auxiliary input z, collected prior to the actual session interactions of $\langle \hat{A}, \hat{B} \rangle$, is independent of the ephemeral messages (e.g., the DH-components) to be generated and exchanged by the instances of $\hat{A}$ and $\hat{B}$.[6]

We denote by $view_{\mathscr{A}}(1^n, \hat{A}, A, \hat{B}, B, z)$ the random variable describing the view of $\mathscr{A}$ in its concurrent interactions with the instances of $\hat{A}$ and $\hat{B}$, which includes the input $(1^n, \hat{A}, A, \hat{B}, B, z)$, its random tape, and all messages received in the $s(n)$ left sessions and the $s(n)$ right sessions (for protocols in the RO model, $\mathscr{A}$'s view also includes the RO). Here, for presentation simplicity, we have assumed that $\mathscr{A}$ concurrently interact with any polynomial number of instances of two players: one is the initiator player $\hat{A}$ and one is the responder player $\hat{B}$. In a way, the two players $\hat{A}$ and $\hat{B}$ (which can be identical) stand for all uncorrupted players in the system. In general, $\mathscr{A}$ can concurrently interact with any polynomial number of instances of any polynomial number of players. Our definitional framework, as well as the security analysis, can be extended to this general setting, by noting that honest players of different public-keys work independently.

Local sessions, session identifiers, and session tags. As in [33, 35], we assume each session (run at a party) in the system is assigned with a unique session-identifier, to bind transmitted messages to their corresponding sessions. As clarified in [33], it is assumed that the session-identifiers are set by some "higher layer" protocol that "calls" the KE protocol and ensures no two sessions (run at a party) are of identical session-identifier. Furthermore, it is left to the calling protocol to make sure that two parties who wish to exchange a key will activate matching sessions of the same session-identifier. Typically, setting session-identifiers may require some communication before the actual key-exchange sessions are activated, which is particularly this case for the IKE protocol of the IPsec standard [66].

To distinguish concurrent sessions of a DHKE protocol locally running at each *uncorrupted* player's side, each uncorrupted player can also simply identify each local session at its side by its identity and its DH-component sent in this session. Each session of a DHKE protocol run

[6] Specifically, we can consider an experiment where the ephemeral DH-components to be exchanged by the instances of uncorrupted players are generated only after the auxiliary string z is fixed.

at the side of an uncorrupted player is also labelled by a tag, which is the concatenation, *in the order of session initiator and then session responder*, of players' identities, public-keys, and DH-components. Sometimes, the components in a session-tag are categorized into the part of public and fixed components (e.g., player's identities and public-keys) succeeded by the part of ephemeral DH-component, where the components in each part are in the order of session initiator and then session responder. For example, $Tag = (\hat{A}, A, X, \hat{B}, B, Y)$ or $Tag = (\hat{A}, A, \hat{B}, B, X, Y)$ for a successfully finished session. Note that one of the players $\hat{A}$ or $\hat{B}$, *but not all*, in the tag may actually be impersonated or just controlled (i.e., corrupted) by the adversary. *We remark that the sessions only involving players controlled or impersonated by the CMIM adversary are actually viewed as activities inside the adversary.* For an incomplete or aborted session (particularly for DHKE in the post-specified-peer model of [35]), its tag is determined by the according elements included in the available current transcript. We call a session-tag a *complete* one, if it consists of all the (ordered) components: players' identities, public-keys and DH-components. Note that a session of a complete session-tag does not necessarily imply the session has already been successfully finished. For example, with respect to the concrete DIKE protocol described in Figure 2, the complete session-tag for a successfully completed session of the DIKE protocol is actually equal to that determined by the first three rounds of the session. Note also that the session-tag of a session does not include the session-identifier that is assumed to be set by some higher-layer calling protocol. In particular, two sessions of different session-tags may be of the same session-identifier.

Definition 4.1 (Tag-based robust non-malleability (TBRNM) for DHKE) *A DHKE protocol, $\langle \hat{A}, \hat{B} \rangle$, is called tag-based robust non-malleable, if for any PPT CMIM adversary $\mathscr{A}$ there exists a PPT simulator/extractor S such that for any sufficiently large n, any pair of uncorrupted players $\hat{A}$ and $\hat{B}$ (of public-key A and B respectively), and any auxiliary string $z \in \{0,1\}^*$, the output of $S(1^n, \hat{A}, A, \hat{B}, B, z)$ consists of two parts (str, sta) such that the following hold, where the auxiliary input z captures the arbitrary (possibly public-key dependent) information collected by $\mathscr{A}$ prior to its actual session interactions of $\langle \hat{A}, \hat{B} \rangle$ but is independent of the ephemeral messages (particularly, the DH-components) to be generated and exchanged by the instances of $\hat{A}$ and $\hat{B}$:*

- ***Statistical simulatability.*** *The following ensembles are statistically indistinguishable:* $\{view_{\mathscr{A}}(1^n, \hat{A}, A, \hat{B}, B, z)\}_{n \in N, \hat{A} \in \{0,1\}^*, A \in G/1_G, \hat{B} \in \{0,1\}^*, B \in G/1_G, z \in \{0,1\}^*}$ *and* $\{S_1(1^n, \hat{A}, A, \hat{B}, B, z)\}_{n \in N, \hat{A} \in \{0,1\}^*, A \in G/1_G, \hat{B} \in \{0,1\}^*, B \in G/1_G, z \in \{0,1\}^*}$, *where* $S_1(1^n, \hat{A}, A, \hat{B}, B, z)$ *denotes (the distribution of) the first output of* $S(1^n, \hat{A}, A, \hat{B}, B, z)$, *i.e., str.*
- ***Knowledge extraction.*** *sta consists of a set of* $2s(n)$ *strings,* $\{\tilde{w}_1^l, \tilde{w}_2^l, \cdots, \tilde{w}_{s(n)}^l, \tilde{w}_1^r, \tilde{w}_2^r, \cdots, \tilde{w}_{s(n)}^r\}$, *satisfying the following:*
  - *For any i, $1 \leq i \leq s(n)$, if the i-th left-session (resp., right-session) in str is aborted or with a tag identical to that of one of the right-sessions (resp., left-sessions), then $\tilde{w}_i^l = \perp$ (resp., $\tilde{w}_i^r = \perp$);*
  - *Otherwise, i.e., the i-th left-session (resp., right-session) in str is successfully completed and is of session-tag different from those of all right-sessions (resp., left-sessions), then $\tilde{w}_i^l = (\tilde{b}_i^l, \tilde{y}_i^l)$ (resp., $\tilde{w}_i^r = (\tilde{a}_i^r, \tilde{x}_i^r)$), where $\tilde{b}_i^l$ (resp., $\tilde{a}_i^r$) is the discrete-logarithm of the public-key $\tilde{B}_i^l$ (resp., $\tilde{A}_i^r$) set and alleged by the CMIM adversary $\mathscr{A}$ for the i-th left-session (resp., right-session) in the name of $\hat{\tilde{B}}_i^l$ (resp., $\hat{\tilde{A}}_i^r$), and $\tilde{y}_i^l$ (resp., $\tilde{x}_i^r$) is the discrete-logarithm of the DH-component $\tilde{Y}_i^l$ (resp., $\tilde{X}_i^r$) set and sent by the CMIM adversary $\mathscr{A}$ in the i-th left-session (resp., right-session).*

Furthermore, we say the DHKE protocol $\langle \hat{A}, \hat{B} \rangle$ is of privacy-preserving TBRNM, if it additionally satisfies: (1) the transcript of each session can be generated merely from the DH-exponents

(along with some public system parameters, e.g., players' public-key and identity information, etc); (2) messages from one party do not bear the identity and public-key information of its peer.

Notes on the (privacy-preserving) TBRNM definition. The TBRNM definition for DHKE is in the spirit of the definition of adaptive tag-based CNMZKAOK, which can actually be viewed as an extended and much strengthened version of the latter.

Recall that, for the auxiliary input z of the CMIM adversary $\mathscr{A}$, we only require it to be independent of the ephemeral messages (particularly, the DH-components) to be generated and exchanged by the instances of $\hat{A}$ and $\hat{B}$. The auxiliary input z can include player's public-keys, static secret-keys, public system parameters, and actually arbitrary information collected/eavesdropped by $\mathscr{A}$ *prior to its actual session interactions with the instances of* $\hat{A}$ *or* $\hat{B}$. That is, the TBRNM formulation implicitly captures the adversarial leakage of players' static secret-keys. But, we reminder that the TBRNM formulation does not take into account the following abilities of the CMIM adversary in: exposing ephemeral private state for incomplete sessions, exposing session-keys for completed sessions, and party corruption.

Note that the (privacy-preserving) TBRNM formulation takes privacy and security as an integrity. On the one hand, the TBRNM definition clearly implies concurrent forward deniability (i.e., concurrent statistical zero-knowledge) for both the initiator $\hat{A}$ and the responder $\hat{B}$; Moreover, *privacy-preserving* TBRNM provides additional privacy protection for the information of players' identities and public-keys. On the other hand, the TBRNM definition essentially ensures that: the CMIM adversary $\mathcal{A}$ can successfully finish a (whether left or right) session of *distinct* session-tag, only if it does "know" both the secret-key and DH-exponent corresponding to the public-key and DH-component alleged and sent by the CMIM adversary in that session; Furthermore, this ensurence holds even if the session is of the same session-identifier with another session (recall the session-tag does not include the session-identifier that is set by some "higher-layer" calling protocol). We suggest this kind of security guarantee is very essential to DHKE protocols, particularly when they are run concurrently over the Internet.

4.3 TBRNM Analysis of the DIKE Protocol

In this section, we prove the following theorem:

Theorem 4.1 *The DIKE protocol depicted in Figure 2 is* privacy-preserving *tag-based robust non-malleable in the restricted random oracle model, under the strong Diffie-Hellman (SDH) assumption and the CKEA assumption.*

Proof (outline). On inputs of $(1^n, \hat{A}, A, \hat{B}, B, z)$, the simulator/extractor S works as follows, by running the CMIM adversary $\mathscr{A}$ as the subroutine and with access to the restricted random oracles H, H_K (note that S does not know the secret-keys a or b). In the following description, we use the notation $\tilde{m}$ to denote a message actually set and sent by the CMIM adversary adaptively based on its view.

Restricted random oracle H, H_K. S can only define the RO answers to queries first made by the CMIM adversary and (the rest part of) the RO is unprogrammable for all other queries. Note that the RO H_K used to generate session-key is trivially restricted (i.e., unprogrammable). S stores the list of all pairs of RO-query and RO-answer, and distinguishes each pair as unprogrammable RO pair or programmable RO pair.

DDH-test list $\mathcal{L}_{DDH}$. During the simulation and knowledge-extraction, S will perform a list of DDH-tests, i.e., checking whether $Z = CDH(X, Y)$ for $(X = g^x, Y = g^y, Z) \in G^3$ where S possesses x or y. For each DDH-test on (X, Y, Z), S stores a record (X, Y, Z, θ) in a list $\mathcal{L}_{DDH}$, where θ is a predicate such that $\theta = 1$ if and only if $Z = CDH(X, Y)$.

Left-session first-round simulation. For each i, $1 \leq i \leq s(n)$, at the first-round of the i-th left session, S acts just as the instance of the honest initiator for the i-th left-session, denoted A_i^l, does. Specifically, it generates $X_i^l = g^{x_i^l}$, and sends $\{sid_i^l, X_i^l\}$ to $\tilde{\hat{B}}_i^l$, where x_i^l is randomly taken from Z_q^* by S, sid_i^l is the session-id for the i-th left-session, and $\hat{B}_i^l$ is the responder player *impersonated* by the CMIM adversary $\mathscr{A}$ for the i-th left-session.

Right-session second-round simulation. For each j, $1 \leq j \leq s(n)$, at the second-round of the j-th right session, after receiving the first-round message $\{sid_j^r, \tilde{X}_j^r\}$ from the initiator player $\tilde{\hat{A}}_j^r$ (*impersonated* by the CMIM adversary $\mathscr{A}$ for the j-th right-session), S first checks whether $\tilde{X}_j^r \in G/1_G$ (and aborts the j-th right session in case $\tilde{X}_j^r \notin G/1_G$). If $\tilde{X}_j^r \in G/1_G$, S does the following: S generates $Y_j^r = g^{y_j^r}$, where y_j^r is taken randomly from Z_q^*; Then, S checks whether the RO H has been defined on the point $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$. This is done by checking all queries to H of the form $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{\vartheta}_j^r)$ for some $\tilde{\vartheta}_j^r \in G$, and testing whether $\tilde{\vartheta}_j^r = CDH(Y_j^r, \tilde{X}_j^r) = (\tilde{X}_j^r)^{y_j^r}$ (all DDH-test records will be stored into $\mathcal{L}_{DDH}$); If H has been defined on $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$, S just sets $NMZK(\hat{B}, y_j^r)$ to be the already defined $H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$, otherwise, S sets $NMZK(\hat{B}, y_j^r)$ to be a random value (taken uniformly at random) in $\{0,1\}^{|q|}$; Finally, S sends to $\mathscr{A}$ $\{sid_j^r, \hat{B}, CERT_{\hat{B}}, Y_j^r, NMZK(\hat{B}, y_j^r)\}$ as the second-round message of the j-th right session.

In case $NMZK(\hat{B}, y_j^r)$ is set to be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary $\mathscr{A}$ queries the RO H with $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r)$ for some value $\tilde{Z}_j^r \in G$, S tests whether $\tilde{Z}_j^r = CDH(Y_j^r, \tilde{X}_j^r) = (\tilde{X}_j^r)^{y_j^r}$ (recall that all DDH-test records will be stored in $\mathcal{L}_{DDH}$). If it is this case, S just defines the RO answer, i.e., $H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r = (\tilde{X}_j^r)^{y_j^r})$, to be the already chosen $NMZK(\hat{B}, y_j^r)$ (otherwise, S defines the RO answer to be a random value in $\{0,1\}^{|q|}$, provided that multiple identical queries are answered with the same answer). Note that if $\mathscr{A}$ never queries the RO with $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{Z}_j^r)$, the random oracle H on this point remains undefined. *In particular, S never defines $H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r))$ on its own (S defines it only when $\mathscr{A}$ makes such a query).*

Left-session second-round knowledge-extraction and third-round simulation. For each i, $1 \leq i \leq s(n)$, after receiving the second-round message, $(sid_i^l, \tilde{\hat{B}}_i^l, CERT_{\tilde{\hat{B}}_i^l}, \tilde{Y}_i^l, \widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l))$, from the CMIM adversary $\mathscr{A}$ in the i-th left session, where $\tilde{\hat{B}}_i^l$ is the player (of public-key $\tilde{B}_i^l$) impersonated by $\mathscr{A}$ for the i-th left-session, S first checks the validity of $CERT_{\tilde{\hat{B}}_i^l}$ and the $G/1_G$ membership of $\tilde{Y}_i^l$, and aborts the i-th left-session in case any check fails. Then, S does the following:

Case-L1. $(sid_i^l, X_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, \widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l)) = (sid_j^r, \tilde{X}_j^r, \hat{B}, Y_j^r, NMZK(\hat{B}, y_j^r))$, for some j, $1 \leq j \leq s(n)$. That is, the transcript of the first two rounds of the i-th left-session (except for the public-key certificate $CERT_{\tilde{\hat{B}}_i^l}$) is identical to that of the first-two rounds of the j-th right-session (except for the public-key certificate $CERT_{\hat{B}}$) for some j, $1 \leq j \leq s(n)$. *Note that this case implies $\tilde{Y}_i^l = Y_j^r$, where $Y_j^r = g^{y_j^r}$ with y_j^r known by S.*

In this case, S checks whether the RO H is defined on the point $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(\tilde{Y}_i^l, A), CDH(\tilde{Y}_i^l, X_i^l))$; This is done by checking all queries to H of the form $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, \tilde{\alpha}_i^l, \tilde{\beta}_i^l)$ for some $(\tilde{\alpha}_i^l, \tilde{\beta}_i^l) \in G^2$, and testing whether $\tilde{\alpha}_i^l = CDH(\tilde{Y}_i^l, A) = CDH(Y_j^r, A) = A^{y_j^r}$ and $\tilde{\beta}_i^l = CDH(\tilde{Y}_i^l, X_i^l) = CDH(Y_j^r, X_i^l) = (X_i^L)^{y_j^r}$ (recall all the DDH-test records are

stored into $\mathcal{L}_{DDH}$). If H has been defined over $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(\tilde{Y}_i^l, A), CDH(\tilde{Y}_i^l, X_i^l))$ then S just sets $NMZK(a, x_i^l)$ to be the already defined $H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(\tilde{Y}_i^l, A), CDH(\tilde{Y}_i^l, X_i^l))$, otherwise, S sets $NMZK(a, x_i^l)$ to be a random value in $\{0,1\}^{|q|}$. Then, S sends $\{sid_i^l, \hat{A}, CERT_{\hat{A}}, NMZK(a, x_i^l)\}$ to $\mathscr{A}$ as the third-round message of the i-th left-session.

In case $NMZK(a, x_i^l)$ is set to be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary $\mathscr{A}$ queries the random oracle H with $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, \tilde{\alpha}_i^l, \tilde{\beta}_i^l)$ for some $(\tilde{\alpha}_i^l, \tilde{\beta}_i^l) \in G^2$, S tests whether $\tilde{\alpha}_i^l = CDH(\tilde{Y}_i^l, A)$ and $\tilde{\beta}_i^l = CDH(\tilde{Y}_i^l, X_i^l)$ (and stores the DDH-test records into $\mathcal{L}_{DDH}$). If it is this case, S just defines the RO answer, i.e., $H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, \tilde{\alpha}_i^l, \tilde{\beta}_i^l)$, to be (the already fixed) $NMZK(a, x_i^l)$. Note that if $\mathscr{A}$ never queries the random oracle H with $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(\tilde{Y}_i^l, A), CDH(\tilde{Y}_i^l, X_i^l))$, the RO on this point remains undefined. *In particular, S never defines $H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(\tilde{Y}_i^l, A), CDH(\tilde{Y}_i^l, X_i^l))$ on its own (S defines it only when $\mathscr{A}$ makes such a query).*

Case-L2. For some j, $1 \le j \le s(n)$, $(sid_i^l, X_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l) = (sid_j^r, \tilde{X}_j^r, \hat{B}, Y_j^r)$, but $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l) \neq NMZK(\hat{B}, y_j^r)$. That is, Case-L2 is identical to Case-L1, except for that the authenticated values $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l)$ and $NMZK(\hat{B}, y_j^r)$ are different. In this case, S simply aborts the i-th left-session.

Case-L3. In all other cases, it implies that no honest player instance ever sent the *correct* value that $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l)$ should be.

In this case, S first checks whether $\mathscr{A}$ has queried the RO H with $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l)$ for $\tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l) = (\tilde{Y}_i^l)^{x_i^l}$. This is done by testing all the RO-queries to H made by $\mathscr{A}$ one by one (and each DDH-test record will be stored into $\mathcal{L}_{DDH}$). If not, S simply aborts the i-th left session; Otherwise, i.e., $\mathscr{A}$ queried H with $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l))$, S further checks whether $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l) = H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l))$, and aborts the i-th left-session if the check fails.

If all the above checks are successful, i.e., $\mathscr{A}$ queried H with $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l))$ and $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l) = H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l))$, S runs the CKEA-extractor $\mathcal{K}$ on inputs of $(1^n, \hat{A}, A, \hat{B}, B, z)$, random coins of S (actually the partial random coins used up to $\mathscr{A}$ outputting $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{Z}_i^l = CDH(\tilde{Y}_i^l, X_i^l))$ *except the coins used to generate the DH-components X_i^l's and Y_j^r's (particularly the DH-exponents x_i^l's and y_j^r's)*, the RO H, H_K (i.e, all the RO queries and RO answers up to now) and the DDH-test list $\mathcal{L}_{DDH}$, to try to extract the discrete logarithm, denoted $\tilde{y}_i^l$, of $\tilde{Y}_i^l = g^{\tilde{y}_i^l}$.

Case-L3.1. If the knowledge-extractor $\mathcal{K}$ fails in extracting $\tilde{y}_i^l$, S outputs "failure" and stops.

Case-L3.2. If $\mathcal{K}$ successfully outputs $\tilde{y}_i^l$, S computes $CDH(A, \tilde{Y}_i^l) = (A)^{\tilde{y}_i^l}$. Then, S checks whether the RO H has been defined on $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$, and if not S makes the (fixed) RO-query $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ to H. Finally, S sets $NMZK(a, x_i^l) = H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$, and sends $\{sid_i^l, \hat{A}, CERT_{\hat{A}}, NMZK(a, x_i^l)\}$ to $\hat{A}$ as the third-round message of the i-th left session.

Right-session third-round knowledge-extraction, and fourth-round simulation. For each j, $1 \le j \le s(n)$, after receiving the third-round message of the j-th right session, denoted $\{sid_j^r, \tilde{\hat{A}}_j^r, CERT_{\tilde{\hat{A}}_j^r}, \widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r)\}$, from the initiator player $\tilde{\hat{A}}_j^r$ of public-key $\tilde{A}_j^r$ (imper-

sonated by the CMIM adversary $\mathscr{A}$ for the j-th right-session), S first checks the validity of the public-key certificate $CERT_{\tilde{A}_j^r}$ (and aborts if the check fails). Then, S works as follows:

Case-R1. $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = NMZK(a, x_i^l)$ for some i, $1 \leq i \leq s(n)$. That is, $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = NMZK(a, x_i^l)$ was sent by the honest player $\hat{A}$ in the i-th left-session. We distinguish several subcases:

Case-R1.1. $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r) \neq (sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l)$, which violates $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = NMZK(a, x_i^l)$ in the RO model with overwhelming probability. In this case, S simply aborts the j-th right-session.

Case-R1.2. $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r) = (sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l)$. We further distinguish several subcases:

Case-R1.2.1 The session-tag of the j-th right-session is different from the session-tag of the i-th left-session. Note that, in this case, the session-tag (determined by the first three-round transcript) of the j-th right session is $(\tilde{\hat{A}}_j^r, \tilde{A}_j^r, \tilde{X}_j^r, \hat{B}, B, Y_j^r)$. As the session-tag (determined by the first three-round transcript) of the i-th left-session is $(\hat{A}, A, X_i^l, \tilde{\hat{B}}_i^l, \tilde{B}_i^l, \tilde{Y}_i^l)$, we conclude that $\tilde{\hat{B}}_i^l \neq \hat{B}$ (recall that the underlying PKI ensures each player identity corresponds to at most one valid registered public-key). In this case, S outputs "failure" and stops.

Case-R1.2.2 The session-tag of the j-th right-session is identical to the session-tag of the i-th left-session. Recall that Case-R1 has assumed $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = NMZK(a, x_i^l)$, and thus Case-R.1.2.2 implies that both the i-th left-session and the j-th right session are not aborted up to now, and the first three-round transcripts (except for public-key certificates) of the j-th right-session and the i-th left-session are identical. Note that in this case, by the definition of tag-based robust non-malleability, S does not need to extract the secret-key $\tilde{a}_j^r$ and the DH-exponent $\tilde{x}_j^r$.

In this case, S checks whether the RO H has been defined on the point $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$. This is done by checking all queries to H of the form $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{\lambda}_j^r, \tilde{\psi}_j^r)$ for some $(\tilde{\lambda}_j^r, \tilde{\psi}_j^r) \in G^2$, and testing whether $\tilde{\lambda}_j^r = CDH(\tilde{X}_j^r, B) = CDH(X_i^l, B) = B^{x_i^l}$ and $\tilde{\psi}_j^r = CDH(\tilde{X}_j^r, Y_j^r) = CDH(X_i^l, Y_j^r) = (Y_j^r)^{x_i^l}$. If H has been defined on $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$, S just sets $NMZK(b, y_j^r)$ to be $H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$, otherwise, S sets $NMZK(b, y_j^r)$ to be a random value in $\{0,1\}^{|q|}$. Finally, S sends $\{sid_j^r, NMZK(b, y_j^r)\}$ to $\mathscr{A}$ as the fourth-round message, thereby finishing the j-th right session.

In case $NMZK(b, y_j^r)$ is set to be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary $\mathscr{A}$ queries the RO H with $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{\lambda}_j^r, \tilde{\psi}_j^r)$ for some $(\tilde{\lambda}_j^r, \tilde{\psi}_j^r) \in G^2$, S tests whether $\tilde{\lambda}_j^r = CDH(\tilde{X}_j^r, B)$ and $\tilde{\psi}_j^r = CDH(\tilde{X}_j^r, Y_j^r)$ (and stores the DDH-test records into $\mathcal{L}_{DDH}$). If it is this case, S just defines the RO answer, i.e., $H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, \tilde{\lambda}_j^r, \tilde{\psi}_j^r)$, to be (the already fixed) $NMZK(b, y_j^r)$. Note that if $\mathscr{A}$ never queries the random oracle H with $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$, the RO on this point remains undefined. *In particular, S never defines the RO on this point on its own (S defines it only when $\mathscr{A}$ makes such a query).*

Case-R2. $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) \neq NMZK(a, x_i^l)$ for all i, $1 \leq i \leq s(n)$. That is, $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r)$ was not sent by $\hat{A}$ in one of left-sessions.

Recall that the correct value that $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r)$ should be is $H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$. By the uniqueness of session-identifiers, i.e., sid_j^rs, of local sessions at the side of $\hat{B}$, the *correct* value that $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r)$ should be also was not sent by $\hat{B}$ in any right-session, even if $\tilde{\hat{A}}_j^r = \hat{B}$ (e.g., the adversary $\mathscr{A}$ is making a reflection attack against $\hat{B}$ in the j-th right-session). We conclude that no honest player ever sent the value $H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$, i.e., the *correct* value that $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r)$ should be, in any existing session.

In this case, S first checks whether $\mathscr{A}$ has queried the RO H with $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, \tilde{\alpha}_j^r, \tilde{\beta}_j^r)$ for $\tilde{\alpha}_j^r = CDH(Y_j^r, \tilde{A}_j^r) = (\tilde{A}_j^r)^{y_j^r}$ and $\tilde{\beta}_j^r = CDH(Y_j^r, \tilde{X}_j^r) = (\tilde{X}_j^r)^{y_j^r}$. This is done by testing all the RO-queries to H made by $\mathscr{A}$ one by one (recall each DDH-test record will be stored into $\mathcal{L}_{DDH}$). If not, S simply aborts the j-th right-session; Otherwise, i.e., $\mathscr{A}$ queried H with $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, \tilde{\alpha}_j^r = CDH(Y_j^r, \tilde{A}_j^r), \tilde{\beta}_j^r = CDH(Y_j^r, \tilde{X}_j^r))$, S further checks whether $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$, and aborts the j-th right-session if the check fails.

Finally, i.e., $\mathscr{A}$ has queried H with $H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$ and $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$, S runs the CKEA-extractor $\mathcal{K}$ on inputs of $(1^n, \hat{A}, A, \hat{B}, B, z)$, random coins of S (actually the partial random coins used up to $\mathscr{A}$ outputting $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(Y_j^r, \tilde{A}_j^r), CDH(Y_j^r, \tilde{X}_j^r))$ *except the coins used to generate the DH-components X_i^l's and Y_j^r's (particularly the DH-exponents x_i^l's and y_j^r's)*, the RO H, H_K (i.e, all the RO queries and RO answers up to now) and the DDH-test list $\mathcal{L}_{DDH}$, to try to extract the discrete logarithms, denoted $\tilde{a}_j^r$ and $\tilde{x}_j^r$, of the public-key $\tilde{A}_j^r = g^{\tilde{a}_j^r}$ and the DH-component $\tilde{X}_j^r = g^{\tilde{x}_j^r}$ (alleged and sent by $\mathscr{A}$ in the j-th right session in the name of $\tilde{\hat{A}}_j^r$).

Case-R2.1. If the knowledge-extractor $\mathcal{K}$ fails in extracting $\tilde{a}_j^r$ or $\tilde{x}_j^r$, S outputs "failure" and stops.

Case-R2.2. If $\mathcal{K}$ successfully outputs $\tilde{a}_j^r$ and $\tilde{x}_j^r$, S sets $\tilde{w}_j^r = (\tilde{a}_j^r, \tilde{x}_j^r)$ and computes $CDH(\tilde{X}_j^r, B) = B^{\tilde{x}_j^r}$ and $CDH(\tilde{X}_j^r, Y_j^r) = (Y_j^r)^{\tilde{x}_j^r}$. Then, S checks whether the RO H has been defined on $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$, and if not S makes the (fixed) RO-query $(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$ to H. Finally, S sets $NMZK(b, y_j^r) = H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(\tilde{X}_j^r, B), CDH(\tilde{X}_j^r, Y_j^r))$, and sends $\{sid_j^r, NMZK(b, y_j^r)\}$ to $\mathscr{A}$ as the fourth-round message of the j-th right-session, thereby finishing the j-th right-session.

Left-session fourth-round knowledge-extraction. For each i, $1 \leq i \leq s(n)$, after receiving $\{sid_i^l, \widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)\}$ from $\mathscr{A}$ in the fourth-round of the i-th left-session (in which $\mathscr{A}$ interacts with $\hat{A}$ in the name of $\tilde{\hat{B}}_i^l$ of public-key $\tilde{B}_i^l$ and sent $\tilde{Y}_i^l$ at the second-round), S does the following:

Case-L4. $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) = NMZK(b, y_j^r)$ for some j, $1 \leq j \leq s(n)$. That is, $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ was sent by the honest player $\hat{B}$ at the fourth-round of the j-th right-session, for some j, $1 \leq j \leq s(n)$. We further distinguish several subcases:

Case-L4.1. $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l) \neq (sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r)$. In this case (which implies $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ is invalid), S simply aborts the i-th left-session.

Case-L4.2. $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l) = (sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r)$. In this case, we further distinguish two subcases:

Case-L4.2.1. The session-tag of the i-th left-session is different from the session-tag of the j-th right-session. Note that, for Case-L4.2, the session-tag of the i-th left-session is $(\hat{A}, A, X_i^l, \tilde{\hat{B}}_i^l, \tilde{B}_i^l, \tilde{Y}_i^l)$, and the session-tag of the j-th right-session is $(\tilde{\hat{A}}_j^r, \tilde{A}_j^r, \tilde{X}_j^r, \hat{B}, B, Y_j^r)$. As Case-L.4.2 has assumed $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l) = (sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r)$, it implies $\hat{A} \neq \tilde{\hat{A}}_j^r$. In this case, S outputs "failure" and stops.

Case-L4.2.2. The session-tag of the i-th left-session is identical to the session-tag of the j-th right-session, which implies S does not need to make knowledge-extraction in this case. In this case, S accepts and finishes the i-th left-session.

Case-L5. $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) \neq NMZK(b, y_j^r)$ for all j, $1 \leq j \leq s(n)$. That is, $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ was *not* sent by the honest player $\hat{B}$ in any right-session.

Recall the correct value that $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ should be is $H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$. Due to the uniqueness of session-identifiers sid_i^l's (of the sessions run by $\hat{A}$), with overwhelming probability the value $H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ cannot be sent by $\hat{A}$ in any left-session other than the i-th left-session. But, in case $\tilde{\hat{B}}_i^l = \hat{A}$ (e.g., $\mathscr{A}$ is making a reflection attack against $\hat{A}$ in the i-th left-session), it is possible that $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ was sent by $\hat{A}$ at the third-round of the i-th left-session. Below, we further distinguish two cases:

Case L5.1 $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) = NMZK(a, x_i^l)$. That is, $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ is just copied from the third-round message sent by $\hat{A}$ in the i-th left-session.

In this case, S checks whether $\tilde{\hat{B}}_i^l = \hat{A}$ and $\tilde{Y}_i^l = X_i^l$. If not (which implies the value $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ cannot be valid), S aborts the i-th left-session. If yes (which implies a successful reflection attack by $\mathscr{A}$ against $\hat{A}$ in the i-th left-session), S stops and outputs "failure".

Case L5.2 $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) \neq NMZK(a, x_i^l)$. This means that no uncorrupted player ever sent the correct value that $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l)$ should be in any existing session.

In this case, S checks whether $\mathscr{A}$ has queried the RO H with $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$. This is done by checking all RO-queries made by $\mathscr{A}$ to H of the form $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, \tilde{\lambda}_i^l, \tilde{\psi}_i^l)$ for some $(\tilde{\lambda}_i^l, \tilde{\psi}_i^l) \in G^2$, and testing whether $\tilde{\lambda}_i^l = CDH(X_i^l, \tilde{B}_i^l) = (\tilde{B}_i^l)^{x_i^l}$ and $\tilde{\psi}_i^l = CDH(X_i^l, \tilde{Y}_i^l) = (\tilde{Y}_i^l)^{x_i^l}$ (recall all DDH-test records will be stored into $\mathcal{L}_{DDH}$). If $\mathscr{A}$ did not make such a query, S aborts the i-th left-session; otherwise, S further checks whether $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) = H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ and aborts the i-th left-session if not.

Finally, i.e., $\mathscr{A}$ made the RO-query $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ to H and $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) = H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$, S runs the CKEA-extractor $\mathcal{K}$ on inputs of $(1^n, \hat{A}, A, \hat{B}, B, z)$, random coins of S (actually the partial random coins used up to $\mathscr{A}$ outputting $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(X_i^l, \tilde{B}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ *except the coins used to generate the DH-components* X_i^l*'s*

and Y_j^r's (particularly the DH-exponents x_i^l's and y_j^r's), the RO H, H_K and the DDH-test list $\mathcal{L}_{DDH}$, to try to extract the discrete logarithms, denoted $\tilde{b}_i^l$ and $\tilde{y}_i^r$, of the public-key $\tilde{B}_i^l = g^{\tilde{b}_i^l}$ and the DH-component $\tilde{Y}_l^l = g^{\tilde{y}_i^l}$ (alleged and sent by $\mathscr{A}$ in the i-th left-session in the name of $\tilde{\hat{B}}_i^l$).

Case-L5.2.1. If the knowledge-extractor $\mathcal{K}$ fails in extracting $\tilde{b}_i^l$ or $\tilde{y}_i^r$, S outputs "failure" and stops.

Case-L5.2.2 If $\mathcal{K}$ successfully outputs $\tilde{b}_i^l$ and $\tilde{y}_i^r$, S sets $\tilde{w}_i^l = (\tilde{b}_i^l, \tilde{y}_i^r)$, and finishes the i-th left-session.

Output of S. During the simulation, whenever $\mathscr{A}$ aborts a session S also automatically aborts that session, and sets the corresponding $\tilde{w}_i^l$ or $\tilde{w}_j^r$ to be $\perp$. For a successfully finished left-session (resp., right-session) of session-tag identical to that of one of right-sessions (resp., left-sessions), S also sets $\tilde{w}_i^l$ (resp., $\tilde{w}_j^r$) to be $\perp$.[7] In case S does not stop outputting "failure", S outputs (str, sta), where sta includes $\{\tilde{w}_1^l, \cdots, \tilde{w}_{s(n)}^l, \tilde{w}_1^r, \cdots, \tilde{w}_{s(n)}^r\}$, and str is set to be the set of the input $(1^n, \overline{x}, \overline{TAG}, z)$, the random coins of $\mathscr{A}$, all messages sent by S emulating the honest player instances of $\hat{A}$ and $\hat{B}$ (Round-1 and Round-3 messages of left-sessions, Round-2 and Round-4 messages of right-sessions), and the RO H and H_K.

Analysis of the simulator/extractor S. First, it is clear that S works in *strict* polynomial-time, and never rewinds the CMIM adversary (run as its subroutine), and thus the simulation and knowledge-extraction of S are *straight-line*. Also, S only defines (programs) the RO H and H_K on queries first made by the CMIM adversary $\mathscr{A}$, and the rest part of the RO is unprogrammable, and thus S works in the *restricted* RO model.

Similar to the proof of Theorem 3.1, it is easy to check that suppose S never stop outputting "failure" the simulated view of str is statistically indistinguishable from the real view of the CMIM adversary. Next, we show the probability S stops outputting "failure" is negligible.

Proposition 4.1 *The probability that S stops outputting "failure", in Case-L5.1, is negligible (in the restricted RO model) under the computational Diffie-Hellman assumption.*

Proof (of Proposition 4.1). Note that the failure of S in Case-L5.1 means a successful reflection attack by $\mathscr{A}$ against $\hat{A}$ in the i-th left-session. In particular, after receiving X_i^l from $\hat{A}$ at the first-round of the i-th left-session, $\mathscr{A}$ successfully sends back, in the same name of $\tilde{\hat{B}}_i^l = \hat{A}$, the same DH-component $\tilde{Y}_i^l = X_i^l$ and the correct value $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l) = \widetilde{NMZK}(\hat{A}, x_i^l) = H(sid_i^l, \hat{A}, X_i^l, X_i^l, CDH(X_i^l, X_i^l))$. Clearly, no honest player ever sent the value $\widetilde{NMZK}(\tilde{\hat{B}}_i^l, \tilde{y}_i^l)$, thus in the RO model with overwhelming probability $\mathscr{A}$ has to query the RO H with $(sid_i^l, \hat{A}, X_i^l, X_i^l, CDH(X_i^l, X_i^l))$. That is, suppose S fails in Case-L5.1 with non-negligible probability p, it implies $\mathscr{A}$ will output $CDH(X_i^l, X_i^l)$ also with the same probability (except a negligible probability).

We remark that the hardness of computing out $CDH(X, X) = X^x$ from a random element $X = g^x \in G$ is equivalent to that of the ordinary CDH problem [97]. Specifically, given $X = g^x$, we can prepare two random values for the CDH solver as follows: $X_1 = Xg^{r_1}$ and $X_2 = Xg^{r_2}$, then the CDH solver for the instance (X_1, X_2) can be used to compute X^x; On the other hand, if one can efficiently compute out X^x from X, we can solve the CDH problem as follows: given two (random) elements X and Y, we first compute out X^x, Y^y, and $(XY)^{x+y} = (g^{x+y})^{x+y}$, then we get $g^{xy} = ((XY)^{x+y}/(X^xY^y))^{1/2}$.

[7] We remark that this treatment is only for fitting the definition of TBRNM and for presentation simplicity. Actually, in this case, the $\tilde{w}_i^l$ or $\tilde{w}_j^r$ may have already been set to be the correctly extracted discrete logarithms (i.e., the corresponding secret-key and DH-exponent) in Case-R.2 and Case-L5.2.2 (when two sessions are of the same session-tag but different session-identifiers).

Now, we consider another algorithm $S'(1^n, \hat{A}, A, a, \hat{B}, B, b, z, X)$, where X is a value taken randomly and independently from G. Note that S' knows the secret-keys of $\hat{A}$ and $\hat{B}$. The goal of S' is to compute $CDH(X, X)$, thus violating the CDH assumption. To this end, S' first chooses i uniformly at random from $\{1, \cdots, s(n)\}$, to guess the left-session with respect to which S will fail in Case-L5.1. Then, S' runs the CMIM adversary $\mathscr{A}(1^n, \hat{A}, A, \hat{B}, B, z)$, and perfectly emulates the players $\hat{A}$ and $\hat{B}$ in all sessions other than the i-th left-session. In the i-th left-session, S' just sends $X_i^l = X$ to $\mathscr{A}$ at the first-round. According to the above clarifications, suppose S fails in Case-L5.1 with non-negligible probability p, with probability $\frac{p}{s(n)}$ (except a negligible probability) S' will output $CDH(X, X)$, which then violates the CDH assumption. □

Proposition 4.2 *The probability that S stops outputting "failure", in Case-R1.2.1 and Case-L4.2.1, is negligible (in the restricted RO model) under the strong Diffie-Hellman assumption.*

Proof (of Proposition 4.2). Note that Case-R1.2.1 assumes that $(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r) = (sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l)$, and $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = NMZK(a, x_i^l)$, but $\hat{B} \neq \tilde{\hat{B}}_i^l$. Further note that Case-R1 (and thus also Case-R1.2.1) implies that the second-round message from $\mathscr{A}$ in the i-th left-session is valid, as the honest player $\hat{A}$ (simulated by S) has sent the third-round message (particularly, $NMZK(a, x_i^l)$) in the i-th left-session. That is: the honest player $\hat{B}$ sends $NMZK(\hat{B}, y_j^r) = H(sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r, CDH(Y_j^r, \tilde{X}_j^r)) = H(sid_j^r, \hat{B}, Y_j^r, X_i^l, CDH(Y_j^r, X_i^l))$ at the second-round of the j-th right-session, but at the second-round of the i-th left-session the CMIM adversary sends $NMZK(\tilde{\hat{B}}_i^l, \tilde{y}_i^l) = H(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l, CDH(\tilde{Y}_i^l, X_i^l)) = H(sid_j^r, \tilde{\hat{B}}_i^l, Y_j^r, X_i^l, CDH(Y_j^r, X_i^l))$ for $\tilde{\hat{B}}_i^l \neq \hat{B}$. Note that with overwhelming probability no honest player instance ever sent the value $H(sid_j^r, \tilde{\hat{B}}_i^l, Y_j^r, X_i^l, CDH(Y_j^r, X_i^l))$ in any existing session. As the simulator does not abort the i-th left-session, the CMIM adversary $\mathscr{A}$ must have queried the RO H with $(sid_j^r, \tilde{\hat{B}}_i^l, Y_j^r, X_i^l, CDH(Y_j^r, X_i^l))$.

The Case-L4.2.1 assumes that $(sid_i^l, \tilde{\hat{B}}_i^l, \tilde{Y}_i^l, X_i^l) = (sid_j^r, \hat{B}, Y_j^r, \tilde{X}_j^r)$, $\widetilde{NMZK}(\tilde{b}_i^l, \tilde{y}_i^l) = NMZK(b, y_j^r)$ but $\hat{A} \neq \tilde{\hat{A}}_j^r$. Further note that Case-L4 (and thus also Case-L4.2.1) implies that the j-th right-session is successfully finished, as the honest player $\hat{B}$ (simulated by S) has sent the fourth-round message (particularly $NMZK(b, y_j^r)$) in the j-th right-session. That is: at the third-round of the i-th left-session, the honest player $\hat{A}$ sends $NMZK(a, x_i^l) = H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l)) = H(sid_i^l, \hat{A}, X_i^l, Y_j^r, CDH(A, Y_j^r), CDH(X_i^l, Y_j^r))$, but at the third-round of the j-th right-session $\mathscr{A}$ sends $\widetilde{NMZK}(\tilde{a}_j^r, \tilde{x}_j^r) = H(sid_j^r, \tilde{\hat{A}}_j^r, \tilde{X}_j^r, Y_j^r, CDH(\tilde{A}_j^r, Y_j^r), CDH(\tilde{X}_j^r, Y_j^r)) = H(sid_i^l, \tilde{\hat{A}}_j^r, X_i^l, Y_j^r, CDH(\tilde{A}_j^r, Y_j^r), CDH(X_i^l, Y_j^r))$ for $\tilde{\hat{A}}_j^r \neq \hat{A}$. Note that with overwhelming probability no honest player instance ever sent the value $H(sid_i^l, \tilde{\hat{A}}_j^r, X_i^l, Y_j^r, CDH(\tilde{A}_j^r, Y_j^r), CDH(X_i^l, Y_j^r))$ in any existing session. As the simulator does not abort the j-th right-session, $\mathscr{A}$ must have queried the RO H with $(sid_i^l, \tilde{\hat{A}}_j^r, X_i^l, Y_j^r, CDH(\tilde{A}_j^r, Y_j^r), CDH(X_i^l, Y_j^r))$.

We show the probability that S fails in Case-L4.2.1 or Case-R1.2.1 is negligible (in the restricted RO model) under the SDH assumption. We present the analysis w.r.t. Case-L4.2.1 (the analysis for Case-R1.2.1 is similar and actually simpler). Specifically, suppose S fails in Case-L4.2.1 with non-negligible probability, we show another efficient algorithm S' that, on inputs $(\hat{A}, A = g^a, a, \hat{B}, B = g^b, b, z, X, Y)$, computes $CDH(X, Y)$ with oracle access to the DDH oracle $\mathcal{O}_Y$ also with non-negligible probability (thus violating the SDH assumption), where X, Y are the challenge elements chosen randomly and independently from G and $\mathcal{O}_Y$ outputs 1 on input $(V, Z) \in G^2$ if and only if $Z = CDH(Y, V)$. Note that, contrary to S, S' takes the secret-keys a and b as its inputs, which are however independent from the challenges (X, Y).

S' first guesses the left and right sessions, with respect to which Case-L4.2.1 failure occurs. This is done by choosing i, j randomly and independently from $\{1, 2, \cdots, s(n)\}$, with success probability

$\frac{1}{(s(n))^2}$. For presentations simplicity, we assume Case-L4.2.1 failure occurs with respect to the i-th left-session and the j-th right-session, and S' successfully guessed these sessions. Then, S' runs $\mathscr{A}(1^n, \hat{A}, A, \hat{B}, B, z)$, and perfectly emulates the honest players $\hat{A}$ and $\hat{B}$ (by knowing the secret-keys a and b, generating DH-components by itself and making only unprogrammable RO-queries) in all sessions other than the i-th left-session and the j-th right-session. In the i-th left-session and the j-th right-session, S' does the following:

- At the first-round of the i-th left-session, S' sends $\{sid_i^l, X\}$ to $\tilde{\hat{B}}_i^l = \hat{B}$ (impersonated by $\mathscr{A}$). That is, S' sets X_i^l just to be the value X given in its input.

- Note that Case-L4.2.1 implies that $\mathscr{A}$ sends, in the name of $\tilde{\hat{A}}_j^r$, the same $\{sid_i^l, X\}$ to $\hat{B}$ (simulated by S') at the first-round of the j-th right session. Then, S' checks whether the RO H has been defined on $(sid_i^l, \hat{B}, Y, X, CDH(Y, X))$, by testing all RO-queries made by $\mathscr{A}$ to H with the assistance of its DDH-oracle $\mathcal{O}_Y$. If yes, S' simply outputs $CDH(X, Y)$, otherwise, S' sets $NMZK(\hat{B}, y)$ to be a random value in $\{0,1\}^{|q|}$, and sends to $\mathscr{A}$ $\{sid_i^l, \hat{B}, CERT_{\hat{B}}, Y, NMZK(\hat{B}, y)\}$ as the second-round message of the j-th right-session. From this point on, whenever $\mathscr{A}$ makes an RO-query to H of the form $(sid_i^l, \hat{B}, Y, X, \alpha)$ for some $\alpha \in G$, S' tests whether $\alpha = CDH(X, Y)$ by querying $\mathcal{O}_Y$ with (X, α); If $\alpha = CDH(X, Y)$, then S' outputs α and stops.

- Case-L4.2.1 further assumes that $\mathscr{A}$ will forward the same message $\{sid_i^l, \hat{B}, CERT_{\hat{B}}, Y, NMZK(\hat{B}, y)\}$ to $\hat{A}$ (simulated by S') as the second-round message in the i-th left-session. Then, S' checks whether the RO H has been defined on $(sid_i^l, \hat{A}, X, Y, CDH(A, Y), CDH(X, Y)$. This is done by checking all queries made by $\mathscr{A}$ to H of the form $(sid_i^l, \hat{A}, X, Y, \alpha_i^l, \beta_i^l)$ for some $(\alpha_i^l, \beta_i^l) \in G^2$, and by testing whether $\alpha_i^l = CDH(A, Y) = Y^a$ (recall that S' knows a) and whether $\beta_i^l = CDH(X, Y)$ by using the DDH-oracle $\mathcal{O}_Y$. If yes, S' stops and outputs $CDH(X, Y)$, otherwise, S' sets $NMZK(a, x)$ to be a random value in $\{0,1\}^{|q|}$, and sends $\{sid_i^l, \hat{A}, CERT_{\hat{A}}, NMZK(a, x)\}$ to $\mathscr{A}$ as the third-round message of the i-th left-session. From this point on, whenever $\mathscr{A}$ makes an RO-query to H of the form $(sid_i^l, \hat{A}, X, Y, \alpha_i^l, \beta_i^l)$ for some $(\alpha_i^l, \beta_i^l) \in G^2$, S' tests whether $\beta_i^l = CDH(X, Y)$ with the assistance of its DDH-oracle $\mathcal{O}_Y$, and stops and outputs β_i^l if it is this case.

- Now, the Case-L4.2.1 failure assumes that $\mathscr{A}$ sends $\{sid_i^l, \tilde{\hat{A}}_j^r, CERT_{\tilde{\hat{A}}_j^r}, \widetilde{NMZK}(\tilde{a}_j^r, x)\}$ to $\hat{B}$ at the third-round of the j-th right-session, where $\tilde{\hat{A}}_i^l \neq \hat{A}$ and $\widetilde{NMZK}(\tilde{a}_j^r, x)$ is supposed to be $H(sid_i^l, \tilde{\hat{A}}_i^l, X, Y, CDH(\tilde{A}_i^l, Y), CDH(X, Y))$. As the Case-L4.2.1 failure assumes $\hat{B}$ does not abort the j-th right-session, this implies that with overwhelming probability $\mathscr{A}$ has to query the RO H with $(sid_i^l, \tilde{\hat{A}}_i^l, X, Y, CDH(\tilde{A}_i^l, Y), CDH(X, Y))$ and thus S' can output $CDH(X, Y)$.

It is easy to check that S' works in the restricted random oracle model, and that the simulation by S' is statistically indistinguishable from the real view of the CMIM adversary $\mathscr{A}$. As the real view of $\mathscr{A}$ is, in turn, statistically indistinguishable from the simulation by S, suppose Case-L4.2.1 failure occurs with non-negligible probability in the simulation by S, with also non-negligible probability it occurs in the simulation by S'. Thus, suppose Case-L4.2.1 failure occurs with non-negligible probability p, S' can solve $CDH(X, Y)$ with oracle access to $\mathcal{O}_Y$ with non-negligible probability $\frac{p}{s(n)^2}$ (except a negligible probability), which violates the SDH assumption. □

By the CKEA assumption and using the same trick (of employing the CKEA assumption) as in the proof of Theorem 3.1, the probability that S fails due to knowledge-extraction failure in

Case-L3.1, Case-R2.1 and Case-L5.2.1 is also negligible. Specifically, all the DH-exponents x_i^l's and y_j^r's are used by S merely for performing DDH-tests. That is, whatever can be done by S can also be done by another machine $\hat{S}$ with DDH-oracle access to the set of the DH-components X_i^l's and Y_j^r (which are also given to $\hat{S}$ as input) but without knowing the corresponding DH-exponents. Then, the CKEA assumption comes into force to ensure correct knowledge-extraction. Putting all together, this establishes the TBRNM security of the DIKE protocol depicted in Figure 2.

Finally, note that the DIKE protocol depicted in Figure 2 is also *privacy-preserving*. Specifically, the transcript of each session (except for players' public-keys and certificates) of the DIKE protocol can be generated merely from the DH-exponents x and y (and the public system parameters); And the messages from one party do not bear the identity and public-key information of its peer. □

4.4 Analysis of DIKE in the Canetti-Krawczyk (CK) Post-Specified Peer Framework

In this section, we further show the DIKE protocol depicted in Figure 2 is secure in the Canetti-Krawczyk framework (CK-framework) for key-exchange protocols with post-specified peer identities [33].

4.4.1 Brief Description of the CK-framework

The following brief description of the CK-framework is almost verbatim from [80] (for complete details the reader is referred to [33, 35]).

A key-exchange (KE) protocol is run in a network of interconnected parties where each party can be activated to run an instance of the protocol called a session. Within a session a party can be activated to initiate the session or to respond to an incoming message. As a result of these activations, and according to the specification of the protocol, the party creates and maintains a session state, generates outgoing messages, and eventually completes the session by outputting a session-key and erasing the session state. A session may also be aborted without generating a session key. A KE session is associated with its holder or owner (the party at which the session exists), a peer (the party with which the session key is intended to be established), and a session identifier. For presentation simplicity, for DHKE protocols, the following two assumptions are made in the basic CK-framework: (i) the activation of a session at a party always specifies the name of the intended peer (i.e., working in the "pre-specified peer model"); and (ii) the session identifier is defined to be $(\hat{A}, \hat{B}, X, Y)$, where $\hat{A}$ is the identity of the holder of the session who sends the DH-component X, $\hat{B}$ the peer who sends the DH-component Y. The peer that sends the first message in a session is called the initiator and the other the responder. It is usually to denote the peers to a session by $\hat{A}$ and $\hat{B}$; either one may act as initiator or responder. The session $(\hat{B}, \hat{A}, Y, X)$ (if it exists) is said to be matching to the session $(\hat{A}, \hat{B}, X, Y)$.

Attacker model. The attacker, denoted $\mathscr{A}$, is an active CMIM adversary with full control of the communication links between parties. $\mathscr{A}$ can intercept and modify messages sent over these links, it can delay or prevent their delivery, inject its own messages, interleave messages from different sessions, etc. (Formally, it is $\mathscr{A}$ to whom parties hand their outgoing messages for delivery.) $\mathscr{A}$ also schedules all session activations and session-message delivery.

In addition, in order to model potential disclosure of secret information, the attacker is allowed to access to secret information via session exposure attacks of three types: state-reveal queries, session-key queries, and party corruption. A state-reveal query is directed at a single session while still incomplete (i.e., before outputting the session key), and its result is that the attacker learns the session state for that particular session (which may include, for example, the DH-exponent corresponding to the DH-component). A session-key query can be performed against an individual

session after completion but before expiration, and the result is that the attacker learns the corresponding session-key. Finally, party corruption means that the attacker learns *all* information in the memory of that party including the long-term static secret-key (corresponding to the public-key) as well as session states and session keys stored at the party; In addition, from the moment a party is corrupted all its actions may be controlled by the attacker (note that the CK-framework does not distinguish between static secret-key reveal and party corruption). A session is called exposed, if this session or its matching session is suffered from *any* of the above three types of session-exposure attacks.

The security of session keys generated in unexposed sessions is captured via the inability of the attacker $\mathscr{A}$ to distinguish the session key of a test session, chosen by $\mathscr{A}$ among all *complete* sessions of the protocol between *uncorrupted* players, from a random value. When $\mathscr{A}$ chooses the test session it is provided with a value v which is chosen as follows: a random bit b is tossed, if $b = 0$ then v is the real value of the session key, otherwise v is a random value chosen under the same distribution of session keys produced by the protocol but independent of the value of the real session key. After receiving v the attacker may continue with the regular actions against the protocol; at the end of its run $\mathscr{A}$ outputs a bit b'. The attacker succeeds in its distinguishing attack if (1) the test-session is not exposed, and (2) the probability that $b = b'$ is significantly larger than $1/2$.

Definition 4.2 (Secure KE (SK-security)) [33] *A polynomial-time attacker with the above capabilities is called a* KE-attacker*. A key-exchange protocol π is called* secure *if for all KE-attackers $\mathscr{A}$ running against π it holds:*

1. *If two uncorrupted parties complete matching sessions in a run of protocol π under attacker $\mathscr{A}$ then, except for a negligible probability, the session key output in these sessions is the same.*

2. *$\mathscr{A}$ succeeds (in its test-session distinguishing attack) with probability not more than that 1/2 plus a negligible function.*

At a high level, the SK-security essentially says that a party that completes a session has the following guarantees: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the (*honest and unexposed*) peer completes a matching session then the two parties have the same shared key.

Adapting SK-security to the post-specified peer setting. The CK-framework [33] assumes that: a party that is activated with a new session knows already at activation the identity of the intended peer to the session. In contrast, in the "post-specified peer" setting (particularly for the IPsec and IKE protocols [77, 66, 76]), the information of who the other party is does not necessarily exist at the session initiation stage. It is actually learnt by the party only after the protocol run evolves.

Specifically, the IPsec and IKE protocols are intended to protect communications in the IP layer, which process the transmission of messages *using the network address (rather than end-uses' IDs)*. This arises the issue that the peer's identity may not be available at the onset of the IKE and may be learned by the parties only after the protocol run evolves. Actually, this is quite a common case for key-exchange protocols in practice. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation; Or, a party may respond to a request (for a key-exchange session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smart-card that authenticates the legitimacy of the card-reader before disclosing its own identity).

Fortunately, adapting the SK-security [33] to the post-specified peer setting only requires some minimal modifications (related to the mechanism of defining matching sessions) [35]: (1) To distinguish concurrent sessions (run at each party), each session bears a unique session-identifier *sid* (recall the SK-security with pre-specified peers does not necessarily assume unique session-identifiers); (2) Let $(\hat{A}, sid)$ be a completed session (at the party $\hat{A}$) with peer $\hat{B}$, the session $(\hat{B}, sid)$ is called the matching session of $(\hat{A}, s)$, if either $(\hat{B}, s)$ is not completed or $(\hat{B}, s)$ is completed with peer $\hat{A}$. For more details, the reader is referred to [35].

SK-security vs. TBRNM. We make some brief comparisons between the SK-security and the tag-binding robust non-malleability (TBRNM) formulated in Section 4.1 .

- At a high level, the SK-security essentially says that a party that completes a session has the following guarantees: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the *unexposed* peer completes a matching session then the two parties have the same shared key.

  Roughly speaking, besides others, TBRNM ensures the enhanced guarantee of the above (2): if the *possibly malicious* peer completes a matching session, then the two parties, not only, have the same shared key, *but also and more importantly, the (possibly malicious) peer does "*know*" both the DH-exponent (and thus the shared session-key) and the secret-key corresponding to the DH-component and public-key sent and alleged by it in the test-session.*

- The TBRNM formulation follows the simulation approach of adaptive tag-based concurrent non-malleable zero-knowledge argument of knowledge, which particularly implies concurrent forward deniability for both the protocol initiator and the responder. The SK-security definition follows the indistinguishability approach, which particularly does not take deniability into account.

- Recall that the TBRNM formulation is w.r.t. any PPT CMIM adversary of *arbitrary* auxiliary input. In particular, the adversary's auxiliary input can be dependent on player's public-key, e.g., consisting of a CDH triple (X, B, g^{xb}) or just the secret-key b.

  What buried in the definition and actual analysis of the SK-security is that: the CMIM-attacker considered in the CK-framework of [33] is "uniform", in the sense that its auxiliary input is assumed to be independent of players' public-keys. The underlying reason is: we will use the ability of the CMIM-attacker in violating the SK-security to compute $CDH(X, Y)$ on a pair of random challenges (X, Y). However, in most cases, one of the inputs (X, Y) is player's public-key in the actual analysis of the SK-security.

- As clarified, the TBRNM formulation allows static secret-key leakage *of uncorrupted players* by allowing arbitrary (maybe public-key dependent) auxiliary input to the CMIM attacker. But, it does not take into account the following abilities of the CMIM adversary in: exposing ephemeral private state for incomplete sessions, exposing session-keys for completed sessions, and party corruption, which are however captured by the SK-security in the CK-framework.

From the above clarifications, it is clear that the TBRNM security and the SK-security are mutually complementary, and thus it is much desirable to have DHKE protocols that enjoy both the SK-security and the TBRNM security simultaneously.

4.4.2 The DIKE Protocol is SK-Secure in the Post-Specified Peer Setting

We consider a version of the DIKE protocol (depicted in Figure 2) with *exposable DH-exponents and pre-computed DH-components*. Specifically, for a state-reveal query against an incomplete session at

a party, the value returned back to the CMIM attacker $\mathscr{A}$ is specified to be the corresponding DH-exponent generated by the party for that session. Furthermore, we assume each uncorrupted player pre-computes (and stores) DH-components, which can be exposed to the CMIM attacker prior to the actual sessions in which these pre-computed DH-components are to be used. We note that this kind of leakage of pre-computed DH-components is not directly captured by the CK-framework.

In this section we prove that the DIKE protocol *with exposable DH-exponents and pre-computed DH-components* is SK-secure in the CK-framework with post-specified peers, under the gap Diffie-Hellman (GDH) assumption in the random oracle model. At the core of the proof is the following lemma, which essentially says that the CMIM attacker can successfully finish an *unexposed* session in the name of some uncorrupted player only if that uncorrupted player (impersonated by the CMIM attacker) does indeed send the authenticated value, say, $NMZK(a, x)$ or $NMZK(b, y)$, in the corresponding matching session. For presentation simplicity, this lemma is described with respect to the CMIM adversary $\mathscr{A}$ considered in the setting of the TBRNM formulation but with the powerful capabilities of secrecy exposure as specified above, and can be trivially extended to the setting of the CK-framework.

Lemma 4.1 *For the DIKE protocol $\langle \hat{A}, \hat{B} \rangle$ (depicted in Figure 2)* with exposable DH-exponents and pre-computed DH-components*, where the players $\hat{A}$ and $\hat{B}$ may be identical, the probability of the following events is negligible under the GDH assumption in the random oracle model:*

Event-1. *The CMIM adversary $\mathscr{A}$ successfully finishes the j-th right-session for some j, $1 \leq j \leq s(n)$, where $\mathscr{A}$ sends $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in the third-round in the name of $\hat{A}$ (actually, any uncorrupted player) with respect to the DH-component Y_j^r sent by the uncorrupted player $\hat{B}$ in the second-round, while the uncorrupted player $\hat{A}$ did not send $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in any left-session and $\mathscr{A}$ does not know the discrete-logarithm of Y_j^r (i.e., $\mathscr{A}$ did not make the state-reveal query against the j-th right-session at the uncorrupted player $\hat{B}$).*

Event-2. *The CMIM adversary $\mathscr{A}$ successfully finishes the i-th left-session for some i, $1 \leq i \leq s(n)$, where $\mathscr{A}$ sends $\widetilde{NMZK}(b, \tilde{y}_i^l)$ in the fourth-round in the name of $\hat{B}$ (actually, any uncorrupted player) with respect to the DH-component X_i^l sent by the uncorrupted player $\hat{A}$ in the first-round, while the uncorrupted player $\hat{B}$ did not send $\widetilde{NMZK}(b, \tilde{y}_i^l)$ in any right-session and $\mathscr{A}$ does not know the discrete-logarithm of X_i^l (i.e., $\mathscr{A}$ did not make the state-reveal query against the i-th left-session at the uncorrupted player $\hat{A}$).*

Proof (outline). First note that, for Event-1, the correct value that $\widetilde{NMZK}(a, \tilde{x}_j^r)$ should be, i.e., $H(sid_j^r, \hat{A}, \tilde{X}_j^r, Y_j^r, CDH(A, Y_j^r), CDH(\tilde{X}_j^r, Y_j^r))$, cannot be sent by the uncorrupted player $\hat{B}$ (even if $\hat{B} = \hat{A}$) in any right-session, due to the uniqueness of the session-identifiers of the right-sessions. Thus, suppose Event-1 occurs, with overwhelming probability $\mathscr{A}$ must have queried the RO H with $(sid_j^r, \hat{A}, \tilde{X}_j^r, Y_j^r, CDH(A, Y_j^r), CDH(\tilde{X}_j^r, Y_j^r))$, which particularly implies that $\mathcal{A}$ can output $CDH(A, Y_j^r)$.

For Event-2, the correct value that $\widetilde{NMZK}(b, \tilde{y}_i^l)$ should be, i.e., $H(sid_i^l, \hat{B}, \tilde{Y}_i^l, X_i^l, CDH(B, X_i^l), CDH(\tilde{Y}_i^l, X_i^l))$, cannot be sent by the uncorrupted player $\hat{A}$ in any left-session other than the i-th left-session, due to the uniqueness of the session-identifiers of the left-sessions. Also, as clarified with the proof of Proposition 4.1, the correct value of $\widetilde{NMZK}(b, \tilde{y}_i^l)$ also cannot be sent by $\hat{A}$ (even if $\hat{A} = \hat{B}$) in the i-th left-session under the CDH assumption. Thus, suppose Event-2 occurs, with overwhelming probability under the CDH assumption $\mathscr{A}$ must have queried the RO H with $(sid_i^l, \hat{B}, \tilde{Y}_i^l, X_i^l, CDH(B, X_i^l), CDH(\tilde{Y}_i^l, X_i^l))$, which particularly implies that $\mathcal{A}$ can output $CDH(B, X_i^l)$.

For presentation simplicity, in the following we only present the proof with respect to Event-1. The proof for the case of Event-2 is similar.

Specifically, suppose the CMIM adversary $\mathscr{A}$ can successfully finish, with non-negligible probability, the j-th right-session for some j, $1 \leq j \leq s(n)$, in which $\mathscr{A}$ sends $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in the third-round in the name of $\hat{A}$ w.r.t. the DH-component Y_j^r sent by the uncorrupted player $\hat{B}$ in the second-round, but the uncorrupted player $\hat{A}$ did not send $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in any left-session and $\mathscr{A}$ does not know the discrete-logarithm of Y_j^r. Then, we construct another efficient algorithm S' that, on inputs $(A, Y) \in G^2$ (where each of (A, Y) is taken randomly and independently from G) and with access to the DDH-oracles $\mathcal{O}_A$ and $\mathcal{O}_Y$, outputs $CDH(A, Y)$ also with non-negligible probability, which violates the GDH assumption. *Moreover, as we shall see, the following actual proof is w.r.t. CMIM adversaries of more powerful capability of secrecy exposure than specified in the lemma statements.*

The algorithm S' runs the CMIM adversary $\mathscr{A}$ as a subroutine, and works as follows:

Key setup. S' generates and sets the public-key and secret-key pairs for all uncorrupted players other than the uncorrupted player $\hat{A}$. For the uncorrupted player $\hat{A}$, S' just sets the public-key of $\hat{A}$ to be A (i.e., the value given to S' as its input). Note that S' does not know the secret-key of $\hat{A}$, i.e., the discrete logarithm of A.

Pre-computation of DH-components and DH-exponents. S' chooses j uniformly at random from $\{1, \cdots, s(n)\}$. S' then pre-computes the DH-components $\{X_1^l = g^{x_1^l}, \cdots, X_{s(n)}^l = g^{x_{s(n)}^l}\}$ to be used in the left-sessions, and $\{Y_1^r = g^{y_1^r}, \cdots, Y_{j-1}^r = g^{y_{j-1}^r}, Y_{j+1}^r = g^{y_{j+1}^r}, \cdots, Y_{s(n)}^r = g^{y_{s(n)}^r}\}$ to be used in all right-sessions *other than the j-th right-session*. The DH-component to be used in the j-th right-session, i.e., Y_j^r, is just set to be Y (i.e., the value given to S' as its input).

Powerful secrecy exposure capability rendered to $\mathscr{A}$. S' give the CMIM adversary $\mathcal{A}$ the secret-keys of all uncorrupted players *other than the uncorrupted player* $\hat{A}$, all the pre-computed DH-components, and all the pre-computed DH-exponents to be used in all right-sessions other than the j-th right-session. We remark that such powerful exposure capabilities are beyond what specified in the CK-framework.

Left-session simulation. For each i, $1 \leq i \leq s(n)$, in the i-th left-session, S' works just as the honest player $\hat{A}$ does at the first-round, the second-round and the fourth-round (by using the DH-exponent x_i^l generated by itself). But at the third-round of the i-th left-session (that is dependent on the secret-key a of $\hat{A}$ which S' however does not know), S' does the following:

Left-session third-round simulation. Denote by $X_i^l = g^{x_i^l}$ the DH-component sent by S' in the first-round of the i-th left-session, and by $\tilde{Y}_i^l$ the DH-component sent by the CMIM adversary $\mathscr{A}$ in the second-round in the name of $\tilde{\hat{B}}_i^l$. S' ensures the consistence of RO-answers related to $NMZK(a, x_i^l) = H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l) = (\tilde{Y}_i^l)^{x_i^l})$, with the assistance of the DDH-oracle $\mathcal{O}_A$, as follows:
S' first checks whether the RO H has been defined on the point $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$. This is done by checking all RO-queries to H of the form $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, \alpha_i^l, (\tilde{Y}_i^l)^{x_i^l})$ for some $\alpha_i^l \in G$, and testing whether $\alpha_i^l = CDH(A, \tilde{Y}_i^l)$ with the assistance of the DDH-oracle $\mathcal{O}_A$. If H has been defined on the point $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$ then S' sets $NMZK(a, x_i^l) = H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l) = (\tilde{Y}_i^l)^{x_i^l})$, otherwise, S' just sets $NMZK(a, x_i^l)$ to be a value taken uniformly at random from

$\{0,1\}^{|q|}$. S' sends $\{sid_i^l, \hat{A}, CERT_{\hat{A}}, NMZK(a, x_i^l)\}$ to $\mathscr{A}$ as the third-round message of the i-th left-session. In case $NMZK(a, x_i^l)$ is set be a random value in $\{0,1\}^{|q|}$, from this point on, whenever the CMIM adversary makes an RO-query to H of the form $(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, \alpha_i^l, (\tilde{Y}_i^l)^{x_i^l})$ for some $\alpha_i^l \in G$, S' checks whether $\alpha_i^l = CDH(A, \tilde{Y}_i^l)$ with the assistance of the DDH-oracle $\mathcal{O}_A$; If it is this case, S' just defines the RO answer, i.e., $H(sid_i^l, \hat{A}, X_i^l, \tilde{Y}_i^l, CDH(A, \tilde{Y}_i^l), CDH(X_i^l, \tilde{Y}_i^l))$, to be (the already fixed) $NMZK(a, x_i^l)$.

Right-session simulation. We distinguish two cases according to whether $\hat{B} = \hat{A}$ or not, for the right-session simulation.

The case of $\hat{B} = \hat{A}$. Note that in this case, the simulator S' does not know the secret-key a of $\hat{B} = \hat{A}$.

Simulation for right-sessions other than the j-th right-session. For each k, $1 \leq k \neq j \leq s(n)$, S' works just as the honest player $\hat{B}$ does in the first three rounds of the k-th right-session. In particular, S' sends the DH-component $Y_k^r = g^{y_k^r}$ at the second-round in the name of $\hat{B} = \hat{A}$ (after receiving $\tilde{X}_k^r$ from $\mathscr{A}$ at the first-round). But at the fourth-round of the k-th right-session, in which S' needs to send the authentication value $NMZK(b, y_k^r) = NMZK(a, y_k^r) = H(sid_j^r, \hat{A}, Y_k^r, \tilde{X}_k^r, CDH(A, \tilde{X}_k^r), (\tilde{X}_k^r)^{y_k^r})$ that S' however does not know the secret-key a (but knows the DH-exponent y_k^r), S' ensures the consistency of the RO answers of H related to the authentication value $NMZK(b, y_k^r) = NMZK(a, y_k^r)$ with the assistance of the DDH-oracle $\mathcal{O}_A$, similar to the left-session third-round simulation described above (details are omitted here).

Simulation for the j-th right-session. S' receives $(sid_j^r, \tilde{X}_j^r)$ from $\mathscr{A}$ at the first-round of the j-th right-session, and works as the honest player does.

Second-round simulation. At the second-round of the j-th right-session, S' just sets the DH-component Y_j^r to be the value Y given as its input. Then, S' sets the value $NMZK(\hat{B}, y)$ and ensures the consistency of the RO-answers of H related to $NMZK(\hat{B}, y) = H(sid_j^r, \hat{B}, Y, \tilde{X}_j^r, CDH(Y, \tilde{X}_j^r))$, *with the assistance of the DDH-oracle* $\mathcal{O}_Y$ (similar to the right-session second-round simulation in the proof of Theorem 4.1).

Third-round output. If $\mathscr{A}$ sends $(sid_j^r, \hat{A}, CERT_{\hat{A}}, \widetilde{NMZK}(a, \tilde{x}_j^r))$ at the third-round of the j-th right-session, and S' never sends $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in any left-session and $\mathscr{A}$ also did not make the state-reveal query against the j-th right-session, S' checks whether $\mathscr{A}$ has queried the RO H with $(sid_j^r, \hat{A}, \tilde{X}_j^r, Y, CDH(A, Y), CDH(\tilde{X}_j^r, Y))$, with the assistance of the DDH-oracle $\mathcal{O}_Y$. Note that S' also never sends the value $H(sid_j^r, \hat{A}, \tilde{X}_j^r, Y, CDH(A, Y), CDH(\tilde{X}_j^r, Y))$, which the valid authentication value $\widetilde{NMZK}(a, \tilde{x}_j^r)$ should be, in any right-session (particularly due to the uniqueness of session-identifiers, i.e., sid_j^r's). If $\mathscr{A}$ made such a query to H, S' stops and outputs $CDH(A, Y)$; In all other cases, S' stops and outputs "failure".

The case of $\hat{B} \neq \hat{A}$. The analysis for the case of $\hat{B} \neq \hat{A}$ is similar to, actually simpler than, that for the case of $\hat{B} = \hat{A}$, by noting that S' knows the secret-key of $\hat{B}$ in this case.

It's easy to see that S' works in polynomial-time, and the simulation by S' is statistically indistinguishable from the real view of $\mathscr{A}$. Thus, suppose Event-1 occurs with non-negligible probability p in the real view of $\mathscr{A}$, with the same probability p (except a negligible probability) $\mathscr{A}$ does the same in the simulation by S'. As S' guesses the j-th right-session uniformly at random

from all the $s(n)$ right-sessions, we conclude that S' succeeds in breaking the GDH assumption with probability $\frac{p}{s(n)}$ (except a negligible probability). □

Theorem 4.2 *The DIKE protocol (depicted in Figure 2),* with exposable DH-exponents and pre-computed DH-components*, is SK-secure in the CK-framework with post-specified peers, under the GDH assumption in the random oracle model.*

Proof (outline). We present the proof with respect to an *unexposed* and successfully finished (left) test-session run by $\hat{A}$ (with peer $\hat{B}$), denoted $(\hat{A}, sid, \hat{B})$, where $\hat{A}$ and $\hat{B}$ are both uncorrupted players and *can be identical* (but $\hat{B}$ may be impersonated by the CMIM attacker $\mathscr{A}$). The proof for the case of an unexposed (right) test-session $(\hat{B}, sid, \hat{A})$ run by $\hat{B}$ (with peer $\hat{A}$) is similar.

For the unexposed test-session $(\hat{A}, sid, \hat{B})$, let $X = g^x$ (resp, $Y = g^y$) be the DH-component sent by $\hat{A}$ (resp., $\hat{B}$), and $NMZK(b, y) = H(sid, \hat{B}, Y, X, X^y, X^b)$ be the authentication value sent by $\hat{B}$ (maybe impersonated by $\mathscr{A}$) in the fourth-round of the test-session. By Lemma 4.1, we have that with overwhelming probability the uncorrupted player $\hat{B}$ does indeed send $NMZK(b, y)$ in one of right-sessions. This implies that in the RO model with overwhelming probability, the (left) test-session has matching (right) session $(\hat{B}, sid)$ in which $\hat{B}$ sends Y in the second-round (after receiving X in the first-round but not necessarily in the peer name of $\hat{A}$) and $NMZK(b, y)$ in the fourth-round.

As the session-key is computed as $H_K(X, Y, g^{xy})$ and H_K is a random oracle, there are only two strategies for the adversary $\mathscr{A}$ to distinguish $H_K(X, Y, g^{xy})$ from a random value:

Key-replication attack: $\mathscr{A}$ succeeds in forcing the establishment of a session (other than the test-session or its matching session) that has the same session-key output as the test-session. In this case, $\mathscr{A}$ can learn the test-session key by simply querying the session to get the same key (without having to expose the test-session or its matching session).

Forging attack: At some point in its run, $\mathscr{A}$ queries the RO H_K with the values (X, Y, g^{xy}).

The possibility of the key-replication attack is trivially ruled out in the RO model, by observing that X is only sent by $\hat{A}$ in the test-session and Y is only sent by $\hat{B}$ in the matching session. Recall that the session-key of the text-session (and its matching session) is $H_K(X, Y, CDH(X, Y))$.

The success of the forging attack says that $\mathscr{A}$ can successfully output $(X, Y, CDH(X, Y))$. Recall that, with overwhelming probability, X and Y are only sent *by uncorrupted players* in the test-session and its matching session. As both the test-session and its matching session are assumed to be unexposed (and thus $\mathscr{A}$ does not know the DH-exponent x or y), similar to the proof of Proposition 4.2, we can exploit the assumed ability of $\mathscr{A}$ in performing the successful forging attack to construct another efficient algorithm S' to break the CDH assumption (with the assistance of the DDH-oracle $\mathcal{O}_X$ or $\mathcal{O}_Y$). Specifically, on inputs (X, Y) and with access to the DDH-oracle $\mathcal{O}_X$ or $\mathcal{O}_Y$, S' runs $\mathscr{A}$ as a subroutine and does the following: (1) S' sets up the public-keys and secret-keys for all uncorrupted players; (2) Then, S' guesses (with non-negligible success probability) the test-session and its matching session, for which the forging attack succeeds; (3) S' perfectly emulates the uncorrupted players in all sessions other than the test-session and its matching session; (4) S' sends X in the first-round of the test-session (in the name of $\hat{A}$), and Y in the second-round of the matching session (in the name of $\hat{B}$). For all the hashed authentication values (exchanged in the test-session and its matching session) with $CDH(X, Y)$ as an input, similar to the proof of Proposition 4.2, S' ensures the consistence of RO answers with the assistance of the DDH-oracle $\mathcal{O}_X$ or $\mathcal{O}_Y$. □

4.5 Discussions on the Resistance of DIKE Against Some Concrete Attacks

In this section, we make some (informal) discussions on a list of concrete but essential security properties of the DIKE protocol (depicted in Figure 2), *some of which are beyond the CK-framework*.

Resistance against "cutting-last-message" attack. Suppose the player $\hat{B}$ sends the last message in the run of a DHKE protocol $\langle \hat{A}, \hat{B} \rangle$, the "cutting-last-message" attack, suffered by SIGMA and IKEv2, works as follows [96] (more details are recalled in Appendix A): A man-in-the-middle $\mathscr{A}$ interacts with the uncorrupted $\hat{B}$ in the name of $\hat{A}$ in a session (referred to as the test-session), while concurrently interacting with the uncorrupted $\hat{A}$ in the name of $\hat{M} \neq \hat{B}$ in another session (referred to as the matching session). $\hat{M}$ just relays messages between $\hat{A}$ and $\hat{B}$ in these two sessions, but aborts the matching session after receiving the last message from $\hat{B}$ in the test-session. Such a simple attack results in authentication failure as follow: $\hat{B}$ is perfectly fooled to believe that it has shared a session key with $\hat{A}$ in the test-session, while $\hat{A}$ thinks it only ever took part in an aborted session with $\hat{M}$ in the matching session.[8]

Such an attack is simply ruled out for our DIKE protocol by the tag-based robust non-malleability of our DIKE protocol. Specifically, for the above cutting-last-message attack, the tag of the completed test-session (i.e., the one in which $\hat{B}$ believes it has shared a session-key with $\hat{A}$) must be *distinct*; In particular, it is different from the tag of the aborted matching session in which $\mathscr{A}$ interacts with $\hat{A}$ in the name of $\hat{M} \neq \hat{B}$. By the tag-based robust non-malleability, it implies that $\mathscr{A}$ has to know $\hat{A}$'s secret-key a and the DH-exponent generated by $\hat{A}$ in the aborted matching session, in order to successfully complete the test-session with $\hat{B}$ in the name of $\hat{A}$. In particular, after receiving the second-round message $(\hat{B}, Y = g^y, NMZ(\hat{B}, y))$ from $\hat{B}$ in the test-session, the CMIM adversary $\mathscr{A}$ cannot compute and send to $\hat{A}$ the message of $(\hat{M}, Y = g^y, NMZK(\hat{M}, y))$ in the name of $\hat{M} \neq \hat{B}$ in the matching session.

Resistance against unknown key share (UKS) attack. Informally speaking, by a successful UKS attack an adversary can successfully make two uncorrupted parties compute the same session-key in two sessions but have different views of who the peer to the exchange was, even if the adversary actually does not know the corresponding session-key.

For a successful UKS attack against our DIKE protocol, between two sessions of different pairs of players, we have the following observations: As the session-key is derived from $H(X, Y, g^{xy})$ and the two sessions are of the same session-key, with overwhelming probability in the RO model these two sessions must be of the same DH-components, say (X, Y), and furthermore, in the same (initiator and responder) order. Note that there are at most two sessions (involving uncorrupted players) of the same DH-components exchanged in the same order, as uncorrupted players generate DH-components randomly and independently. In other words, besides the two sessions suffering from the UKS attack, there exist no other sessions of the same (ordered) DH-components (X, Y). This implies that each one of the two sessions (suffering from the UKS attack) is of a *distinct* tag, i.e., different from the tags of all other sessions. By the tag-based robust non-malleability, the adversary must know both of the corresponding DH-exponents x and y (and also the secret-keys corresponding to the public-keys alleged by the adversary in the two sessions). This particularly implies that the adversary does know the session-key $H(X, Y, g^{xy})$, which violates the assumed success of the UKS attack.

Resistance against key-compromise impersonation (KCI) attack. Informally, security against KCI attacks says that: even if the attacker knows the static secret-key of an uncorrupted party, e.g., the secret-key a of $\hat{A}$, it still cannot compromise the session-key (i.e., distinguishing it from a random value) of the session of DH-components (X, Y) exchanged between $\hat{A}$ and another uncorrupted $\hat{B}$, as long as the attacker is not actively controlling or observing the secret DH-

[8] As suggested in [35], this "cutting-last-message-attack" can be prevented by adding an additional fifth-round of "acknowledgement" from $\hat{A}$ to $\hat{B}$, but increasing the system complexity.

exponent x or y [80]. In particular, the knowledge of the secret-key of an uncorrupted player (e.g., a of $\hat{A}$) does not help the adversary to impersonate another uncorrupted player (e.g., $\hat{B}$) to this uncorrupted *yet secret-key compromised* player (i.e., $\hat{A}$).

Note that the resistance against KCI attacks is not directly implied by the SK-security in the CK-framework [33]. Fortunately, the resistance against KCI attacks does hold for our DIKE protocol, by the following observations:

- For the case of the test-session between a pair of different (uncorrupted) players $(\hat{A}, \hat{B})$ where $\hat{A} \neq \hat{B}$, the proofs Lemma 4.1 and Theorem 4.2 have already explicitly allowed he CMIM attacker to expose the secret-key of the uncorrupted player that is running the test-session.
- For the case of the test-session between the same single player, i.e., $\hat{A} = \hat{B}$, with exchanged DH-components (X, Y) (w.o.l.g., suppose the player running the test-session sends the DH-component X), recall that the successful KCI attack against the test-session $(\hat{A}, sid, \hat{B}) = (\hat{A}, sid, \hat{A})$ has already assumed another (whether matching or not) session in which an uncorrupted player sent the DH-component Y and against which the CMIM-attacher does not make the state-reveal query (as the KCI attack assumes the attacker does not know the DH-exponent x or y). Then, similar to the proof of Theorem 4.2, the assumed successful KCI attack by the CMIM-attacker can be used to break the GDH assumption.

In addition, note that secret-key leakage is also captured in the formulation and analysis of the TBRNM security of our DIKE protocol.

Implication of perfect forward secrecy (PFS). Informally, a key-exchange protocol is of the PFS property, if the leakage of the static secret-key of an uncorrupted player does not compromise the security of the session-keys established by the player for unexposed yet expired sessions, which have been erased from memory before the leakage occurred [80]. In other words, once an unexposed session is expired and the session-key is erased from its holder's memory, then the session-key cannot be learned by the attacker even if the player is subsequently corrupted. The PFS property of our DIKE protocol is from the observation that: the computation of the session-key $H_K(X, Y, g^{xy})$ does not involve players' secret-keys.

Resistance against reflection attack. In some scenarios, a party may want to establish a secure channel with itself (i.e., $\hat{A} = \hat{B}$ in this case). For example, a mobile user that communicates to its desktop computer, while both the mobile device and the desktop have the same identity in the form of the same public-key certificate [80]. In a *reflection* attack, an attacker simply copies the authentic messages from $\hat{A}$ and sends them back to $\hat{A}$ as the messages coming from the other copy of $\hat{A}$. Note that we have shown in Proposition 4.1 that under the CDH assumption our DIKE protocol is resistant against reflection attacks.

A Note on the role of $NMZK(\hat{B}, y)$ in frustrating reflection attacks. With respect to the protocol structure of our deniable IKE, to mount a successful reflection attack against the DIKE protocol an adversary has to set $Y = X$ and $\hat{B} = \hat{A}$ so that $(X, Y) = (Y, X)$ and $NMZK(b, y) = NMZK(a, x)$. Note that this is an effective reflection attack against the variant of SIGMA (depicted in Appendix A) *when "0" and "1" are removed from inputs of SIG and MAC*, where the bits 1 and 0 indicate the roles of the protocol initiator and the responder. But, this play is frustrated with our deniable IKE, by briefly noting that the adversary cannot provide the proof-of-knowledge of $y = x$, i.e., $NMZK(\hat{B}, y) = NMZK(\hat{A}, x)$, in the second-round.

4.6 Deniable Internet Key-Exchange: the Aggressive Model

In accordance with the aggressive model of IKE [66, 76], we present the 3-round variant of our deniable IKE protocol that is depicted in Figure 3 (page 43).

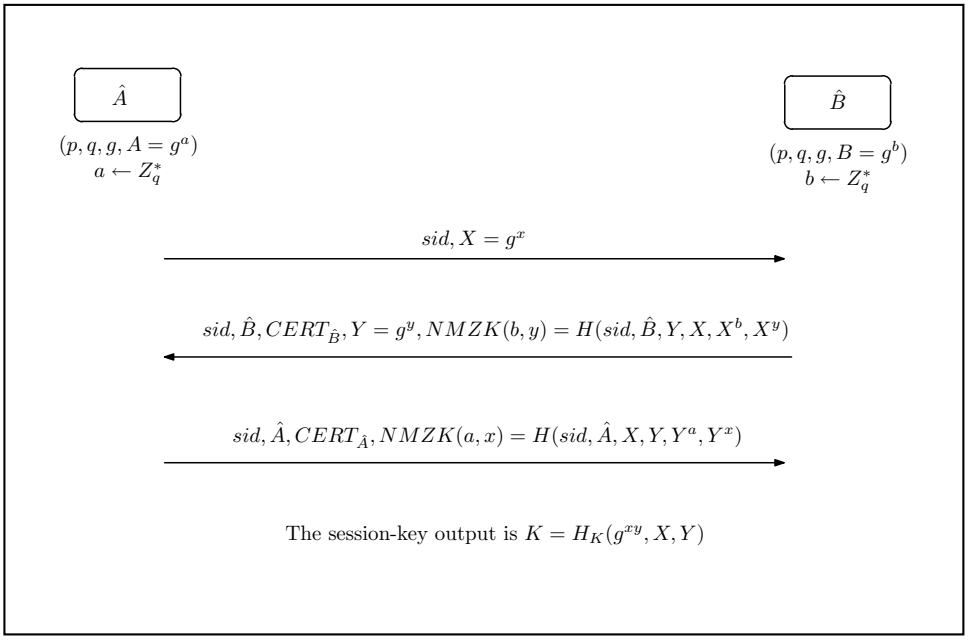


Figure 3: Deniable Internet key-exchange: the aggressive model

Most security properties of the deniable IKE of the main model are essentially inherited by this 3-round protocol variant in the aggressive model, except for the full deniability for the responder player $\hat{B}$. Specifically, the player $\hat{B}$ only enjoys *completed-session deniability*, in the sense that if a malicious player $\hat{A}$, denoted as $\hat{A}^*$, completes the session then $\hat{B}$'s deniability will be guaranteed. But if the (possibly malicious) player $\hat{A}^*$ just aborts the session after receiving the second-round message, the deniability for $\hat{B}$ is not ensured. Note that the initiator player $\hat{A}$ still has full deniability. We remark that such kind of completed-session deniability for the responder is still very useful and reasonable. For instance, consider the scenario where $\hat{A}$ is a client and $\hat{B}$ is a (bank or shop) server: in such a scenario it is the client $\hat{A}$ who cares more about its privacy and full deniability does guarantee for it, while the server cares less about deniability and the completed-session deniability may still be deemed to be good enough for it.

4.7 Comparisons with SIGMA

We briefly make some comparisons between our DIKE protocol and the SIGMA protocol of [79] on which the current IKEv2 standard is based.

Communication complexity and (online) computational complexity. As clarified in Section 4.1, the total computational complexity of our DIKE protocol at each player side is about 3.3 exponentiations, and the on-line computational complexity at each player side can be only 1.3 exponentiations! In comparison, with the underlying signature implemented by the Digital Signature Standard (DSS), the computational complexity of SIGMA is about 4.5 exponentiations at each player side in total and the online complexity is about 3.5 exponentiations at each player side without offline partial signature generation (or about 2.5 exponentiations with offline partial signature generation).

For communication complexity, by waiving the use and exchanges of signatures, our deniable IKE is also of improved communication complexity, in comparison with that of SIGMA.

Privacy protection. Our DIKE enjoys much better privacy protection than SIGMA. In particular, our DIKE protocol is of *privacy-preserving* TBRNM property, which particularly implies concurrent forward deniability (for both players) against any efficient CMIM adversary of arbi-

trary auxiliary input. Note also that all the authentic messages exchanged in a session can be computed merely from the DH-exponents. SIGMA does not enjoy these privacy properties, due to the underlying signatures used. (A *variant* of the SIGMA protocol was recently proved to satisfy some *plausible partial* deniability assuming key-aware derivation, typically with the random oracle assumption of hash functions [45].)

Besides, as clarified with the resistance against reflection attacks in Section 4.5, our deniable IKE also provides seemingly better privacy protection of players' roles than (the basic version of) SIGMA, by waiving the explicit role indicators (i.e., the bits 0 and 1) used in SIGMA. Specifically, players' roles cannot be *publicly* checked from the authentic messages, particularly, $NMZK_{(a,x)}$ or $NMZK_{(b,y)}$, of our DIKE protocol; the whole transcript may reveal the roles of initiator and responder, but the whole transcript is deniable.

Robust security. Our DIKE protocol is of both TBRNM secure and SK-secure, which particularly implies most security vulnerabilities (particularly the "cutting-last-message" attack) not captured by the SK-security. As clarified, the TBRNM security and the SK-security are mutually complementary, and it is thus much desirable to have DHKE protocols that enjoy both of them simultaneously. The SIGMA protocol is particularly not TBRNM secure.

But, we also reminder that the TBRNM security and the SK-security of our DIKE are based on the random oracle assumption and the CKEA and GDH assumptions; The security of SIGMA relies on the security of the underlying building tools it employs: signatures, MACs, the key-derivation function and also the CDH assumption, without directly relying on the RO assumption.

We remark that, as we are aiming at achieving *highly practical* DHKE protocols of both the TBRNM security (that particularly implies concurrent non-malleable statistical ZKAOK) and the SK-security, the RO assumption and also the CKEA assumption (that essentially implies the GDH assumption) may be unavoidable. Also note that when the SIGMA protocol and the IKEv2 standard are deployed in practice, in most cases, (some of) the underlying building tools (say, signatures, MACs, key-derivation functions) are actually implemented with provable security in the RO model or based on some non-standard assumptions (or, even without provable security like the DSS). *Particularly, the RO and/or KEA assumptions are common in the stringent situation of deniable key-exchange and authentication [10, 9, 44, 80, 42, 45, 117].* As mentioned above, partial deniability for a variant of SIGMA [45], with 0 and 1 removed from inputs of signatures, is proved, assuming key-aware derivation (typically, with the RO assumption and/or some other types of non-standard assumptions).

Compatibility with SIGMA and (H)MQV. Our DIKE protocol is of well compatibility with the widely deployed and standardized SIGMA [79] (i.e., the basis of IKEv2) and (H)MQV [83, 80], which eases the deployment and use of our deniable IKE in practice.

By compatibility with SIGMA and IKEv2, we mean that in case some players are not of DL public-keys, they still can use the Sign-then-MAC mechanism of SIMGA and IKEv2 to authenticate messages from them. In more details, in this case, any one of the last two messages in our deniable IKE (both for the main model and for the aggressive model) can be replaced by the corresponding message flow in IKEv2 and SIGMA. By compatibility with (H)MQV, we mean that both (H)MQV and our deniable IKE work for players of DL public-keys, and can be of the same system parameters.

4.8 Identity-Based Deniable IKE

Let $\hat{e} : G \times G \to G_T$ be an admissible pairing [15, 3], where G is a cyclic multiplicative (or additive) group of order q generated by an element g. *To be compatible with the presentation and notations in Section 3, 4 and 6, the following descriptions of ID-YZ protocols is w.r.t. a is a cyclic multiplicative group G.*

In this section, we present the identity-based version, referred to as ID-DIKE, of the deniable IKE protocol described in Section 4.

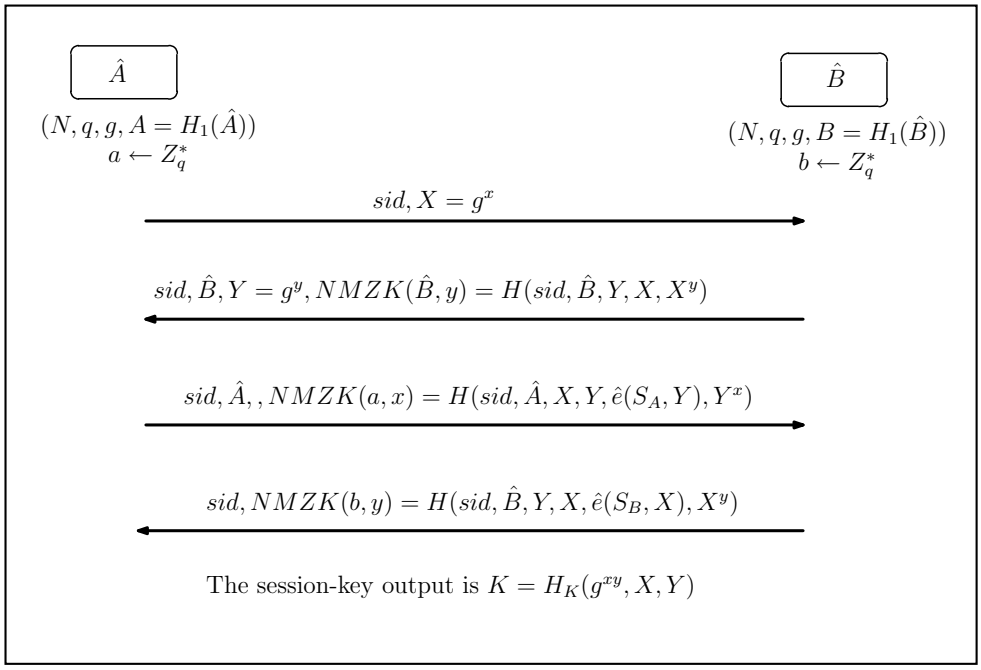


Figure 4: ID-based deniable IKE: the main model

Setup: The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. Besides the functions H and H_K (that are the same in the description of the DIKE protocol in Section 4), PKG also specifies a map-to-point hash function $H_1 : \{0,1\}^* \rightarrow G$. The public parameters are: $(G, G_T, \hat{e}, g, S, H_1, h, H_K)$.

User secret-key extract: For a user with identity $\hat{A}$ the public-key is given by $A = H_1(\hat{A})$, and the PKG generates the associated secret-key of the user as $S_A = A^s$. Similarly, a user of identity $\hat{B}$ is of public-key $B = H_1(\hat{B})$ and secret-key $S_B = B^s$.

ID-based DIKE between two users $\hat{A}$ and $\hat{B}$. The 4-round ID-based DIKE protocol for the main model is depicted in Figure 4 (page 45), and the 3-round ID-based DIKE protocol for the aggressive model is depicted in Figure 5 (page 46).

Notes on ID-DIKE: Most advantageous features of the DIKE protocols described in Section 4 are inherited by the above ID-DIKE protocols. Below, we explicitly highlight some features of the ID-DIKE protocols.

- **Deniability.** Note that the value $NMZK_{(a,x)}$ (resp., $NMZK_{(b,y)}$) can be computed merely from peer's DH-exponent y (resp., x). In particularly, $\hat{e}(S_A, Y) = \hat{e}(A, S)^y$ (resp., $\hat{e}(S_B, X) = \hat{e}(B, S)^x$).

- **Subgroup test.** $\hat{A}$ (resp., $\hat{B}$) can make explicit subgroup test, by computing Y^x and Y^q (resp., X^y and X^q) in parallel that amounts to about 1.2 exponentiations. For embedded subgroup test (particularly when the cofactor t is small), we can set $\hat{e}(S_A, Y)$ (resp. $\hat{e}(S_B, X)$) to be $\hat{e}(S_A^t, Y)$ (resp., $\hat{e}(S_B^t, X)$, where t is the cofactor. For this protocol variant, $\hat{A}$ (resp., $\hat{B}$) will simply abort if $\hat{e}(S_A^t, Y) = 1_{G_T}$ (resp., $\hat{e}(S_B^t, X) = 1_{G_T}$. *Note that $\hat{A}$ (resp., $\hat{B}$) can offline pre-compute S_A^t (resp., S_B^t).* Another approach for embedded subgroup test is to set $X^y = Y^x$ to be $X^{yt} = Y^{xt}$. We prefer to the former paring-based approach as it allows more efficient online efficiency.

- **Online efficiency.** In case of pre-specified peer identity, $\hat{A}$ (resp., $\hat{B}$) can offline pre-compute $\hat{e}(B, S)$ and $\hat{e}(B, S)^x$ (resp., $\hat{e}(A, S)$ and $\hat{e}(A, S)^y$). That is, the online computational com-

plexity of each user is essentially 1 pairing and 1 modular exponentiation. Note that users do not need to *online* perform the map-to-point hash operations of H_1 (that can be relatively expensive [16].

- **Modular composability and compatability with DIKE and SIGMA.** In case one of the players, $\hat{A}$ or $\hat{B}$, works in the normal public-key model and does not support the ID-based implementation, the third-round or fourth-round message in ID-DIKE can be replaced by the corresponding round message in DIKE (without using signatures) or SIGMA (using signatures).

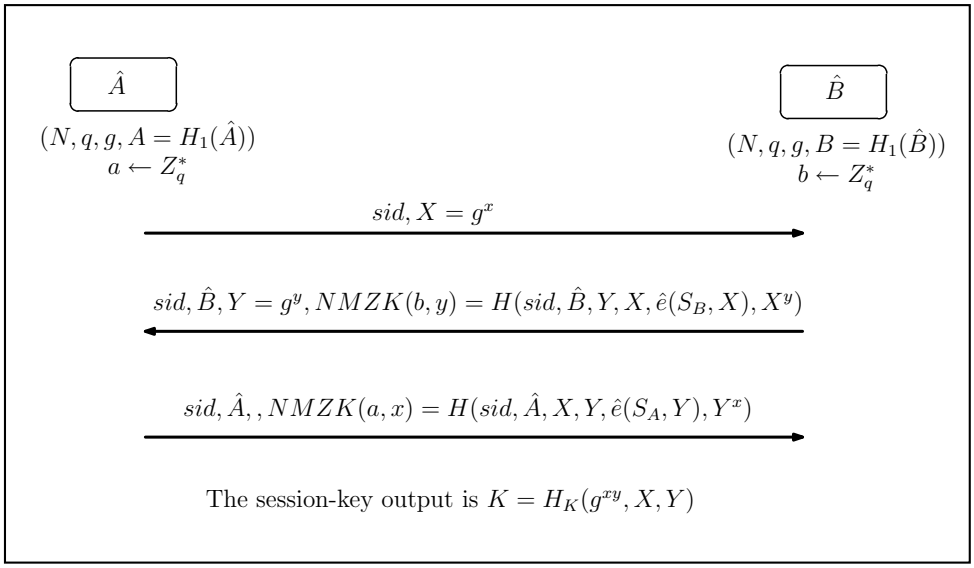


Figure 5: ID-based deniable IKE: the aggressive model

5 Prove DH-Knowledge Jointly, Non-Malleably, and (Online) Efficiently

Recall the core authentication mechanism underlying the DIKE protocol is as follows: the challenger player, e.g., $\hat{A}$, presents the DH-component $X = g^x$, and the responder player, e.g., $\hat{B}$, responds with $\{Y = g^y, H(aux, X^b, X^y)\}$ as the proof-of-knowledge of its secret-key b and DH-exponent y, where aux consists of some public system parameters. For this authentication mechanism, besides the computations of DH-components and $G/1_G$ membership tests, $\hat{B}$ needs to compute X^b and X^y that amounts to about 1.2 exponentiations in parallel, $\hat{A}$ has to perform about two exponentiations B^x and Y^x to verify the correctness of the authentication.

In order to improve the computational complexity of this authentication mechanism, one natural attempt is as follows: the values X^b, X^y are replaced by X^{b+y}. By such a modification, $\hat{B}$ only needs to perform one exponentiation to compute X^{b+y}, and $\hat{A}$ needs to compute $B^x Y^x (= X^{b+y})$ that amounts to 1.5 exponentiations. Moreover, as B^x can be offline pre-computed by $\hat{A}$, the online computational complexity of $\hat{A}$ can be only one exponentiation.

But, such a modification turns the originally sound authentication mechanism into a fundamentally flawed one. Consider the following attack: an adversary $\mathscr{A}$ can simply impersonate $\hat{B}$ by setting $Y = B^{-1} \cdot g^z$ for any value $z \in Z_q$. Note that $y = -b + z$ in this case, and $\mathscr{A}$ can trivially compute out $X^{b+y} = X^z$, but actually *without knowing the knowledge of either b or y*. That is,

$\mathscr{A}$ can malleate B and Y into X^{y+b} *by maliciously correlating the values of* y *and* b, but actually without knowing either of them.

This motivates us to investigate the essential properties for highly practical mechanisms of proving multiple DH-knowledges jointly and non-malleably, with multiple shared DH-secrets (e.g., the $X^b = CDH(X, B)$ and $X^y = CDH(X, Y)$ in the above illustration) involved (particularly, *multiplied*).We make the investigations in the spirit of achieving tag-based robust non-malleability (TBRNM) proposed in Section 4.1, which essentially says that a party finishing a session of a DHKE protocol with a (possibly malicious) peer has the guarantee that the (possibly malicious) peer does indeed "know" both the secret-key and the DH-exponent alleged and sent by the peer for that session.

5.1 Tag-Binding Self-Sealed Diffie-Hellman

As clarified, in many applications, the shared DH-secret is in turn used as the building block, *particularly multiplied with other DH-secrets*, to build more advanced and complex cryptographic objects (e.g., group Diffie-Hellman key exchange, etc). To our knowledge, it is not clear (at least not explicitly addressed) previously what are the necessary properties to securely ensure such modular uses of DH-secrets.

We observe that one intuition here for non-malleable modular uses of shared DH-secrets is: for a DHKE protocol $\langle \hat{A}, \hat{B} \rangle$, no matter how the possibly malicious $\hat{A}$ or the possibly malicious $\hat{B}$ does, and no matter what the outside components (of the advanced and complex cryptographic objects) that are composed with the DH-secret are, the shared DH-secret for a successful-finished session should be bounded to that session, in the sense that the it should not be malleated *into* or *from* other sessions. This is captured by the following definition:

Definition 5.1 (tag-binding self-seal (TBSS)) *Denote by* Z_{Tag} *the random variable of the shared DH-secret for any session specified by a* complete *(not partial)* Tag *(e.g., the session is successfully-finished). We say a DHKE protocol is* tag-binding self-sealed*, if for any value* $\alpha \in G/1_G$ *and any complete* Tag, $\Pr[Z_{Tag} = \alpha] \leq \frac{1}{2^{l-1}}$, *where* l *is the security parameter. (In the RO model, the probability is taken over only the choice of the random function.)*

Notes on the definition of TBSS. Note that the definition of TBSS does *not* require *random* DH-component contributions in each session. It holds for any (arbitrary) complete session-tag, particularly for any arbitrary DH-components X and Y (as long as $X, Y \in G/1_G$). That is, the TBSS property is formulated with respect to the set of all *complete* session-tags. The definition of TBSS particularly implies that: given a complete session-tag and the shared DH-secret (Tag, Z_{Tag}), no efficient algorithm can output, with non-negligible probability, a different $Tag' \neq Tag$ such that $Z_{Tag'}$ and Z_{Tag} *collide*, in the sense $Z_{Tag'} = Z_{Tag}$. In a sense, the DH-secret determined by a complete session-tags is "bounded" to this specific session, and is essentially "independent" of the outside world composed concurrently composed with the current session. Some particular cases of TBSS are: (1) for any value $\alpha \in G/1_G$ and any complete session-tag Tag, $\Pr[Z_{Tag} = \alpha]$ is just either 0 or $\frac{1}{2^{l-1}}$; (2) for any complete Tag, Z_{Tag} is uniformly distributed over $G/1_G$.

With respect to the original DH-protoocl (depicted in Figure 6, page 48), to render the tag-binding self-seal property to it, we define the DH-secret to be $Y^{xh(Tag)} = g^{xyh(Tag)} = X^{yh(Tag)}$, where $Tag = (\hat{A}, \hat{B}, X, Y)$ and $h : \{0,1\}^* \rightarrow \{0,1\}^l / \{0\} \subseteq Z_q^*$ is a hash function. This protocol variant is named tag-binding self-sealed Diffie-Hellman (TBSS-DH), which is depicted in Figure 6 (page 48), and also graphically illustrated in Figure 8-a (page 52). The TBSS property can be easily checked in the random oracle (RO) model by assuming the hash function h as a truly random function. Specifically, for any given $\alpha \in G/1_G$ and any given session-tag $Tag = (\hat{A}, \hat{B}, X, Y)$, the

probability of $g^{xyh(Tag)}$ being equal to α is either 0 or $\frac{1}{2^l-1}$, where the probability is taken over the choice of the random function of h.

The TBSS-DH protocol also enjoys a remarkable fairness property to the players, in the sense that: on any values $X \in G/1_G$ and $\beta \in Z_q^*$, the probability that a malicious responder player $\hat{B}$ can produce a DH-component $Y \in G/1_G$ such that $yh(\hat{A}, \hat{B}, X, Y) = \beta$ is negligible. The probability is taken on the random coins of $\hat{B}$ and the choice of the random function h. This property essentially says that a malicious responder player $\hat{B}$ cannot make the value $yh(\hat{A}, \hat{B}, X, Y)$ dependent (in an essential way) upon the initiator's DH-component. Note that, in contrast, in the original DH-protocol the responder has seemingly more advantageous power to affect the DH-secret than the initiator, by generating its DH-component Y depending upon the DH-component X sent by the initiator (a phenomenon also appears in coin flipping [14]).

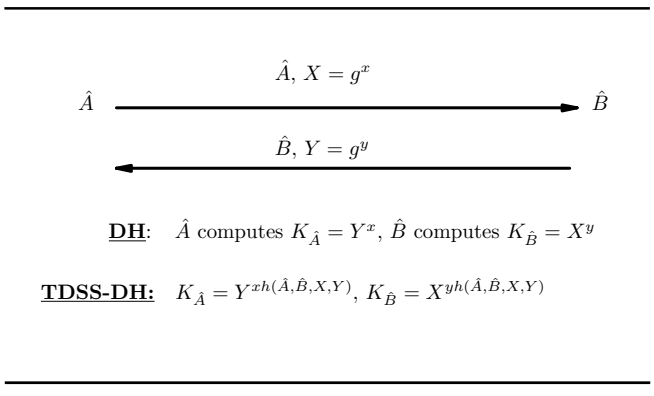


Figure 6: Specifications of DH and TBSS-DH

Notes on the hash function h. Note that if the range of h is just Z_q^*, then for any complete session-tag, the shared DH-secret $g^{xyh(Tag)}$ is distributed uniformly at random over $G/1_G$. This property holds for all DHKE protocols proposed in this work. Recall that modeling the hash function h to be a random oracle from $\{0,1\}^*$ to Z_q^* is quite common in the literature of applied cryptography, which also essentially simplifies the underlying security analysis. But, *in general*, it may not be easy to have a practical cryptographic hash function (even modelled as a random oracle) with its range just to be Z_q^* in practice. Some special cases are: $q = 2^l - 1$, or, q is negligibly close to 2^l in the sense that $|2^l - q|/2^l$ is negligible (in this case, we can define $h(x)$ of range $\{0,1\}^l$ to be $h(x) \mod q$). Also note that we assume the range of h is $\{0,1\}^l/\{0\}$. For any value α such that $h(\alpha) = 0$, we can define $h(\alpha)$ to be some non-zero value in a pre-determined manner (e.g., defining $h(\alpha)$ to be a value between $2^l + 1$ and $q - 1$, etc). In the security analysis in the RO model, for presentation simplicity, we simply treat the range of h is $\{0,1\}^l$ (rather than $\{0,1\}^l/\{0\}$).

5.2 Self-Sealed Joint Proof-of-Knowledge (SSJPOK)

Now, we return back to the attempts to further improve the efficiency of the core authentication mechanism of DIKE. Recall that replacing the *separate* values (X^b, X^y) with a single value X^{b+y} turns the originally sound authentication mechanism of DIKE into a fundamentally flawed one. The underlying reason is: $\mathscr{A}$ can malleate B and Y into X^{y+b} *by maliciously correlating the values of y and b*, but actually without knowing either of them.

In view of the above observations and discussions, one natural attempt to further remedy the situation is to mask the exponents b and y by some random values. In this case, the proof is

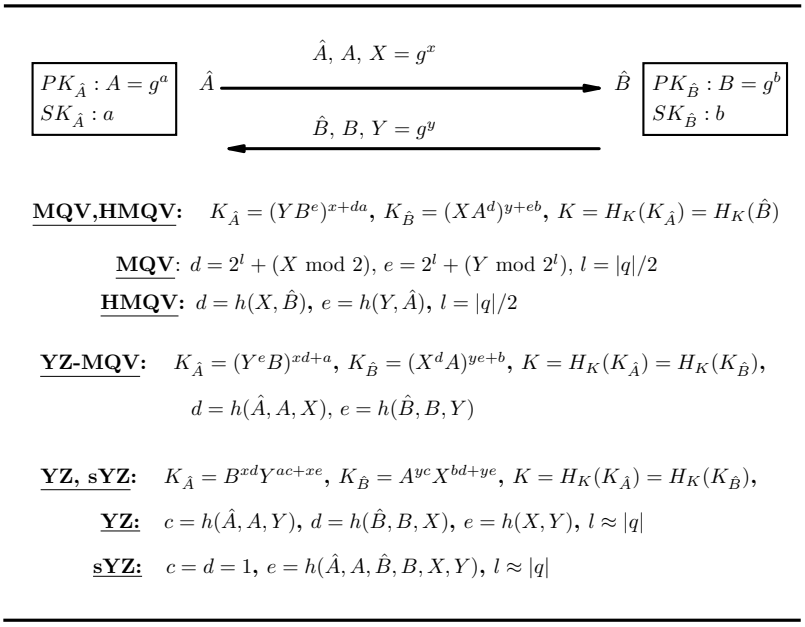


Figure 7: Specifications of (H)MQV, YZ-MQV and YZ/sYZ

denoted as $JPOK_{(b,y)} = X^{db+ey}$, where d and e are random values and JPOK stands for "joint proof-of-knowledge". This intuition turns out also to be wrong. With the values $d = h(B, \hat{A})$ and $e = h(X, \hat{B})$ as an illustrative example, after receiving X a malicious $\hat{B}$ generates and sends $Y = B^{-d/e}$, and in this case $JPOK_{(b,y)} = X^{db+ey} = 1_G$. Note that, as the underlying PKI does not require POK/POP during key registration, the malicious $\hat{B}$ does not necessarily know b and thus also y. This attack shows that masking b and y by random values is not sufficient for ensuring the non-malleability of the JPOK. The key point here is that they are *not* necessarily *independent*.

Denote by $JPOK_{(b,y)} = f_0^h(X^b, aux_0) \cdot f_1^h(X^y, aux_1)$ the general structure of JPOK of the knowledge both b and y w.r.t. a challenge DH-component X, where f_1 and f_2 are some functions from $\{0,1\}^*$ to $G/1_G$ with oracle access to an RO h, each aux_δ for $\delta \in \{0,1\}$ is a subset of public values (e.g., a subset of the session-tag). Denote $S_0 = \{X, B\}$, $Z_0 = CDH(X, B)$, $F_0 = f_0^h(Z_0, aux_0)$, $S_1 = \{X, Y\}$, $Z_1 = CDH(X, Y)$ and $F_1 = f_1^h(Z_1, aux_1)$. Here, informally speaking, by "independent" we mean that the inside multiplied components F_1 and F_2 of $JPOK_{(b,y)}$ should be *computationally independent*, no matter how a malicious knowledge-prover $\hat{B}$ does. That is, the adversarial attempts at Z_δ for any $\delta \in \{0,1\}$ should be essentially *sealed*, i.e., localized, to F_δ, and are isolated, i.e., "independent", from the adversarial attempts at $Z_{1-\delta}$.

Definition 5.2 (computational independence) *We formulate two types of "computational independence" w.r.t. $JPOK_{(b,y)}$:*

Self-sealed computational independence. *Given arbitrary values $(\alpha, \beta) \in G^2$, no matter how a malicious PPT knowledge-prover $\hat{B}$ does, the probabilities $\Pr[F_0 = \alpha]$ and $\Pr[F_1 = \beta]$ are both negligible. (With respect to the specific structure of $JPOK_{(b,y)} = X^{db+ey}$, this means that given X and arbitrary $(\beta, \gamma) \in (Z_q^*)^2$, no polynomial-time prover $\hat{B}$ can present the values $(B, Y) \in (G/1_G)^2$ such that any one of the probabilities $\Pr[db = \beta]$ and $\Pr[ey = \gamma]$ is non-negligible.)*

Committed computational independence. *There exists $\delta \in \{0,1\}$ such that for any $\alpha \in G$,*

the probability $\Pr[F_\delta = \alpha]$ *is negligible, no matter how a malicious PPT prover* $\hat{B}$ *does. This captures the "computational independence" of* F_δ *on* $F_{1-\delta}$*;*

On the other hand, the value $F_{1-\delta}$ *is committed to* F_δ*, in the sense that* $\{S_{1-\delta}, aux_{1-\delta}\} \subseteq aux_\delta$*, and given* (Z_δ, aux_δ) *no efficient knowledge-prover can provide, with non-negligible probability,* $\{S'_{1-\delta}, aux'_{1-\delta}\} \subseteq aux'_\delta$ *(w.r.t. the same challenge DH-component* $X = S_{1-\delta} \cap S'_{1-\delta}$*) such that* $\{S'_{1-\delta}, aux'_{1-\delta}\} \neq \{S_{1-\delta}, aux_{1-\delta}\}$ *but* $f^h_\delta(Z_\delta, aux_\delta) = f^h_\delta(Z_\delta, aux'_\delta)$*. This implies the "computational independence" of* $F_{1-\delta}$ *on* F_δ*.*

The probabilities are taken over the random coins used by (the possibly malicious) $\hat{B}$ *and (the honest)* $\hat{A}$ *(and the choice of the random function* h *in the random oracle model).*

To ensure computational independence between db and ey, one way is: we put the values $(B, 0)$ into the input of d, and put the values $(Y, 1)$ into the input of e. By this way and viewing h as a random oracle, it is easy to check that the values db and ey are computationally independent, no matter how the adversary maliciously forms B and Y.

It might be okey if we only consider the security of the honest knowledge verifier $\hat{A}$ against the possibly malicious knowledge prover $\hat{B}$. But, this solution is still insufficient for our purpose, specifically, for the non-malleable modular use of X^{db+ey} as a building block to achieve higher-level cryptographic objects. For example, to further strengthen the security of the honest knowledge-verifier and to provide mutual authentications, we consider the following modular use of the X^{db+ey}: define the shared DH-secret to be $A^{cy}X^{db+ey} = B^{dx}Y^{ca+ex}$, where $c = h(Y)$, $d = h(B, 0)$ and $e = h(Y, 1)$; and require $\hat{B}$ send $H_K(1, A^{cy}X^{db+ey})$ and $\hat{A}$ send $H_K(0, B^{dx}Y^{ca+ex})$ for mutual authentications and identifications. Clearly, this strengthens the security of the honest player $\hat{A}$ against possibly malicious $\hat{B}$, and provides seemingly reasonable mutual authentications and identifications. But, the intuition about mutual authentications/identifications is also wrong. Specifically, consider the following attacks: (1) KCI vulnerability: if the static secret-key b is leaked to a malicious player $\hat{A}$, $\hat{A}$ can set $X = A^{-c/e}$ and then compute out $B^{dx}Y^{ca+ex} = B^{dx} = X^{db}$, but actually without knowing either a or x. (2) If $\hat{A}$ can set its public-key based its peer's DH-component (e.g., when $\hat{A}$ works in the post-ID model), $\hat{A}$ can set $X = Y$ and $A = (B^dY^e)^{-c^{-1}}$, and then compute out $B^{dx}Y^{ca+ex} = 1_G$, again without knowing either a or x. We remark that the underlying reason beneath the above attacks is partially due to the lack of tag-binding self-seal of the DH-secret $JPOK_{(b,y)} = X^{db+ey}$ for $d = h(B, 0)$ and $e = h(Y, 1)$ (that is in turn multiplied by other DH-secrets to compute the actual shared key). For example, the values X^{db+ey} and X'^{db+ey} derived from two different tags $(\hat{A}, \hat{B}, B, Y, X)$ and $(\hat{A}, \hat{B}, B, Y, X')$, where $X' \neq X$ but correlated (e.g., $X' = X^2$), are clearly not independent.

All the above clarifications and discussions are boiled down to that, to fit the goal of tag-based robust non-malleability, the joint proof-of-knowledge $JPOK_{(b,y)} = X^{db+ey}$ should ensure:

1. db and ey (specifically, X^{db} and X^{ey}) are computationally independent, no matter how a malicious knowledge-prover $\hat{B}$ does. A stronger requirement is: the values dbx and eyx are computationally independent, no matter how a malicious $\hat{B}$ or a malicious $\hat{A}$ does. [9]

2. The shared DH-secret, i.e., X^{db+ey}, is tag-binding self-sealed. Or, d and e at least commit to the partial session-tag known by $\hat{B}$ at the point of sending X^{db+ey} (e.g., when $\hat{A}$ works in the

[9] In certain scenarios or applications, e.g., it is ensured that one's public-key is independent of the ephemeral DH-component of its honest peer (e.g., each player possesses an authenticated copy of its peer's public-key certificate prior to the protocol run, and generates *fresh* ephemeral DH-components during the evolution of the session), and/or explicit POK/POP of key registration, the above independence requirements may be relaxed to hold only for any (non-identity) *ephemeral* DH-components of that player, i.e., no matter how the non-identity ephemeral DH-components are generated by that (possibly malicious) player.

post-ID model).[10]

3. To ease online efficiency, we can additionally require: the value B^{xd} can be pre-computed by $\hat{A}$ offline.

Under the above observations, we propose two preferable solutions for non-malleable joint proof-of-knowledge.

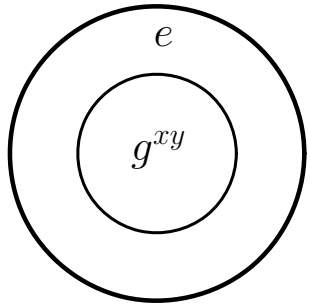
- **Self-sealed joint proof-of-knowledge (SSJPOK).** $SSJPOK_{(b,y)} = X^{db+ey}$, where $d = h(\hat{A}, \hat{B}, B, X)$ and $e = h(X, Y)$. It is easy to check that db and ey are *self-sealed* computationally independent. Specifically, for any given values (B, Y) (which determine (b, y)) and any value $\beta \in Z_q^*$, $\Pr[db = \beta]$ (resp., $\Pr[ey = \beta]$) is constant: either 0 or $\frac{1}{2^l - 1}$ in the RO model, which then implies (by a straightforward calculation) the tag-binding self-seal property, i.e., for any given session-tag $(\hat{A}, A, \hat{B}, B, X, Y)$ and any value $\alpha \in G/1_G$, $\Pr[SSJPOK_{(b,y)} = \alpha] \leq \frac{1}{2^l - 1}$. An interesting graphic illustration of the SSJPOK is depicted in Figure 8-b (page 52). In case the verifier $\hat{A}$ does not reveal its identity until the point of sending X^{db+ey} (e.g., for preserving its privacy), we set $d = h(\hat{B}, B, X)$.
- **Single-hash SSJPOK (sSSJPOK).** $sSSJPOK_{(b,y)} = X^{db+ey}$, where $d = 1$ and $e = h(\hat{A}, \hat{B}, B, X, Y)$. Note that for sSSJPOK db and ey are *committed* computationally independent. Here, we note that for sSSJPOK, the TBSS property holds that: for any given session-tag $(\hat{A}, A, \hat{B}, B, X, Y)$ and any value $\alpha \in G/1_G$, $\Pr[SSJPOK_{(b,y)} = \alpha]$ is constant: either 0 or $\frac{1}{2^l - 1}$ (the probability is taken over only the choice of the random function h).

The $SSJPOK_{(b,y)}$ is actually the product of two *self-sealed POKs* (SSPOK): X^{db} for proving the knowledge of the static secret-key b with $d = h(\hat{A}, \hat{B}, B, X)$, and X^{ey} for proving the knowledge of the ephemeral DH-exponent y with $e = h(X, Y)$. But, by the setting mechanism of d and e, the adversarial attempts (whether by malicious $\hat{B}$ or by $\hat{A}$) at $X^b = B^x$ (resp., X^y) are essentially *sealed*, i.e., localized, to X^b (resp., X^y), and are isolated (i.e., computationally independent) from the adversarial attempts at another SSPOK $X^{ey} = Y^{ex}$ (resp., X^{db}). It is the reason we name it as *self-sealed joint proof-of-knowledge*. In particular, no matter how the possibly malicious $\hat{B}$ does, the knowledge-prover $\hat{B}$ has to compute two *independent* DH-secrets X^{db} and X^{ey} w.r.t. the fresh random challenge X. By the fresh-challenge knowledge-of-exponent (KEA) assumption against CMIM adversaries of arbitrary auxiliary input, clarified in Section 2.1, this particularly implies that $\hat{B}$ does indeed "know" both b and y.

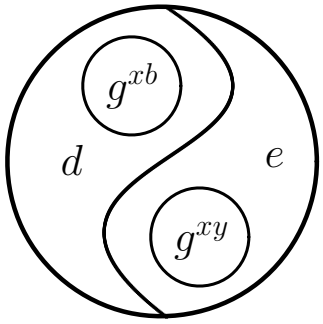
Needless to say, there are other SSJPOK candidates. But the above explicitly proposed two solutions enjoy the following advantageous properties, which make them more desirable:

- Post-ID and modular computability of SSJPOK. SSJPOK allows the knowledge-verifier $\hat{A}$ to work in the post-ID model for preserving its privacy. Specifically, as the input of e does not include $\hat{A}$'s identity and public-key, $\hat{A}$ can first send X without revealing its identity information. In this case, $\hat{B}$ can first compute X^{ey}, and then X^{db} only after learning $\hat{A}$'s identity and public-key. Also, note that without inputting Y into d allows $\hat{A}$ to pre-compute $B^{dx} (= X^{db})$ prior to the protocol run.

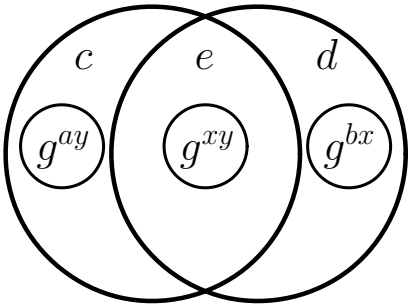
[10] In certain scenarios or applications, e.g., it is ensured that one's public-key is independent of the ephemeral DH-component of its honest peer, and/or POK/POP is explicitly required during key registration, that player's public-key may not be committed to d and e. In case players' identities are not committed to d and e, they should be put into the input of authentication and/or key-derivation functions. In practice, we suggest each player's public-key certificate itself includes the hashed value of player's identity and public-key, and only the hashed value is committed to d and e.



(a) TBSS-DH



(b) SSJPOK



(c) YZ

Figure 8: Graphic and philosophic illustrations of TBSS-DH, SSJPOK and YZ

- Efficiency and minimal RO use of sSSJPOK. sSSJPOK is preferable because of its more efficient computational complexity and the minimal use of random oracle.

Extension. Note that SSJPOK and sSSJPOK are presented here with respect to the proof-of-knowledge of only two DH-exponents, say b and y. In general, this approach can be extended to the general case of proving the knowledge of multiple (polynomially many) DH-exponents. We omit further details here.

5.2.1 Viewing SSJPOK as a New Assumption

We suggest that the SSJPOK/sSSJPOK can be viewed as a new assumption, named "*joint*-knowledge-of-exponents assumption (JKEA)". The reasonability and soundness of the JKEA assumption is justified by the above discussions, and is to be further justified by the analysis presented in Section 8.1 and Section 7.

Definition 5.3 (Joint Knowledge-of-Exponent Assumption (JKEA)) *Let G be a cyclic group of prime order q generated by an element g, and consider algorithms that on input a pair $(g, X = g^x \in G/1_G)$ output a triple $(B, Y, Z) \in (G/1_G)^3$. Such an algorithm $\mathcal{A}$ is said to be a JKEA algorithm if with non-negligible probability (over the choice of g, x and $\mathcal{A}$'s random coins) $\mathcal{A}(g, g^x)$ outputs (B, Y, Z) satisfying: (1) $Z = B^{dx}Y^{ex}$; (2) db and ey are computationally independent, no matter how $\mathcal{A}$ maliciously chooses $B \in G/1_G$ and $Y/1_G$.*

We say that the JKEA assumption holds over G, if for every efficient (probabilistic polynomial-time) JKEA algorithm $\mathcal{A}$ in G there exists another efficient algorithm $\mathcal{K}$ for which the following property holds except for a negligible probability: let (g, g^x) be an input to $\mathcal{A}$ and ρ a vector of random coins for $\mathcal{A}$ on which $\mathcal{A}$ outputs (B, Y, Z), then on the same inputs and random coins $\mathcal{K}(g, g^x, \rho)$ outputs (b, y) where $B = g^b$ and $Y = g^y$.

JKEA vs. KEA. By the KEA assumption, it only guarantees that the algorithm $\mathcal{A}$ knows $db + ey$ from the ability of computing $(B, Y, Z = B^{dx}Y^{ex})$, but it does not ensures that $\mathcal{A}$ does indeed know both b and y. Actually, the above discussions and motivations of SSJPOK/sSSJPOK indicate that knowing $db + ey$ does not necessarily imply the possession of the knowledge of b or y.

On using JKEA for achieving robust non-malleability. To use the new assumption in practice, particularly for achieving robust concurrent non-malleability against a CMIM adversary $\mathscr{A}$ of arbitrary auxiliary input, some key points should be kept in mind:

- The joint proof-of-knowledge (JPOK), provided by one party in a session, should be bounded to that session, in the sense that the JPOK should not be malleated *into* or *from* other sessions. This essentially requires the tag-binding self-seal (TBSS) property of the JPOK.[11]
- The transcript of the concurrent run of the protocol does not render any CDH solver oracle to the adversary $\mathscr{A}$, which normally is easily achievable in the RO model in practice (with some care of protocol design).
- The auxiliary input of the CMIM adversary $\mathscr{A}$ can be dependent on, i.e., any polynomial-time computable function of, all the public data (particularly, the public-keys) in the system, and the arbitrary transcript of other (different) protocol runs prior to the run of the protocol at hand (on which the CMIM $\mathscr{A}$ is to issue attacks).

[11] In case the JPOK itself loses TBSS, add the TBSS property to the session-key output by putting the (possibly partial) session tag into the input of session-key derivation function.

Approaches for weakening the JKEA assumption, and for mutual fairness. Note that, in the above JKEA assumption formulation, the JKEA algorithm $\mathcal{A}$ is given the challenge X before it provides $(B, Y, Z = B^{dx}Y^{ex})$. A seemingly weaker assumption is to require the JKEA algorithm $\mathcal{A}$ to first give (B, Y), and only after that the challenge X is then known to $\mathcal{A}$. But, for DH-based cryptographic applications, there is always a party who has to first reveal its DH-component, and thus maybe being in a disadvantageous position to its (possibly malicious) peer.

A general approach for weakening the JKEA assumption, as well as for mutual fairness, is to require the JKEA algorithm $\mathcal{A}$ (the responder) to first commit to its DH-components (B, Y) before learning X from the challenger. In practice, in case B is a public-key that is known to the challenger before presenting its challenge DH-component X, we can require the JKEA algorithm to only commit to Y, e.g., by sending $h(Y)$, on the top of the protocol run. This achieves mutual fairness and weakens the JKEA assumption, but at the price of incurred additional round-complexity. This general approach is applicable to all the protocols developed in this work.

5.3 Reinvestigations of Some DHKE Protocols, and the Exponent-Dependent Attacks

To further justify the principles observed and clarified in Section 5.1 and 5.2 for the non-malleability of cryptographic systems involving multiple DH-secrets (particularly when they are multiplied to build the more advanced and complex cryptographic objects), next, under the observations and clarifications made in Section 5.1 and 5.2 we re-investigate a number of DHKE protocols with the multiplication of multiple DH-secrets. Some salient examples of such DHKE protocols are the Burmester-Desmedt (BD) protocol [18] that is an elegant extension of the original DH-protocol into the group setting, the Chen-Kudla (CK) protocol [38] that is an elegant identity-based key-exchange protocol, and the (H)MQV protocol [100, 83, 80, 98] that is currently one of the most widely-standardized DHKE protocols.

In particular, we develop a family of attacks, named "exponent-dependent attacks (EDA)", against DHEK protocols involving multiplied multiple DH-secrets. The goal of an EDA attack is to make the multiplied multiple DH-secrets maliciously correlated, so that an adversary can successfully finish a session but without knowing the knowledge of the corresponding DH-exponents. Though we demonstrate this kind of attacks w.r.t. several well-known DHKE protocols [18, 38, 100, 83, 80, 98], it can be applied to investigate any DH-based cryptographic systems with multiplied multiple DH-secrets in general. With respect to the concrete protocol examples investigated below, we also present some remedies to the vulnerabilities caused by EDA attacks, under the principles clarified in Section 5.1 and 5.2. We stress that the following EDA attacks do not violate the security proofs of the protocols in [18, 38, 83, 80], as this kind of attacks is not captured in the underlying security models there.

5.3.1 Re-Examination of the Burmester-Desmedet (BD) Group Diffie-Hellman

Suppose $U_1, U_2, \cdots, U_n, n > 2$, be a group of parties who want to share a common group key among them. The BD-protocol [18] works as follows: Each $U_i, 1 \leq i \leq n$ computes and broadcasts $X_i = g^{x_i}$ for x_i randomly taken from Z_q^*; After that, each U_i computes and broadcasts $Z_i = (X_{i+1}/X_{i-1})^{x_i}$ (where the indices are taken in a cycle); Finally, U_i computes the shared DH-secret $K_i = (X_{i-1})^{nx_i} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2} (= g^{x_1x_2+x_2x_3+\cdots+x_nx_1})$. The tag for a successfully finished BD-protocol session is defined to be $(U_1, U_2, \cdots, U_n, X_1, X_2, \cdots, X_n)$.

We demonstrate an EDA attack against the BD-protocol with $n = 3$, where U_1 colludes with U_2 to be against the honest U_3 (the attack can be easily extended to the general case of $n > 3$, where some malicious players collude to be against sets of honest players). U_2 sets X_2 to be X_1^{-1} (i.e., $x_2 = -x_1$), then the shared DH-secret is $K_1 = K_2 = K_3 = g^{-x_1^2}$, no matter what DH-exponent

x_3 is chosen by the honest U_3. Note that, as x_1 may be maliciously generated by U_1, the shared DH-secret $g^{-x_1^2}$ can be arbitrary value in G with no guarantee on its randomness and independence. Furthermore, suppose the colluding U_1 and U_2 use the same X_1 and $X_2 = X_1^{-1}$ in different sessions, then the shared DH-secrets among different sessions are the same, i.e., always $g^{-x_1^2}$, no matter what efforts are made desperately by the player U_3 (actually any other player). This is clearly unfair to the honest player U_3. This attack indicates that the BD-protocol is not tag-binding self-sealed, and the lack of the TBSS property can cause essential security concern for DHKE protocols. We note that even the universally composable (UC) version of the BD-protocol, proposed in [75, 74], still does not frustrate the above attack (specifically, the TBSS and fairness issues and particularly the above attack were not captured by the UC framework there).

We present a variant of the BD-protocol, named (mutually) contributable BD (cBD), in the spirit of the TBSS-DH (depicted in Figure 7). The cBD protocol works as follows: Each $U_i, 1 \leq i \leq n$, computes and broadcasts $X_i = g^{x_i}$ for x_i randomly taken from Z_q^*; After that, U_i computes and broadcasts $Z_i = X_{i+1}^{x_i h(U_i, X_i, X_{i+1})} / X_{i-1}^{x_i h(U_{i-1}, X_{i-1}, X_i)}$; Finally, U_i computes the DH-secret $K_i = (X_{i-1})^{n x_i h(U_{i-1}, X_{i-1}, X_i)} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2} (= g^{x_1 x_2 h(U_1, X_1, X_2) + x_2 x_3 h(U_2, X_2, X_3) + \cdots + x_n x_1 h(U_n, X_n, X_1)})$.

It's easy to check that our cBD protocol ensures the following properties in the random oracle model: (1) For any value $\alpha \in G/1_G$ and any i, $1 \leq i \leq n$, as long as U_i is honest, i.e., x_i is distributed uniformly at random over Z_q^*, it is guaranteed that the probability that the shared DH-secret K_i is equal to α is negligible, no matter how the rest players collude against it. (2) The shared DH-secret K_i is tag-binding self-sealed. We remark that the cBD protocol can be converted into an authenticated group KE by the general technique of [75], and password-based group KE by the technique of [2].

5.3.2 Re-Examination of the Chen-Kudla (CK) Identity-Based KE Protocol

Let $\hat{e} : G \times G \to G_T$ be an admissible pairing [15, 3], where G is a cyclic multiplicative (or additive) group of order q generated by an element g. *To be compatible with the presentation and notations in Section 3, 4 and 6, the following descriptions of ID-YZ protocols is w.r.t. a is a cyclic multiplicative group G.* Here, an admissible pairing $\hat{e}$ satisfies the following three properties:

- Bilinear: If $g_1, g_2 \in G_1$, and $x, y \in Z_q$, then $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1_{G_T}$ (particularly, $\hat{e}(g, g)$ is the generator of G_T in case G_T is also a cyclic group of the same order q).
- Computable: If $g_1, g_2 \in G_1$, one can compute $\hat{e}(g_1, g_2)$ in polynomial-time.

The (basic version of) CK-protocol (with escrow) works as follows [38]:

Setup: The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. PKG also specifies a map-to-point hash function $H_1 : \{0,1\}^* \to G$ and a key-derivation function KDF (e.g., the hash function H_K). The public parameters are: $(G, G_T, \hat{e}, g, S, H_1, , H_K)$.

User secret-key extract: For a user with identity $\hat{A}$ the public-key is given by $A = H_1(\hat{A})$, and the PKG generates the associated secret-key of the user as $S_A = A^s$. Similarly, a user of identity $\hat{B}$ is of public-key $B = H_1(\hat{B})$ and secret-key $S_B = B^s$.

Key agreement between two users $\hat{A}$ and $\hat{B}$:

1. $\hat{A}$ picks $x \in Z_q^*$ at random and computes $X = A^x$ and sends X to $\hat{B}$.
2. $\hat{B}$ picks $y \in Z_q^*$ at random and computes $Y = B^y$ and sends Y to $\hat{A}$.

3. $\hat{A}$ computes $K_{\hat{A}} = \hat{e}(S_A, YB^x)$. Similarly, $\hat{B}$ computes $K_{\hat{B}} = \hat{e}(XA^y, S_B)$. Note that if $\hat{A}$ and $\hat{B}$ follow the protocol, they will compute the same shared secret: $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A, B)^{s(x+y)}$. Then, the actual session-key is derived from $K = KDF(K_{\hat{A}}) = KDF(K_{\hat{B}})$.

An EDA attack on the CK-protocol when $\hat{A} = \hat{B}$ (or, a reflection attack): As clarified in Section 4.5, in some scenarios, a party may want to establish a secure channel with itself (i.e., $\hat{A} = \hat{B}$ in this case). Below, we show an EDA attack on the CK-protocol, by which an adversary $\mathcal{A}$ can successfully finish a session with $\hat{A}$ in the same name of $\hat{A}$ (i.e., impersonating $\hat{B} = \hat{A}$) but without knowing the corresponding DH-exponent y (i.e., the discrete logarithm of Y) or the secret-key S_A.

After receiving $X = A^x$ from $\hat{A}$, the adversary $\mathcal{A}$ (impersonating $\hat{B} = \hat{A}$) randomly selects $\alpha \in Z_q$ and sends back $Y = g^\alpha X^{-1}$ in the same name $\hat{B} = \hat{A}$. *Note that, denote $Y = A^y = B^y$ (recall $A = B$), $\hat{A}$ does not know the DH-exponent y.* Finally, $\mathcal{A}$ computes $K_{\hat{B}} = \hat{e}(S, A)^\alpha$, and then derives the session-key from $K_{\hat{B}}$. Note that, as $B = A$ and $Y = A^y = B^y$ and $XY = g^\alpha$, $\hat{e}(S, A)^\alpha = \hat{e}(g^\alpha, S_A) = \hat{e}(XY, S_A) = \hat{e}(XA^y, S_B) = K_{\hat{B}}$. This shows that $\mathcal{A}$ successfully finishes the session but without knowing either the DH-exponent y or the secret-key S_A.

Patching approaches: Under the observations and clarifications of SSJPOK/sSSJPOK, we present some patching approaches to resist the above EDA attacks. The fixing approach is to set $K_{\hat{A}} = \hat{e}(S_A, B^{xc}Y^d)$, $K_{\hat{B}} = \hat{e}(X^c A^{yd}, S_B)$. Note that $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A, B)^{s(xc+yd)}$. We suggest two ways, referred to as SSJPOK-CK and sSSJPOK-CK respectively, to set the values of c and d:

- SSJPOK-CK: where $c = h(X)$ and $d = h(Y)$, where $h : \{0,1\}^* \to Z_q^*$ is a hash function.[12]
- sSSJPOK-CK: where $c = 1$ and $d = h(X, Y)$ [13]; Or, $c = h(X, Y)$ and $d = 1$. Note that for sSSJPOK-CK, the computation of $K_{\hat{A}}$ and that of $K_{\hat{B}}$ are not symmetric. Specifically, if $c = 1$ (resp., $d = 1$), then the computation of $K_{\hat{B}}$ (resp., $K_{\hat{A}}$) is (slightly) more efficient. In practice, we can decide to use which version of sSSJPOK-CK according to the application scenarios. For example, if sSSJPOK-CK is run in the clint-server setting, client efficiency should take priority over that of the server.

5.3.3 Re-Examination of (H)MQV

The (H)MQV protocol family [100, 83, 80, 98] may currently be the most efficient authenticated Diffie-Hellman protocols based on public-key authentication, and is one of the most widely standardized. Let $\hat{A}$ (resp., $\hat{B}$) be of public-key $A = g^a$ (resp., $B = g^b$) and secret-key a (resp., b), and H_K be a hash function of k-bit output, the (H)MQV protocol family is recalled in Figure 7 (page 49) (and the (H)MQV variants are recalled in Appendix B), where l is set to be $|q|/2$. (H)MQV has been widely standardized [4, 5, 68, 69, 104, 105, 115]. In particular, it has been announced by the US National Security Agency (NSA) as the key exchange mechanism underlying "the next generation cryptography to protect US government information", which includes the protection of "classified or mission critical national security information" [105, 80].

Despite an arguable evolution history [100, 71, 83, 80, 98], it is commonly believed that the current version of (H)MQV, resulted from [80, 98] with security analysis and discussions in the Canetti-Krawczyk framework [33] and the UC framework [25], should be robust enough and should have no other security weaknesses. Particularly in view of the nearly optimal efficiency of (H)MQV, it may be suggested that (H)MQV achieves the best for DHKE in the integrity of security and protocol efficiency.

[12] In general, we can also put the identity $\hat{A}$ (resp., $\hat{B}$) to the input of c (resp., d).

[13] Again, we can put both the identities of $\hat{A}$ and $\hat{B}$ into the input of d.

We re-investigate (H)MQV in the sprit of TBRNM proposed in Section 4.1, and show that MQV and HMQV (particularly when players' public-keys are not put into the input of d and e in HMQV[14]) fail in achieving the goal of tag-based robust non-malleability *in general*. This is demonstrated by concrete EDA attacks.

An asymmetrical DoS-like attack against (H)MQV. Given a value $X \in G$ for which the malicious player $\hat{A}$ (e.g., a client) does not necessarily know the discrete logarithm of X, $\hat{A}$ computes d and sets $A = X^{-1d^{-1}} \cdot g^t$ where $t \in Z_q^*$ and $d = h(X, \hat{B})$ for HMQV or $d = 2^l + (X \mod 2^l)$ for MQV. Note that $XA^d = X(X^{-1d^{-1}} \cdot g^t)^d = XX^{-1}g^{td} = g^{td}$. Note that shared DH-secret $K_{\hat{A}} = (XA^d)^{y+eb} = g^{tdy}g^{tdeb} = Y^{td}B^{tde}$. If $\mathcal{A}$ sets $t = 0$ then the shared DH-secret $K_{\hat{A}}$ is always 1_G; If $\mathcal{A}$ sets $t = d^{-1}$, then $K_{\hat{A}} = YB^e$. In any case, the computational complexity in computing the shared DH-secret of $\hat{A}$ is much lesser than that of its peer $\hat{B}$, which indicates some unfairness in computing the shared DH-secret. We note that the basic version of HMQV [80] does not check whether the shared DH-secret is 1_G or not. A revised version of HMQV proposed in [81], as well as MQV with implicit subgroup membership test, does check whether the shared DH-secret is 1_G, but cannot find the second attack with $t = d^{-1}$. We also note MQV does not explicitly mandate checking the shared DH-secret is not 1_G for the case of explicit subgroup membership test (e.g., for MQV implementations over groups of large cofactors). The key point is that the malicious $\mathcal{A}$ can successfully finish sessions of (H)MQV *but without knowing either static secret-keys or ephemeral DH-exponents*. Even if the above attacks are not serious, it is highly desirable to have DHKE protocols without such drawbacks. Also, it is desirable to have DHKE protocols *not* to explicitly check whether the shared DH-secret is 1_G, in order to prevent the particular attack when $t = 0$, as such additional check increases the system complexity and also computational complexity.

Besides asymmetry in computing the shared DH-secret (with adversary knowing neither static secret-key nor ephemeral DH-exponent), such drawbacks also allow more effective DoS attacks. Though an adversary can send arbitrary messages to an honest party to issue DoS attacks against the honest party (i.e., player $\hat{B}$ in the above attacks), i.e., to consume the computational resource of $\hat{B}$. But, such DoS attacks can be easily detected by the authentication mechanism of (H)MQV, e.g., by checking the validity of the underlying MAC values. But, with our above attacks, the honest player $\hat{B}$ is hard to distinguish an attack from an honest execution of (H)MQV. That is, the adversary does indeed know the shared DH-secret, and thus the session-key, and thus can communicate with $\hat{B}$ further with the session-key (e.g., to effectively consume the computational resource of $\hat{B}$). Note that there are many choices of setting the value t by the adversary, and thus explicitly checking whether the shared DH-secret is YB^e also does not work, as the adversary can choose different t in different sessions.

In general, given a list of values $X_1, \cdots, X_k$, the DoS attacker can register a list of parties with public-keys $A_1, \cdots, A_k$, such that for any $i, 1 \leq i \leq k$, $A_i = X_i^{-1d_i^{-1}} g^{t_i}$ where $d_i = h(X_i, \hat{B})$ or $2^l + (X_i \mod 2^l)$. The above attacks can also be trivially modified (actually simplified) to be against the one-round HMQV variant (recalled in Appendix B).

As the malicious $\hat{A}$ can honestly generate its public-key $A = g^a$ (and thus learning the secret-key), and $\hat{A}$ does compute out the shared DH-secret (i.e., 1_G), explicitly requiring POP/K of secret-key during public-key registration and explicit key-confirmation and mutual authentication (as required by the 3-round (H)MQV) do not prevent above attacks. Also, explicitly putting players' identities into key derivation cannot prevent the attacks as well. We note that POP/K of secret-keys are abandoned by HMQV, while POP/K of secret-keys is commonly assumed for MQV. We suggest a secure DHKE protocol should be a joint proof-of-knowledge of both static secret-key and ephemeral DH-exponent, without relying upon some external mechanisms. We also note that the in [80], checking whether the shared DH-secret is 1_G is also not mandated (though

[14]We note that it was not mandated in [80] to put players' identities into the input of d and e in HMQV.

some standards requires such a check). We stress that such asymmetric DoS-like attacks do not violate the security analysis of HMQV in [80], as such attacks are beyond the Canetti-Krawczyk security model.

We suggest that it is highly preferable for a DHKE protocol itself to ensure the essential security properties (expected for DHKE), in a way as robust as possible while without much resorting to additional requirements on the system environments beyond the protocol itself (or relying on such additional requirements as minimal as possible). Below, we present some fixing approaches without requiring POK/POP or pre-knowing peer's ID and public-key, in the spirit of adding tag-based self-seal to the shard DH-secrets. Specifically, for the two-round and three-round (H)MQV variants, set (1) $d = h(\hat{A}, A, X)$ and $e = h(\hat{B}, B, Y)$; or, (2) $d = h(X, \hat{B}, A)$ and $e = h(Y, \hat{A}, B)$; or (3) $d = h(\hat{A}, A, \hat{B}, B, X, Y)$ and $e = h(d)$, etc. For the one-round (H)MQV variant, set $d = h(\hat{A}, A, \hat{B}, B, X)$. Other components remain unchanged. *The key point here is to commit players' public-keys to* d, e. We prefer the first fixing proposal above. Here, we make some further modifications, as follows: specifically, we define $K_{\hat{A}} = (Y^e B)^{xd+a}$, $K_{\hat{B}} = (X^d A)^{ye+b}$, where $d = h(\hat{A}, A, X)$ and $e = h(\hat{B}, B, Y)$; The session-key is still $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$. This protocol variant is named as YZ-MQV, which is depicted in Figure 7. (In case of POK/POP is explicitly required during key registration and/or each player possesses an authenticated copy of its peer's public-key certificate, the public-key inputs could also be removed from the inputs of d and e). In practice, we suggest each player's public-key certificate itself includes the hashed value of the player's identity and public-key, which will be verified and authorized by the CA. In this case, $(\hat{A}, A)$ and $(\hat{B}, B)$ in the inputs of d, e can just be replaced by their hashed values already included in the certificates.

We briefly note all the above attacks fail with YZ-MQV. But, here, we did not try to analyze YZ-MQV, and the above other fixed protocol variants of MQV/HMQV (actually, we do not know how to prove the security of YZ-MQV). YZ-MQV enjoys the following features:

- YZ-MQV is tab-binding self-sealed.
- The value $xd + a$ (resp., $ye + b$) can be pre-computed by $\hat{A}$ (resp., $\hat{B}$) without knowing its peer's information. Moreover, the value B^{xd+a} (resp., A^{ye+b}) can be pre-computed by $\hat{A}$ (resp., $\hat{B}$) offline.

6 Towards (Online) Optimal Diffie-Hellman Key-Exchange

We suggest the SSJPOK and sSSJPOK, clarified in Section 5, as a practical paradigm of proving multiple DH-knowledges jointly and non-malleably or just a new *non-black-box* assumption for achieving concurrent non-malleability in a practical and conceptually simple/natural way, may be of independent interest, and of wide applications (particularly to authentication and key-exchange). In particular, they can be applicable to any cryptographic applications using the sign-then-MAC authentication mechanism [79] or involving the multiplications of multiple DH-secrets. One core application of SSJPOK/sSSJPOK, as demonstrated in this section, is a new family of DHKE protocols, referred to as the YZ protocol family, which, as we shall analyze, may essentially (maybe arguably) achieve the optimal in the integrity of security, privacy, efficiency and easy deployment. More applications are investigated in Section 9.

6.1 The YZ Protocol Family

Note that the $SSJPOK_{(b,y)}$ with respect to the random challenge X only ensures the player $\hat{A}$ that the player $\hat{B}$ does know both b and y. Denote by $SSJPOK_{(a,x)}$ the joint proof-of-knowledge

of a and x given by $\hat{A}$ with respect to the random challenge Y. In order to achieve a secure DHKE protocol (particularly, with mutual identifications and key-confirmations), a natural way is: we replace the separate POK mechanism underlying the DIKE protocol depicted in Figure 2, i.e., (X^b, X^y) (resp., (Y^b, Y^a)), by $SSJPOK_{(b,y)}$ (resp., $SSJPOK_{(a,x)}$). The session-key is still derived from g^{xy}. Though this would render us a still sound DHKE protocol, the resultant DHKE protocol essentially obtains no advantages over the DIKE protocol.

Recall that the value X^{db+ey} of $SSJPOK_{(b,y)}$ actually can be used to demonstrate the knowledge of x, as $X^{db+ey} = B^{dx}Y^{ex}$. The key observation now is: in order for $\hat{A}$ to additionally prove the knowledge of the secret-key a, we can multiply the SSJPOK $X^{db+ey} = B^{dx}Y^{ex}$ by another self-sealed POK Y^{ca} for $c = h(\hat{B}, \hat{A}, A, Y)$. This yields $\sigma_{\hat{A}} = B^{dx}Y^{ca+ex} = (B^{dx}Y^{ex}) \cdot Y^{ca} = (X^{db+ey}) \cdot Y^{ca} = (X^{db+ey}) \cdot A^{cy} = A^{cy}X^{db+ey} = \sigma_{\hat{B}}$, where $\sigma_{\hat{A}}$ is computed by $\hat{A}$ and $\sigma_{\hat{B}}$ is computed by $\hat{B}$, $c = h(\hat{B}, \hat{A}, A, Y)$, $d = h(\hat{A}, \hat{B}, B, X)$, $e = h(X, Y)$ and the tag determined is $(\hat{A}, \hat{B}, A, B, X, Y)$. Further note that the identity $\hat{A}$ (resp., $\hat{B}$) in the input of c (resp., d) is redundant to ensure the TBSS property of the shared DH-secret $\sigma_{\hat{A}} = \sigma_{\hat{B}}$, and thus can be removed. All the observations boil down to the YZ protocol, which is depicted in Figure 7 (page 49) and graphically illustrated in Figure 8-c (page 52). Recall that the underlying PKI requires no proof-of-knowledge (POP) or proof-of-possession (POP) of secret-key during key registration, but the CA will check the (non-identity) sub-group membership for the registered public-keys (i.e., make sure the registered public-keys are in $G/1_G$). Also, with our YZ protocol family, the hash functions H_K, h can be identical.

Ephemeral private values exposable to adversary. The ephemeral private values exposable to adversary, generated by the honest $\hat{B}$ (resp., $\hat{A}$) during the protocol run, are: y (resp., x) if $\hat{B}$ (resp., $\hat{A}$) does not pre-compute A^{cy} (resp., B^{dx}), or (y, A^{cy}) (resp., (x, B^{dx})) if $\hat{B}$ (resp., $\hat{A}$) pre-computes A^{cy} (resp., B^{dx}). Other ephemeral private values are erased promptly after use. We remark all ephemeral private values, except for the session-key in case the session is successfully finished, generated by an honest party within the protocol run are erased after the session is completed (whether finished or aborted). For expired sessions, the session-keys are also erased.

About output length of h. Recall that $h : \{0,1\}^* \rightarrow \{0,1\}^l/\{0\} \subseteq Z_q^*$. For reasonable tradeoff between security and efficiency, as clarified in [80], l can be shorter, e.g., $|q|/2$ where $|q| = \lfloor \log_2 q \rfloor$. Note that whether $l = |q|$ or $|q|/2$ only has a minor impact on the performance of YZ. But, setting $l = |q$ provides more robust resistance to birthday or colliding attacks against the hash function h (particularly with pre-computed or reused DH-components). Thus, for our YZ protocol family, we recommend the output length of h, i.e., l, to be approximately $|q|$. In particular, with the YZ protocol family, h and the KDF H_K can be identical.

YZ with embedded subgroup test. The basic technique to check the DH-component, e.g. X, is in G is to verify $X^q = 1_G$ (and $X \in G'/1_G$) that needs performing one modular exponentiation. But, if the cofactor t is small, e.g., $G' = Z_N^*$ such that $N = 2q + 1$ or G is the subgroup of an elliptic curve over a finite field (in this case the cofactor t is usually a small constant and can be 1), the subgroup test of X can be essentially reduced to: (1) check $X \in G'$; (2). $X^t \neq 1_G$. In general, $X \in G'$ and $X^t \neq 1_G$ guarantees that X is not in a (small) subgroup of G' with the order that is a factor of t, but it does not fully guarantee $X \in G$ (e.g., consider that $X = -g^x$). This leads to the following YZ protocol variant (the technique is also applicable to SSJPOK and other instantiations of the YZ structure):

$K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, the session-key $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$, where $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$, $e = h(X, Y)$. The subgroup test is performed as follows: each player verifies that its peer's DH-component is in G'; for the party $\hat{A}$ (resp., $\hat{B}$), if B^{dxt} and $Y^{cat+ext}$ (resp., A^{cyt} and $X^{dbt+eyt}$) are computed separately, $\hat{A}$ (resp., $\hat{B}$) checks that $Y^{cat+ext} \neq 1_G$ (resp., $X^{dbt+eyt} \neq 1_G$); In case of no separate computation, $\hat{A}$ (resp., $\hat{B}$) verifies $K_{\hat{A}} \neq 1_G$ (resp., $K_{\hat{B}} \neq 1_G$). Note that the checking of $K_{\hat{A}} \neq 1_G$ and $K_{\hat{B}} \neq 1_G$, just as done in the elliptic curve based MQV, does

not *fully* guarantee $X^t \neq 1_G$ or $Y^t \neq 1_G$ (i.e., not fully exclude the possibility of DH-components from small subgroups), but it still provides reasonable assurance in the elliptic curve setting.

We note that the damage caused by ignoring subgroup test of peer's DH-component (but still with the supergroup G' check) can be much relieved (and even waived), if the ephemeral private values generated by the honest party within the protocol run are well-protected. For example, even if the adversary learns some partial information about $db + ey$ by issuing a small subgroup attack against the honest $\hat{B}$ (by setting X to be in a small subgroup), it still cannot derive the value of b without compromising the ephemeral value y. Further note that the adversary actually cannot derive the full value of $db + ey$ by small subgroup attacks, as the DH-exponent y is independent at random in each session.

For presentation simplicity and unity, in the rest of this paper, it is assumed that $t = \frac{N}{q}$ for implementations with embedded subgroup test (e.g., for elliptic curve based implementations), and $t = 1$ if each player makes sure the $G/1_G$ membership of its peer's DH-component.

High-level overview of the security of YZ. For the YZ-protocol depicted in Figure 7, at a high level, to correctly compute the shared session-key, no matter what adversarial strategies are employed by the CMIM adversary $\mathscr{A}$, $\mathscr{A}$ has to compute out three *computationally independent* DH-secrets: A^{cy}, X^{db} and X^{ey}. As A^{cy} is computationally independent from (i.e., cannot be made maliciously correlated to) X^{db+ey} or X^{b+ey}, to successfully finish the test-session, the adversary $\mathscr{A}$ has to compute out X^{db+ey} or X^{b+ey} *w.r.t. the fresh random challenge* X. As the random challenge X is fresh, the arbitrary auxiliary input possessed by the CMIM adversary $\mathscr{A}$, which captures arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from the executions of arbitrary (*possibly different*) protocols prior to its interactions with the protocol instances of YZ, does not help $\mathscr{A}$ to compute X^{db+ey} or X^{b+ey}. On the other hand, the transcript of the concurrent run of YZ does not render any CDH solver oracle to the adversary. Further, the exposable ephemeral private values also do not render the adversary any additional advantages other than what can be derived merely from the exposed ephemeral DH-exponents, and thus no CDH solver oracle is rendered as well. This implies that $\mathscr{A}$ has to "know" both b and y to correctly compute the shared session-key of YZ. Detailed security analysis of YZ are presented in Section 7 and Section 8.

6.2 YZ Variants

In this section, we investigate some useful variants of the YZ protocol depicted in Figure 7, which are themselves of independent interest. More variants within the general YZ protocol structure are discussed in Appendix C.

Single-hash YZ (sYZ): $K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, the session-key is $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$, where $c = d = 1$ and $e = h(\hat{A}, A, \hat{B}, B, X, Y)$.

Note that sYZ is designed with the single-hash SSJPOK, and is aimed at minimal use of random oracles. Also note that, in contrast to YZ where the components A^{cy}, X^{db} and X^{ey} are all computationally independent, for sYZ, it is clear that A^y and X^b are not independent, but X^{ey} is still computationally independent from each of them. The detailed security analysis of sYZ is also presented in Section 7 and Section 8.

Robust YZ (rYZ): $K_{\hat{A}} = B^{fa+dxt}Y^{cat+ext}(\neq 1_G)$ and $K_{\hat{B}} = A^{fb+cyt}X^{dbt+eyt}(\neq 1_G)$, the session-key is $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$, where $c = h(\hat{A}, A, \hat{B}, B, Y)$, $d = h(\hat{A}, A, \hat{B}, B, X)$, $e = h(\hat{A}, A, \hat{B}, B, X, Y)$.

Actually, the function e can be set to be $h(c, d)$ or $h(c, X)$ or $h(d, Y)$. The key point is to put both X and Y into the input of the function e. In particular, if f=0, e also can be set to be $h(X, Y)$. But if $f = 1$ and $e = h(X, Y)$, there are the following (seemingly unreasonable) weaknesses: the player $\hat{B}$ sets $Y = A^{-1}$ and $B = X^{et}$, then $K_{\hat{B}} = A^{fb+cyt}X^{dbt+eyt}$ is actually equal to $A^{cyt}X^{dbt}$.

It's true that rYZ when $f = 0$ provides the security as strong as that of the YZ protocol (depicted in Figure 7), at the price of sacrificing the advantage of post-ID and modular computability. But, for rYZ when $f = 1$, the situation is a bit subtler. Specifically, suppose a CMIM adversary $\mathscr{A}$ interacts with $\hat{A}$ as the responder in the name $\hat{M}_1$ in a session (referred to as the test-session), and concurrently interacts with $\hat{B}$ as the initiator in the name of $\hat{M}_2$ in another session (referred to as the second-session). After receiving X from $\hat{A}$ in the first session, $\mathscr{A}$ registers X as the public-key of $\hat{M}_2$ in the second-session. In case *both* of the ephemeral private values y and X^{b+cyt} (pre-computed by $\hat{B}$ in the second-session) are exposed to $\mathscr{A}$, $\mathscr{A}$ can compute out $X^b = (X^{b+cyt}/X^{cyt})$, where $c = h(\hat{M}_2, X, \hat{B}, B, Y)$. Now, further suppose $\mathscr{A}$ also learns the static secret-key a of $\hat{A}$, $\mathscr{A}$ can then successfully impersonate the player $\hat{B}$ (i.e., $\hat{M}_1 = \hat{B}$) in the test-session *but actually without knowing the secret-key* b (to this end, $\mathscr{A}$ will generate the DH-component honestly in the test-session). That is, in this case, $\mathscr{A}$ computes $K_{\hat{M}_1} = K_{\hat{B}} = B^a A^{cyt} (X^b)^{dt} X^{eyt} (= A^{b+cyt} X^{dbt+eyt})$ using the knowledge (a, y, X^b) but without knowing b.

We remark that the underlying reason beneath the above attacks is: the leakage of the ephemeral values of *both* y and X^{b+cyt} can render a CDH solver oracle, say X^b, to the adversary. This concrete attack can be prevented by explicitly requiring POK/POP of secret-key during public-key registration. We remark that the POP/POK of secret-keys, or special care of ephemeral private values, for rYZ with $f = 1$ and pre-computation of A^{fb+cy} and B^{fa+dx}, is only against the powerful adversary who can expose *both* the static secret-key of $\hat{A}$ and the ephemeral private values of $\hat{B}$. Without exposing either of them, the above demonstrated attack will fail.

One-round YZ (orYZ): The player $\hat{A}$ sends $X = g^x$ to $\hat{B}$. Normally, $\hat{A}$ is a client machine and $\hat{B}$ is a server machine. Let $K_{\hat{A}} = B^{a+ex}$ and $K_{\hat{B}} = A^b X^{eb}$, where $e = h(\hat{A}, A, \hat{B}, B, X)$ and the session-key is $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$. For orYZ, it is also recommend to set the output length of h to be shorter, e.g., $|q|/2$, to ease the computation of $K_{\hat{B}} = A^b X^{eb} = (AX^e)^b$ in some application scenarios (e.g., when the pre-computation of A^b is inconvenient). This technique is also applicable to the single-hash SSJPOK.

Note that all the computational complexity of $\hat{A}$ is 2 exponentiations in total and all the computation of $\hat{A}$ can be offline. To improve the on-line efficiency of $\hat{B}$, the player $\hat{B}$ can pre-compute A^b in an off-line way (and store it in a database entry corresponding to the player $\hat{A}$), and only on-line computes X^{eb} and X^q which amounts to about 1.2 exponentiations (it is recommended for $\hat{B}$ to explicitly check the subgroup membership of X). In case of embedded subgroup test, $\hat{B}$ should explicitly check $X \in G'$ and $X^{ebt} \neq 1_G$ (only checking $K_{\hat{B}} \neq 1_G$ is not sufficient to prevent the small subgroup attack). We remark that the one-round YZ intrinsically suffers from the KCI vulnerability in case $\hat{B}$'s static secret-key b is compromised, and lacks perfect forward secrecy.

About key derivation and mutual authentication. For seemingly more robust security (particularly when the output length of h is short, e.g., $|q|/2$, and/or, with pre-computed or reused DH-components), it is recommended (for resisting birthday or colliding attacks) to put both players' identities (and public-key information), or even just the session-tag, into the input of the key-derivation function. Another way is to put (c, d) into the input of H_K for the key derivations of the YZ, and to put e into the input of H_K for the key derivations of the rYZ, sYZ and orYZ protocols. These methods are applicable to the key derivations in various applications of the YZ protocol family.

For adding mutual authentication to the YZ protocol family, besides the session-key K we also need a MAC-key K_m to be used within the protocol run (but erased after the protocol run). There are various implementations of the key-derivation function (KDF) to generate the session-key and the MAC-key: (1) $(K, K_m) \leftarrow H(K_{\hat{A}}, 1)H(K_{\hat{A}}, 2) \cdots H(K_{\hat{A}}, i)$, where i is a counter such that it increases until the concatenation of hash outputs is longer than the length of (K, K_m); (2). $K_m = H_K(c, d, K_{\hat{A}})$ and $K = H_K(K_{\hat{A}}, c, d)$ for YZ, and $K_m = H_K(e, K_{\hat{A}})$ and $K = H_K(K_{\hat{A}}, e)$ for rYZ, sYZ and orYZ; (3). $K_m = H_K(K_{\hat{A}}, 0)$ and $K = H_K(K_{\hat{A}}, 1)$; (4). The hash functions used for

session-key derivation and for MAC-key derivation are different; etc. Again, for all possible key-derivation methods, the values (c, d) or e can be put into the input of the key derivation functions. In general, KDF can be implemented with a pseudorandom function (PDF).

For mutual authentications of YZ, rYZ and sYZ, $\hat{B}$ sends $t_B = MAC_{K_m}(1)$ in the second-round, and $\hat{A}$ sends $t_A = MAC_{K_m}(0)$ in an additional third-round. For mutual authentication of the one-round YZ, the player $\hat{A}$ can additionally send $t_A = MAC_{K_m}(0)$ in the first-round, and the player $\hat{B}$ responds back $MAC_{K_m}(1)$ in the subsequent round. In practice, the MAC and PDF functions can be instantiated with HMAC [7].

About putting public-keys and identities into the input of c, d and/or e. The underlying rational of putting the public-key A (resp., B) into the input of c (resp., d) for YZ, and putting (A, B) into the input e for sYZ, is to ensure the non-malleability of SSJPOK. As clarified in Section 5.2 and particularly demonstrated with the weak YZ variants discussed in Appendix C, this is necessary for deploying the YZ protocol family in the settings of post-ID and/or key-registration without POK/POP. But, to ease protocol deployments, in case of key registration with POP/POK and each player knows and makes sure the validity of its peer's public-key prior to the protocol run, the public-keys in the input of c, d, e may also be removed.

Putting player identity $\hat{A}$ (resp. $\hat{B}$) into the input of c (resp. d) for YZ, $(\hat{A}, \hat{B})$ into the input of e for sYZ, plays an essential role for achieving tag-based robust non-malleability (e.g., resistance against UKS attacks, etc). In case one interests with variants with players' identities are removed from c, d and/or e, it is strongly recommend to put player's identities into the input of KDF and explicitly require POK/POK of key registration.

In practice, we suggest each player's public-key certificate itself includes the hashed value of the player's identity and public-key, which will be verified and authorized by the CA. In this case, $(\hat{A}, A)$ and $(\hat{B}, B)$ in the input of c, d, e can just be replaced by their hashed values already included in the certificates.

About pre-computing and reusing DH-components. In some scenarios, e.g., for online efficiency or for resistance against denial-of-service (DoS) attacks, the player may wish to pre-compute and even reuse the same DH-components in multiple sessions within a reasonable (short) time period. To ease such DH-component reuses, as well as to strengthen the security of the protocol, we can replace X (resp., Y) in the input d and/or e (resp., c and/or e) to be $(r_{\hat{A}}, X)$ (resp., $(r_{\hat{B}}, Y)$), where $r_{\hat{A}}$ (resp., $r_{\hat{B}}$) is the random nonce sent by $\hat{A}$ (resp., $\hat{B}$). This ensures that even each player reuses its DH-components in multiple sessions, the session-keys are still independent at random.

6.2.1 Client Privacy-Preserving and Server DoS-Resistant YZ

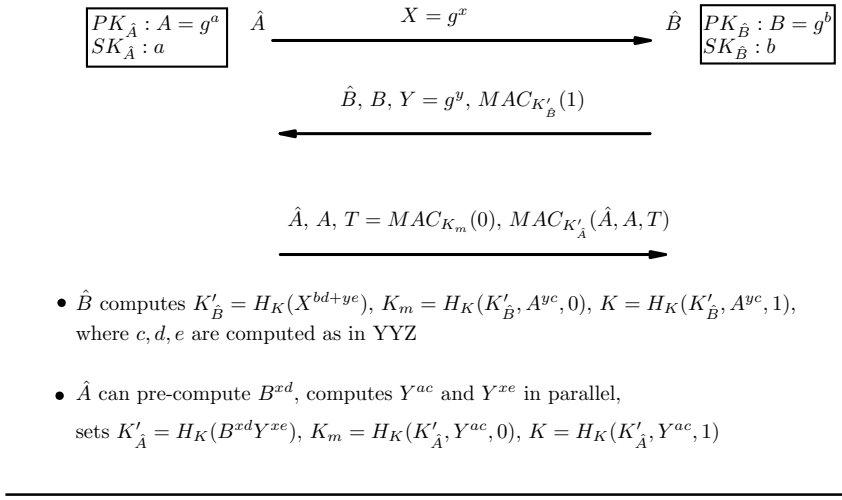
We present a variant of the YZ protocol in the distributed client-server setting, which is named as client-server YZ (csYZ). The rough protocol structure of csYZ is depicted in Figure 9 (page 63).

The following are the detailed description of the csYZ protocol.

Round-1. The client (denoted $\hat{A}$) sends $X = g^x$ to the server (denoted B).

After receiving X, if $\hat{B}$ performs the explicit subgroup test it will abort in case $X \notin G$. If $\hat{B}$ performs embedded subgroup test, $\hat{B}$ first verifies $X \in G'$ (and aborts if the verification fails), and then computes $X^{dbt+eyt}$, where $d = h(\hat{B}, B, X)$, $e = h(X, Y)$ and $Y = g^y$ is the DH-component of $\hat{B}$. $\hat{B}$ verifies $X^{dbt+eyt} \neq 1_G$ (and aborts in case of verification failure). Then, $\hat{B}$ computes $K'_{\hat{B}} = H_K(X^{dbt+eyt})$, erases $X^{dbt+eyt}$ from the ephemeral private state (but keeps $K'_{\hat{B}}$ and y in secret), and moves to Round-2.

Round-2. $\hat{B}$ sends $\{Cert_{\hat{B}}, Y, t'_B = MAC_{K'_{\hat{B}}}(1)\}$ to $\hat{A}$, where $Cert_{\hat{B}}$ is $\hat{B}$'s public-key certificate.



- $\hat{B}$ computes $K'_{\hat{B}} = H_K(X^{bd+ye})$, $K_m = H_K(K'_{\hat{B}}, A^{yc}, 0)$, $K = H_K(K'_{\hat{B}}, A^{yc}, 1)$, where c, d, e are computed as in YYZ
- $\hat{A}$ can pre-compute B^{xd}, computes Y^{ac} and Y^{xe} in parallel, sets $K'_{\hat{A}} = H_K(B^{xd}Y^{xe})$, $K_m = H_K(K'_{\hat{A}}, Y^{ac}, 0)$, $K = H_K(K'_{\hat{A}}, Y^{ac}, 1)$

Figure 9: The csYZ variant

$\hat{A}$ first verifies $Y \in G'$ and aborts if the verification fails. $\hat{A}$ then verifies the validity of $Cert_{\hat{B}}$ and aborts if the verification fails. Then, $\hat{A}$ does one of the following two actions:

Action-1. $\hat{A}$ computes Y^{ext} and Y^{cat} (and Y^q for explicit sub-group test) in parallel (which amounts to about 1.2 exponentiations), checks that $Y^{ext} \neq 1_G$ or $Y^{cat} \neq 1_G$ (or $Y^q \neq 1_G$), and aborts in case any of the checks failures. $\hat{A}$ then computes $K'_{\hat{A}} = H_K(B^{dxt}Y^{ext}) = K'_{\hat{B}}$, where B^{dxt} can be pre-computed offline prior to the protocol run. Then, $\hat{A}$ verifies the validity of t'_B and aborts in case of verification failure. Finally, $\hat{A}$ computes the MAC-key $K_m = H_K(K'_{\hat{A}}, Y^{act}, 0)$ and the session-key $K = H_K(K'_{\hat{A}}, Y^{act}, 1)$.

Action-2. $\hat{A}$ computes $B^{dxt}Y^{ext}$ (which amounts to 1.5 exponentiations), checks $B^{dxt}Y^{ext} \neq 1_G$ (in case for embedded sub-group test) and aborts in case of check failure, and then computes $K'_{\hat{A}} = H_K(B^{dxt}Y^{ext}) = K'_{\hat{B}}$; $\hat{A}$ verifies the validity of t'_B and aborts in case of verification failure. Then, $\hat{A}$ computes Y^{cat} (possibly and Y^q in parallel in case for explicit sub-group test), where $c = h(\hat{A}, A, Y)$, verifies $Y^{act} \neq 1_G$ (in case for embedded sub-group test) and aborts in case of verification failure. Finally, $\hat{A}$ computes the MAC-key $K_m = H_K(K'_{\hat{A}}, Y^{act}, 0)$ and the session-key $K = H_K(K'_{\hat{A}}, Y^{act}, 1)$.

$\hat{A}$ computes $T = MAC_{K_m}(0)$ and $t'_A = MAC_{K'_{\hat{A}}}(Cert_{\hat{A}}, T)$, where $Cert_{\hat{A}}$ is $\hat{A}$'s certificate. Then, $\hat{A}$ keeps the session-key in secret, erases all other ephemeral private values generated within the protocol run, and moves to Round-3.

We remark that the ephemeral private values generated by $\hat{A}$ within the protocol run, which are potentially exposable to adversary, are: (x, B^{dxt}) in case of pre-computation of B^{dxt}, or just x without pre-computation of B^{dxt}. Other ephemeral private states are erased promptly after use. Note that all ephemeral private values generated by $\hat{A}$ within the protocol run, except for the session-key in case the session is successfully finished, are erased after the session is completed (whether finished or aborted); And for expired sessions the underlying session-keys are also erased.

Round-3: $\hat{A}$ sends $\{Cert_{\hat{A}}, T = MAC_{K_m}(0), t'_A = MAC_{K'_m}(Cert_{\hat{A}}, T)\}$ to $\hat{B}$, and finishes the session.

After receiving the Round-3 messages, $\hat{B}$ first uses $K'_{\hat{B}}$ to verify the validity of t'_A (and aborts in case of verification failure). Then, $\hat{B}$ verifiers the validity of $Cert_{\hat{A}}$ (and aborts in case of verification failure). $\hat{B}$ then computes A^{yct}, computes the MAC-key $K_m = H_K(K'_{\hat{B}}, A^{yct}, 0)$ and the session-key $K = H_K(K'_{\hat{B}}, A^{yct}, 1)$. $\hat{B}$ verifies the validity of T (and aborts in case of verification failure). $\hat{B}$ keeps the session-key in secret, erases all ephemeral private values generated within the protocol run, and finishes the session.

We remark that the ephemeral private values generated by $\hat{B}$ within the protocol run, which are potentially exposable to adversary, are: $\{y, K'_{\hat{B}}\}$. Other ephemeral private states are erased promptly after use. All ephemeral private values generated by $\hat{B}$ within the protocol run, except for the session-key in case the session is successfully finished, are erased after the session is completed (whether finished or aborted); For expired sessions the underlying session-keys are also erased.

Advantageous features of csYZ. The csYZ enjoys the following advantageous features, which make the csYZ protocol itself very attractive and being of independent interest:

Preserving client privacy: post-ID and zero-knowledge. Only after the server authenticates to the client, the client reveals its identity and authenticates itself to the server. That is, the client works in the post-ID model [35].

The interaction transcript of the protocol run is zero-knowledge (ZK) against possibly malicious server of arbitrary auxiliary input (that captures any arbitrary information collected by the server prior to the protocol run). The ZK property is from the following observations: (1) The transcript of the protocol run can be generated merely from the DH-exponents (x, y) *without using either the static secret-key a or b*; (2) Before the client $\hat{A}$ sends the authentication message at Round-3, the server has proved its knowledge of y by t'_B at Round-2 via the underlying building tool SSJPOK.

Server capability of denial-of-service (DoS) resistance, and reasonable deniability. The server $\hat{B}$ can perform only one on-line exponentiation, i.e., $K'_{\hat{B}}$, to detect and resist against potential DoS attacks. Note that Y can be pre-computed and can be reused in multiple sessions in case of facing DoS attacks; Also note that $\hat{B}$ computes A^{cy} and verifies the validity $Cert_{\hat{A}}$ *only after verifying the validity of* t'_A (that does not need performing additional exponentiations).

The reasonable deniability to $\hat{B}$ is still from the observation: the transcript of the protocol run can be generated merely from the DH-exponents (x, y) *without using either the static secret-key a or b*.

Nearly optimal (online) efficiency. $\hat{B}$ performs 3 exponentiations in total. $\hat{A}$ performs about 3.5 exponentiations in total with the computation of $K'_{\hat{B}} = B^{dxt}Y^{ext}$, or, about 3.2 exponentiations in total with the computation of Y^{ext} and Y^{cat} in parallel. As the computation of B^{dxt} and DH-components can be pre-computed offline, the on-line efficiency of csYZ can be 1.2 exponentiations for $\hat{A}$ and 2 exponentiations for $\hat{B}$. Also, note that in case $\hat{B}$ knows $\hat{A}$'s identity prior to the protocol run, the online efficiency of $\hat{B}$ can be just one exponentiation as A^{cyt} can also be pre-computed in this case.

Notes on another (natural) implementation of csYZ. For better compatibility with other variants of YZ, one natural implementation of the csYZ protocol is to derive the MAC-key and

the session-key from the same shared DH-secret as is derived in YZ (depicted in Figure 7). For example, the MAC-key and the session-key are set to be: $K_m = H_K(K_{\hat{A}}, 0) = H_K(K_{\hat{B}}, 0)$ and $K = H_K(K_{\hat{A}}, 1) = H_K(K_{\hat{B}}, 1)$, where $K_{\hat{A}} = B^{xtd}Y^{atc+xte}$ and $K_{\hat{B}} = A^{ytc}X^{btd+yte}$ are the same as in YZ (depicted in Figure 7).

For this natural implementation of csYZ, in spite of the above compatibility advantage it however has the following disadvantages: (1) It is more inefficient. Specifically, each player needs to perform one additional multiplication to compute out the MAC-key and the session-key; (2) The server, i.e., $\hat{B}$, may demand a more careful or involved management of its ephemeral private state, whose exposure can violate the spirit of TBRNM security. This subtlety is further addressed below.

Note that for this natural implementation of csYZ, the ephemeral private values kept by $\hat{B}$ within the protocol run, which are possibly exposable to adversary, are $(y, X^{btd+yte})$. We show the exposure of $(y, X^{btd+yte})$ can cause potential security vulnerabilities, by demonstrating a concrete attack. For presentation simplicity, we assume $t = 1$ in the following attack description. Specifically, suppose a CMIM adversary $\mathscr{A}$ interacts with $\hat{A}$ as the responder in the name $\hat{M}_1$ in a session (referred to as the test-session), and concurrently interacts with $\hat{B}$ as the initiator in the name of $\hat{M}_2$ in another session (referred to as the second-session). After receiving X from $\hat{A}$ in the first session, $\mathscr{A}$ forwards X to $\hat{B}$ in the second-session. In case the ephemeral private values $(y, X^{d_2b+e_2y})$ kept by $\hat{B}$ for the second-session are composed to $\mathscr{A}$, $\mathscr{A}$ can compute out $X^b = (X^{d_2b+e_2y}/X^{e_2y})^{d_2^{-1}}$, where $d_2 = h(\hat{B}, B, X)$ and $e_2 = h(X, Y)$. Now, $\mathscr{A}$ has two strategies:

- $\mathscr{A}$ sets $\hat{M}_1 = \hat{B}$ (and thus the public-key of $\hat{M}_1 = \hat{B}$ is B), generates and sends $Y' = g^{y'}$ to A in the second-round of the test-session (note that $\mathscr{A}$ knows the DH-exponent y'); Then, $\mathscr{A}$ uses X^b and y' to compute out the session-key of the test-session, *while without knowing the static secret-key b*.

- $\mathscr{A}$ sends B as its DH-component in the second-round of the test-session but generates the public-key of $\hat{M}_1$ honestly. Suppose $\mathscr{A}$ further learns the static secret-key a of $\hat{A}$, $\mathscr{A}$ can then successfully finish the test-session but without knowing the DH-exponent b in this case.

We remark that the underlying reason beneath above attacks is: the leakage of the ephemeral values of *both* y and $X^{d_2b+e_2y}$ can render a CDH solver oracle, say (X, B, X^b), to the adversary.

6.3 Advantageous Features of YZ

We highlight some advantageous features of our YZ protocol family, in comparison with the (H)MQV protocol family. The detailed security analysis and comparisons are deferred pending Section 7 and Section 8.

Essentially optimal (on-line) efficiency. One key advantageous feature of YZ is its essentially optimal on-line efficiency. Specifically, the player $\hat{A}$ can pre-compute X and B^{dxt}, and the player $\hat{B}$ can pre-compute Y and A^{cyt}. Thus, the online complexity is just one exponentiation for each player, besides the subgroup test of peer's DH-component that however can be embedded without performing additional exponentiation. In comparison, note that the on-line complexity of (H)MQV is about 1.5 exponentiations. We remark that online efficiency is particularly important for smart-card based applications.

The total computational complexity of YZ is essentially the same as that of (H)MQV: each player performs about 1.5 exponentiations, besides the DH-component computation and sub-group test. Actually, the computation of $K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ (resp., $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$) amounts to about 1.3 exponentiations [99, 62, 47].[15] In particular, the sYZ protocol is slightly more efficient than

[15] We note that the computation of $K_{\hat{A}}$ (resp., $K_{\hat{B}}$) of (H)MQV can also be performed in 1.3 exponentiations by the technique of simultaneous exponentiations. But, the computation of $K_{\hat{A}}$ (resp., $K_{\hat{B}}$) of HMQV is still slightly more inefficient than sYZ.

HMQV. Note also that in case of *fixed* peer of public-key A or B (particularly for *smart-card-based* applications in the distributed client/server setting), the computation A^{cy} or B^{dx} amounts to about 0.2 exponentiation. Note also that, with pre-computed A^{cy} (resp., B^{dx}), the computation X^{db+ey} and X^q (resp., Y^{cx+xe} and Y^q) can be in parallel, which amounts to about 1.2 exponentiations (rather than 2 separate exponentiations).

Recall that the output length of the hash function, i.e., l, is set to be $|q|$ for YZ/sYZ, and $|q|/2$ for HMQV. When working on the same subgroup of order q, HMQV is more invulnerable to collision attacks against the underlying hash function h than YZ/sYZ in practice. To strengthen the security of (H)MQV, some standards specify larger subgroups (e.g., $|q| = 255$ in [105]) to use. We remark that (H)MQV working in a subgroup of order about 2^{255} is significantly inefficient than YZ/sYZ that are recommended to work in a subgroup of order about 2^{160}. Note that in memory-restricted environments (like smart-cards or other portable electronic tokens), subgroup size is an influential parameter in favor of a given algorithmic solution. Particularly, it is the partial reason underlying the popularity of Digital Signature Standard (DSS) and most elliptic curve based cryptographic solutions, where the security parameter $|q|$ is most often specified to be 160.

Parallel, modular and post-ID computability. B^{dxt} and $Y^{cat+ext}$ and the explicit subgroup test Y^q by $\hat{A}$, respectively A^{cyt} and $X^{dbt+eyt}$ and X^q by $\hat{B}$, can be computed in a parallel, modular and post-ID way, which much eases the YZ protocol deployments in various scenarios.

Specifically, the parallel computation of $Y^{cat+ext}$ and Y^q (resp., $X^{dbt+eyt}$ and X^q) amounts to 1.2 exponentiations. More importantly, observe that $Y^{cat+ext}$ (resp., $X^{dbt+eyt}$) can be computed before learning peer's identity and public-key information. This post-ID computability is remarkable for preserving players' privacy [35]. The advantage of the parallel, modular and post-ID computability is particularly demonstrated by the privacy-preserving (zero-knowledge+post-ID for the client) and DoS-resistant (for the server) protocol variant of YZ, i.e., the csYZ protocol proposed in Section 6.2.1, for the distributed client-server setting. We note that the (H)MQV protocol family does not enjoy all these advantageous properties, particularly the post-ID computability.

Deniability. The protocol transcript, particularly $K_{\hat{A}} = K_{\hat{B}} = A^{cy}B^{dx}g^{xy}$, of YZ can be computed out merely from the DH-exponents x and y (and some public system parameters including players' public-key certificates), *without involving any player's static secret-key a or b*. This renders reasonable and useful deniability privacy to both protocol players.

Furthermore, for the 3-round YZ and particularly the csYZ protocol variant, the initiator $\hat{A}$ is of perfect deniability (concurrent zero-knowledge) against concurrent malicious $\hat{B}$ of arbitrary auxiliary input. Specifically, the transcript of the concurrent run of YZ or csYZ (between the concurrent adversary $\hat{B}$ and the instances of the honest $\hat{A}$) can be efficiently simulated by a simulator, by an extended version of the assumption related to SSJPOK in the RO model, i.e., JKEA assumption (formulated in Section 5.2.1), in the concurrent setting. Details are omitted here. The responder $\hat{B}$ enjoys the *completed-session deniability*, similar to that in the DIKE of the aggressive model depicted in Figure 3, which is still very useful and reasonable particularly in the client-server setting with $\hat{B}$ being a server.

We note (H)MQV does not enjoy such deniability privacy advantages: the use of the session-key of (H)MQV can be definitely traced back to the group of the two players, particularly because the DH-secret g^{ab} involved in the computation of the shared secret $K_{\hat{A}} = K_{\hat{B}}$ in (H)MQV.

Easy deployment and minimal setup. We suggest YZ and sYZ enjoy easier deployment in practice, than (H)MQV, *particularly for smart-card-based applications.*

- The features of pre-computability, parallel, modular and post-ID computability, as well as flexible embedded subgroup tests, much ease protocol deployments of YZ/sYZ, allowing for various trade-offs among security, privacy and efficiency in various application scenarios.

- YZ and particularly sYZ support public-computation model [82] (to be clarified in Section

8.2), which is particularly important for smart-card-based applications.

- YZ and sYZ allow smaller security parameters, and are well compatible with popular applications involving DSS or based on ECC, with the security parameter $|q|$ typically set to be 160.
- YZ does not mandate POP/POK for key registration, while POK/POP is now commonly assumed for MQV.

Other notable features. The design rational of YZ is new and of conceptual simplicity (with SSJPOK as the core building tool), the protocol structure is of simplicity, and the computation is symmetrical.

7 Implications of YZ, and Analysis in the Canetti-Krawczyk Framework

In this section, we analyze the YZ protocol family in the Canetti-Krawczyk framework [33], which particularly justifies the soundness of the design rational of YZ as well as the underlying building tools. Further security properties of the YZ protocol family beyond the CK-framework are analyzed in Section 8.

One of main conceptual contributions of the analysis of HMQV in the CK-framework [80] is to cast the design of HMQV in terms of Hashed Dual challenge-Response (HDR) signatures and Hashed Challenge-Response (HCR) signatures, which are in turn based on Dual Challenge-Response (DCR) signatures and eXponential Challenge-Response (XCR) signatures and can be traced back to Schnorr's identification scheme [112]. We show that YZ and sYZ all can be casted in terms of HDR/HCR signatures. Moreover, the HDR/HCR signatures implied by the YZ/sYZ protocols, referred to as YZ-HDR/HCR and sYZ-HDR/HCR, are both *online efficient* and *strongly secure*. This provides extra security strength of the underlying building tools, say SSJOPK and sSSJPOK, used in YZ/sYZ. To this end, we also show the relation of YZ-HDR/HCR to a family of signature scheme, which may itself be of independent interest.

7.1 Challenge-Divided Fiat-Shamir Paradigm and Challenge-Divided Schnorr Signatures

A common paradigm, known as the Fiat-Shamir paradigm [53], of obtaining signatures is to collapse a 3-round public-coin honest-verifier zero-knowledge, known as Σ-protocol, into a non-interactive scheme with hash functions that are modeled to be random oracles [11].

Definition 7.1 (Σ-protocol [39]) *A three-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ-protocol for an $\mathcal{NP}$-relation $\mathcal{R}$ if the following hold:*

- *Completeness. If P, V follow the protocol, the verifier always accepts.*
- *Special soundness. From any common input U of length n and any pair of accepting conversations on input U, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(U, w) \in \mathcal{R}$ with overwhelming probability. Here a, e, z stand for the first, the second and the third message respectively and e is assumed to be a string of length l (that is polynomially related to n) selected uniformly at random in $\{0,1\}^l$.*
- *Perfect/statistical SHVZK (Special honest verifier zero-knowledge). There exists a probabilistic polynomial-time (PPT) simulator S, which on input U (where there exists an $\mathcal{NP}$-witness*

w such that $(U, w) \in \mathcal{R}$) and a random challenge string $\hat{e}$, outputs an accepting conversation of the form $(\hat{a}, \hat{e}, \hat{z})$, with the same probability distribution as that of the real conversation (a, e, z) between the honest $P(w)$, V on input V.

The first Σ-protocol (for an $\mathcal{NP}$-language) in the literature can be traced back to the HVZK protocol for Graph Isomorphism [59] (but the name of Σ-protocol is adopted much later in [39]), and a large number of Σ-protocols for various languages have been developed now. Σ-protocols have been proved to be a very powerful cryptographic tool, and are widely used in numerous important cryptographic applications. Below, we briefly recall the Σ-protocol examples for DLP and RSA.

Σ-Protocol for DLP [112]. The following is a Σ-protocol $\langle P, V \rangle$ proposed by Schnorr [112] for proving the knowledge of discrete logarithm, w, for a common input of the form (p, q, g, U) such that $U = g^w \ mod \ p$, where p, q are primes g is an element in Z_p^* of order q. Normally, the length of q, $|q|$, is denoted as the security parameter.

- P chooses r at random in Z_q and sends $a = g^r \ mod \ p$ to V.
- V chooses a challenge e at random in Z_{2^l} and sends it to P. Here, l is fixed such that $2^l < q$.
- P sends $z = r + ew \ mod \ q$ to V, who checks that $g^z = aU^e \ mod \ p$, that p, q are prime and that g, h are of order q, and accepts iff this is the case.

Σ-Protocol for RSA [64]. Let n be an RSA modulus and q be a prime. Assume we are given some element $y \in \mathbf{Z}_n^*$, and P knows an element w such that $w^q = y \mod n$. The following protocol is a Σ-protocol for proving the knowledge of q-th roots modulo n.

- P chooses r at random in Z_n^* and sends $a = r^q \mod n$ to V.
- V chooses a challenge e at random in Z_{2^l} and sends it to P. Here, l is fixed such that $2^l < q$.
- P sends $z = rw^e \mod n$ to V, who checks that $z^q = ay^e \mod n$, that q is a prime, that $gcd(a, n) = gcd(y, n) = 1$, and accepts iff this is the case.

The Fiat-Shamir paradigm and its provable security. Given *any* Σ-protocol (a, e, z) on common input U (which will be viewed as signing public-key), the Fiat-Shamir paradigm collapse the Σ-protocol into a signature scheme as follows: $(a, e = h(a, m), z)$, where m is the message to be signed and h is a hash function. Note in actual signature scheme with the Fiat-Shamir paradigm, the generated signature only consists of (e, z) as the value a can be computed from (e, z). The provable security of the general Fiat-Shamir paradigm is shown by Pointcheval and Stern [111] in the random oracle model (assuming h to be an idealized random function). The core of the security arguments of Pointcheval and Stern [111] is a forking lemma.

On-line/off-line signature. The notion of *on-line/off-line signature* is introduced in [52]. The idea is to perform signature generation into two phases: the off-line phase and the on-line phase. On-line/off-line signature schemes are useful, since in many applications the signer (e.g., a smart-card) has a very limited response time once the message is presented (but it can carry out costly computations between consecutive signing requests). The on-line phase is typically very fast, and hence can be executed even on a weak processor. On-line/off-line signature schemes are particularly remarkable in smart-card based applications [113]: the off-line phase can be implemented either during the card manufacturing process or as a background computation whenever the card is connected to power.

Note that for signature schemes obtained via the Fiat-Shamir scheme, the signer can pre-compute and store a list of values $(a = g^r, r)$. Then, to sign a message m, it simply computes $e = h(a, m)$ and z. With Schnorr's signature as an illustrative example, in this case, the signer only

needs to perform $z = r + h(m, a)w$ online, where $a = g^r$ and r are offline pre-computed and stored. Some general transformation from any signature scheme to secure off-line/off-line signature scheme are know (e.g., [52, 113]), but typically are not as efficient (for both computational complexity and space complexity of the signer) as the signature resultant directly via the Fiat-Shamir paradigm.

The Digital Signature Standard (DSS). The DSS scheme [54] is a variant of Schnorr's signature [112] via the Fiat-Shamir paradigm. The general structure of DSS is as follows:

- Public-key: $U = g^w \in G'$, where $w \in Z_q^*$. Typically, w is a 160-bit prime.
- Secret-key: w.
- Signature generation: Let $m \in \{0,1\}^*$ be the message to be signed.
  1. Compute $a = g^r \mod p$, where r is taken randomly from Z_q. Compute $d = f(a)$, where $f : G' \to Z_q^*$ is a conversion function.
     Typically, for DSS with $G' = Z_p^*$, f is just the "mod q" operation; for DSS with G' being some elliptic curve group over a finite field (i.e., a stands for an elliptic curve point (x, y)), $f(a)$ is to take the x-coordinate of a.
  2. Compute s from the equation $h(m) = sr - dw \mod q$, as follows:
     - Compute $\hat{r} = r^{-1}$.
     - Compute $s = (h(m) + dw)\hat{r}$, or $s = h(m)\hat{r} + dw\hat{r}$ with offline pre-computed $dw\hat{r}$, where h is a hash function.
  3. Output (d, s) as the signature.
- Signature verification: Given $(e = h(m), d, s)$ where $d, s \in Z_q^*$, the verifier verifies the signature as follows:
  - Compute $\hat{s} = s^{-1}$.
  - Verify $f(g^{e\hat{s}} U^{d\hat{s}}) = d$, where $e = h(m)$.

Recall that in the DSS scheme, the signature is generated as: $(d, s = er^{-1} + dwr^{-1})$, where $e = h(m)$, $d = f(a)$ and $a = g^r$, w is the secret-key. In general, the conversion $f : G' \to Z_q^*$ also can be viewed as RO. Observe that the value m (i.e., the message to be signed) and the value $a = g^r$ are not put into the input of a single RO in the DSS scheme, contrary to signature schemes via the Fiat-Shamir scheme where (m, a) is put into the single RO h.

Note that, with DSS, the signer can pre-compute a list of values a's (just as in signature schemes via the Fiat-Shamir paradigm), but contrary to signature schemes via the Fiat-Shamir paradigm, the signer of DSS does not need to store these pre-computed values. Specifically, for each pre-computed value $a = g^r$, the DSS signer can off-line compute $d = f(a)$, r^{-1} and dwr^{-1}, and only stores (d, r^{-1}, dwr^{-1}) (note that it is unreasonable to assume the message to be signed is always known beforehand). *Actually, for smart-card based applications, the values (d, r^{-1}, dwr^{-1})'s can be stored during the card manufacturing process.* Note that $d, r^{-1}, dwr^{-1} \in Z_q$ while $a \in G'$. Suppose $G' = Z_p^*$ (where p is typically of 1024 bits while q is of 160 bits) and the signer pre-computes k values of a, then in comparison with Schnorr's signature scheme the space complexity (of storing pre-computed values) is reduced from $(|p| + |q|)k$ to $3|q|k$. We remark that, for the implementations of Schnorr and DSS based on ECC, the above offline space efficiency advantage of DSS over Schnorr disappears.

Challenge-divided Σ-protocols and challenge-divided Fiat-Shamir paradigm. Next, we show a modified Fiat-Shamir paradigm, named *challenge-divided Fiat-Shamir paradigm*, that

is applicable to a variant of Σ-protocol with *divided* random challenges (that is referred to as *challenge-divided* Σ-protocol). The transformed signature schemes by applying the challenge-divided Fiat-Shamir paradigm on challenge-divided Σ-protocols enjoy, besides others, remarkable advantage of online-offline efficiency. Below, we first describe the challenge-divided Σ-protocols for DLP and RSA.

Challenge-divided Σ-Protocol for DLP. The common input is the same as that of Schnorr's protocol for DLP: (p, q, g, U) such that $U = g^w \ mod \ p$.

- P chooses r at random in Z_q and sends $a = g^r \ mod \ p$ to V.
- V chooses a *pair* of challenges d, e at random in $Z_{2^l} \times Z_{2^l}$ and sends (d, e) to P. Here, l is fixed such that $2^l < q$.
- P sends $z = er + dw \ mod \ q$ (resp., $z = dr + ew$) to V, who checks that $g^z = a^e U^d \ mod \ p$ (resp., $g^z = a^d U^e \ mod \ p$), that p, q are prime and that g, h are of order q, and accepts iff this is the case.

Challenge-divided Σ-Protocol for RSA. Let n be an RSA modulus and q be a prime. The common input is (n, q, y), and the private input is w such that $y = w^q \mod n$.

- P chooses r at random in Z_n^* and sends $a = r^q \mod n$ to V.
- V chooses a *pair* of challenges d, e at random in $Z_{2^l} \times Z_{2^l}$ and sends (d, e) to P. Here, l is fixed such that $2^l < q$.
- P sends $z = r^d w^e \mod n$ (resp., $z = r^e w^d \mod n$) to V, who checks that $z^q = a^d y^e \mod n$ (resp., $z^q = a^d y^e \mod n$), that q is a prime, that $gcd(a, n) = gcd(y, n) = 1$, and accepts iff this is the case.

The challenge-divided Fiat-Shamir paradigm for challenge-divided Σ-protocols. Let F be a one-way function (e.g., DLP) admitting challenge-divided Σ-protocols, i.e., the range of the OWF has a challenge-divided Σ-protocol for proving the knowledge of the corresponding preimage w.r.t. the $\mathcal{NP}$-relation $\{(U, w)|U = F(w)\}$. Denote by d, e the (challenge-divided) random challenges, and let $U = F(w)$ be signer's public-key and w the secret-key. To sign a message m, the signer computes a, $d = f(a)$, $e = h(m)$, and z, and then outputs (d, z) as the signature on m, where h and f are conversion functions from $\{0, 1\}^*$ to Z_q^*.

***Challenge-divided* Schnorr signature scheme.** With Schnorr's linear Σ-protocol for DLP as an illustrative instance, the transformed signature via the above challenge-divided Fiat-Shamir paradigm is called *challenge-divided* Schnoor's signature. (The signature based on the above challenge-divided Σ-protocol for RSA can be derived similarly.) In the following, we directly describe the online/offline version of challenge-divided Schnorr's signature. *We remark that the main purpose of developing challenge-divided Schnorr is to demonstrate the divided forking lemma, to be used later in the analysis of (s)YZ.*

- Public-key: $U = g^{-w} \in G'$, where $w \in Z_q^*$.
- Secret-key: w.
- Message to be signed: m.
- Offline pre-computation: the signer pre-computes and stores (r, d, dw) (resp., (d, rd)), where r is taken randomly by the signer from Z_q^*, $a = g^r$, $d = f(a)$. The signature verifier can pre-compute $e = h(m)$ and $\hat{e} = e^{-1}$, in case it knows m before receiving the signature. (For the purpose of signature of knowledge of w, we can set $d = h(a, U)$ or $e = h(m, U)$).

- Online signature generation: After receiving the message m to be signed, the signer computes $e = h(m)$, retrieves the pre-stored value (r, d, dw) (resp., (d, dr)), and computes $z = er + dw$ (resp., $z = dr + ew$). The signer outputs (d, z) as the signature on m.
- Signature verification: given a signature $(e = h(m), d, z)$ where $d, z \in Z_q^*$, check that $d, z \in Z_q^*$ and $f(g^{z\hat{e}}U^{d\hat{e}}) = d$ (resp., $f(g^{z\hat{d}}U^{e\hat{d}}) = d$), where $\hat{e} = e^{-1}$ (resp., $\hat{d} = d^{-1}$). Note that $\hat{e} = e^{-1}$ can be offline pre-computed by the verifier, in case it knows the message m before receiving the signature.

Challenge-divided Schnorr vs. DSS. We note all performance advantages of DSS are essentially preserved with the challenge-divided Schnorr scheme. We also note the techniques proposed in [101] for improving the performance of DSS in certain scenarios, e.g., signature batch verification and compression, etc, are also applicable to challenge-divided Schnorr. In addition, the challenge-divided Schnorr scheme has the following advantages over DSS:

- Same or better offline space complexity. Suppose k values a's are pre-computed, the offline space complexity of challenge-divided Schnorr with $z = er + dw$ is $3k|q|$ (which is the same as that of DSS); But, for challenge-divided Schnorr with $z = dr + ew$, the online space complexity is only $2k|q|$ (which further improves the offline space complexity of DSS).
- More efficient signature generation in total: To compute out the value s in the DSS-signature, the signer of DSS needs to perform 1 modular inverse (i.e., $\hat{r} = r^{-1}$) and 2 modular multiplications (without offline pre-computation of dwr^{-1}) in total. In comparison, to compute out the value z in the challenge-divided Schnorr signature, the signer only needs to perform 2 modular multiplications in total (without performing the modular inverse operation). We remark that modular inverse is a relatively expensive operation (which is typically performed by the Euclid algorithm), and is thus much preferable to dispense with (particularly for smart-card-based implementations).
- More efficient offline pre-computation: Besides the same other pre-computations, the signer of DSS needs to perform 1 modular inverse r^{-1} and 2 modular multiplications for computing dwr^{-1}, but the signer of challenge-divided Schnorr needs to offline perform only 1 modular multiplication dw or dr.
- More efficient online signature verification (for the case of $z = er + dw$): For verifying a DSS-signature (d, s), the verifier has to compute $\hat{s} = s^{-1}$ online, as the value s is known to the verifier only when the signature comes to it. As mentioned, the operation of modular inverse is relatively expensive and is much preferable to be dispensed with. In comparison, for verification of challenge-divided Schnorr signature with $z = er + dw$, the verifier only needs to compute the inverse $\hat{e} = e^{-1}$ where $e = h(m)$. In case the verifier learns the message to be signed prior to receiving the signature from the signer, which is quite common in practice, the values e and e^{-1} can be offline pre-computed by the verifier of challenge-divided Schnorr. For challenge-divided Schnorr with $z = dr + ew$, signature verification is of the same computational complexity as that of DSS.
- Provable security in the random oracle model: We show that, assuming both h and f are random oracles, the challenge-divided Schnorr scheme is existentially unforgeable against adaptive chosen message attacks [61] under the DLP assumption in the RO model. Note that the provable security of DSS is still unknown.

Theorem 7.1 *Assuming $h, f : \{0,1\}^* \rightarrow \{0,1\}^l / \{0\} \subseteq Z_q^*$ are random oracles where l is the security parameter (for presentation simplicity, we assume the range of ROs is $\{0,1\}^l$ rather than*

$\{0,1\}^l/\{0\}$*), the challenge-divided Schnorr scheme is existentially unforgeable against adaptive chosen message attacks under the DLP assumption.*

Proof. We mainly provide the proof for challenge-divided Schnorr with $z = er + dw$, the proof for the case of $z = dr + ew$ is similar.

Given a polynomial-time and successful forger $\mathcal{F}$, i.e., $\mathcal{F}$ successfully outputs (after polynomially many adaptively chosen queries to the signing oracle and random oracles), with non-negligible probability in polynomial-time, a valid signature on a new message that is different from those queried to the signing oracle, we build an efficient solver $\mathcal{C}$ for the DLP problem, namely, $\mathcal{C}$ gets as input a random element $U = g^w$ in G and outputs the corresponding discrete logarithm w also with non-negligible probability. For presentation simplicity, we assume the random oracles h, f are identical, namely we use the unique RO h to handle all RO queries $e = h(m)$ and $d = h(a)$. The algorithm $\mathcal{C}$ is presented in Figure 10.

For the description of $\mathcal{C}$ in Figure 10, suppose $\mathcal{F}$ makes Q RO queries and R signing oracle queries (where Q and R are some polynomials in the security parameter l), we have the following proposition:

Proposition 7.1 *With probability at most* $(RQ + R^2/2)/(q-1)$ *(that is negligible),* $\mathcal{C}$ *fails in one of Step S3 of signature simulations (note that, assuming* $\mathcal{F}$ *never fails at Step S3 in signature simulations, the signature simulations are perfect).* $\mathcal{C}$ *fails at Step F3 with probability at most* $3 \cdot 2^{-l}$

Proof (of Proposition of 7.1). It is easy to check that suppose $\mathcal{C}$ never fails at Step S3, the signature simulations by $\mathcal{C}$ are of identical distribution with that of real signatures by using the secret-key w.

Next, we limit the upper-bound of Step S3 failure. Note that for each a_i generated by $\mathcal{C}$ at Step S2, it is distributed uniformly in $G/1_G$. In the RO model, there are two cases for $\mathcal{C}$ fails at Step S3:

Case 1. For *some* i, $1 \le i \le R$, $\mathcal{F}$ ever successfully guessed the value a_i in one of its Q random oracle queries. Thus, the probability that $\mathcal{C}$ fails in Case 1 is at most $RQ/(q-1)$.

Case 2. For some i, $1 \le i \le R$, the value a_i has ever been generated in dealing with the j-th signing oracle query, $j < i$. The probability that $\mathcal{C}$ fails in Case 2 is at most $C_R^2/(q-1) \le (R^2/2)/(q-1)$.

Finally, it is easy to check that $\mathcal{C}$ fails at Step F3 with probability at most $3 \cdot 2^{-l}$. To see this, first note that the probability that $\mathcal{F}$ outputs $d = h(a)$ without making RO query $h(a)$ is 2^{-l}; That is, with probability $1 - 2^{-l}$, $\mathcal{F}$ knows a, and from (a, d, z) the value $log_a^{g^z U^d}$ is (which should be equal to $h(m) = e$) is then determined. Conditioned on this, the probability $e = h(m) = log_a^{g^z U^d}$ is 2^{-l}, as e is distributed uniformly over $\{0,1\}^l$. Thus, $\mathcal{F}$ does not query $h(m)$ with probability at most $2^{-l} + (1 - 2^{-l}) \cdot 2^{-l} < 2 \cdot 2^{-l}$. □

Thus, suppose the forger $\mathcal{F}$ succeeds (i.e., outputs a valid signature (d, z) for a new message m different from those queried) with non-negligible probability in its real attack against the signer of public-key U, $\mathcal{F}$ succeeds in the first run of $\mathcal{C}$ in Figure 10 also with non-negligible probability (up to a gap at most $(QR + R^2/2)/(q-1)$). Then, with non-negligible probability (with a gap at most $(QR + R^2/2)/(q-1) + 3 \cdot 2^{-l}$ to the success probability of $\mathcal{F}$ in its real attack), $\mathcal{C}$ does the repeated second run.

For presentation simplicity, we write the signature of challenge-divided Schnorr on a message m as $(m, e = h(m), a, d = h(a), z)$. Note that given a pair of different signatures on the same m (and a): $\{(m, e, a, d, z), (m, e', a, d, z')\}$ that corresponds to Case C1 in Figure 10, or, $\{(m, e, a, d, z), (m, e, a, d', z')\}$ that corresponds to Case C2 in Figure 10, the value w computed by $\mathcal{C}$ is correct (i.e., the discrete logarithm of U). Thus, to finish the theorem, what left is to show

Building the DLP solver $\mathcal{C}$ from the challenge-divided Schnorr forger $\mathcal{F}$

Setup: The input to $\mathcal{C}$ is a random element $U = g^{-w}$ in G, and its goal is to compute out w. To this end, $\mathcal{C}$ provides $\mathcal{F}$ with a random tape, and runs the forger $\mathcal{F}$ as the challenge-divided Schnorr signer of public-key U.
RO queries: $\mathcal{C}$ provides random answers to queries to the random oracle h, under the limitation that if the same h query is presented more than once, $\mathcal{C}$ answers it with the same response as in the first time.
Signature query simulation: Each time $\mathcal{F}$ queries the signing oracle for a challenge-divided Schnorr signature on message m_i, $1 \leq i$, chosen by $\mathcal{F}$ adaptively, $\mathcal{C}$ answers the query as follows (note that $\mathcal{C}$ does not know the secret-key w corresponding to the public-key $U = g^w$):

S1. Chooses $z_i \in_{\mathrm{R}} Z_q^*$ and $d_i \in_{\mathrm{R}} \{0,1\}^l \subseteq Z_q^*$ where l is the output length of the RO h. If $h(m)$ has been defined by previous query to h, then sets $e_i = h(m)$, otherwise chooses $e_i \in_{\mathrm{R}} \{0,1\}^l$ and defines $h(m) = e_i$.

S2. Computes $a_i = g^{z_i e_i^{-1}} U^{d_i e_i^{-1}}$.

S3. If $h(a_i)$ has been previously defined, $\mathcal{C}$ aborts its run and outputs "fail". Otherwise, sets $h(a_i) = d_i$. Recall that, for presentation simplicity, we have assumed $f = h$.

S4. $\mathcal{C}$ responds to $\mathcal{F}$'s signing query m_i with the simulated signature (d_i, z_i).

When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs (m, d, z) such that (d, z) is a valid signature on m. That is, d, z are in Z_q^*, $e = h(m)$ $a = g^{ze^{-1}} U^{de^{-1}}$, and $d = h(a)$.

F2. m was not queried by $\mathcal{F}$ to the signing oracle previously.

F3. The values $h(m)$ and $h(a)$ were queried from the RO h.

If these three conditions hold, $\mathcal{C}$ proceeds to the "repeat experiments" below; in all other cases $\mathcal{C}$ halts and outputs "fail".
The repeat experiments. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time, under the same public-key U and using the same coins for $\mathcal{F}$. There are two cases according to the order of the RO queries of $h(m)$ and $h(a)$:

C1. $h(m)$ posterior to $h(a)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(m)$, responds back a new independent value $e' \in_{\mathrm{R}} \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a valid signature (d, z') for the message m, i.e., $e' = h(m)$, $d = h(a)$ and $a = g^{z'e'^{-1}} U^{de'^{-1}}$, $\mathcal{C}$ computes $w = (z'e'^{-1} - ze^{-1})/(de'^{-1} - de^{-1}) \mod q$.

C2. $h(a)$ posterior to $h(m)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(a)$, responds back a new independent value $d' \in_{\mathrm{R}} \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a valid signature (d', z') for the message m, i.e., $e = h(m)$, $d' = h(a)$ and $a = g^{z'e^{-1}} U^{d'e^{-1}}$, $\mathcal{C}$ computes $w = (z' - z)/(d' - d) \mod q$.

Figure 10: Reduction from DLP to challenge-divided Schnorr forgeries

that conditioned $\mathcal{F}$ succeeds in outputting the valid (m, e, a, d, z) in the first run of $\mathcal{C}$, with non-negligible probability $\mathcal{F}$ will also succeeds in Case C1 or Case C2 of the repeated second run. We note that this can be shown by an *extended* version of the Pointcheval-Stern forking lemma [111] (that was originally developed to argue the security of digital signature schemes via the Fiat-Shamir paradigm). For completeness, we reproduce the forking lemma tailored for signature schemes via the challenge-divided Fiat-Shamir paradigm (on linear Σ-protocols), referred to as *divided* forking lemma, which might also be used elsewhere.

Suppose $\mathcal{F}$ produces, with probability ε', a valid signature (m, e, a, d, z), within the time bound T in its real attack against the signer of public-key U, then with probability at least $\varepsilon = (\varepsilon' - (QR + R^2/2)/(q-1) - 3 \cdot 2^{-l})/2$ $\mathcal{F}$ outputs a valid signature (m, e, a, d, z) in the first run of $\mathcal{C}$ described in Figure 10 such that $\mathcal{F}$ made both $h(m) = e$ and $h(a) = d$ queries to the RO with the order of $h(m)$ *being posterior to* $h(a)$ or the order of $h(a)$ *being posterior to* $h(m)$. Without loss of generality, we assume it is the former case, i.e., the RO query $h(m)$ is posterior to $h(a)$ (the analysis of the case of $h(a)$ being posterior to $h(m)$ is similar). We have the following lemma, from which the theorem is then established.

Lemma 7.1 (divided forking lemma) *Suppose $\mathcal{F}$ produces, with probability ε, a valid signature (m, e, a, d, z) within the time bound T in the first run of $\mathcal{C}$ such that $\mathcal{F}$ made both $h(m) = e$ and $h(a) = d$ RO queries with the order $h(m)$* being posterior to $h(a)$*, then within time $T' \leq (2/\varepsilon + (\varepsilon/4Q - 2^{-l})^{-1}) \cdot T$ and with probability at least $\frac{1}{9}$, a replay of $\mathcal{F}$ outputs a valid signature (m, e', a, d, z') for $e' \neq e$.*

Proof (of Lemma 7.1). The proof of Lemma 7.1 is essentially identical to that of Lemma 2 in [111], which we re-produce here for completeness.

Denote by ω the random tape of $\mathcal{F}$, and assume $\mathcal{F}$ makes at most Q RO queries $\mathcal{Q}_1, \cdots, \mathcal{Q}_Q$ (for presentation simplicity, we assume all RO queries are distinct), and denote by $\rho = (\rho_1, \cdots, \rho_Q)$ the Q RO answers. It is clear a random choice of the random function h (i.e., the RO) corresponds to a random choice of ρ.

Define $\mathcal{S}$ to be the set of (ω, h) such that $\mathcal{F}^h(\omega)$ outputs a valid signature (m, e, a, d, z) in the first run of $\mathcal{C}$, such that $\mathcal{F}$ made both $h(m)$ and $h(a)$ RO queries with the order of $h(m)$ being posterior to $h(a)$. That is, $\Pr[\mathcal{S}] = \varepsilon$. Define $Ind(\omega, h)$ to be the index of the RO query $h(m)$, i.e., $m = \mathcal{Q}_{Ind(\omega,h)}$. Define $\mathcal{S}_i$ be the subset of $\mathcal{S}$ such that $Ind(\omega, h) = i$ for $1 \leq i \leq Q$. That is, the set $\{\mathcal{S}_1, \cdots, \mathcal{S}_Q\}$ is a partition of $\mathcal{S}$. Define $I = \{i | Pr[\mathcal{S}_i|\mathcal{S}] \geq 1/2Q\}$, i.e., $\Pr[\mathcal{S}_i | i \in I] \geq \varepsilon/2Q$. For each $i \in I$, define by h_i the restriction of h to queries of index strictly less than i, they by applying the Splitting Lemma (Lemma 1, page 12 in [111]), there exists a subset Ω_i (of $\mathcal{S}$) such that: (1) for any $(\omega, h) \in \Omega_i$, $\Pr_{h'}[(\omega, h') \in \mathcal{S}_i | h'_i = h_i] \geq \varepsilon/4Q$; (2) $\Pr[\Omega_i | \mathcal{S}_i] \geq \frac{1}{2}$. As all the subsets $\mathcal{S}_i$ are disjoint, it is calculated that $\Pr_{\omega,h}[\exists i (\omega, h) \in \Omega_i \cap \mathcal{S}_i | \mathcal{S}] \geq \frac{1}{4}$ (for more details, the reader is referred to [111]).

By the Lemma 3 (page 14) in [111], we get $\Pr[Ind(\omega, h) \in I | \mathcal{S}] \geq \frac{1}{2}$. Now, run $\mathcal{F}$ $2/\varepsilon$ times with random ω and h, with probability $1 - (1 - \varepsilon)^{2/\varepsilon} \geq \frac{4}{5}$ we get one successful pair $(\omega, h) \in \mathcal{S}$. Denote by β the index $Ind(\omega, h)$ corresponding to the successful pair. We know with probability at least $\frac{1}{4}$, $\beta \in I$ and $(\omega, h) \in \mathcal{S}_\beta \cap \Omega_\beta$. Consequently, with probability at least $\frac{1}{5}$, the $2/\varepsilon$ runs have provided a successful pair $(\omega, h) \in \mathcal{S}_\beta \cap \Omega_\beta$ where $\beta = Ind(\omega, h)$. As $\Pr_{h'}[(\omega, h') \in \mathcal{S}_\beta | h'_\beta = h_\beta] \geq \varepsilon/4Q$ in this case, we get $\Pr_{h'}[(\omega, h') \in \mathcal{S}_\beta \wedge \rho_\beta \neq \rho'_\beta | h'_\beta = h_\beta] \geq \varepsilon/4Q - 2^{-l}$, where $\rho_\beta = h(\mathcal{Q}_\beta)$ and $\rho'_\beta = h'(\mathcal{Q}_\beta)$. Now, we replay $\mathcal{F}$ with fixed ω but randomly chose h' such that $h'_\beta = h_\beta$, for $(\varepsilon/4Q - 2^{-l})^{-1}$ times, with probability at least $\frac{3}{5}$, we will get another success. That is, after less than $2/\varepsilon + (\varepsilon/4Q - 2^{-l})^{-1}$ repetitions of $\mathcal{F}$'s attack, with probability at lease $\frac{1}{5} \times \frac{3}{5} \geq \frac{1}{9}$, we have obtained two valid signatures (m, e, a, d, z) and (m, e', a, d, z') for $e \neq e'$. □ □

Note: As clarified in [80], the output length, i.e., l, of the random oracle h in the above security analysis can be set to be $|q|/2$ for reasonable tradeoff between security and efficiency.

7.2 Casting (s)YZ in Terms of *Online-Efficient* and *Strongly-Secure* HDR/HCR Signatures

Informally speaking, a HDR signature scheme is an *interactive* signature scheme between two parties in the public-key model. The two parties generate the *same* signature, which is actually a *hashed* value of the DH-secret shared between the two parties, with the *dual* roles of signer and challenger: each party generates the signature with private values of its static secret-key and the secret DH-exponent with respect to its peer's DH-component and public-key as the challenges. With a HDR signature, we are only interested to ensure verifiability of the signature by the two intended parties, and thus we make no assumptions or requirements regarding the transferability or verifiability of the signature by a third party. Roughly speaking, a HDR signature scheme is secure if the signature cannot be generated by any other parties other than the two intended (honest) parties.

Definition 7.2 ((s)YZ-HDR signature schemes) *Let $\hat{A}$, $\hat{B}$ be two parties with public-keys $A = g^a$, $B = g^b$, respectively. Let $m_{\hat{A}}$, $m_{\hat{B}}$ be two messages. The* YZ-HDR, sYZ-HDR *signatures of $\hat{B}$ on messages $(m_{\hat{A}}, m_{\hat{B}}, \hat{A}, A, \hat{B}, B, X, Y)$ are defined as a vector of values (the signatures of $\hat{A}$ are defined straightforwardly):*

YZ-HDR. $\{\hat{A}, A, m_{\hat{A}}, m_{\hat{B}}, X, Y, HSIG^{YZ}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y) = H_K(A^{yc}X^{bd+ye})\}$, *where $X = g^x$, $Y = g^y$ are chosen by $\hat{A}$, $\hat{B}$ respectively as the random* challenge *and* response*, $x, y \in_{\mathrm{R}} Z_q^*$, $c = h(m_{\hat{A}}, \hat{A}, A, Y)$, $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$.*

Another form of YZ-HDR is to set $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$ and $e = h(m_{\hat{A}}, m_{\hat{B}}, X, Y)$. Both of these two versions are secure.

sYZ-HDR. $\{\hat{A}, A, m_{\hat{A}}, m_{\hat{B}}, X, Y, HSIG^{sYZ}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y) = H_K(A^{yc}X^{bd+ye})\}$, *where $c = d = 1$, $e = h(m_{\hat{A}}, m_{\hat{B}}, \hat{A}, A, \hat{B}, B, X, Y)$.*

YZ/sYZ in a nutshell. Actually, the above YZ-HDR/sYZ-HDR can be viewed as a general structure of the YZ/sYZ protocols. Specifically, YZ and sYZ are instantiated with YZ-HDR and sYZ-HDR respectively, with the special $m_{\hat{A}}$ and $m_{\hat{B}}$ that are set to be the empty string. In general, $m_{\hat{A}}$ (resp., $m_{\hat{B}}$) can include some values sent to $\hat{A}$ (resp., $\hat{B}$) from $\hat{B}$ (resp., $\hat{A}$), which does not affect the pre-computability of YZ/sYZ. In particular, in practice with pre-computed and reused DH-components, $m_{\hat{A}}$ (resp., $m_{\hat{B}}$) can include a random nonce generated and sent by $\hat{B}$ (resp., $\hat{A}$).

In the following, we show the security of YZ-HDR, sYZ-HDR with off-line pre-computed DH-exponents, DH-components, and the values A^{yc} or B^{xd} (that may be potentially exposed to the forger *even prior to the session involving these pre-computed values*), on which the security of YZ and sYZ in the CK-framework will be based. In particular, we show that our YZ-HDR and sYZ-HDR satisfy a significantly stronger security definition (than the definition given in [80]), which is defined below.

Definition 7.3 (*Strong* security of HDR signature scheme (with off-line pre-computation)) *We say a HDR signature scheme (of $\hat{B}$) is* strongly *secure, if no polynomial-time machine $\mathcal{F}$ can win the game in Figure 11 with non-negligible probability* with respect to any uncorrupted party $\hat{A}$ of public-key $A = g^a$ such that the secret-key a was not chosen by the attacker $\mathcal{F}$.

Notes: For presentation simplicity, in the above HDR signature description we assume the CA in the underlying PKI will check registered public-keys are in $G/1_G$, and each player checks its peer's DH-component is also in $G/1_G$. These subgroup tests may not be necessary for the security

Forger $\mathcal{F}$ in Definition 7.3

1. Forger $\mathcal{F}$ is given values B, X_0, where $B, X_0 \in_{\mathrm{R}} G$.
2. $\mathcal{F}$ is given access to a signing oracle $\hat{B}$ (of public-key $B = g^b$ and secret-key b).
3. Each signature query from $\mathcal{F}$ to $\hat{B}$ consists of the following interactions:
   (a) $\mathcal{F}$ presents $\hat{B}$ with messages $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}})$. Here, $\hat{Z}$ can be any (even corrupted) party chosen by $\mathcal{F}$, and $Z = g^z \in G/1_G$ is the public-key of $\hat{Z}$. Note that $\mathcal{F}$ may not necessarily know the corresponding secret-key z of $\hat{Z}$.
   (b) $\hat{B}$ generates $y \in_{\mathrm{R}} Z_q^*$ and $Y = g^y$, and computes Z^{cy}, where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$ for YZ-HDR or $c = 1$ for sYZ-HDR. Then, $\hat{B}$ responds with $(y, Y = g^y, Z^{cy})$ to $\mathcal{F}$ (this captures the powerful exposure capability to the forger), and stores the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, Z^{cy})$ as an "incomplete session". Here, (y, Y, Z^{cy}) can be offline pre-computed by $\hat{B}$, and leaked to $\mathcal{F}$ prior to the session involving (y, Y, Z^{cy}).
   (c) $\mathcal{F}$ presents $\hat{B}$ with $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$, and a *challenge* X.
   (d) $\hat{B}$ checks that $X \in G/1_G$ (if not, it aborts) and that $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$ is in one of its incomplete sessions (if not, it ignores). $\hat{B}$ then computes $r = H_K(Z^{cy} X^{db+ey})$, where $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$ for YZ-HDR (resp., $d = 1$ and $e = (m_{\hat{Z}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$ for sYZ-HDR). $\hat{B}$ responds $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, Y, r)$ to $\mathcal{F}$, and marks the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, Z^{cy})$ as a "*complete session*", and stores with it the signature values $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, r)$.
4. $\mathcal{F}$ is allowed a polynomial number of queries to $\hat{B}$, where the queries are chosen adaptively by $\mathcal{F}$. We remark that the interaction steps from different signature queries (sessions) can be *arbitrarily* interleaved.
5. $\mathcal{F}$ halts with output "fail" or with a *guess* in the form of a triple $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$. $\mathcal{F}$'s guess is called a successful *forgery* if the following two conditions hold:
   (a) $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$ is a valid HDR-signature of $\hat{B}$ on the messages $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, where $\hat{A}$ is an uncorrupted player of public-key $A = g^a$, m_1 corresponds to $m_{\hat{A}}$ (that is an arbitrary message sent by the adversary $\mathcal{F}$ impersonating the signer $\hat{B}$ to the honest player $\hat{A}$), and m_0 corresponds to $m_{\hat{B}}$ (that is chosen by the honest player $\hat{A}$). Note that the value X_0 is the one received by $\mathcal{F}$ as input.
   (b) *The vector* $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ *did not appear in any one of the responses of* $\hat{B}$ *to* $\mathcal{F}$*'s queries.*

   We say that $\mathcal{F}$ wins the game if it outputs a successful forgery (w.r.t. any value $A = g^a$ not chosen by $\mathcal{F}$).

Figure 11: Forgery Game for HDR Signatures (with offline pre-computation)

of HDR in general, assuming no ephemeral private state is exposed, and thus can be relaxed in some scenarios (see [80, 98] for more details).

On the *strong* security of HDR. The *strong* security of our definition for HDR lies in that:

- We assume (y, Y, A^{cy}) are off-line pre-computed, and the forger can get them prior to the session run involving them.

  This particularly renders stronger capability to the attacker to perform colliding (birthday) attacks against the hash function h (that is of length $|q|/2$ for HMQV). To deal with this subtlety, the actual HMQV implementation needs some changes in practice (to be clarified later).

- In the forging game defined in Figure 11, the successful forgery requires that the *whole* vector $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the responses of $\hat{B}$ to $\mathcal{F}$'s queries. The definition for the security of HCR in [80] only requires that the pair (Y_0, m_0) did not appear in responses from the signer. As we shall see, the HMQV-HDR scheme may not be strongly secure in general.

(s)YZ-HDR vs. HMQV-HDR. In [80], the HMQV-HDR (of $\hat{B}$) is defined to be $\{X, Y, DSIG^{HMQV}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y) = H_K((XA^d)^{y+be})\}$, where $d = h(m_{\hat{A}}, X)$, $e = h(m_{\hat{B}}, Y)$. For building HMQV with HMQV-HDR, $m_{\hat{B}}$ (resp., $m_{\hat{A}}$) is set to be its *peer's* identity $\hat{A}$ (resp., $\hat{B}$). The underlying HMQV-XCR-signature is defined to be X^{y+be}, where $e = h(m_{\hat{B}}, Y)$. The following are some brief comparisons between YZ-HDR and HMQV-HDR:

- One notable advantageous feature of YZ-HDR and sYZ-HDR is the online efficiency. Specifically, the online efficiency of YZ-HDR and sYZ-HDR, for each player, can be only one exponentiation. In comparison, each player of HMQV-HDR performs at least 1.5 online exponentiations.

- As we shall see, the YZ-HDR and sYZ-HDR are *strongly* secure in accordance with Definition 7.3. We note that the HMQV-XCR underlying HMQV-HDR is not *strongly* secure. For example, to forge a HMQV-XCR signature $(X, Y, \sigma = X^{b+ey})$ on message m, where $e = h(m, Y)$, the forger can first query the signer with $(m, X' = X^2)$, gets back $(X', Y, \sigma' = X'^{b+ey})$, and then outputs $(X, Y, \sigma = \sigma'^{\frac{1}{2}})$ as the XCR signature on m. Note that the triple (X, Y, σ) did not appear in any one of the responses from the HMQV-XCR signer $\hat{B}$. We note that one way to remedy this security vulnerability of HMQV-XCR is to commit X also to e by defining $e = h(m, X, Y)$.

- The security of YZ-HDR/sYZ-HDR against uncorrupted players *other than the signer itself*, with offline pre-computed (y, Y, A^{cy}) that can be exposable to the adversary even prior to the session involving (y, Y, A^{cy}), is based on the gap Diffile-Hellman (GDH) assumption.

  The security of HMQV-HDR against uncorrupted players *other than the signer itself*, with offline pre-computed DH-component Y, is based on both the GDH assumption and the non-standard KEA assumption [41], even if the pre-computed DH-exponent y is not exposable and only the pre-computed DH-component Y is exposable. Furthermore, for robust security of HMQV-HDR with pre-computed DH-components, when the number of messages in the system is large, HMQV-HDR needs to make the following modifications [80]: (1) Increase the output length, i.e., l, of the hash function h, e.g., from $|q|/2$ to $|q|$, which may bring negative impact on the performance of HMQV. (2) Add random nonces into the input of d and e, or, put the message to be signed also into H_K, which increases the system complexity.

- The generation of the sYZ-HDR signature uses minimal (i.e., only one) random oracle (in computing the value of e).
- The HMQV-HDR signature is actually an XCR signature w.r.t. the challenge XA^d. In comparison, YZ-HDR and sYZ-HDR in general cannot be viewed as a structure of XCR w.r.t. some challenge $f(X, A)$ for some function f.
- As we shall see, the special protocol structure of YZ-HDR and sYZ-HDR also much simplifies, in certain scenarios, the security analysis of YZ and sYZ in the CK-framework, and tightens the security reductions (to the underlying assumption).

Next, we show the strong security of YZ-HDR, sYZ-HDR under the Gap Diffie-Hellman (GDH) assumption in the random oracle model.

Theorem 7.2 *Under the GDH assumption, YZ-HDR and sYZ-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), are* strongly *secure in the random oracle model, with respect to any uncorrupted player other than the signer $\hat{B}$ itself* even if the forger is given the private keys of all uncorrupted players in the system other than b of $\hat{B}$

Proof (of Theorem 7.2). Given an efficient and successful forger $\mathcal{F}$ against YZ-HDR or sYZ-HDR, i.e., $\mathcal{F}$ wins the forgery game in Figure 11 with respect to some uncorrupted player $\hat{A} \neq \hat{B}$ with non-negligible probability, we build an efficient solver $\mathcal{C}$ for GDH problem also with non-negligible probability. The algorithm $\mathcal{C}$ for YZ-HDR is presented in Figure 12, and the algorithm $\mathcal{C}$ for sYZ-HDR is presented in Figure 13.

For the description of $\mathcal{C}$ in Figure 12, suppose $\mathcal{F}$ makes Q_h RO queries to h, Q_H queries to H_K, Q_s signing oracle queries, where Q_q, Q_H, Q_s are polynomial in the security parameter l (i.e., the output length of h). We have the following observations:

- The signature simulation at steps S1-S3 is perfect.
- Now, suppose $\mathcal{F}$ outputs a successful forgery $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, which particularly implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0 c_0} X_0^{bd_0 + y_0 e_0}$, $X_0 = U$, $Y_0 = g^{y_0}$, $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$. We investigate the probability that $\mathcal{C}$ aborts at step F3. We have the following observations:
  - With probability at most $\frac{1}{2^l - 1} + 2^{-k}$, $\mathcal{F}$ can succeed with undefined any one of $\{c_0, d_0, e_0\}$. Here, $\frac{1}{2^l - 1}$ is the probability that $\mathcal{F}$ guesses σ_0 with undefined c_0 or d_0 or e_0, and 2^{-k} is the probability that $\mathcal{F}$ simply guesses the value r_0.
  - With defined c_0 and d_0 and e_0, there are two cases for $\mathcal{F}$ to succeed without querying $H_K(\sigma_0)$:

    Case-1. $\mathcal{F}$ simply guesses the value r_0. This probability is 2^{-k}.

    Case-2. r_0 is the value r set by $\mathcal{C}$ at one of S3.1 steps, where r is supposed to be $H_K(\sigma)$ w.r.t. a stored vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$. Recall that for the value r set at step S3.1, $\mathcal{C}$ does not know σ (as it does not know b), and thus in this case both $\mathcal{C}$ and $\mathcal{F}$ may not make the RO-query $H_K(\sigma_0) = H_K(\sigma)$. In this case, with probability $1 - 2^{-k}$, $\sigma_0 = \sigma$, i.e., $A^{c_0 y_0} X_0^{d_0 b + e_0 y_0} = Z^{cy} X^{db + ey}$, where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$, $d = h(m_{\hat{B}}, \hat{B}, B, X)$, $e = h(X, Y)$, $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, Y_0)$, and $(m_0, m_1, \hat{A}, A, \hat{B}, B, X_0, Y_0) \neq (m_{\hat{A}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$.
    By the tag-binding self-seal property of YZ, for any value $\sigma \in G/1_G$ and any $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, the probability $\Pr[A^{c_0 y_0} X_0^{d_0 b + e_0 y_0} = \sigma] \leq \frac{1}{2^l - 1}$, where

X_0 are the given random element in $G/1_G$, $\hat{A}$ and $\hat{B}$ are uncorrupted players. This is true, even if the public-key A (resp., B) is removed from c_0 (resp., d_0), as the public-keys A and B are generated by the uncorrupted players $\hat{A}$ and $\hat{B}$ independently at random. Then, we can get that $\mathcal{F}$ succeeds in Case-2 with probability at most $O(\frac{Q_h^2}{2^l-1} + \frac{Q_s}{2^k})$.

Note: To rule out the possibility of Case-2, the analysis of HMQV-HCR requires the KEA assumption [41]. Furthermore, to resist to birthday attacks in Case-2, some modifications of HMQV are recommended in [80]: (1) increase the output length, i.e., l, of h, e.g., from $|q|/2$ to $|q|$. But, this remedy cause serious negative impact on the performance of HMQV; (2) Add random and *fresh* nonces (which cannot be offline pre-computed) to the input of h, or put the messages to be signed $m_{\hat{A}}, m_{\hat{B}}$ into the input of H_K.

– With probability at most $\frac{1}{2^l-1}$, the query $H_K(\sigma_0)$ is prior to any one of the queries $\{c_0, d_0, e_0\}$.

- It is easy to check that, in case the forger $\mathcal{F}$ successfully outputs another different forge satisfying the conditions F1-F3 in the repeat experiment C1 or C2, the output of $\mathcal{C}$ is the correct value of $CDH(X_0, B)$.

The similar observations can be easily checked for the algorithm $\mathcal{C}$ for sYZ-HDR described in Figure 13. Putting all together, we have that: suppose for some uncorrupted player $\hat{A} \neq \hat{B}$, the forger $\mathcal{F}$ provides, with non-negligible probability, a successful forgery w.r.t. $\hat{A}$ in its real interactions with the signer of YZ-HDR/sYZ-HDR, then with also non-negligible probability (up to a negligible gap specified by the above observations) $\mathcal{F}$ succeeds under the run of $\mathcal{C}$. Then, by applying the forking lemma, specifically, the divided forking lemma (Lemma 7.1) for YZ-HDR and the normal forking lemma of [111] for sYZ-HDR, the theorem is established. □

On the role of putting players' public-keys into the inputs of c, d for YZ-HDR and e for sYZ-HDR. We remark that the players' public-keys in the inputs of c, d, e for YZ-HDR/sYZ-HDR actually play *no* role in the above security analysis. That is, the above security analysis is actually with respect to a (*public-key free*) variant of YZ-HDR/sYZ-HDR, with public-keys are removed from the inputs of c, d, e. Recall that, players' public-keys are only used for arguing the TBSS property of YZ-HDR/sYZ-HDR. Specifically, for any value $\sigma \in G/1_G$ and any $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0)$, the probability $\Pr[\sigma_0 = A^{c_0 y_0} X_0^{d_0 b + e_0 y_0} = \sigma] \leq \frac{1}{2^l-1}$, where $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, Y_0)$ and the probability is taken over only the choice of the random function h. But, as we assume $\hat{A}$ and $\hat{B}$ are both uncorrupted players, their public-keys are generated independently at random. Also, the value X_0 is the given random DH-component (not generated by the attacker). To affect the distribution of σ_0, the only freedom of the attacker is to maliciously choose (Y_0, m_0, m_1), which however does not change the distribution of σ_0. In particular, for any value $\sigma \in G/1_G$ and for any (Y_0, m_0, m_1) chosen maliciously by the attacker w.r.t. the fixed $(\hat{A}, A, \hat{B}, B, X_0)$, it still holds that $\Pr[\sigma_0 = \sigma] \leq \frac{1}{2^l-1}$.

Security of YZ-HDR/sYZ-HDR against the signer itself. The above security analysis considers the security of YZ-HDR/sYZ-HDR against any other uncorrupted players other than the signer itself, i.e., the (in)feasibility of outputting a successful forgery $(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0, Y_0, r_0)$ where $\hat{A}$ is an uncorrupted player and $\hat{A} \neq \hat{B}$. But, the forger $\mathcal{F}$ may also be against the signer $\hat{B}$ itself. That is, $\mathcal{F}$ may output a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, Y_0, r_0)$ (i.e., $\hat{A} = \hat{B}$). Here, we further investigate the feasibility of successful forgeries of this form. We distinguish two cases: (1) $Y_0 = X_0$, i.e., the successful forgery is of the form $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$. For this case, we show YZ-HDR and sYZ-HDR are still secure under the traditional CDH assumption (not the stronger GDH assumption) in the RO model; (2) $Y_0 \neq X_0$. For this case, we show

Building the CDH solver $\mathcal{C}$ from the YZ-HDR forger $\mathcal{F}$

Setup: The inputs to $\mathcal{C}$ are random elements $U = g^u, V = g^v$ in G, and its goal is to compute $CDH(U,V) = g^{uv}$ with oracle access to a DDH oracle $\mathcal{O}$. To this end, $\mathcal{C}$ sets $B = V$ and $X_0 = U$, and sets the public-keys and secret-keys for all other uncorrupted players in the system. $\mathcal{C}$ runs the forger $\mathcal{F}$ on input (B, X_0) against the signer $\hat{B}$ of public-key B. $\mathcal{C}$ provides $\mathcal{F}$ with a random tape, and provides the secret-keys of all uncorrupted players other than the signer $\hat{B}$ itself (the attacker $\mathcal{F}$ may register arbitrary public-keys for corrupted players, based on the public-keys and secret-keys of uncorrupted players other than signer's secret-key b).

Signature query simulation: Each time $\mathcal{F}$ queries $\hat{B}$ for a signature on values $(\hat{Z}, Z, m_{\hat{B}}, m_{\hat{A}})$, $\mathcal{C}$ answers the query for $\hat{B}$ as follows (note that $\mathcal{C}$ does not know b):

S1. $\mathcal{C}$ generates $y \in_R Z_q^*$, $Y = g^y$ and Z^{cy}, where $c = h(m_{\hat{Z}}, \hat{Z}, Z, Y)$ (that may be pre-defined, otherwise $\mathcal{C}$ defines c with the RO h). Actually, (y, Y, Z^{cy}) can be pre-computed by $\mathcal{C}$ and leaked to $\mathcal{F}$ prior to the session. Then, $\mathcal{C}$ responds $(y, Y = g^y, Z^{cy})$ to $\mathcal{F}$, and stores the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, A^{cy})$ as an "incomplete session".

S2. $\mathcal{F}$ presents $\mathcal{C}$ with $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$, and a *challenge* X.

S3. $\hat{B}$ checks that $X \in G/1_G$ (if not, it aborts) and that $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$ is in one of its incomplete sessions (if not, it ignores the query). Then, $\mathcal{C}$ checks for every value $\sigma \in G/1_G$ *previously used by* $\mathcal{F}$ as input to H_K whether $\sigma = Z^{cy} X^{bd+ye}$, where $d = h(m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$ (in case d, e undefined, $\mathcal{C}$ defines them with h): it does so using the DDH-oracle $\mathcal{O}$, specifically, by checking whether $CDH(X, B) = (\sigma / Z^{cy} X^{ye})^{d^{-1}}$. If the answer is positive, then $\mathcal{C}$ sets r to the already determined value of $H_K(\sigma)$.

S3.1. In any other cases, r is set to be a random value in $\{0,1\}^k$, where k is the output length of H_K. Note that, in this case, $\mathcal{C}$ does not know $\sigma = Z^{cy} X^{db+ey}$, as it does not know b, which also implies that $\mathcal{C}$ does not make (actually realize) the RO-query $H_K(\sigma)$ *even if the value* σ *has been well-defined and known to* $\mathcal{F}$.

Finally, $\mathcal{C}$ marks the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy})$ as a "*complete session*", stores $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$ and responds $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, Y, r)$ to $\mathcal{F}$.

RO queries: $\mathcal{C}$ provides random answers to queries to the random oracles h and H_K (made by $\mathcal{F}$), under the limitation that if the same RO-query is presented more than once, $\mathcal{C}$ answers it with the same response as in the first time. But, for each *new* query σ to H_K, $\mathcal{C}$ checks whether $\sigma = Z^{cy} X^{db+ey}$ for any one of the stored vectors $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^{cy}, r)$ (as before, this check is done using the DDH-oracle). If equality holds then the corresponding r is returned as the predefined $H_K(\sigma)$, otherwise a random r is returned.

Upon $\mathcal{F}$'s termination. When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs a valid HDR-signature $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, where $\hat{A} \neq \hat{B}$ is an uncorrupted player. In particular, it implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0 c_0} X_0^{bd_0 + y_0 e_0}$, $Y_0 = g^{y_0}$ (chosen by $\mathcal{F}$), $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$.

F2. $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the above responses of the simulated YZ-HDR signatures.

F3. The values $c_0 = h(m_1, \hat{A}, A, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$ and $e_0 = h(X_0, Y_0)$ were queried from the RO h, and the value $H_K(\sigma_0)$ was queried from H_K *being posterior to the queries* c_0, d_0, e_0. Otherwise, $\mathcal{C}$ aborts.

If these three conditions hold, $\mathcal{C}$ proceeds to the "repeat experiments" below, else it aborts.

The repeat experiments. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time, under the same input (B, X_0) and using the same coins for $\mathcal{F}$. There are two cases according to the order of the queries of $h(m_0, \hat{B}, B, X_0)$ and $h(X_0, Y_0)$

C1. $h(m_0, \hat{B}, B, X_0)$ posterior to $h(X_0, Y_0)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(m_0, \hat{B}, B, X_0)$, responds back a new independent value $d_0' \in_R \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0 c_0'} X_0^{bd_0' + y_0 e_0}$, $\mathcal{C}$ computes $CDH(U,V) = CDH(X_0, B) = [(\sigma_0 / Y_0^{ac_0}) / (\sigma_0' / Y_0^{a' c_0'})]^{(d_0 - d_0')^{-1}}$, where a and a' are the private keys of the uncorrupted $\hat{A}$ and $\hat{A}'$ (different from $\hat{B}$, which are assumed to be known to $\mathcal{C}$). Note that $(\hat{A}', A', m_1')$ need not necessarily to equal $(\hat{A}, A, m_1)$.

C2. $h(X_0, Y_0)$ posterior to $h(m_0, \hat{B}, B, X_0)$: $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(X_0, Y_0)$, responds back a new independent value $e_0' \in_R \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0 c_0'} X_0^{bd_0 + y_0 e_0'}$, $\mathcal{C}$ computes $X_0^{y_0} = ((\sigma_0 / Y_0^{ac_0}) / (\sigma_0' / Y_0^{a' c_0'}))^{(e_0 - e_0')}$, and then $CDH(U,V) = CDH(X_0, B) = (\sigma_0 / ((X_0^{y_0})^{e_0} \cdot Y_0^{ac_0}))^{d_0^{-1}}$.

Figure 12: Reduction from CDH to YZ-HDR forgeries

Building the CDH solver $\mathcal{C}$ from the sYZ-HDR forger $\mathcal{F}$

Setup: $\mathcal{C}$ does the same as it does for the forger $\mathcal{F}$ against YZ-HDR.
Signature query simulation: Each time $\mathcal{F}$ queries $\hat{B}$ for a signature on values $(\hat{Z}, Z, m_{\hat{B}}, m_{\hat{A}})$, $\mathcal{C}$ answers the query for $\hat{B}$ as follows (note that $\mathcal{C}$ does not know b):

S1. $\mathcal{C}$ generates $y \in_R Z_q^*$, $Y = g^y$ and Z^y. Again, (y, Y, Z^y) can be pre-computed by $\mathcal{C}$ and leaked to $\mathcal{F}$ prior to the session. Then, $\mathcal{C}$ responds $(y, Y = g^y, Z^y)$ to $\mathcal{F}$, and stores the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, y, Y, A^y)$ as an "incomplete session".

S2. $\mathcal{F}$ presents $\mathcal{C}$ with $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$, and a *challenge* X.

S3. $\hat{B}$ checks that $X \in G/1_G$ (if not, it aborts) and that $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, Y)$ is in one of its incomplete sessions (if not, it ignores the query). $\mathcal{C}$ checks for every value $\sigma \in G/1_G$ *previously used by* $\mathcal{F}$ as input to H_K whether $\sigma = Z^y X^{b+ye}$, where $e = h(m_{\hat{Z}}, m_{\hat{B}}, \hat{Z}, Z, \hat{B}, B, X, Y)$ (in case of undefined e, $\mathcal{C}$ defines it with the RO h). It does so using the DDH-oracle $\mathcal{O}$, specifically, by checking whether $CDH(X, B) = (\sigma / Z^y X^{ye})$. If the answer is positive, then $\mathcal{C}$ sets r to the already determined value of $H_K(\sigma)$.

S3.1. In any other cases, r is set to be a random value in $\{0,1\}^k$, where k is the output length of H_K. Note that, in this case, $\mathcal{C}$ does not know $\sigma = Z^y X^{b+ey}$, as it does not know b, which also implies that $\mathcal{C}$ does not make (actually realize) the RO-query $H_K(\sigma)$ *even if the value* σ *has been well-defined (with predetermined* d *and* e*) and known to* $\mathcal{F}$.

Finally, $\mathcal{C}$ marks the vector $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^y)$ as a "*complete session*", stores $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^y, r)$ and responds $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, Y, r)$ to $\mathcal{F}$.

RO queries: $\mathcal{C}$ provides random answers to queries to the random oracles h and H_K (made by $\mathcal{F}$), under the limitation that if the same RO-query is presented more than once, $\mathcal{C}$ answers it with the same response as in the first time. But, for each *new* query σ to H_K, $\mathcal{C}$ checks whether $\sigma = Z^y X^{b+ey}$ for any one of the stored vectors $(\hat{Z}, Z, m_{\hat{Z}}, m_{\hat{B}}, X, y, Y, Z^y, r)$ (as before, this check is done using the DDH-oracle). If equality holds then the corresponding r is returned as the predefined $H_K(\sigma)$, otherwise a random r is returned.
Upon $\mathcal{F}$'s termination. When $\mathcal{F}$ halts, $\mathcal{C}$ checks whether the following conditions hold:

F1. $\mathcal{F}$ outputs a valid HDR-signature $(\hat{A}, A, m_1, m_0, X_0, Y_0, r_0)$, where $\hat{A} \neq \hat{B}$ is an uncorrupted player. In particular, it implies that r_0 should be $H_K(\sigma_0)$, where $\sigma_0 = A^{y_0} X_0^{b+y_0 e_0}$, $Y_0 = g^{y_0}$ (chosen by $\mathcal{F}$), and $e_0 = h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0 Y_0)$.

F2. $(\hat{A}, A, m_1, m_0, X_0, Y_0)$ did not appear in any of the above responses of the simulated sYZ-HDR signatures.

F3. The value $e_0 = h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0 Y_0)$ was queried from the RO h, and the value $H_K(\sigma_0)$ was queried from H_K *being posterior to the query* e_0. Otherwise, $\mathcal{C}$ aborts.

If these three conditions hold, $\mathcal{C}$ proceeds to the "repeat experiment" below, else it aborts.
The repeat experiment. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time, under the same input (B, X_0) and using the same coins for $\mathcal{F}$. $\mathcal{C}$ rewinds $\mathcal{F}$ to the point of making the RO query $h(m_1, m_0, \hat{A}, A, \hat{B}, B, X_0 Y_0)$, responds back a new independent value $e_0' \in_R \{0,1\}^l$. All subsequent actions of $\mathcal{C}$ (including random answers to subsequent RO queries) are independent of the first run. If in this repeated run $\mathcal{F}$ outputs a successful forgery $(\hat{A}', A', m_1', m_0, X_0, Y_0, r_0')$ satisfying the conditions F1-F3 (otherwise, $\mathcal{C}$ aborts), which particularly implies that $r_0' = H_K(\sigma_0')$, $\sigma_0' = A'^{y_0} X_0^{b+y_0 e_0'}$, $\mathcal{C}$ computes $CDH(X_0, Y_0) = g^{x_0 y_0} = [(\sigma_0 / Y_0^a)/(\sigma_0' / Y_0^{a'})]^{(e_0 - e_0')^{-1}}$, where a and a' are the private keys of the uncorrupted $\hat{A}$ and $\hat{A}'$ (different from $\hat{B}$, which are assumed to be known to $\mathcal{C}$). Note that $(\hat{A}', A', m_1')$ need not necessarily to equal $(\hat{A}, A, m_1)$. Finally, $\mathcal{C}$ computes $CDH(U, V) = CDH(X_0, B) = \sigma_0 / ((g^{x_0 y_0})^{e_0} \cdot Y_0^a)$.

Figure 13: Reduction from CDH to sYZ-HDR forgeries

YZ-HDR and sYZ-HDR are secure under the GDH assumption, and additionally *the KEA assumption*, in the RO model. We remark that the KEA assumption is only used to rule out the feasibility of successful forgeries in this case of $Y_0 \neq X_0$ and $\hat{A} = \hat{B}$.

Lemma 7.2 Under the computational Diffie-Hellman (CDH) assumption*, YZ-HDR and sYZ-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), are* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself with $Y_0 = X_0$.*

Proof. This case implies that the forger $\mathcal{F}$ can output, with non-negligible probability, a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$, where $r_0 = H_K(\sigma_0)$, $\sigma_0 = B^{c_0 x_0} X_0^{d_0 b + e_0 x_0} = (X_0^{c_0} X_0^{d_0})^b X_0^{e_0 x_0}$, $c_0 = h(m_1, \hat{B}, B, X_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, X_0)$ for YZ-HDR (for sYZ-HDR, $c_0 = d_0 = 1$ and $e_0 = h(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0)$). Note that from σ_0 and $\hat{B}$'s secret-key b, we can compute out $X_0^{x_0}$. But, as mentioned, the hardness of computing out X^x from random X is equivalent to that of the CDH problem [97].

With the above observations, we modify the algorithm $\mathcal{C}$ depicted in Figure 12 and Figure 13 as follows:

- $\mathcal{C}$ knows (sets) also the private key b for $\hat{B}$. By knowing the private key b, $\mathcal{C}$ dispenses with the DDH-oracle in order to make the answers to RO-queries to be consistent.
- After $\mathcal{F}$ outputs a successful forgery of the form $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, X_0, r_0)$, satisfying the conditions F1-F3, $\mathcal{C}$ simply computes out $X_0^{x_0}$ from σ_0 and the private-key b. Note that $\mathcal{C}$ does not need to perform the rewinding experiments at all in this case.

The analysis show that, in case of successful forgery against the signer itself with $Y_0 = X_0$, the security not only is based on the weaker hardness assumption (say, the CDH assumption rather than the GDH assumption), but also of tighter security reduction (to the underlying hardness assumption, say the CDH assumption here). □

Now we consider the case of $Y_0 \neq X_0$. As mentioned, it is the only place we need to additionally use the KEA assumption. Recall that the KEA assumption says that, given $B = g^b$, the only way to output $(C, D = C^b) \in G^2$ is to know the discrete logarithm c of C such that $C = g^c$.

Lemma 7.3 *Under the GDH assumption, and additionally the KEA assumption, YZ-HDR and sYZ-HDR signatures of $\hat{B}$, with offline pre-computed and exposable (y, Y, A^{cy}), are* strongly *secure in the random oracle model, with respect to the signer $\hat{B}$ itself with $Y_0 \neq X_0$.*

Proof. The proof of Lemma 7.3 follows the same outline of that of Theorem 7.2. We highlight the main differences, and how the KEA assumption comes into force in the security analysis. The analysis is mainly w.r.t. YZ-HDR (the similar, and actually simpler, hold also for sYZ-HDR).

The main difference between the proof of Lemma 7.3 and that of Theorem 7.2 is that, here, the forger outputs with non-negligible probability a successful forgery of the form: $(m_1, m_0, \hat{B}, B, \hat{B}, B, X_0, Y_0, r_0)$, where $r_0 = H_K(\sigma_0)$, $\sigma_0 = B^{c_0 y_0} X_0^{d_0 b + e_0 y_0}$, $c_0 = h(m_1, \hat{B}, B, Y_0)$, $d_0 = h(m_0, \hat{B}, B, X_0)$, $e_0 = h(X_0, Y_0)$. The key point is that, by performing the rewinding experiments, we cannot directly output the $CDH(B, X_0)$, as we do not know the private key b of $\hat{B}$ (recall that we are going to compute $CDH(B, X_0)$ by running the forger $\mathcal{F}$). Note that in the security analysis of Theorem 7.2, we heavily relied on the fact that we know the private key of any uncorrupted player other than the signer itself.

We modify the algorithm $\mathcal{C}$ depicted in Figure 12 and Figure 13 as follows: the actions of $\mathcal{C}$ remain unchanged until the rewinding experiments; $\mathcal{C}$ performs the rewinding experiments according to the order of the RO-queries c_0, d_0, e_0.

d_0 **posterior to** c_0, e_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $d_0 = h(m_0, \hat{B}, B, X_0)$, and redefines $h(m_0, \hat{B}, B, X_0)$ to be a new independent d'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c_0 y_0} X_0^{d'_0 b + e_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets that $CDH(B, X_0) = (\sigma/\sigma'_0)^{(d_0 - d'_0)^{-1}}$. *Note that, in this case,* $\mathcal{C}$ *does not rely on the KEA assumption for breaking the CDH assumption* (but still with the DDH-oracle).

c_0 **posterior to** d_0, e_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $c_0 = h(m_1, \hat{B}, B, Y_0)$, and redefines $h(m_1, \hat{B}, B, Y_0)$ to be a new independent c'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c'_0 y_0} X_0^{d_0 b + e_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets $CDH(B, Y_0) = B^{y_0} = (\sigma/\sigma'_0)^{(c_0 - c'_0)^{-1}}$. That is, given B, $\mathcal{C}$ can output (Y_0, B^{y_0}). By the KEA assumption, it implies that $\mathcal{F}$ knows y_0 (which can be derived from the internal state of $\mathcal{F}$). More formally, there exists an algorithm that, given B and X_0 and the random coins of $\mathcal{C}$ and $\mathcal{F}$ can successfully output y_0. Now, with the knowledge of y_0, $CDH(B, X_0)$ can be derived from σ_0 (or σ'_0).

e_0 **posterior to** c_0, d_0**.** In this case, by rewinding $\mathcal{F}$ to the point of making the query $e_0 = h(X_0, Y_0)$, and redefines $h(X_0, Y_0)$ to be a new independent e'_0, $\mathcal{C}$ will get $\sigma'_0 = B^{c_0 y_0} X_0^{d_0 b + e'_0 y_0}$. Then, from σ_0 and σ'_0, $\mathcal{C}$ gets $CDH(X_0, Y_0) = X_0^{y_0} = (\sigma/\sigma'_0)^{(e_0 - e'_0)^{-1}}$. Then, by the KEA assumption, the knowledge of y_0 can be derived, with which $CDH(X_0, B)$ can then be computed from either σ_0 or σ'_0. □

7.2.1 (s)YZ-HCR Signatures

The YZ-HCR and sYZ-HCR signatures, implied by the YZ-HDR and sYZ-HDR, are defined as follows:

Definition 7.4 ((s)YZ-HCR signature family) *Let $\hat{A}$, $\hat{B}$ be two parties, and $\hat{B}$ (the signer) is of public-key $B = g^b$ (the signature verifier $\hat{A}$ does not necessarily possess public-key for HCR signatures). Let $m_{\hat{A}}$, $m_{\hat{B}}$ be two messages. The* YZ-HCR, sYZ-HCR*, signatures of $\hat{B}$ on messages $(m_{\hat{B}}, \hat{A}, \hat{B}, B, X, Y)$ are defined as a vector of values (the signatures of $\hat{A}$, in case of possessing public-key $A = g^a$, are defined straightforwardly):*

YZ-HCR. $\{\hat{A}, m_{\hat{B}}, X, Y, HSIG^{YZ}_{\hat{A},\hat{B}}(m_{\hat{B}}, X, Y) = H_K(X^{bd+ye})\}$*, where $X = g^x$, $Y = g^y$ are chosen by $\hat{A}$, $\hat{B}$ respectively as the random challenge and response, $x, y \in_{\mathrm{R}} Z_q^*$, $d = h(\hat{A}, m_{\hat{B}}, \hat{B}, B, X)$ and $e = h(X, Y)$. Note that $\hat{A}$ can offline pre-compute B^{dx} for verifying YZ-HCR signatures.*

sYZ-HCR. $\{\hat{A}, m_{\hat{B}}, X, Y, HSIG^{sYZ}_{\hat{A},\hat{B}}(m_{\hat{B}}, X, Y) = H_K(X^{bd+ye})\}$*, where $e = h(m_{\hat{B}}, \hat{A}, \hat{B}, B, X, Y)$ and $d = 1$. Again, $\hat{A}$ can offline pre-compute B^x for verifying sYZ-HCR signatures.*

General YZ-HCR (gYZ-HCR). $\{\hat{A}, m_{\hat{B}}, X, Y, HSIG^{sYZ}_{\hat{A},\hat{B}}(m_{\hat{B}}, X, Y) = H_K(X^{bd+ye})\}$*, where db and ey are computationally independent no matter how malicious $\hat{B}$ does, and $(m_{\hat{B}}, \hat{A}, \hat{B}, B, X, Y)$ are committed to (d, e).*

Strong **security of (s)YZ-HCR.** The definition of strong security of HCR signature scheme is similar to Definition of 7.3 for the strong security of HDR, with the following modifications in the forger game depicted in Figure 11: (1) The signing query from the forger $\mathcal{F}$ to the signer now does not include $(A, m_{\hat{A}})$; (2) A HCR signature forgery, denoted $(\hat{A}, m_1, X_0, Y_0, r_0)$, where X_0 is the given random challenge and Y_0 is generated by the forger itself, output by the forger $\mathcal{F}$ at the end of its attack, is deemed to be *successful* if both it is a valid HCR signature, i.e., $r_0 = H_K(\sigma_0)$ and $\sigma_0 = X_0^{d_0 b + e_0 y_0}$, and the vector $(\hat{A}, m_1, X_0, Y_0)$ did not appear in any of the responses of $\hat{B}$ to $\mathcal{F}$'s queries.

We note that both YZ-HCR and sYZ-HCR are strongly secure under the GDH assumption (without relying on the KEA assumption). The security analysis essentially follows the proof of Theorem 7.2. Dispensing with the KEA assumption in the analysis of YZ/sYZ-HCR is from the fact that: the value A^{cy} is removed from the computation of HCR signature, and thus we get no trouble with computing the value B^{y_0} in the analysis of YZ-HDR for the case of $\hat{A} = \hat{B}$ (in order to compute out X_0^b with the rewinding experiments). We state the following result, with details omitted here.

Lemma 7.4 *Under the GDH assumption, YZ-HCR and sYZ-HCR, with offline pre-computed and exposable* (y, Y)*, are* strongly *secure in the random oracle model.*

7.3 Analysis of (s)YZ with Offline Pre-computation in the CK-Framework

In this section, we present the analysis of YZ and sYZ protocols in the CK-framework *with pre-specified peers* (recalled in Section 4.4.1), with offline pre-computed and exposable DH-exponents, DH-components, and DH-secrets derived from one's DH-component and its peer's public-key (say, A^{cy} and B^{dx}) which may be exposed to the adversary prior to the session involving these pre-computed values.

Using the terminology of HDR signatures, a session of YZ (resp., sYZ), for the basic protocol version without explicit mutual identifications and key confirmations, between two parties $\hat{A}$ and $\hat{B}$ consists of a basic Diffie-Hellman exchange of DH-components $X = g^x$ and $Y = g^y$; And the session-key K is then computed as the corresponding HDR-signatures, specifically, $K = HSIG^{YZ}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y)$ for YZ and (resp., $K = HSIG^{sYZ}_{\hat{A},\hat{B}}(m_{\hat{A}}, m_{\hat{B}}, X, Y)$ for sYZ), where $m_{\hat{A}}$ *and* $m_{\hat{B}}$ *are the empty string for both YZ and sYZ.*

During a session of YZ/sYZ within the CK-framework, with offline pre-computation, a party can be activated with three types of activations (for presentation simplicity, we assume $\hat{A}$ denotes the identity of the party being activated and $\hat{B}$ the identity of the intended peer to the session):

Initiate$(\hat{A}, \hat{B})$ (i.e., $\hat{A}$ is activated as the initiator): $\hat{A}$ generates a value $X = g^x$, $x \in_{\mathrm{R}} Z_q^*$, creates a local session of the protocol which it identifies as (the incomplete) session $(\hat{A}, \hat{B}, X)$, and outputs the DH-component X as its outgoing message.

Here (X, x, B^{dx}), where $d = h(\hat{B}, B, X)$ for YZ or $d = 1$ for sYZ can be offline pre-computed by $\hat{A}$, which may be exposed to the adversary prior to the session involving them.

Respond$(\hat{A}, \hat{B}, Y)$ (i.e., $\hat{A}$ is activated as the responder): $\hat{A}$ checks $Y \in G/1_G$, if so it generates a value $X = g^x$, $x \in_{\mathrm{R}} Z_q^*$, outputs X, computes the session-key and then completes the session $(\hat{A}, \hat{B}, X, Y)$.

Again, (X, x, B^{dx}) can be offline pre-computed by $\hat{A}$, which may be exposed to the adversary prior to the session involving them.

Complete$(\hat{A}, \hat{B}, X, Y)$ (i.e., the initiator $\hat{A}$ receives Y from the responder peer $\hat{B}$): $\hat{A}$ checks that $Y \in G/1_G$ and that it has an open session with identifier $(\hat{A}, \hat{B}, X)$. If any of these conditions fails $\hat{A}$ ignores the activation, otherwise it computes the session-key and completes the session $(\hat{A}, \hat{B}, X, Y)$.

With the above notation, it is ensured that if $(\hat{A}, \hat{B}, X, Y)$ is a complete session at $\hat{A}$, then its matching session (if it exists) is unique, which is $(\hat{B}, \hat{A}, Y, X)$ owned by the player $\hat{B}$. In the following analysis, we specify that the values, exposable to the adversary via session-state query (against an incomplete session), include the DH-component and DH-exponent and the DH-secret of one's DH-component and its peer's public-key, e.g., (Y, y, A^{cy}).

Theorem 7.3 *Under the GDH assumption in the RO model, the YZ and sYZ protocols (actually, the variants with public-keys removed from the inputs of c, d, e), with offline pre-computed DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say A^{cy} and B^{dx}), are SK-secure in the CK-framework w.r.t. any test-session between a pair of different players.*

Proof. According to the SK-security definition in the CK-framework, we need to prove YZ and sYZ satisfy the following two requirements:

Requirement-1. If two parties $\hat{A}, \hat{B}$ complete matching sessions, then their session-keys are the same.

Requirement-2. Under the CDH assumption, there is no feasible adversary that succeeds in distinguishing the session-key of an unexposed session with non-negligible probability.

The Requirement-1 can be trivially checked for both YZ and sYZ. In the following, we focus on establishing the Requirement-2.

Denote by $(\hat{A}, \hat{B}, X_0, Y_0)$ the unexposed test-session between a pair of *uncorrupted* players $\hat{A}$ and $\hat{B}$ where $\hat{A} \neq \hat{B}$, and by $H_K(v)$ the session-key of the test-session that is referred to as the *test* HDR-signature, where $v = A^{cy}X^{db+ey} = B^{dx}Y^{ca+ex}$. As H_K is a random oracle, there are only two strategies for the adversary $\mathscr{A}$ to distinguish $H_K(v)$ from a random value:

Key-replication attack. $\mathscr{A}$ succeeds in forcing the establishment of a session (other than the test-session or its matching session) that has the same session-key output as the test-session. In this case, $\mathscr{A}$ can learn the test-session key by simply querying the session to get the same key (without having to learn the value of the test HDR-signature).

Forging attack. At some point in its run, $\mathscr{A}$ queries the RO H_K with the value v. This implies that $\mathscr{A}$ succeeds in computing or learning the test HDR-signature (i.e., the session-key of the test-session) via its attacks. For presentation simplicity, we assume $\mathscr{A}$ directly outputs the session-key of the test-session, referred to as the test-signature, via a successful forging attack.

The possibility of key-replication attack is trivially ruled out *unconditionally* in the RO model, by the tag-based self-seal (TBSS) property of YZ and sYZ. Specifically, for any session-tag $(\hat{A}, A, \hat{B}, B, X, Y)$ and for any value $\sigma \in G/1_G$, the probability $\Pr[K_{\hat{A}} = K_{\hat{B}} = \sigma] \leq \frac{1}{2^{l-1}}$ for both YZ and sYZ, where the probability is taken over only the choice of the random function h. We note that, however, the analysis of HMQV to rule out key-replication attack in [80] is quite complicated, and is still reduced to the underlying hardness assumptions (to be precisely, to the unforgeability of HMQV-HDR). That is, the analysis of YZ/sYZ in order to rule out the key-replication attacks is not only much simpler, but also does not go through costly security reductions.

Then, in the following analysis, we only focus on ruling out the forging attack. Recall that $\hat{A} \neq \hat{B}$ for the test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ held by $\hat{A}$. In the rest, we make analysis mainly with respect to the YZ protocol, the similar and actually simpler hold also for sYZ.

Now, suppose there is an efficient KE-attacker $\mathscr{A}$ who succeeds, by forging attacks, against the test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ with $\hat{A} \neq \hat{B}$ (particularly, $A \neq B$), we present an efficient forger $\mathcal{F}$ against the underlying YZ-HDR signature, which contradicts the security of the underlying YZ-HDR signature scheme (that is based on the GDH assumption), and thus establishing the theorem. $\mathcal{F}$ works as follows, by running $\mathscr{A}$ as a subroutine.

1. We assume $\mathcal{F}$ successfully guessed the *exposed* test-session $(\hat{A}, \hat{B}, X_0, Y_0)$ held at $\hat{A}$, where $\hat{A} \neq \hat{B}$.

2. The inputs of $\mathcal{F}$ are (B, X_0), and $\mathcal{F}$ has oracle access to the YZ-HDR signer $\hat{B}$ of public-key B.

3. $\mathcal{F}$ sets the inputs to all parties other than $\hat{B}$, and thus can perfectly emulate these parties. In particular, $\mathcal{F}$ can deal with state-reveal queries, session-key queries by $\mathscr{A}$ on any session other than the test-session and its matching session, and party corruption queries on any party other than $\hat{A}$ and $\hat{B}$.

4. When $\mathscr{A}$ activates a session at $\hat{B}$, either as a responder or initiator, with peer identity $\hat{P}$ of public-key P and incoming message X, then $\mathcal{F}$ feeds $\hat{B}$ the value $(\hat{P}, P, X)$. In response, $\mathcal{F}$ gets values (y, Y, P^{cy}) from $\hat{B}$, and then $\mathcal{F}$ hands $\mathscr{A}$ the value Y as the outgoing message from $\hat{B}$. Actually, the values (y, Y, P^{cy}) can be offline pre-computed by $\hat{B}$, and leaked to $\mathcal{F}$ (and $\mathscr{A}$) prior to the session involving them.

5. When $\mathscr{A}$ issues a state-reveal query against an incomplete session $(\hat{B}, \hat{P}, Y)$ (*not matching to the test-session*) held at $\hat{B}$, then $\mathcal{F}$ returns the values (Y, y, P^{cy}) to $\mathscr{A}$.

6. When $\mathscr{A}$ issues a session-key query to a session $(\hat{B}, \hat{P}, Y, X)$ (*not matching to the test-session*) held at $\hat{B}$, then $\mathcal{F}$ queries the session-signature from its signing oracle $\hat{B}$ by presenting the signing oracle with $(\hat{P}, P, X, Y)$, and returns the HDR-signature from $\hat{B}$ to $\mathscr{A}$.

7. When $\mathscr{A}$ halts with a valid test-signature, denoted σ_0, $\mathcal{F}$ stops and outputs σ_0.

Suppose there are n parties in total in the system, and each party is activated at most m times (where n and m are polynomials in the security parameter), in actual analysis $\mathcal{F}$ guesses the test-session by choosing uniformly at random a triple $(\hat{P}_i, \hat{P}_j, t)$ (hoping that $\hat{P}_i = \hat{A}$ and $\hat{P}_j = \hat{B}$ and the test-session is the t-th session activated at $\hat{A}$ with peer $\hat{B}$), where $1 \leq i \neq j \leq n$ and $1 \leq t \leq m$. Thus, with probability $(n^2 m)^{-1}$, $\mathcal{F}$ successfully guesses the test-session. It is easy to check that, conditioned on $\mathcal{F}$ successfully checks the test-session, the view of $\mathscr{A}$ under the run of $\mathcal{F}$ is identical to that in the real run of $\mathscr{A}$. Suppose $\mathscr{A}$ successfully outputs, with non-negligible probability ε, the valid test-signature via forging attack in its real run, with still non-negligible probability $(n^2 m)^{-1}\varepsilon$ $\mathscr{A}$ (and thus $\mathcal{F}$) outputs the valid test-signature under the run of $\mathcal{F}$.

We need then to check whether the valid test HDR-signature outputted by $\mathcal{F}$ is a *successful* YZ-HDR forgery. As the test-signature output by $\mathscr{A}$ is valid, according to Definition 7.3, we only need to show the vector $\{\hat{A}, A, X_0, Y_0\}$ did not appear in any one of the responses from the signing oracle $\hat{B}$. We distinguish three cases, according to the appearance of Y_0:

Case-1. Y_0 was never output in any one of the signatures issued by $\hat{B}$. In this case, the test HDR-signature output by $\mathscr{A}$ (and thus $\mathcal{F}$) is clearly a successful forgery against YZ-HDR.

Case-2. Y_0 was output in one of the signatures issued by $\hat{B}$ in a session *non-matching* to the test-session. Denote by $(\hat{B}, \hat{P}, Y_0, X)$ this non-matching session, we have that $\hat{P} \neq \hat{A}$ or $X \neq X_0$. That is, $(\hat{P}, P, X) \neq (\hat{A}, A, X_0)$. As $\hat{B}$ uses random and independent DH-components in each session, the value Y_0 is only used in this non-matching session $(\hat{B}, \hat{P}, Y_0, X)$, and thus does not appear (except for a negligible probability of accidental repetition) in any other signatures issued by $\hat{B}$ in other sessions different from $(\hat{B}, \hat{P}, Y_0, X)$. Putting all together, we get that $\{\hat{A}, A, X_0, Y_0\}$ did not appear in any of the HDR-signatures issued by $\hat{B}$, and thus the test HDR-signature output by $\mathcal{F}$ is a successful forgery against YZ-HDR.

Case-3. Y_0 was generated by $\hat{B}$ in the *matching* session $(\hat{B}, \hat{A}, Y_0, X_0)$. However, this matching session was never queried by $\mathscr{A}$ via session-key query or session-state query (recall we assume the test-session and its matching session are unexposed in the CK-framework), which in

turn implies that $\mathcal{F}$ never queries $\hat{B}$ for the HDR-signature of this matching session. Also, the random value Y_0 was used by $\hat{B}$ only for this matching session (except for a negligible probability of accidental repetition). This implies that, in Case-3, the values $\{\hat{A}, A, X_0, Y_0\}$ also did not appear in any one of the responses from the signing oracle $\hat{B}$, and thus the test HDR-signature output by $\mathcal{F}$ is a successful forgery against YZ-HDR. □

Notes on the security analysis of YZ/sYZ in the CK-framework. For the above security analysis of YZ/sYZ in the CK-framework, we have the following observations and notes:

- For the same security level (actually, whenever the DH-component is offline pre-computed and exposable, no matter whether the secret DH-exponent is exposable or not), the security of HMQV in the CK-framework relies on both the GDH assumption and the KEA assumption. In contrast, for the security of YZ/sYZ *even with the additional powerful exposure of DH-exponents and* A^{cy} *or* B^{dx}, the KEA assumption is dispensed with.

- The security reduction (from the security of YZ/sYZ to the security of the underlying HDR signatures) is tighter than that of HMQV.

- Note that the above security analysis is actually w.r.t. the *public-key free* variants of YZ/sYZ, with players' public-keys removed from the inputs of the functions of c, d, e. The reason is that the security of the underlying YZ-HDR/sYZ-HDR signatures does not rely on them.

- The analysis shows that YZ and sYZ remain their security in the CK-framework, even if the attacker $\mathscr{A}$ exposes the private values (y, A^{cy}) of the matching session (but not the session-key itself). This provides extra security guarantee of YZ/sYZ that is beyond the CK-framework. The reason is that, even if these pre-computed private values are used by $\hat{B}$ in the matching session $(\hat{B}, \hat{A}, Y_0, X_0)$ and exposed to $\mathscr{A}$, the forger $\mathcal{F}$ never queries the *full* HDR-signature corresponding to this matching session as the underlying attacker $\mathscr{A}$ is not allowed to make the session-key query against the matching session (note that $\mathcal{F}$ queries the HDR signer for a full session-signature only when $\mathscr{A}$ makes the session-key query against this session), and thus $(\hat{A}, A, X_0, Y_0)$ still did not appear in any one of the signatures issued by $\hat{B}$.

Using Lemma 7.2 and Lemma 7.3, we have the following lemmas about the security of YZ/sYZ in the CK-framework w.r.t. any test-session between the identical players $\hat{A} = \hat{B}$. The proofs are straightforward adaptations of the proof of Theorem 7.3, and details are omitted here.

Lemma 7.5 *Under the CDH assumption in the RO model, the YZ and sYZ protocols (actually, the variants with public-keys removed from the inputs of* c, d, e*), with offline pre-computed and exposable DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say* A^{cy} *and* B^{dx}*), are SK-secure in the CK-framework w.r.t. any test-session of identical peer and identical DH-component (i.e.,* $\hat{A} = \hat{B}$ *and* $X = Y$*).*

Lemma 7.6 *Under the GDH assumption and additionally the KEA assumption in the RO model, the YZ and sYZ protocols (actually, the variants with public-keys removed from the inputs of* c, d, e*), with offline pre-computed and exposable DH-components, DH-exponents, and the DH-secrets of one's DH-component and its peer's public-key (say* A^{cy} *and* B^{dx}*), are SK-secure in the CK-framework w.r.t. any test-session of identical peer but different DH-components (i.e.,* $\hat{A} = \hat{B}$ *but* $X \neq Y$*).*

Notes on some inherent security limitations. The reader should beware of some inherent security limitations for any one-round and two-round implicitly-authenticated DHKE protocols,

e.g., the PFS vulnerability for any two-round implicitly-authenticated DHKE and the KCI vulnerability for any one-round DHKE (more details are referred to [80]). Even for the three-round version of YZ (as well as HMQV) with explicit mutual authentications, there are also some inherent limitations. For example, the protocol responder may not be able to get deniability in a fair way, in case the malicious protocol initiator just aborts after receiving the second-round message; Also, both the three-round YZ and (H)MQV suffer from the cutting-last-message attack [96], etc. We remark that losing deniability fairness to protocol responder and lacking correct delivery guarantee of the last message are inherent to the protocol structure of YZ and (H)MQV and do not violate the definition of the SK-security in the CK-framework, which though can be easily remedied but at the price of ruining the performance advantages and/or adding additional system complexity.

8 Security of YZ Beyond the CK-framework

In this section, we investigate some security properties not captured by the CK-framework, which further strengthens the security guarantee of the YZ protocol family.

8.1 Arguments for Tag-Based Robust Non-Malleability Against Powerful Secrecy Exposure

By "tag-based robust non-malleability (TBRNM) against powerful secrecy exposure", we mean the remarkable robust security goal that a party completing a session of some *distinct* session-tag, referred to as the test-session, has the following guarantee: as long as the private values generated by the party during the run of the session are not exposed (in other words, the attacker did not make state-reveal query against this session within the CK-framework), the peer to this session (that may be controlled or impersonated by the attacker) has to "know" both the secret-key and DH-exponent corresponding to the public-key and DH-component alleged and sent by this peer. Note that this holds even if the attacker learns the secret-keys of all parties, the ephemeral private values and session-keys of all sessions other than the test-session. For presentation simplicity, we call the session, against which the attacker did not make the state-reveal query, an *internally-unexposed* session.

We argue that the YZ and sYZ protocols essentially achieve this robust security goal. At a high level, the TBRNM security of YZ/sYZ is essentially inherited from that of the underlying building tools SSJPOK/sSSJPOK, which is clarified and justified in Section 5.2. We reminder that as TBRNM (formulated in Section 4.1) implies concurrent statistical zero-knowledge, and as the YZ/sYZ protocols cannot be provably zero-knowledge, the following security arguments are far from being a formal security proof. However, the informal security arguments and discussions, for this goal of TBRNM against powerful secrecy exposure, does indeed further demonstrate and justify (as well as provide extra confidence and witnesses on) the soundness and robustness of the YZ/sYZ protocols. Recall that, as clarified in Section 5.3.3, (H)MQV is not TBRNM secure even without secrecy exposure (without POK/P of secret-keys).

8.1.1 High-level Discussion of the TBRNM Security

We first present some high-level discussions on the TBRNM security against powerful secrecy exposure for the YZ protocol. At a high level, the analysis of TBRNM of YZ can be essentially reduced to the following adversarial scenario (the analysis extension is discussed later). The concurrent man-in-the-middle (CMIM) adversary $\mathscr{A}$ interacts with the uncorrupted player $\hat{A}$ of public-key $A = g^a$ in the name $\hat{M}_1$ of alleged public-key M_1 in one session (which is referred to as the test-session); simultaneously, $\mathscr{A}$ concurrently interacts with the uncorrupted player $\hat{B}$ of public-key $B = g^b$ in the name of $\hat{M}_2$ of alleged public-key M_2 in another session (which is referred to as the

second session for presentation simplicity). Suppose $\hat{A}$ sends $X = g^x$ in the first-round of the first session, $\mathscr{A}$ forwards X' to $\hat{B}$ in the first-round of the second-session; After receiving Y from $\hat{B}$ in the second-session, $\mathscr{A}$ sends Y' in the second-round of the first session, where the DH-components X' and Y' and the alleged public-keys M_1 and M_2 can be any elements in $G/1_G$ and particularly can be dependent on the values (A, B, X, Y).

We assume the test-session at the side of $\hat{A}$, referred to as the session $(\hat{A}, X)$, is internally-unexposed and is of distinct tag, but the adversary $\mathscr{A}$ still can learn the static secret-key a of $\hat{A}$; Furthermore, $\mathscr{A}$ can also learn the static secret-key b of $\hat{B}$ and can learn any private values generated by $\hat{A}$ and $\hat{B}$ in any other sessions other than the test-session. The CMIM adversary also possesses some arbitrary auxiliary input $z \in \{0,1\}^*$, which captures arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from the executions of arbitrary (*possibly different*) protocols prior to its interactions with the instances of $\hat{A}$ or $\hat{B}$. (Actually, the adversary is also allowed to expose the session-key outputs of any completed sessions of different tags. But, as session-key outputs are independent at random for completed sessions of different tags, the exposure of session-key outputs of other sessions does not help the adversary to successfully finish the test-session, and thus is omitted in the following analysis.) The goal of the adversary $\mathscr{A}$ is to compute $K_{\hat{A}} (= K_{\hat{M}_1})$ of the test-session that is internally-unexposed and of distinct tag, but without knowing either the secret-key m_1 (corresponding to the public-key $M_1 = g^{m_1}$ alleged by $\mathscr{A}$) or the DH-exponent y' (i.e., the discrete logarithm of $Y' = g^{y'}$).

With respect to the above adversarial scenario, the tag of the second session is $(\hat{M}_2, M_2, X', \hat{B}, B, Y)$ and the tag of the test-session is $(\hat{A}, A, X, \hat{M}_1, M_1, Y')$. As we assume the test-session is of *distinct* tag, $(\hat{A}, A, X, \hat{M}_1, M_1, Y') \neq (\hat{M}_2, M_2, X', \hat{B}, B, Y)$. Note that $K_{\hat{A}} = M_1^{d_1 x}(Y')^{c_1 a + e_1 x} = A^{c_1 y'} X^{d_1 m_1 + e_1 y'} = K_{\hat{M}_1}$ for the test-session, where $c_1 = h(\hat{A}, A, Y')$ and $d_1 = h(\hat{M}_1, M_1, X)$ and $e_1 = h(X, Y')$. Suppose $\mathscr{A}$ learns the secret-key a of $\hat{A}$ (but knows neither m_1 nor y' *for now*), then the computation of $K_{\hat{M}_1}$ by the adversary $\mathscr{A}$ is actually reduced to the computation of $X^{d_1 m_1 + e_1 y'}$. Note that the values $A^{c_1 y'} = Y'^{c_1 a}$ and $X^{d_1 m_1 + e_1 y'}$ are computationally independent and cannot be made maliciously correlated by $\mathscr{A}$, and thus the knowledge of a and $A^{c_1 y'}$ does not help $\mathscr{A}$ to compute out $X^{d_1 m_1 + e_1 y'}$ suppose $\mathscr{A}$ does not "know" m_1 and y'. Also note that, as we assume the test-session is internally-unexposed, $\mathscr{A}$ cannot learn the ephemeral private values generated by $\hat{A}$ during the run of the test-session, which particularly includes the ephemeral DH-exponent x.

We assume that from the information got by the adversary, which includes all the exposable ephemeral private values (other than those of the test-session), all the exposable static secret-keys and session-keys, the adversary $\mathscr{A}$ cannot efficiently derive *both* m_1 *and* y'. Otherwise, it will not violate TBRNM by definition, as the adversary does indeed know both m_1 and y' w.r.t. the test-session. Now, we further assume that the adversary can get one of the values m_1 and y'. For example, the adversary sets $\hat{M}_1 = \hat{B}$ and then learns $m_1 = b$ by static secret-key exposure at $\hat{B}$; or, $\mathscr{A}$ sets $Y' = Y^2$ and then learns y by exposing the ephemeral private values generated by $\hat{B}$ during the run of the second session.

Suppose the adversary $\mathscr{A}$ knows m_1, but does not know y' (the analysis for the case of knowing y' but not m_1 is similar). Then, the computation of $X^{d_1 m_1 + e_1 y'}$ is reduced to compute $X^{e_1 y'}$. To this end, $\mathscr{A}$ can set its DH-component Y' arbitrarily based on all messages it gets and collects during its attack until the point of sending the second-round message in the test-session. The key observation is: whenever the value Y' is fixed, the value $e_1 = h(X, Y')$ is then determined (that is a random and independent value), which guarantees that the value $X^{e_1 y'}$ will be computationally independent from (i.e., cannot be maliciously correlated to) all other messages got and collected by the adversary. Suppose the adversary does not know y', there are essentially only two ways for $\mathscr{A}$ to successfully compute out $X^{e_1 y'}$: (1) knowing x, which is ruled out as we assume the test-session in internally-unexposed; (2) the adversary can derive $X^{y'}$, from which it can then form $X^{e_1 y'}$. So, we only need to consider the last possibility.

The last possibility of deriving $X^{y'}$ implies that the adversary $\mathscr{A}$ can, by means of its adversarial capability allowed, access to a CDH solver oracle, i.e., for (X, Y') it can find somewhere the value $g^{xy'}$ by means of its adversarial capability. The observation here, as we shall clarify, is the adversary has no means to obtain $X^{y'}$ (other than it can indeed derive y' or violate the CDH assumption).

- On the one hand, by the fresh random challenge KEA assumption against CMIM of arbitrary auxiliary input (clarified in Section 2.1), the arbitrary auxiliary input z possessed by $\mathscr{A}$ does not help it to compute $X^{y'}$ w.r.t. the fresh random challenge X sent by the honest $\hat{A}$.
- On the other hand, as we shall show, the exposable static secret-keys, exposable ephemeral private values of other sessions (other than the test-session) and exposable session-keys also do not render a CDH solver oracle to the adversary. Clearly, exposable static secret-keys and session-key outputs do not render any CDH solver oracle. The subtle point here is about the exposable ephemeral private values. We clarify the subtlety with the ephemeral values exposable from $\hat{B}$ for the second-session. There are two cases:
  - $\hat{B}$ does not pre-compute $M_2^{c_2 b}$, where $c_2 = h(\hat{M}_2, M_2, Y)$. In this case, the exposable ephemeral value is just the DH-knowledge y, and thus no CDH solver oracle is rendered.
  - If $\hat{B}$ pre-computes $M_2^{c_2 y}$, the exposable ephemeral values are $(y, M_2^{c_2 y})$. As $M_2^{c_2 y}$ can be computed from the knowledge y, the leakage of $(y, M_2^{c_2 y})$ also does not render the adversary the CDH solver oracle i.e., any additional advantage other than what can be derived merely from the DH-knowledge y (here we assume the capability of exposing $M_2^{c_2 y}$ implies the capability of exposing y).

8.1.2 Non-Black-Box Argument of TBRNM with the JKEA Assumption

We further give a more technical *non-black-box* (*still informal*) argument of the TBRNM security for YZ/sYZ with the new non-black-box JKEA assumption (proposed in Section 5.2.1). In the following security analysis, we distinguish several cases according to the forms of the session-tag of the internally-unexposed test-session.

The test-session is of (distinct) tag $(\hat{A}, A, X, \hat{B}, B, Y)$ for $\hat{B} \neq \hat{A}$. In this case, if $\hat{B}$ is corrupted or $\hat{B}$ is uncorrupted but the CMIM adversary $\mathscr{A}$ has already exposed the static secret-key b of $\hat{B}$, we construct a simulator S that sets the public-key and secret-key pairs to all uncorrupted players (including $\hat{A}$ and $\hat{B}$) and runs $\mathscr{A}$ as a subroutine by mimicking all uncorrupted players. Clearly, the run of $\mathscr{A}$ under S is identical to its real run. In other words, in this case, what can be done by $\mathscr{A}$ against the test-session $(\hat{A}, A, X, \hat{B}, B, Y)$ can also be done by the simulator. Recall that $K_{\hat{B}} = A^{cy} X^{db+ey}$, where $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$ and $e = h(X, Y)$ for YZ (resp., $c = d = 1$ and $e = h(\hat{A}, A, \hat{B}, B, X, Y)$ for sYZ). As the simulator S knows a it can compute out A^{cy}. Then, by the JKEA assumption (and the fact that the transcript and exposable secrecy do not render CDH solver oracle), it implies that the adversary $\mathscr{A}$ does indeed know both b and y in this case.

The subtle case here is that: $\mathscr{A}$ may not make query to expose the static secret-key b of the uncorrupted player $\hat{B}$ during its CMIM attack against the test-session. Recall that we assume an uncorrupted player can still expose its secret-key to the attacker. In this case, the simulator S runs $\mathscr{A}$ as a subroutine and works as follows: It sets the public-key and secret-key pairs to all uncorrupted players *other than the uncorrupted player* $\hat{B}$, and thus S can perfectly mimic the uncorrupted players except $\hat{B}$; When $\mathscr{A}$ initiates a session with $\hat{B}$ (either as the initiator or responder) with an identity $\hat{M}$ of public-key M and DH-component $Z = g^z \in G/1_G$, we further distinguish two cases according to whether $\hat{A}$ "knows" the discrete logarithm z of Z:[16]

[16] We remark that in an actual formal security proof, we cannot explicitly distinguish whether $\hat{A}$ "knows" z or not.

- $\mathscr{A}$ "knows" z for $Z = g^z$ (i.e., z can be derived from the internal state of $\mathscr{A}$). In this case, the simulator S can perfectly mimic $\hat{B}$ by selecting $y \in Z_q^*$, forwarding $Y = g^y$ and then computing $K_{\hat{B}} = M^{cy} Z^{db+ey} = M^{cy}(B^d Y^e)^z$.

- $\mathscr{A}$ does *not* "know" z for $Z = g^z$. In this case, the simulator S mimics $\hat{B}$ as follows: it responds back $Y = g^y$ (where y is generated by S itself), and then defines the session-key to be a random value r in the range of H_K. (In case the value z is known to $\mathscr{A}$ in a later stage, S can compute $K_{\hat{B}}$ as above and then define $H_K(K_{\hat{B}})$ to be r to ensure the consistency of the RO.) The *non-black-box* simulation of S in this case is still perfect.

According to the *non-black-box* perfect simulation of S, it implies again that what can be done by $\mathscr{A}$, against the internally-unexposed test-session of *distinct* session-tag $(\hat{A}, A, X, \hat{B}, B, Y)$, can also be done by the simulator S. Recall that $K_{\hat{B}} = A^{cy} X^{db+ey}$ for the test-session, as the simulator S knows a it can compute out X^{db+ey}. Then, by the JKEA assumption (and the fact that the transcript and exposable secrecy do not render CDH solver oracle), it implies that the adversary $\mathscr{A}$ does indeed know both b and y (where $Y = g^y$) for the case of secret-key-unexposed $\hat{B}$.

The test-session is of (distinct) tag $(\hat{A}, A, X, \hat{A}, A, Y)$. We note that the TBRNM security in this case can essentially be reduced to the normal (fresh-challenge) KEA assumption (rather than the JKEA assumption). In this case, we first observe that, by the KEA assumption, the adversary $\mathscr{A}$ has to know y (and $A^{cy} X^{da+ey}$) for computing out the session-key $H_K(A^{cy} X^{da+ey}) = H_K((Y^c X^d)^a X^{ey}$ of the test-session. The reason is that X^{ey} is computationally independent of $(Y^c X^d)^a$, and the run of $\mathscr{A}$ can be perfectly simulated by a simulator who sets the public-key/secret-key pairs to all uncorrupted players (including $\hat{A}$), and thus the simulator can output X^{ey} as well. Then, by the (fresh-challenge) KEA assumption, it implies that the adversary $\hat{A}$ has to know y in order to successfully finish the test-session. But, from the knowledge of y and $A^{cy} X^{da+ey}$, we can then compute out X^{da}, which in turn implies that $\hat{A}$ also has to know a.

8.2 Security with Public Computations

The work of [82] considers a new attack scenario for key-exchange protocols with public computations, where it is convenient to split an entity (performing a run of KE-protocol) into two parts: a trusted authentication device, and an untrusted computing device. The authentication device enforces the confidentiality of the authentication data, while some computing operations required by the protocol are *publicly* carried out by the (possibly untrusted) computing device. This allows to use an authentication device with little computing power, and to make computing devices independent from users [82].

The work [82] gives some concrete applications that might be benefited from public computations: (1) Mobile phones include smart cards which store the user authentication data; the handsets themselves are the computing devices. (2) PCs (corresponding to the computing device) equipped with a crypto token (corresponding to the authentication device) have a lot more computing power than the token itself, but may be plagued by spyware or virus. For more details, the reader is referred to [82].

(H)MQV with public computations. With the computation of $\hat{B}$ as an example (the same holds for $\hat{A}$), a natural split of authentication computation and public computation is as follows: The authentication device generates (y, Y), forwards Y to the computation device; After getting $(\hat{A}, X)$ from the computation device, the authentication device computes $s = y + eb$, where $e = h(Y, \hat{A})$, and then forwards s to the computation device; After getting s from the authentication device,

That is the (partial) reason we view the security analysis informal. But, the analysis of this kind is still instrumental to understand and indicate the soundness and robustness of the YZ/sYZ protocols.

the computation device computes $K_{\hat{B}} = (XA^d)^s$, and then the session-key, and then communicate with $\hat{A}$ with the session-key.

One key point is: as we assume the computation device may not be trustful, once the value s is leaked to an attacker (who may compromise the computation device), then the attacker can definitely impersonate $\hat{B}$ to $\hat{A}$ in any sessions. Note that, by only compromising the computation device, the attacker does not learn the DH-exponent y and the private-key b. This shows (H)MQV is totally insecure in such scenarios [82].

YZ/sYZ with public computations. For applications in such scenarios, the natural split of authentication computation and public computation for YZ/sYZ is as follows, with the computation of $\hat{B}$ as an example (the similar hold for $\hat{A}$: (1) The authentication device generates (y, Y) and possibly A^{cy} (in case the authentication device has learnt the peer identity $\hat{A}$) where $c = 1$ for sYZ or $c = h(\hat{A}, A, Y)$ for YZ, and then forwards Y and possibly A^{cy} to the computation device; (2) After getting X from the computation device, the authentication device computes $s = db + ey$, where $d = h(\hat{B}, B, Y)$ and $e = h(X, Y)$ for YZ (resp., $d = 1$ and $e = h(\hat{A}, A, \hat{B}, B, X, Y)$ for sYZ), and then forwards s to the computation device; (3) After getting s from the authentication device, the computation device computes $K_{\hat{B}} = A^{cy}X^s$, and then the session-key, and then communicate with $\hat{A}$ with the session-key. *Note that y, Y, c, d, A^{cy}, db can be offline pre-computed by the authentication device, and the authentication device can only online compute ey and s. Also, the computation device essentially needs to compute only one exponentiation X^s.*

Below, we make some discussions about the security of sYZ and YZ in the public computation model.

Security of sYZ with public computations. We note that, under the DLP assumption, the knowledge of (A^y, s) of a session of sYZ, learnt by the adversary by compromising the computation device, is useless for the attacker to violate other sessions other than the matching session $(\hat{B}, \hat{A}, Y, X)$. The reason is that $s = b + ey$ for sYZ, where $e = h(\hat{A}, A, \hat{B}, B, X, Y)$, commits to the whole session-tag. Thus, the value s cannot be used by the attacker to violate a non-matching session, unless it can compute out y from A^y (and thus b from s) which however is infeasible by the DLP assumption.

Security of YZ with public computations. The knowledge $(A^c y, s)$ of a session of YZ, where $s = db + ey$, $d = h(\hat{B}, B, Y)$ and $e = h(X, Y)$, is useless under the DLP assumption for the attacker to violate other sessions other than sessions of the tag $(\hat{A}^*, A^*, \hat{B}, B, X, Y)$ where $(\hat{A}^*, A^*)$ may be different from $(\hat{A}, A)$. As the DH-component X is generated by uncorrupted players randomly and independently, it implies that the knowledge of (A^{cy}, s) can only help the attacker to violate the security of *at most one unexposed non-matching session*.

For example, consider that the attacker interacts concurrently with $\hat{A}$ (in the name of $\hat{B}$) and $\hat{B}$ (in the name of $\hat{A}^* \neq \hat{A}$ but of the same public-key A); the attacker faithfully relays the DH-components X and Y in the two sessions; in case the attacker learns both s and the private-key a of $\hat{A}$, then it can impersonate $\hat{B}$ to $\hat{A}$ in *the unique session in which $\hat{A}$ sends X*.

We remark this weakness is at the price of supporting the advantageous post-ID computability offered by YZ (well clarified with the csYZ protocol variant). Though this weakness can be trivially remedied (by putting $\hat{A}$ into d and $\hat{B}$ into c), but at the price of sacrificing the advantage of post-ID computability. Even with this (seemingly unreasonable) weakness in the public computation model for YZ in mind, the potential damage caused is still significantly mitigated in comparison with the total insecurity of (H)MQV in such scenarios.

8.3 Resistance to KCI, and Weak PFS

Recall that the security of DHKE protocols in the CK-framework is w.r.t. an unexposed test-session $(\hat{A}, \hat{B}, X_0, Y_0)$, where $\hat{A}$ and $\hat{B}$ are uncorrupted parties (which implies both the private-keys a, b are

not exposed to the attacker) but the value Y_0 may be generated by the attacker impersonating $\hat{B}$ (in this case, the matching session does not exist). In this section, we consider the security damage caused by compromising static secret-keys of players, i.e., one or both of the secret-keys a, b of the test-session are exposed to the attacker.

Firstly, we note that if both the peer $\hat{B}$ (in the test-session) is corrupted and the value Y_0 is generated by the attacker itself, then no security can be guaranteed for the test-session within the CK-framework (as the attacker can now compute out the session-key by itself). In this section, we mainly investigate the resistance against key-compromise impersonation (KCI) attacks, and perfect forward security (PFS). Roughly speaking, a key-compromise impersonation attack is deemed successful if the attacker, knowing the private key a of a party $\hat{A}$ (which of course allows the attacker to impersonate $\hat{A}$), is able to impersonate another *different* uncorrupted party $\hat{B} \neq \hat{A}$ (for which the attacker does not know the secret-key b) to $\hat{A}$. Note that for KCI attacks, the attacker still can generate the DH-component Y_0 for the test-session (without the matching session then). The PFS property says that the leakage of the static secret-key of a party should not compromise the security of session-keys ever established by that party, and erased from memory before the leakage occurred.

Definition 8.1 (clean session [80]) *We say that a complete session of a key-exchange protocol is* clean*, if the attacker did not have access to the session's state at the time of session establishment (i.e., before the session is complete), nor it issued a session-key query against the session after completion.*

Note that, for a *clean* session at an uncorrupted party, the attacker did not issue a state-reveal query while the session was incomplete or a session-key query after completion; Moreover, the attacker was not actively controlling or impersonating the party during the session establishment (neither by making any choices on behalf of that party in that session or eavesdropping into the session's state).

Definition 8.2 *[80] We say that a KE-attacker $\mathscr{A}$ that has learned the static secret-key of $\hat{A}$ succeeds in a KCI attack against $\hat{A}$, if $\mathscr{A}$ is able to distinguish from random the session-key of the a complete session at $\hat{A}$ for which the session peer $\hat{B} \neq \hat{A}$ is uncorrupted (which implies the private-key of $\hat{B}$ is not exposed to $\mathscr{A}$) and the session and its matching session (if it exists) are clean.*

In other words, the definition says that, as long as the attacker is not actively controlling or observing the secret choices (particularly the ephemeral DH-exponent x) of the test-session, then even the knowledge of $\hat{A}$'s private-key still does not allow $\mathscr{A}$ to compromise the session-key. In particular, in such a protocol $\mathscr{A}$ cannot impersonate an uncorrupted party $\hat{B}$ to $\hat{A}$ in a way that allows $\mathscr{A}$ to learn any information about the resultant session-key [80] (even if the attacker impersonates $\hat{B}$ and generates the DH-component, say Y_0, by itself).

Proposition 8.1 *Under the GDH assumption in the random oracle model, the YZ and sYZ protocols (actually, the public-key free variants), with offline pre-computation, resist KCI attacks in the CK-framework.*

The resistance of YZ/sYZ to KCI attacks is essentially implied by the proof of Theorem 7.2 and the proof of Theorem 7.3, from the observations that: for KCI attacks the test-session is of different uncorrupted peers $\hat{A} \neq \hat{B}$, and the security of the underlying YZ-HDR/sYZ-HDR hold even if the forger learns the private-key of the uncorrupted peer (the party $\hat{A}$ here).

Weak PFS (wPFS). It is clarified in [80] that, no 2-round DHKE protocols with implicit key confirmation can fully render PFS security (the 3-round versions of HMQV and YZ/sYZ, with

explicit key-confirmation and mutual authentications, do fully provide PFS property). The work [80] formulates a weak notion of PFS, named weak PFS (wPFS), and shows that HMQV satisfies this wPFS property. Roughly speaking, wPFS property says that *if the attacker is not actively involved with the choices of X, Y at a session* (particularly if it does not get to choose or learn the DH-exponent x or y), then the resultant session-key does enjoy forward security. Formally,

Definition 8.3 *[80] A key-exchange protocol provides wPFS, if an attacker $\mathscr{A}$ cannot distinguish from random the key of any clean session $(\hat{A}, \hat{B}, X, Y)$, where Y is also generated by an uncorrupted party in a clean session, even if $\mathscr{A}$ has learned the private keys of both $\hat{A}$ and $\hat{B}$.*

Proposition 8.2 *Under the CDH assumption (rather than the stronger GDH assumption), the YZ and sYZ protocols provide wPFS property in the random oracle model.*

For establishing the wPFS property for YZ/sYZ, we do not need here to construct a YZ-HDR/sYZ-HDR forger from the attacker violating the wPFS property. Actually, we can directly reduce the loss of wPFS to the CDH assumption, from the following observations: given the knowledge of both a and b, the computation of $K_{\hat{A}}$ or $K_{\hat{B}}$ is reduced to the computation of g^{xye} from the random DH-components X, Y. Recall that, for wPFS property, we assume the attacker is not actively involved with the choice of X, Y. Then, we can simply guess the test-session, and set the DH-components as some random elements X, Y, and then reduce the ability of the attacker to violate wPFS *directly* to the CDH assumption. More details are omitted here.

8.4 Summary of the Security Comparisons Between YZ vs. (H)MQV

We remark that, though both YZ/sYZ and HMQV can be proved secure in the CK-framework, they are actually designed with fundamentally different building tools and are designed under fundamentally different guiding principles.

- YZ is designed with the self-sealed joint-POK (SSJPOK), introduced and justified in this work, *which we suggest is fundamental and of independent value.* It's also interesting to note that the design of YZ is in turn traced back to the DIKE protocol proposed in Section 4.

  The design of HMQV is based on exponential challenge-response (XCR) signatures [80], and the design of MQV is based on implicit signatures [100].

- YZ is designed for the robust TBRNM security goal against powerful secrecy exposure, with the philosophy of achieving security as robust as possible while relying on secrecy as minimal as possible. Specifically, a party completing a session (of distinct tag) has the guarantee that, as long as the session is not internally-exposed, its peer (whether honest or malicious) has to "know" both the DH-exponent and secret-key (corresponding to the DH-component and public-key sent and alleged by the peer for that session).

  Note that, as demonstrated in Section 5.3.3, (H)MQV does not achieve so robust security guarantee in general, even without any secrecy exposure. HMQV was analyzed according to the SK-security in the CK-framework [33], while MQV still lacks provable security in the CK-framework.

Arguably, we suggest the underlying building tool and design principle of (s)YZ may be more sound and robust than those of (H)MQV. Below, we present brief summary about the security comparisons between YZ/sYZ vs. MQV/HMQV.

SK-security in the CK-framework. For provable security in the CK-framework, we have the following observations:

- SK-security against test-session of different peers. In this case, the security of YZ/sYZ protocols, with pre-computed and exposable DH-component, DH-exponent and (A^{cy}, B^{dx}), is based on only the GDH assumption and dispenses with the KEA assumption.

  The SK-security of HMQV in this case, with only pre-computed and exposable DH-component (*even if the pre-computed DH-exponent is unexposable to the attacker*), is based on both the GDH assumption and the KEA assumption.
- SK-security against test-session of identical peers and identical DH-components. In this case, the SK-security of both YZ/sYZ and HMQV is based on the traditional CDH assumption.
- SK-security against test-session of identical peers but different DH-components. In this case, the SK-security of both YZ/sYZ and HMQV is based on both the GDH assumption and the KEA assumption.
- In general, the security reduction (to the underlying assumptions) of YZ/sYZ can be tighter than that of (H)MQV.
- Assuming each party only generates fresh ephemeral DH-components *on the fly* in each session (i.e., without pre-computation), and assuming the ephemeral DH-exponents are unexposable to the attacker, the SK-security of HMQV can be based on the CDH assumption, while we do not know how to prove this property with YZ/sYZ. The security of YZ/sYZ in this case is still based on the GDH assumption but with a tighter security reduction.

  We remark that even if it is reasonable to assume unexposable DH-exponents, it is unreasonable or unrealistic to assume non-precomputation and non-exposure of DH-components for *all* uncorrupted parties in the system. Note that offline pre-computation of DH-components is common for applying DHKE protocols in practice. Furthermore, as the DH-components are actually public values to be exchanged in plain, they are much less protected in practice, and thus, in reality, security of DHKE should not depend upon the privacy of DH-components. Moreover, for DHKE protocols running concurrently in settings like the Internet, even one player follows the instruction of not pre-computing DH-components, it is unrealistic to require *all* uncorrupted parties to strictly adhere to the same rule. Note that the CDH-based security of HMQV essentially requires all uncorrupted parties in the system not to offline pre-compute DH-components, in the sense that the DH-components should not be exposable to the attacker prior to the sessions involving them.[17] Thus, the advantage of HMQV over YZ/sYZ in this special case is insignificant in reality. Recall that the advantage of tighter security reduction of YZ/sYZ still remains in this case.

Security beyond the CK-framework. YZ/sYZ have advantages over (H)MQV for a list of security properties beyond the CK-framework.

- Tag-based robust non-malleability against powerful secrecy exposure. YZ/sYZ essentially achieves the following robust security goal, with the philosophy of achieving security as robust as possible while relying on secrecy as minimal as possible: a party completing a session (of distinct tag) has the guarantee that, as long as the session is not internally-exposed (i.e., the attacker did not make the state-reveal query against this session), its peer (whether honest or malicious) has to "know" both the DH-exponent

[17] Suppose some uncorrupted players pre-compute DH-components that are exposed to the attacker prior to the sessions involving them, the attacker can choose the sessions run by those players as the test-sessions. In this case, the CDH-based analysis of HMQV does not work.

and secret-key (corresponding to the DH-component and public-key sent and alleged by the peer for that session).

As clarified in Section 5.3.3, (H)MQV does not achieve so robust security goal in general, even if without any secrecy exposure.

- Security with public computations. YZ and particularly sYZ provide very reasonable security with public computations, while MQV and HMQV are definitely insecure in the public computation model [82].
- Resistance to DoS attacks and collision attacks. MQV and HMQV suffer from effective DoS attacks, as demonstrated in Section 5.3.3. The DoS attack demonstrated there does not work against YZ/sYZ, particularly by the essential TBRNM security of YZ/sYZ.

  For HMQV and YZ/sYZ working on the same subgroup, e.g., $|q| = 160$ that is popular, YZ and sYZ provide more robust resistance to collision attacks against the underlying hash function h.

In view of the various other advantageous features of YZ/sYZ, discussed in Section 6.3, we may conclude that YZ/sYZ may achieve better tradeoff, than MQV/HMQV, among security, privacy, efficiency, and easy deployment.

9 Diffie-Hellman Key-Exchange in the Distributed Client-Server Setting

In this section, we investigate and discuss some (partial) applications of the YZ protocol family and its underlying building tool SSJPOK. Besides, we also investigate and improve the applications of password-based DHKE in the distributed client-server setting, e.g., for TLS/SSL. These applications are possibly of independent value and interest, particularly, to applied cryptographic engineering.

9.1 More Applications of SSJPOK

9.1.1 SSJPOK-Based Secure Shell (SSH)

The Secure Shell (SSH) is a public-key based authentication protocol suite, which enables a user to securely login onto a remote server host machine from a client machine through an insecure network, to securely execute commands in the remote host and to securely move files from one host to another. The protocol is a de facto industrial standard, and is in wide use for server machines of running UNIX and Linux operating systems and client machines of running any operating systems [96].

The SSH protocol suite consists of three protocols: (1) the SSH Transport Layer (SSH/TL) protocol [119, 120], which provides a unilateral authenticated secure channel from the server to the client; (2) The SSH User Authentication (SSH/UA) protocol, which provides unilateral authenticated secure channel from the client to the server; (3) The SSH Connection Protocol, which runs over the mutually authenticated secure channel established by the above two protocols.

The core authentication protocol of the SSH/TL protocol [119, 120] is: $\hat{A}$ (i.e., the client) sends $X = g^x$, and $\hat{B}$ (i.e., the server of public-key B) responds with $\{sid, B, Y = g^y, Sig_B(H(sid, B, X, Y, X^y))\}$, where sid is the session identifier (which is normally the concatenation of two exchanged random nonces), $Sig_B(H(sid, B, X, Y, X^y))$ denote the signature by $\hat{B}$ on message $H(sid, B, X, Y, X^y)$, and H is a cryptographic hash function (e.g., SHA-1). The session-key is derived from g^{xy}.

Suppose the signature is generated using the Digital Signature Standard (DSS), which is the common case now in practice, the computational complexity at the side of the player $\hat{A}$ (i.e., the

client) is about 3.5 exponentiations, and about 3 exponentiations at the side of the player $\hat{B}$ (i.e., the server). *Note that, besides the DH-components X and Y (and possibly a part of signature generation), the players cannot pre-compute or verify the signature and the session-key.* That is, the *on-line* computational complexity at the side of the player $\hat{A}$ is about 2.5 exponentiations, and about 1-2 exponentiations at the side of the player $\hat{B}$. For communication complexity, beside other necessary transmission, the length of the DSS signature (i.e., $Sig_B(H(sid, B, X, Y, X^y))$) is $2|q|$.

We note that our SSJPOK and sSSJPOK proposed in Section 5.2 can be directly applied to SSH/TL in this case. Specifically, let $K_{\hat{B}} = X^{dbt+eyt} = B^{dxt}Y^{ext} = K_{\hat{A}}$, where $d = h(\hat{B}, B, X)$ and $e = h(sid, X, Y)$, or, $d = 1$ and $e = h(sid, \hat{B}, B, X, Y)$. Note that, to preserve its privacy, the client $\hat{A}$ may not reveal its identity in the first-round, which is thus not included in the input of d and e (of course, if $\hat{A}$'s identity is revealed in the first round, it can be put into the input of d or e). Let (k_1, k_2) be the key pair derived from $K_{\hat{A}} = K_{\hat{B}}$ (or $(sid, K_{\hat{A}} = K_{\hat{B}})$ in case sid is not put into the input of e). Then, we can replace the second-round message of the above SSH/TL by $\{sid, B, Y = g^y, MAC_{k_2}(1)\}$, and define the session-key to be k_1.

For the computational complexity of the SSJPOK-based SSH/TL protocol, if the player $\hat{B}$ explicitly checks $X^q = 1_G$ (i.e., performs explicit subgroup test), then the computational complexity at the side of $\hat{B}$ is about 2.2 exponentiations, or just 2 exponentiations with embedded sub-group test (in this case it is recommended for $\hat{B}$ to well protect the ephemeral DH-exponent y). As Y can be pre-computed offline, the online efficiency of $\hat{B}$ is 1.2 exponentiations with explicit sub-group test or just 1 exponentiation with embedded sub-group test. For the computational complexity at the side of $\hat{A}$, it is about 2.5 exponentiations in total, but the on-line efficiency can be only 1 exponentiation (specifically, X and B^{dxt} can be computed offline). This indicates significant (online) efficiency improvements, in comparison with the DSS-based implementation of SSH/TL. In particular, the online efficiency of SSJPOK-based SSH/TL is essentially optimal. Further note that the length of $MAC_{k_2}(1)$ is $|q|$, which also improves the communication complexity of the DSS-based SSH/TL.

9.1.2 Online Efficient Signcryption

Given a DH-based public-key encryption (PKE) scheme (e.g., the widely standardized DHIES scheme [1] or PSEC scheme [57], etc), where the receiver is $\hat{B}$ of the public-key B and the sender $\hat{A}$ sends X (as a part of the ciphertext) to generate and share the DH-key B^x with $\hat{B}$ (normally B^x is in turn used to derive a symmetric encryption key and a MAC key). One application of the one-round YZ, i.e., orYZ, is to replace the DH-key B^x of the underlying DH-based PKE scheme by $K_{\hat{A}}$, which yields an online efficient signcryption variant [6].

9.2 Password-Based TLS/SSL

The Secure Sockets Layer (SSL) protocol [67, 56] is an important and widely deployed authentication protocol, mainly for World Wide Web (WWW). SSL evolves into the de facto standards for Web security until it further evolves into the Transport Layer Security (TLS) [43], which is an Internet standard for Web security developed by the industrial standardization body Internet Engineering Task Force (IETF). The TLS/SSL protocol runs under the application-layer protocols such as Hypertext Transfer Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), Internet Messaging Access Protocol (IMAP), etc, and above the TCP/IP layers. When the transport-layer communications are secured (ensured by TLS/SSL), communications in all application-layer protocols will be secured in the same manner [96].

The typical run of TLS/SSL, particularly for e-Banking and e-Commerce over Internet, is in the distributed client-server setting. In this setting, each client $\hat{A}$ pre-registers a password $pw_{\hat{A}} \in D$ at the server $\hat{B}$ who possesses a public-key, where D is a low-entropy dictionary. The server maintains

a database, where each registered user has an entry that stores the user's identity and password information.

The core authentication mechanism for the typical run of TLS/SSL in the client-server setting is as follows: Suppose r_A and r_B be two random nonces sent by $\hat{A}$ and $\hat{B}$ respectively before the following protocol run; In the first-round, $\hat{A}$ sends to $\hat{B}$ $\{\hat{A}, PKE(r), H(r, \hat{A}, r_A, r_B)\}$, where PKE is the public-key encryption scheme supported by $\hat{B}$, r is a random string and H is a hash function (e.g., SHA-1); In the second-round, $\hat{B}$ decrypts r, and sends back $\{H(r, \hat{B}, r_B, r_A)\}$, which authenticates $\hat{B}$ to $\hat{A}$; The session-key is just set to be r. To authenticate $\hat{A}$ to $\hat{B}$, $\hat{A}$ sends back $E_r(pw_{\hat{A}})$ to $\hat{B}$, where E is a symmetric encryption scheme using r as the secret-key.

Suppose the server $\hat{B}$ is of DL public-key $B = g^b$, e.g., the server uses the widely standardized PKE scheme of DHIES [1] or PSEC [57], etc, we present some combinatorial optimizations, which make the password-based TLS/SSL both more efficient and more secure in this case. The core of the combinatorial optimization is a password-based (plaintext-aware) public-key encryption scheme that is in turn based on a password-based DHKE for the client-server setting.

Password-based DHKE for the client-server setting. For presentation simplicity, we only describe the implementation with embedded subgroup test. When $\hat{A}$ registers its password $pw_{\hat{A}}$ to $\hat{B}$, $\hat{B}$ computes (B^{tbw}, B^{-tbw}), and stores $(\hat{A}, B^{tbw}, B^{-tbw})$ at the database entry corresponding to $\hat{A}$, where t is the cofactor, and $w = h_l(\hat{A}, \hat{B}, pw_{\hat{A}})$ denotes the prefix of length l of the output of $h(\hat{A}, \hat{B}, pw_{\hat{A}})$ (typically, l is small, e.g., the length of $pw_{\hat{A}}$). Note that even the values (B^{tbw}, B^{-tbw}), pre-computed and stored by the server, are compromised, they do not leak the client's password $pw_{\hat{A}}$. Specifically, as clarified in Section 7.3 for the security of sYZ, the hardness of computing out B^b from $B = g^b$ is equivalent to that of the standard CDH problem. Thus, under the CDH assumption, the possibility of computing out w (and thus $pw_{\hat{A}}$), *by the means of computing* B^b *from the public-key* B, is rule out.

$\hat{A}$ sends $\{\hat{A}, X' = g^x B^w\}$ to $\hat{B}$. Note that X' is actually a DH-based *perfectly-hiding* commitment to the value of w. This implies that w can just be set to be the plain password $pw_{\hat{A}}$, but setting $w = h_l(\hat{A}, \hat{B}, pw_{\hat{A}})$ as suggested eases the key derivation and mutual (key confirmation) authentications.[18] The shared DH-secret is computed as follows: $\hat{A}$ computes $K_{\hat{A}} = B^{xt}$. Note that all the computation of $\hat{A}$ can be off-line, and the computation of $X' = g^x B^w$ essentially accounts for one exponentiation (as w is very small); After receiving the messages from $\hat{A}$, $\hat{B}$ retrieves (B^{tbw}, B^{-tbw}) from its database, checks that $X' \in G'$, computes X'^{tb} and checks that $X'^{bt} \neq B^{bwt}$ that implies $X^t \neq 1_g$ (otherwise, $\hat{B}$ aborts), and finally sets the DH-secret to be $K_{\hat{B}} = X'^{bt} B^{-bwt}$. Note that the on-line computation of $\hat{B}$ is only one exponentiation. It is easy to check that $K_{\hat{A}} = K_{\hat{B}}$, from which the shared secret-keys will be derived.

Password-based (plaintext-aware) PKE. Given a DH-based public-key encryption (PKE) scheme, e.g., the widely standardized DHIES scheme [1] or PSEC scheme [57], etc, where the decryptor $\hat{B}$ is of public-key $B = g^b$ and the encryptor $\hat{A}$ sends X (as a part of the ciphertext) to generate and share the DH-key $B^x = X^b$ with $\hat{B}$. Normally, (X, B^x) are in turn used to derive the actual secret, denoted $KDF(X, B^x)$, to be used in the underlying PKE [1, 57]. We convert it into a password-based (plaintext-aware) PKE, using the above password-based DHKE, as follows:

- The DH-component X is changed to be $X' = g^x B^w$.
- The shared DH-secret $B^x = X^b$ is replaced by $K_{\hat{A}} = B^{xt} = X'^{bt} B^{-bwt} = K_{\hat{B}}$ computed as specified above. $(X, B^x = X^b)$ in the input of the key-derivation function KDF are replaced by $(X', B^{xt} = X'^{bt} B^{-bwt})$.

[18] In general, in case of l is small, e.g., the length of password, we suggest still to put players' identities into the inputs of key-derivation and (key-confirmation) authentication functions for extra security guarantee; but if l is large and w has already committed to players' identities, they can be waived from the inputs of the key-derivation and authentication functions.

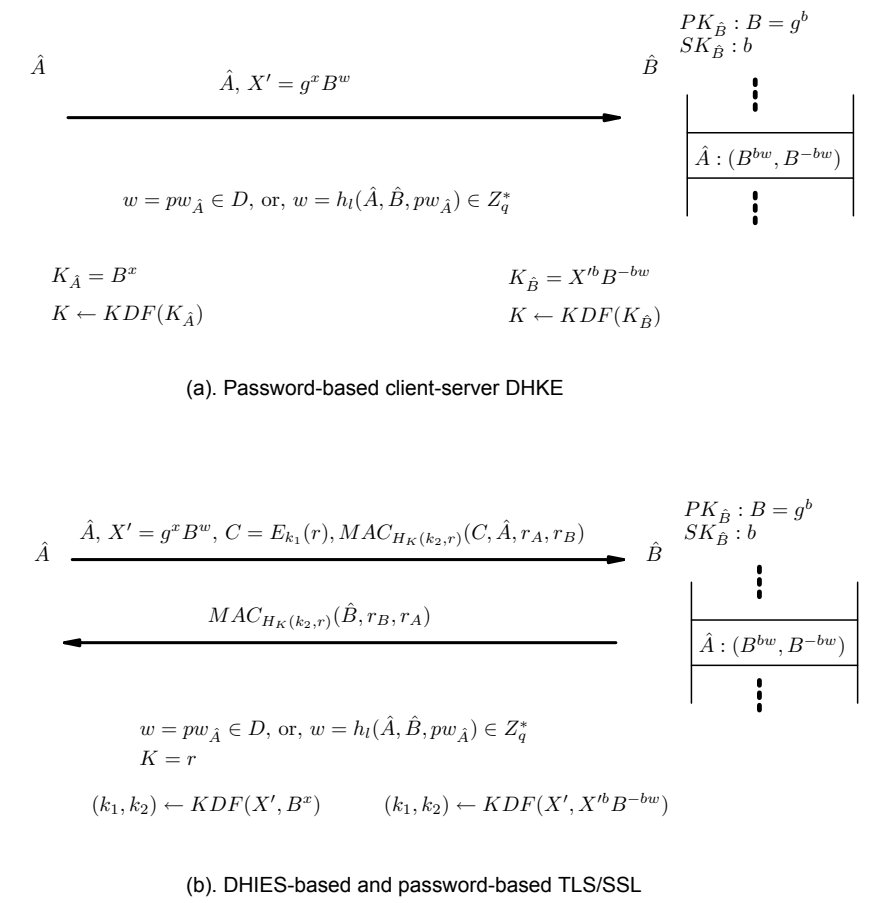


Figure 14: Password-based DHKE and TLS/SSL

- Suppose the ciphertext of a message of m in the original DH-based PKE (e.g., the DHIES and PSEC) consists of $C = E_{k_1}(m)$ and $MAC_{k_2}(C)$, we can add plaintext awareness by changing $MAC_{k_2}(C)$ to be $MAC_{H_K(k_2,m)}(C)$.

The resultant PKE scheme by the above modifications is denoted by $PWPKE$. We remark that the above modifications do not weaken the security of the original PKE.

Improved password-based TLS/SSL. In the first-round, $\hat{A}$ sends to $\hat{B}$ $\{\hat{A}, PWPKE(r)\}$. Here, we can put (r_A, r_B) into the input of the MAC included in the ciphertext of $PWPKE(r)$. Specifically, $MAC_{H_K(k_2,r)}(C)$ is set to be $MAC_{H_K(k_2,r)}(C, \hat{A}, r_A, r_B)$. In the second-round, $\hat{B}$ decrypts r, and sends back $MAC_{H_K(k_2,r)}(\hat{B}, r_B, r_A)$. The session-key is still set to be r.

The following is a detailed instantiation based on the widely-standardized DHIES scheme [1] (the instantiation based on PSEC of [57] can also be derived similarly). In the first-round, $\hat{A}$ sends to $\hat{B}$ $\{\hat{A}, X' = g^x B^w, C = E_{k_1}(r), MAC_{H_K(k_2,r)}(C, \hat{A}, r_A, r_B)\}$, where $(k_1, k_2) \leftarrow KDF(X', B^{tx})$. After receiving the messages from $\hat{A}$, $\hat{B}$ does the following: it retrieves (B^{tbw}, B^{-tbw}) from its database, checks that $X' \in G'$, computes X'^{tb} and checks that $X'^{bt} \neq B^{bwt}$ (otherwise, $\hat{B}$ aborts), computes $K_{\hat{B}} = X'^{bt} B^{-bwt}$, and derives the keys $(k_1, k_2) \leftarrow KDF(X', K_{\hat{B}})$; Then, $\hat{B}$ decrypts C to get r, and checks the validity of $MAC_{H_K(k_2,r)}(C, \hat{A}, r_A, r_B)$ (if not $\hat{B}$ aborts). If all the above checks are successful, $\hat{B}$ computes and returns back $MAC_{H_K(k_2,r)}(\hat{B}, r_B, r_A)$ in the second-round.

We remark that all the security of the original password-based TLS/SSL is preserved with our above improved password-based TLS/SSL (needless to say, the TLS/SSL protocol is not of

full-fledged security, e.g., the PFS and KCI vulnerabilities). But, our improved password-based TLS/SSL enjoys the following remarkable advantages, which might make it more attractive and desirable in practice (particularly for e-Banking and e-Commerce over Internet that are the main applications of TLS/SSL):

- Clients' passwords are well protected at the side of the server, even if the database of the server is compromised.
- Ruling out offline attacks, *even if the session-key is compromised.*

  Specifically, the first-round message perfectly hides $pw_{\hat{A}}$ and the second-round message is independent of $pw_{\hat{A}}$, which rules out any potential off-line attacks. Moreover, the offline attacks are impossible even if the session-key r is compromised. Note that, with the original password-based TLS/SSL, the password can always be derived from $E_r(pw)$ in case the session-key r is compromised. We remark that, as the session-key can be maintained by the client and the server for several hours in practice, it is indeed more susceptible to adversarial exposure than other ephemeral private values (e.g., the freshly generated DH-exponent x that may be maintained only for several seconds).
- Improved (communication) efficiency. Specifically, the round complexity is reduced from 3 to 2, and the value $H(r, \hat{A}, r_A, r_B)$ sent in the first-round of the original TLS/SSL is also removed. The computational complexity remains essentially the same as that of the original password-based TLS/SSL.

9.3 Wireless Group Key-Exchange for IEEE 802.11

Bresson, et al [17] proposed a very efficient group key-exchange protocol, referred to as the BCEP-protocol, for a group of low-power mobile devices (clients) $\{\hat{A}_1, \cdots, \hat{A}_n\}$ interacting with a wireless gateway (the server or base-station) $\hat{B}$. The BCEP-protocol was aimed for fulfilling the key-exchange mechanism behind the Wired Equivalent Privacy (WEP) protocol that is part of the IEEE 802.11 standard. In the BCEP-protocol, the server $\hat{B}$ is of public-key $B = g^b$, and each client $\hat{A}_i$ is of public-key $A_i = g^{a_i}$. The core key-exchange building block of the BCEP-protocol is as follows: each $\hat{A}_i$ generates $X_i = g^{x_i}$, computes $K_i = B^{x_i}$, generates a signature σ_i with its signing-key a_i on K_i, and finally sends (K_i, σ_i) to $\hat{B}$.

Our one-round YZ (orYZ) protocol can be directly applied here to significantly improve the efficiency of the BCEP-protocol. Specifically, we replace (K_i, σ_i) by $B^{a_i + e_i x_i} = A_i^b X_i^{e_i b}$ (other components of BCEP can remain unchanged), where $e_i = h(\hat{A}_i, A_i, \hat{B}, B, X_i)$. In comparison with the standard DSS-based implementation of BCEP, the orYZ-based implementation has the following remarkable advantages, which are critical for deploying group KE in the stringent wireless setting (of *low-power* mobile device and *centralized* base-stations):

- The computational complexity of each $\hat{A}_i$ is reduced from 3 exponentiations to 2 exponentiations. Note also that all the computation of $\hat{A}_i$ can be pre-computed offline.
- Ignoring the complexity of sub-group test of DH-components (for both the original protocol of [17] and the orYZ-based implementation), the computational complexity of $\hat{B}$ can be reduced from $n \times 2.5$ exponentiations to $n \times 1.5$ exponentiations in total. But, if $\hat{B}$ pre-computes and stores A_i^b, $1 \leq i \leq n$, the on-line efficiency of $\hat{B}$ is reduced from $n \times 2.5$ to n! Note that, in the BCEP-protocol [17], $\hat{B}$ cannot pre-compute K_i and pre-verifies the signature σ_i, which takes 2.5 exponentiations for each (K_i, σ_i), $1 \leq i \leq n$.
- By waiving signatures, the communication complexity is reduced from $n \times (|p| + 2|q|)$ to $n \times |p|$. Note that the length of each (K_i, σ_i) is $|p| + 2|q|$.

Recall that orYZ intrinsically suffers from the KCI vulnerability, under the (whether reasonable or not) assumption that the server's static secret-key b is compromised. However, in case $\hat{B}$'s secret-key b is compromised, all group session-keys of the BCEP protocol have already been compromised. We remark that, in comparison with the original BCEP-protocol, the orYZ-based implementation does not incur essential security negatives, but gains significant efficiency advantages (or seemingly much better tradeoff between security and efficiency) in the stringent wireless setting.

Another way to improve the BCEP-protocol is to generate the shared DH-secret K_i by running the password-based DHKE for the client-server setting proposed in Section 9.2. In this case, the clients do not possess public-keys any longer, but register passwords at the server (which are well-protected even if the server's database is compromised). This solution has the following additional advantages:

- The computational complexity of the server is further reduced to only n exponentiations in total.
- It should be easier for the server and the clients to maintain and manage passwords (rather than public-keys). In particular, note that clients' passwords are well-protected even if the server's database is compromised.
- The KCI vulnerability of orYZ-based solution is also ruled out.

10 Formulation of Non-Malleable Extractable (Joint) OWF (NME-JOWF)

We formulate four notions of NME-OWF, *self*-non-malleable extractable joint OWF (SNME-JOWF), resettable SNME-JOWF (RSNME-JOWF), universal non-malleable extractable joint OWF (UNME-JOWF) and resettable UNME-JOWF (RUNME-JOWF).

10.1 Formulating Self-Non-Malleable Extractable Joint One-Way Function

On a security parameter n, consider a polynomial-time (in n) computable *deterministic* function $f : \{0,1\}^* \times (\mathcal{D})^t \rightarrow \{0,1\}^* \times \mathcal{R}$ of the form $f(m, x_1, \cdots, x_t) = (m, f'(m, x_1, \cdots, x_t))$, where $m \in \{0,1\}^*$,[19] $\mathcal{D}$ and $\mathcal{R}$ are finite sets, t is a positive polynomial in n, $x_i \in \mathcal{D} \cap \{0,1\}^n$ for each i, $1 \leq i \leq t$, $f' : \{0,1\}^* \times (\mathcal{D})^t \rightarrow \mathcal{R}$ is another polynomial-time computable deterministic function.

Note: The input m is *public*, in the sense it is copied into the output. In general, the inputs of f can contain a set of (rather than only one) public inputs. If m is the empty string, then f and f' become identical; If m is the empty string and $t = 1$, then f is degenerated to the normal function of one input.

The function f is called *preimage verifiable*, if there exists an efficient, i.e., probabilistic polynomial-time (PPT), algorithm $\mathcal{V}$ such that, for sufficiently large n's and any element $(m, \delta) \in \{0,1\}^* \times \mathcal{R}$, $\mathcal{V}(1^n, m, \delta) = 1$ if and only if there exist $(x_1, \cdots, x_t) \in (\mathcal{D} \cap \{0,1\}^n)^t$ such that $(m, \delta) = f(m, x_1, \cdots, x_t)$.

The function f is called *joint one-way* (JOW) if for any PPT algorithm $\mathcal{A}$ there exists a negligible function $\nu_{\mathcal{A}}$ such that, for any $t \geq 1$ and any $i, 1 \leq i \leq t$ and sufficiently large n's, it holds that: $\Pr[y = f(m, x_1, \cdots, x_{i-1}, \hat{x}, x_{i+1}, \cdots, x_t) : \mathcal{A}(1^n) = m; (x_1, \cdots, x_t) \leftarrow (\mathcal{D} \cap \{0,1\}^n)^t; y = f(m, x_1, \cdots, x_t); \mathcal{A}(1^n, y, x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_t) = \hat{x}] \leq \mu_{\mathcal{A}}(n)$. That is, given the output of $f(m, x_1, \cdots, x_t)$, where each x_i is taken uniformly at random from $\mathcal{D} \cap \{0,1\}^n$ and m is set by the adversary, no PPT algorithm $\mathcal{A}$ can compute out x_i for any i, $1 \leq i \leq t$, with non-negligible probability, even if $\mathcal{A}$ is given all the other $t-1$ preimages $(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_t)$.

[19] Particularly, m could be an empty string. In general, m could be a set of public values.

Note: Joint one-way function (JOWF) is stronger than the traditional OWF. In particular, when m is the empty string and $t = 1$, JOWF is degenerated to the traditional OWF.

The function f is called *resettably joint one-way* (RJOW), if for every PPT adversary $\mathcal{A}$ there exists a negligible function $\nu_{\mathcal{A}}$ such that, for any $t \geq 1$ and any $i, 1 \leq i \leq t$ and sufficiently large n's, it holds that: $\Pr[y = f(m, x_1, \cdots, x_{i-1}, \hat{x}, x_{i+1}, \cdots, x_t) : (x_1, \cdots, x_t) \leftarrow (\mathcal{D} \cap \{0,1\}^n)^t; \mathcal{A}^{f(\cdot, x_1, \cdots, x_t)}(1^n, x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_t) = \hat{x}] \leq \mu_{\mathcal{A}}(n)$, where $\mathcal{A}^{f(\cdot, x_1, \cdots, x_t)}$ denotes that $\mathcal{A}$ gets oracle access to $f(\cdot, x_1, \cdots, x_t)$ in the following way: $\mathcal{A}$ queries m_i and gets back $\delta_i = f(m_i, x_1, \cdots, x_t)$, $1 \leq i \leq q$ where q is polynomial in n. That is, even if getting polynomially many $\delta_1 = f(m_1, x_1, \cdots, x_t), \cdots, \delta_q = f(m_q, x_1, \cdots, x_t)$ *w.r.t. the same fixed* $(x_1, \cdots, x_t)$, where $m_1, \cdots, m_q$ are taken adaptively by $\mathcal{A}$, still no efficient algorithm $\mathcal{A}$ can output x_i for any i, $1 \leq i \leq t$ with non-negligible probability, even if $\mathcal{A}$ knows all the other $t-1$ preimages $(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_t)$. Clearly, resettably jointly one-way function (RJOWF) is further stronger than JOWF.

The function f is called *non-uniformly (resettably) joint one-way*, if the (resettably) joint one-wayness property holds also for non-uniform polynomial-time algorithms (equivalently, families of polynomial-size circuits).

Denote by $(m, \delta) \leftarrow \mathcal{A}^f(1^n, z)$ the procedure that: on input $(1^n, z)$ where $z \in \{0,1\}^*$ is some arbitrary (auxiliary) information, $\mathcal{A}$ gets oracle access to the function f in the following way: when $\mathcal{A}$ queries f with m_i, $1 \leq i \leq q$ where q is polynomial in n, a vector of values $(x_1^{(i)}, \cdots, x_t^{(i)})$ are chosen uniformly at random from $(\mathcal{D} \cap \{0,1\}^n)^t$, and then the value $f(m_i, x_1^{(i)}, \cdots, x_t^{(i)}) = (m_i, \delta_i)$ is returned back to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs (m, δ). Denote by $\mathcal{Q} = \{(m_1, \delta_1), \cdots, (m_q, \delta_q)\}$ the set of oracle answers got by $\mathcal{A}$.

The function f is called *self-non-malleable* and *extractable*, if

1. It is *preimage verifiable* and *joint one-way*.

2. For any efficient (i.e., PPT) algorithm $\mathcal{A}$ there exists an efficient (i.e., PPT) algorithm $\mathcal{E}$ such that for arbitrary $z \in \{0,1\}^*$ and sufficiently large n, it holds that: if $\mathcal{A}^f(1^n, z)$ outputs (m, δ), satisfying $(m, \delta) \notin \mathcal{Q}$ and $\mathcal{V}(1^n, m, \delta) = 1$ and z is *independent* of $\hat{m}$ that is a substring of m,[20] then with overwhelming probability $\mathcal{E}(1^n, z, \mathcal{Q}, m, \delta, \rho)$ outputs $(x_1, \cdots, x_t) \in (\mathcal{D} \cap \{0,1\}^n)^t$ satisfying $(m, \delta) = f(m, x_1, \cdots, x_t)$, where ρ is the random coins used by $\mathcal{A}$ to generate (m, δ).

Extractable function essentially says that: given a list of the outputs of f, i.e., $\mathcal{Q}$ adaptively obtained by $\mathcal{A}$, and arbitrary auxiliary input z, an efficient algorithm $\mathcal{A}$ can output a *new valid* output (m, δ) of f, where m is independent of z, only if $\mathcal{A}$ does "*know*" the corresponding pre-images in the sense that such values can be efficiently extracted in a *non-black-box* way.

Notes: Note that, by joint one-wayness property, we have that the output of E, i.e., $\{x_1, \cdots, x_t\}$ has no intersection with $\{x_1^{(i)}, \cdots, x_{t_i}^{(i)}\}$ for any $i, 1 \leq i \leq q$. Also note that the preimage-extractor E works in strict polynomial-time (rather than expected polynomial-time).

On requiring $\hat{m}$ to be independent of z. The reason is, as noted in [27, 28], if the output of $\mathcal{A}$, i.e., (m, δ), is arbitrarily dependent of z, then $\mathcal{A}$ does not necessarily know the corresponding pre-images of (m, δ) in general and thus knowledge extraction may fail in such cases. But, our independence requirement is very weak, and accordingly knowledge-extraction property ensured is very strong. Specifically, we only require m includes a substring $\hat{m}$ that is independent of z, but m and $\hat{m}$ can still be dependent on $\mathcal{Q}$ (particularly, $m = m^{(i)}$ for some i, $1 \leq i \leq l(n)$) and δ may even be dependent on both $\mathcal{Q}$ and z. Moreover, z can be arbitrarily dependent upon the description of the (single) function f. Note that if the adversary takes no auxiliary input, then the independency requirement can be trivially achieved (waived). Also, this weaker independence requirement eases

[20] In general, if we view m as the encoding of a set, then m at least contains a value $\hat{m}$ that is independent of z.

the modular use of SNME-OWF in achieving more complex interactive systems, where m can just be a random challenge.

10.2 Formulating Resettable Self-Non-Malleable Extractable Joint OWF

Denote by $(m, \delta) \leftarrow \mathcal{A}^{f(\cdot, \bar{x}_1, \cdots, \bar{x}_{\bar{t}})}(1^n, z)$, where $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}}) \leftarrow (\mathcal{D} \cap \{0,1\}^n)^{\bar{t}}$, $\bar{t} \geq 1$ and $z \in \{0,1\}^*$ is some arbitrary (auxiliary) information that is *independent* of $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}})$, the procedure that: on input $(1^n, z)$ $\mathcal{A}$ gets oracle access to the function $f(\cdot, \bar{x}_1, \cdots, \bar{x}_t)$, w.r.t. the fixed $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}})$, in the following way: $\mathcal{A}$ queries m_i, $1 \leq i \leq q$ where q is polynomial in n, and gets back the value $\delta_i = f(m_i, \bar{x}_1, \cdots, \bar{x}_{\bar{t}}) = (m_i, \delta_i)$. Finally, $\mathcal{A}$ outputs (m, δ). Denote by $\mathcal{Q} = \{(m_1, \delta_1), \cdots, (m_q, \delta_q)\}$ the set of oracle answers got by $\mathcal{A}$.

The function f is called *RSNME-OWF*, if

1. It is *preimage verifiable* and *resettably joint one-way.*
2. For any efficient algorithm $\mathcal{A}$ there exists an efficient algorithm $\mathcal{E}$ such that for sufficiently large n's and any vector of values $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}}) \leftarrow (\mathcal{D} \cap \{0,1\}^n)^{\bar{t}}$ and arbitrary $z \in \{0,1\}^*$ that is *independent* of $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}})$, it holds that: if $\mathcal{A}^{f(\cdot, \bar{x}_1, \cdots, \bar{x}_{\bar{t}})}(1^n, z)$ outputs (m, δ), satisfying $(m, \delta) \notin \mathcal{Q}$ and $\mathcal{V}(1^n, m, \delta) = 1$ and z is *independent* of $\hat{m}$ that is a substring of m, then with overwhelming probability $\mathcal{E}(1^n, z, \mathcal{Q}, m, \delta, \rho)$ outputs $(x_1, \cdots, x_t) \in (\mathcal{D} \cap \{0,1\}^n)^t$ satisfying $(m, \delta) = f(m, x_1, \cdots, x_t)$, where ρ is the random coins used by $\mathcal{A}$.

Again, by the resettably joint one-wayness, the preiamges $(x_1, \cdots, x_t)$ extracted have no intersection with $(\bar{x}_1, \cdots, \bar{x}_{\bar{t}})$.

10.3 Formulating Collections of (Resettable) Non-Malleable Extractable JOWF

We consider a collection of functions $F = \{f_{i,t_i} : \{0,1\}^* \times (\mathcal{D}_i)^{t_i} \rightarrow \{0,1\}^* \times \mathcal{R}_i\}_{i \in \mathcal{I}, t_i \in \mathcal{T}}$, where $\mathcal{I}$ and $\mathcal{T}$ are a polynomial-time samplable index set. We first (re)specify the following procedures:

Denote by $(t, m, \delta) \leftarrow \mathcal{A}^{f_i}(1^n, z)$ the procedure that: on input $(1^n, z)$ where $z \in \{0,1\}^*$ is some arbitrary auxiliary information, $\mathcal{A}$ gets oracle access to the function f_i in the following way: when $\mathcal{A}$ queries f_i with (t_{i_k}, m_k), where $1 \leq k \leq q$ and t_{i_k} and q are polynomials in n, a vector of values $(x_1^{(k)}, \cdots, x_{t_{i_k}}^{(k)})$ are chosen uniformly at random from $(\mathcal{D}_i \cap \{0,1\}^n)^{t_{i_k}}$, and then the value $f_{i,t_{i_k}}(m_k, x_1^{(k)}, \cdots, x_{t_{i_k}}^{(k)}) = (m_k, \delta_k)$ is returned back to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs (t, m, δ), where t may not be in $\{t_{i_1}, \cdots, t_{i_q}\}$. Let $\mathcal{Q} = \{(t_{i_1}, m_1, \delta_1), \cdots, (t_{i_q}, m_q, \delta_q)\}$. In general, $\mathcal{A}$ can have oracle access to multiple (polynomially many) functions (rather than only a single function f_i). The current presentation is only for presentation simplicity.

Denote by $(m, \delta) \leftarrow \mathcal{A}^{f_{i,t_i}(\cdot, \bar{x}_1, \cdots, \bar{x}_{t_i})}(1^n, z)$, where $(\bar{x}_1, \cdots, \bar{x}_{t_i}) \leftarrow (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ and $z \in \{0,1\}^*$ is some arbitrary (auxiliary) information that is *independent* of $(\bar{x}_1, \cdots, \bar{x}_{t_i})$, the procedure that: on input $(1^n, z)$ $\mathcal{A}$ gets oracle access to the function $f_{i,t_i}(\cdot, \bar{x}_1, \cdots, \bar{x}_{t_i})$, w.r.t. the fixed $(\bar{x}_1, \cdots, \bar{x}_{t_i})$, in the following way: $\mathcal{A}$ queries m_k, $1 \leq k \leq q$ where q is polynomial in n, and gets back the value $f_{i,t_i}(m_k, \bar{x}_1, \cdots, \bar{x}_{t_i}) = (m_k, \delta_k)$. Finally, $\mathcal{A}$ outputs (m, δ). Let $\mathcal{Q} = \{(m_1, \delta_1), \cdots, (m_q, \delta_q)\}$. Note $\mathcal{A}$ gets oracle access to f_{i,t_i} for a fixed t_i. Here, in general, $\mathcal{A}$ can have oracle access to multiple functions (w.r.t. multiple input parameters). The current presentation is only for presentation simplicity.

Definition 10.1 (Collection of (resettable) NME-JOWF) *A collection of (resettable) UNME-JOWF is a set $F = \{f_{i,t_i} : \{0,1\}^* \times (\mathcal{D}_i)^{t_i} \rightarrow \{0,1\}^* \times \mathcal{R}_i\}_{i \in \mathcal{I}, t_i \in \mathcal{T}}$, where $\mathcal{I}$ is a function index set, $\mathcal{T}$ is input number index set, $\mathcal{D}_i$ and $\mathcal{R}_i$ are finite sets (specified by the index i and t_i) and t_i is polynomial in $|i|$, satisfying the following conditions:*

- *There exists a PPT algorithm S_1 which, on input security parameter 1^n, outputs an index $i \in \mathcal{I} \cap \{0,1\}^{poly(n)}$.*
- *There exists a PPT algorithm S_2 which, on input 1^n and $i \in \mathcal{I}$ and any t_i that is polynomial in n, outputs $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$.*
- *There exists a* deterministic *polynomial-time algorithm A_1 which, on input 1^n and $i \in \mathcal{I}$ and $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ and $m \in \{0,1\}^*$, outputs $A_1(i, m, x_1, \cdots, x_{t_i}) = f_i(m, x_1, \cdots, x_{t_i}) = f_i(m, f'_i(m, x_1, \cdots, x_{t_i})) = (m, \delta)$.*
- *For any $i \in \mathcal{I}$ and $t_i \in \mathcal{T}$, f_{i,t_i} is preimage verifiable. That is, there exists an PPT algorithm $\mathcal{V}$ such that, for sufficiently large n's, any $i \in \mathcal{I}, t_i \in \mathcal{T}$ and any $(m, \delta) \in \{0,1\}^* \times \mathcal{R}_i$, $\mathcal{V}(1^n, i, t_i, m, \delta) = 1$ if and only if there exist $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ such that $(m, \delta) = f_{i,t_i}(m, x_1, \cdots, x_{t_i})$.*
- *For any $i \in \mathcal{I}$ and $t_i \in \mathcal{T}$, f_{i,t_i} is (resettably) joint one-way.*
- *(Resettable) non-malleable extractability (w.r.t. the fixed function f_i). For any PPT algorithm $\mathcal{A}$ there exists a PPT algorithm $\mathcal{E}$ such that, for sufficiently large n, any string $z \in \{0,1\}^*$, any q that is polynomial in n and any $i \leftarrow S_1(1^n)$ (resp., any $i \leftarrow S_1(1^n)$ and any t_i that is polynomial in n), it holds that: if $(t, m, \delta) \leftarrow \mathcal{A}^{f_i}(1^n, z)$ (resp., $(m, \delta) \leftarrow \mathcal{A}^{f_{i,t_i}(\cdot, \bar{x}_1, \cdots, \bar{x}_{t_i})}(1^n, z)$), satisfying $(t, m, \delta) \notin \mathcal{Q}$ (resp., $(m, \delta) \notin \mathcal{Q}$) and $\mathcal{V}(1^n, i, t, m, \delta) = 1$ (resp., $\mathcal{V}(1^n, i, t_i, m, \delta) = 1$) and z is* independent *of $\hat{m}$ that is a substring of m, then with overwhelming probability $\mathcal{E}(1^n, z, i, t, m, \delta, \mathcal{Q}, \rho)$ (resp., $\mathcal{E}(1^n, z, i, t_i, m, \delta, \mathcal{Q}, \rho)$) outputs $(x_1, \cdots, x_t) \in (\mathcal{D}_i \cap \{0,1\}^n)^t$ (resp., $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$) satisfying $(m, \delta) = f_{i,t}(m, x_1, \cdots, x_t)$ (resp., $(m, \delta) = f_{i,t_i}(m, x_1, \cdots, x_{t_i})$), where ρ is the random coins used by $\mathcal{A}$.*

10.4 Formulating (Resettable) Universal Non-Malleable Extractable JOWF

Motivation for UNME-OWF: The formulation of SNME-OWF is w.r.t. a single function f. When employing a *single* (not necessarily non-malleable and extractable) function in achieving more complex systems, it is implicitly assumed that the parameters of this single function is known to all players in the systems in an authentic way (before the complex system in question takes place). Typically, all the players run some interactive negotiation and authentication processes to ensure each player getting an authentic copy of these parameters, or rely upon some trust mechanisms to distribute function parameters. However, ensuring all players in the system to obtain an authentic copy of function parameters can be non-trivial, particularly when facing MIM in open networks like the Internet. It is more desirable to achieve non-malleable systems without parameter negotiation or distribution prior to each session run, where, in particular, MIM adversaries can adaptively select functions (on the right) as it wishes (as long as the function chosen by MIM on the right belongs to a well-described function collection).

Now, we consider a collection of functions $F = \{f_{i,t_i} : \{0,1\}^* \times (\mathcal{D}_i)^{t_i} \to \{0,1\}^* \times \mathcal{R}_i\}_{i \in \mathcal{I}, t_i \in \mathcal{T}}$, where $\mathcal{I}$ and $\mathcal{T}$ are a polynomial-time samplable index set. We first (re)specify the following procedures:

Denote by $(i', t_{i'}, m, \delta) \leftarrow \mathcal{A}^{f_i}(1^n, z)$ the procedure that: on input $(1^n, z)$ where $z \in \{0,1\}^*$ is some arbitrary auxiliary information, $\mathcal{A}$ gets oracle access to the function f_i in the following way: when $\mathcal{A}$ queries f_i with (t_{i_k}, m_k), where $1 \le k \le q$ and t_{i_k} and q are polynomials in n, a vector of values $(x_1^{(k)}, \cdots, x_{t_{i_k}}^{(k)})$ are chosen uniformly at random from $(\mathcal{D}_i \cap \{0,1\}^n)^{t_{i_k}}$, and then the value $f_{i,t_{i_k}}(m_k, x_1^{(k)}, \cdots, x_{t_{i_k}}^{(k)}) = (m_k, \delta_k)$ is returned back to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(i', t_{i'}, m, \delta)$, where $(i', t_{i'})$ may not be in $\{(i, t_{i_1}), \cdots, (i, t_{i_q})\}$. Let $\mathcal{Q} = \{(i, t_{i_1}, m_1, \delta_1), \cdots, (i, t_{i_q}, m_q, \delta_q)\}$. In general,

$\mathcal{A}$ can have oracle access to multiple (polynomially many) functions (rather than only a single function f_i). The current presentation is only for presentation simplicity.

Denote by $(i', t_{i'}, m, \delta) \leftarrow \mathcal{A}^{f_{i,t_i}(\cdot,\bar{x}_1,\cdots,\bar{x}_{t_i})}(1^n, z)$, where $(\bar{x}_1, \cdots, \bar{x}_{t_i}) \leftarrow (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ and $z \in \{0,1\}^*$ is some arbitrary (auxiliary) information that is *independent* of $(\bar{x}_1, \cdots, \bar{x}_{t_i})$, the procedure that: on input $(1^n, z)$ $\mathcal{A}$ gets oracle access to the function $f_{i,t_i}(\cdot, \bar{x}_1, \cdots, \bar{x}_{t_i})$, w.r.t. the fixed $(\bar{x}_1, \cdots, \bar{x}_{t_i})$, in the following way: $\mathcal{A}$ queries m_k, $1 \leq k \leq q$ where q is polynomial in n, and gets back the value $f_{i,t_i}(m_k, \bar{x}_1, \cdots, \bar{x}_{t_i}) = (m_k, \delta_k)$. Finally, $\mathcal{A}$ outputs $(i', t_{i'}, m, \delta)$. Let $\mathcal{Q} = \{(i, t_i, m_1, \delta_1), \cdots, (i, t_i, m_q, \delta_q)\}$. Note $\mathcal{A}$ gets oracle access to f_{i,t_i} for a fixed t_i. Here, in general, $\mathcal{A}$ can have oracle access to multiple functions (w.r.t. multiple input parameters). The current presentation is only for presentation simplicity.

Definition 10.2 (Collection of (resettable) UNME-JOWF) *A collection of (resettable) UNME-JOWF is a set $F = \{f_{i,t_i} : \{0,1\}^* \times (\mathcal{D}_i)^{t_i} \to \{0,1\}^* \times \mathcal{R}_i\}_{i \in \mathcal{I}, t_i \in \mathcal{T}}$, where $\mathcal{I}$ is a function index set, $\mathcal{T}$ is input number index set, $\mathcal{D}_i$ and $\mathcal{R}_i$ are finite sets (specified by the index i and t_i) and t_i is polynomial in $|i|$, satisfying the following conditions:*

- *There exists a PPT algorithm S_1 which, on input security parameter 1^n, outputs an index $i \in \mathcal{I} \cap \{0,1\}^{poly(n)}$.*
- *There exists a PPT algorithm S_2 which, on input 1^n and $i \in \mathcal{I}$ and any t_i that is polynomial in n, outputs $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$.*
- *There exists a* deterministic *polynomial-time algorithm A_1 which, on input 1^n and $i \in \mathcal{I}$ and $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ and $m \in \{0,1\}^*$, outputs $A_1(i, m, x_1, \cdots, x_{t_i}) = f_i(m, x_1, \cdots, x_{t_i}) = f_i(m, f'_i(m, x_1, \cdots, x_{t_i})) = (m, \delta)$.*
- *For any $i \in \mathcal{I}$ and $t_i \in \mathcal{T}$, f_{i,t_i} is preimage verifiable. That is, there exists an PPT algorithm $\mathcal{V}$ such that, for sufficiently large n's, any $i \in \mathcal{I}, t_i \in \mathcal{T}$ and any $(m, \delta) \in \{0,1\}^* \times \mathcal{R}_i$, $\mathcal{V}(1^n, i, t_i, m, \delta) = 1$ if and only if there exist $(x_1, \cdots, x_{t_i}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_i}$ such that $(m, \delta) = f_{i,t_i}(m, x_1, \cdots, x_{t_i})$.*
- *For any $i \in \mathcal{I}$ and $t_i \in \mathcal{T}$, f_{i,t_i} is (resettably) joint one-way.*
- *(Resettable) universal non-malleable extractability (w.r.t. any function $f_{i',t_{i'}}$ chosen by the adversary itself). For any PPT algorithm $\mathcal{A}$ there exists a PPT algorithm $\mathcal{E}$ such that, for sufficiently large n, any string $z \in \{0,1\}^*$, any q that is polynomial in n and any $i \leftarrow S_1(1^n)$ (resp., any $i \leftarrow S_1(1^n)$ and any t_i that is polynomial in n), it holds that: if $(i', t_{i'}, m, \delta) \leftarrow \mathcal{A}^{f_i}(1^n, z)$ (resp., $(i', t_{i'}, m, \delta) \leftarrow \mathcal{A}^{f_{i,t_i}(\cdot,\bar{x}_1,\cdots,\bar{x}_{t_i})}(1^n, z)$), satisfying $(i', t_{i'}, m, \delta) \notin \mathcal{Q}$ and $\mathcal{V}(1^n, i', t_{i'}, m, \delta) = 1$ and z is* independent *of $\hat{m}$ that is a substring of m, then with overwhelming probability $\mathcal{E}(1^n, z, i', t_{i'}, m, \delta, \mathcal{Q}, \rho)$ outputs $(x_1, \cdots, x_{t_{i'}}) \in (\mathcal{D}_i \cap \{0,1\}^n)^{t_{i'}}$ satisfying $(m, \delta) = f_{i',t_{i'}}(m, x_1, \cdots, x_{t_{i'}}))$, where ρ is the random coins used by $\mathcal{A}$.*

Note: The universal non-malleable extraction w.r.t. function collection is much stronger, in the sense that extractability holds even for arbitrary function chosen by the adversary itself.

10.5 Comparisons with Related Notions

10.5.1 Brief Comparisons with Extractable Functions

We briefly point out some main difference between our non-malleable extractable functions and the extractable functions considered in [27, 28].

- The extractable OWFs considered in [27, 28] are not necessarily non-malleable. For example, consider the extractable perfect OWF proposed in [27] based upon the KEA assumption: $F_{p,g,g,g^a}(x,r) = (g^r, g^{ar}, g^{rx}, g^{arx})$, where r is a random string. Note that given the output of $F_{p,g,g,g^a}(x,r)$ can output $F_{p,g,g,g^a}(2x,2r) = (g^{2r}, g^{a2r}, g^{4rx}, g^{a4rx})$ but without knowing neither $2x$ nor $2r$, which shows this extractable function is not non-malleable. In particular, the extractable functions proposed in [27, 28] have essential limitations for achieving more complex non-malleable cryptographic systems. For example, only 3-round ZK arguments were proposed in [27], while with our non-malleable extractable functions, we can achieve much robust 3-round CNMZK with straight-line simulation/extraction and with full input selection and with adaptive corruption, and even with simultaneous resettability.

- The extractable functions considers in [27, 28] are not publicly verifiable, i.e., the validity of function outputs are not publicly verifiable.

- The extractable functions considered in [27, 28] are mainly probabilistic functions and are not necessarily resettably secure, while our extractable functions are deterministic and resettably secure.

- Our extractable functions are robust, in the sense that extractability still holds even for functions chosen by the adversary, while for extractable functions considered in [27, 28] extractability only holds for honestly generated functions.

- The non-interactive extractable functions considered in [27, 28] are mainly w.r.t. adversaries of *independent* auxiliary input, where "independent" means that adversary's auxiliary input is independent of the description of the function, while for our extractable function the adversary is allowed auxiliary input that can be arbitrarily dependent upon the function description (we only require the adversary's auxiliary input should be independent of the public value, i.e, m, in the output of the adversary).

- For interactive extractable functions considered in [27, 28], extractability is achieved with rewindings, which essentially limits the modular use for achieving high-level more complex cryptographic systems, while for our extractable functions extractability is never achieved with rewindings. In particular, when our extractable functions are used in interactive settings, the verifier only needs to send a random challenge m (that is supposed to appear in the output of adversary).

- The extractable functions considered in [27, 28] are not necessarily of negligible extraction error, while our extractable functions insist negligible extraction error.

- The dependent auxiliary input considered in [27] is mainly restricted to function images, while no such restriction is posed on our non-malleable extractable functions.

- For extractable functions considered in [27, 28], a single function cannot be extractable OWF. While for our extractable functions, a single function can be extractable OWF against an adversary $\mathcal{A}$ of arbitrary auxiliary input z *as long as the public value* m *in the output of* $\mathcal{A}$ *is independent of* z.

10.5.2 Brief Comparisons with Non-Malleable OWF

Roughly speaking, the non-malleable OWF proposed in [21] comprises the perfect one-way hash function (POWHF) [24, 32] and a simulation-sound non-interactive zero-knowledge proof-of-knowledge

(SSNIZKPOK), with the parameters of POWHF and the common reference string (CRS) of SSNIZKPOK serving as the parameters of the composed NMOWF. Below, we make some brief comparisons between the formulation of our NME-JOWF and that of NMOWF of [21].

- The NMOWF formulated in [21] is probabilistic, while our NME-JOWF is determistic.
- The NMOWF of [21] does not take extractability into account.
- The NMOWF of [21] is not universal non-malleable. Specifically, if an adversary is allowed to choose function parameters on its own, then NMOWF is meaningless (as the adversary can just run the SSNIZKPOK simulator).
- The NMOWF of [21] does not take resettability into account.
- We also note that the NMOWF of [21] lacks highly practical instantiations, while we give direct practical instantiations (candidates) of NME-JOWF. We also demonstrate much wider range of cryptographic applications of NME-JOWF.

11 Generic Applications of NME-OWF

11.1 3-Round (Statistical) CNMZKAOK for $\mathcal{NP}$ with Full Adaptive Security

In this section, we formulate (transcript-based) CNMZK with full adaptive security (i.e., both adaptive input selection and adaptive corruption), and then present the a 3-round CNMZKAOK $\mathcal{NP}$ with straight-line simulation and full adaptive input selection in the plain model. The protocol can be extended to be *statistical* CNMZKAOK or fully adaptively secure.

11.1.1 Formulating CNMZK with Full Adaptive Security

Following [122], we briefly describe the CMIM setting for any two-party protocol $\langle P, V\rangle$, for an $\mathcal{NP}$-language L, with full adaptive security (adaptive input selection and adaptive corruption).

The CMIM adversary $\mathcal{A}$. $\mathcal{A}$ can concurrently interact with any polynomial number of instances of the honest prover P in the *left CMIM interaction part*. The interactions with each instance of the honest prover P is called a *left session*, in which $\mathcal{A}$ plays the role of the verifier V; *Simultaneously*, $\mathcal{A}$ interacts with any polynomial number of instances of the honest verifier V in the *right CMIM interaction part*. The interactions with each instance of the honest verifier is called a *right session*, in which $\mathcal{A}$ plays the role of the prover. For CMIM adversary with adaptive input selection, $\mathcal{A}$ can further set statements to all sessions (at the beginning of each session) *adaptively base tn on its view*. For CMIM adversary with adaptive corruption, at any point $\mathcal{A}$ can corrupt one of the honest players. Upon corruption, the whole internal state (including its input, random coins, etc) of the corrupted party is known to $\mathcal{A}$, and the corrupted party's future actions are controlled by $\mathcal{A}$. We call a CMIM adversary capable of both full adaptive input selection and adaptive corruption a *fully adaptive CMIM adversary*. A CMIM adversary is called s-CMIM adversary, for a positive polynomial $s(\cdot)$, if the adversary involves, on security parameter 1^n, at most $s(n)$ concurrent sessions in each CMIM interaction part.

For any n and $z \in \{0,1\}^*$, we denote by $view_{\mathcal{A}}(1^n, z)$ the random variable describing the view of $\mathcal{A}$, which includes its random tape, the auxiliary string $z \in \{0,1\}^*$, and all messages it receives.

We next describe a more detailed experiment, referred to as $\mathsf{Expt}^{\mathcal{A}}_{CMIM}(1^n, z)$, for the fully adaptive CMIM setting w.r.t. an interactive protocol $\langle P, V\rangle$ and an $s(n)$-CMIM adversary $\mathcal{A}$. At *any time* during this stage, $\mathcal{A}$ can do one of the following actions.

- Deliver to V a message for an already started right session, or deliver to P a message for an already started left session.
- Adaptive statement selection: $\mathcal{A}$ sets a statement $\tilde{x}_i \in \{0,1\}^{poly(n)}$ for the i-th left or right session (at the beginning of each session), $1 \leq i \leq s(n)$. We stress that if $\tilde{x}_i \in \{0,1\}^{poly(n)}$ is set to be the i-th left session, it is required that $\tilde{x}_i \in L \cup \{0,1\}^{poly(n)}$, i.e., $\mathcal{A}$ is restricted to set true statements to left sessions (otherwise, the experiment may render an $\mathcal{NP}$-membership oracle to $\mathcal{A}$), and then a witness $\tilde{w}_i$ such that $(\tilde{x}_i, \tilde{w}_i) \in \mathcal{R}_L$ is given (e.g., by a (possibly randomized) exponential-time machine [86]) to the prover instance of P.
- Adaptive corruption: $\mathcal{A}$ corrupt one of the honest players. Upon corruption, the whole internal state (including its input, random coins, etc) of the corrupted party is known to $\mathcal{A}$, and the corrupted party's future actions are controlled by $\mathcal{A}$.
- Session initiation: $\mathcal{A}$ starts a new i-th left (resp., right) session, $1 \leq i \leq s(n)$. We remark that full adaptive input selection and session initiation can be merged, if the statement to a session is mandated to be presented at the start of the session.
- Output a special "end attack" symbol within time polynomial in n.

Definition 11.1 ((transcript-based) CNMZK (with full adaptive security)) *We say that a protocol $\langle P, V \rangle$, is* (transcript-based) concurrently non-malleable zero-knowledge *(against fully adaptive CMIM) for an $\mathcal{NP}$-language L, if for any positive polynomial $s(\cdot)$, any s-CMIM adversary $\mathcal{A}$ (defined above), there exist a pair of (expected) polynomial-time algorithms S (the simulator) and E (the extractor) such that for any sufficiently large n, any auxiliary input $z \in \{0,1\}^*$, the following hold, in accordance with the experiment $\mathbf{Expt}_{\mathbf{CNMZK}}(1^n, z)$ described below (page 108):*

$\mathbf{Expt}_{\mathbf{CNMZK}}(1^n, z)$

$(str, sta) \longleftarrow S^{\mathcal{A}(1^n, z)}(1^n, z)$. That is, on inputs $(1^n, z)$ and with oracle access to $\mathcal{A}(1^n, z)$ (by providing random coins to $\mathcal{A}$ and running $\mathcal{A}$ as a subroutine) the simulator S outputs a simulated transcript str, and some state information sta to be transformed to the knowledge-extractor E. Here, we remark that during the simulation, if $\mathcal{A}$(run by S as the subroutine) corrupts the prover of the i-th left-session on input x_i, $1 \leq i \leq s(n)$, a witness w_i for $x_i \in L$ will be given to S.

For any n and any $z \in \{0,1\}^*$, we denote by $S(1^n, z)$ the random variable describing the first output (i.e., str) of $S^{\mathcal{A}(1^n,,z)}(1^n, z)$.

The knowledge-extractor E:
$\widehat{W} \longleftarrow E(1^n, sta, str)$. On (sta, str), E outputs a list of witnesses to (right) statements whose validations are successfully conveyed in right sessions in str, where the transcript of each of these successful right sessions is different from those of left sessions.

- **Simulatability.** *The following ensembles are indistinguishable: $\{S(1^n, z)\}_{n \in N, z \in \{0,1\}^*}$ and $\{view_{\mathcal{A}}(1^n, z)\}_{n \in N, z \in \{0,1\}^*}$.*
- **Knowledge-extraction.** *Specifically, E outputs a list of strings $\widehat{W} = (\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_{s(n)})$, satisfying the following:*
  - *$\hat{w}_i$ is set to be $\perp$, if the i-th right session in str is not accepting (due to abortion or verification failure) or the transcript of the i-th right session is identical to that of one of left sessions,* or the verifier of the i-th session is corrupted before this session is completed, *where $1 \leq i \leq s(n)$, .*
  - *In all other cases, with overwhelming probability $(\hat{x}_i, \hat{w}_i) \in \mathcal{R}_L$, where $\hat{x}_i$ is the statement set by $\mathcal{A}$ for the i-th right session in str and $\mathcal{R}_L$ is the $\mathcal{NP}$-relation for the language L.*

The probabilities are taken over the randomness of S, the randomness of E, and the randomness of $\mathcal{A}$.

If the simulatability property holds for any (unbound) distinguisher, then we say the protocol $\langle P, V\rangle$ is *statistical* CNMZK. If the simulator S and/or the extractor E works in a straight-line way *without rewinding the underlying* $\mathcal{A}$, we say $\langle P, V\rangle$ is CNMZK *straight-line* simulation and/or extraction. If the simulator and the extractor E only work for CMIM only capable adaptive input selection (but not adaptive corruption), we say the protocol is CNMZK with adaptive input selection (in this case, we do not need the require

11.1.2 Building Tools

Our 3-round CNMZK uses NME-OWF (and one-time signatures), and is based upon the Lapidot-Shamir (LS) WIPOK for DHC. Here, we mainly recall the Lapidot-Shamir WIPOK for DHC.

The Lapidot-Shamir protocol for DHC [84]. The n-parallel repetitions of the Lapidot-Shamir basic protocol for proving the knowledge of Hamiltonian cycle on a given directed graph G [84] is a 3-round public-coin WIPOK for $\mathcal{NP}$ (with knowledge error 2^{-n}) under any one-way permutation (as the first round of it involves one-round perfectly-binding commitments of a random permutation of G). It can be easily modified into a 4-round public-coin WIPOK for $\mathcal{NP}$ under any OWF by employing Naor's two-round (public-coin) perfectly-binding commitment scheme, or a 3-round statistical WI argument of knowledge by employing non-interactive statistically-hiding commitments based upon collision-resistant hashing.

Round-1. The prover P commits a adjacency matrix for a randomly-labeled cycle C of size q (without knowing the Hamiltonian graph to be proved). The commitment is done bit-by-bit using the one-round perfectly-binding (resp., statistically-hiding) commitment scheme for computational (resp., statistical) WI proof (resp., argument) of knowledge system.

Round-2. The verifier V responds with a randomly chosen bit b

Round-3. Now, P is given the Hamiltonian graph $G = (V, E)$ with size $q = |V|$ to be proved and a Hamiltonian cycle C_G in G as its private input. If $b = 0$, then P opens all commitments (and V checks the revealed graph is indeed a q-cycle). If $b = 1$, then P sends a random permutation π mapping C_G (i.e., its private witness) to C (committed to its first-round message), and for each non-edge of G $(i, j) \notin E$ $(1 \leq i, j \leq q)$, P opens the value (that should be 0) committed to the $(\pi(i), \pi(j))$ entry of the adjacency matrix sent in the first-round message (and V checks all revealed values are 0 and the *unrevealed* entries in the committed adjacency matrix constitute a graph that is isomorphic to G via the permutation π).

For presentation simplicity, denote by a, e, z the first, second and third message of the LS-protocol respectively. The key feature of the Lapidot-Shamir protocol is that, the prover sends the first-round message with only the knowledge of the size of the Hamiltonian graph to be proved. Furthermore, it can be easily extended to the case that the prover knows only the lower-bound $l(n)$ and the upper-bound $u(n)$ of the size of the graph to be proved [123]. In this case, in the first-round P commits $(u(n) - l(n) + 1)$ many adjacency matrices for $(u(n) - l(n) + 1)$ many cycles with sizes ranging from $l(n)$ to $u(n)$. In the third-round, after the size of G is clear, P only decommits with respect to the unique cycle of according size.

The WI property of FS protocol for HC relies on the hiding property of the underlying commitment scheme (used in its first-round). If commitment used in its first-round is replaced by a non-interactive statistically-hiding commitment scheme (with known constructions based upon on collision-resistant hash functions), the resultant protocol is a 3-round statistical WI argument of knowledge.

11.1.3 Three-Round CNMZKAOK for $\mathcal{NP}$

We first present a 3-round CNMZKAOK with only straight-line simulation and adaptive input selection for the language of HC, which is described below:

Common input: $x \in L$, where L is an $\mathcal{NP}$-language.

Witness: w

Round-1: Prover P generates (pk_{ots}, sk_{ots}), and sends $\{a, pk_{ots}\}$, where pk_{ots} (resp., sk_{ots}) is the public-key (resp., secret-key) of a one-time signature scheme.

Round-2: Verifier V responds back $\{v = f(pk_{ots}, x_1, \cdots, x_t), e\}$.

Round-3: P checks the validity of v (by the preimage public verifiability of f), and aborts if v is not valid (i.e., not in the range of f). Otherwise, P sends back $\{z, sig\}$, where $sig = sig_{sk_{ots}}(x, a, pk_{ots}, v, e, z)$ is the signature using sk_{ots} on the transcript $(x, a, pk_{ots}, v, e, z)$, (a, c, z) is the transcript of executing LS-protocol for proving the knowledge of w such that $(x, w) \in R_L$ or the knowledge of $(x_1, \cdots, x_t)$ such that $v = f(pk_{ots}, x_1, \cdots, x_t)$. Here the actual statement to LS-protocol is got by reducing (x, pk_{ots}, v) to HC.

Verifier Decision: V accepts if (1) the one-time signature sid is valid and (2) the LS-proof (a, e, z) is accepting.

Note that one-time signature can be based upon on any OWF, and non-interactive statistically-binding commitments can be based upon any one-way permutation.

Theorem 11.1 *Assuming non-interactive commitments and NME-OWF, the above protocol is 3-round (transcript-based) CNMZKAOK (with straight-line simulation).*

Proof (outline). Soundness of the above protocol is from the joint one-wayness of f and AOK property of LS-protocol.

Straight-line simulation is direct from the non-malleable extractability of f. Specifically, if (pk_{ots}, v) in one of left sessions is identical to that of one of right sessions, then the value (pk_{ots}, v) are generated by the simulator S itself (in simulating the right session), and thus S knows the pre-image of v to successfully compute z (as well as sig) in the third-round of the left-session. If (pk_{ots}, v) in one of left sessions is distinct, then by the non-malleable extractability of f, the pre-images of v can be efficiently extracted from the view of the CMIM adversary $\mathcal{A}$. Note, in particular, that the one-time signing key pk_{ots} sent by the honest prover (accordingly the simulator) in each left session is independent of the auxiliary input of the CMIM adversary $\mathcal{A}$.

By the unforgeability of one-time signature scheme, for each successful right session of transcript that is different from those of all left sessions we have that: the one-time signing-key used in this right-session must be different from the signing keys of all left sessions. For each successful right session of different signing public-key $\tilde{pk}_{ots}$, the simulator applies the underlying witness-extractor to extract a witness $\tilde{w}$ (by rewinding $\mathcal{A}$ and resending the second-round message). The extracted value $\tilde{w}$ is either a witness to the statement $\tilde{x}$ to this session or the preimages of v sent by the simulator in the second-round (in this case the witness extraction is failed for this right-session). However, suppose there exists a right session for which witness extraction fails, this violates the joint one-wayness of f. Specifically, S can simply guess this right session, and then gets the value v in accordance with the definition of joint one-wayness of f and forwards the got value v to $\mathcal{A}$. □

On achieving 3-round statistical CNMZKAOK. If we employ non-interactive statistically-hiding commitments in the first-round of the above protocol, we get a 3-round (transcript-based)

statistical CNMZKAOK for $\mathcal{NP}$ (with straight-line simulation). For this protocol variant, computational soundness, as well as correct witness extraction for successful right-session of distinct transcript, is from the joint one-wayness of f, the AOK property of LS-protocol and the computational binding property of the underling statistically-hiding commitments.

On extension to CNMZKAOK with adaptive corruption. When dealing with a CMIM adversary $\mathcal{A}$ of adaptive corruption, the difficulty is the corruption of the prover in a left-session (say, the i-th left-session) *after this left-session is finished*. In this case, the simulator S receives the witness w_i for $x_i \in L$ (where x_i is the statement being proved by the i-th left-session). The problem is that, in the simulation, S used the fake witness (i.e., the extracted preimages of f) as the witness in forming the third-round message of LS-protocol (in particular, the random permutation π). But, now after $\mathcal{A}$ gets the real witness upon corruption, it can distinguish the simulated view from its real view. To solve this problem, the idea is to employ the HC-based adaptive instance-dependent commitment (AIDC) scheme as defined and achieved in [88].

11.2 Non-Malleable Extractable Commitments, and Straight-Line CNMZK

In this section, we formulate and achieve (3-round public-coin) non-malleable extractable commitments (NINMEC). Then, using NINMEC we present a constant-round CNMZKAOK for $\mathcal{NP}$ with adaptive input selection and both straight-line simulation and extraction. The proposed CNMZK protocol with both straight-line simulation and extraction is also UCZK in the framework of UC [25].

11.2.1 Two-Round Public-Coin Non-Malleable Extractable Commitments

We consider a one-left-one-right MIM adversary $\mathcal{A}(1^n, z)$ that concurrently interacts with the honest committer instances in one left sessions and the honest receiver in one right session. We assume the left-session is committing to value $v \in \{0,1\}^n$, and $\mathcal{A}$ gets some a priori information z of v. Denote by c the transcript of the left session and $\tilde{c}$ the transcript of the right session.

Definition 11.2 *A commitment scheme $Com = \langle C, R\rangle$ is (transcript-based) non-mallealbe extractable (w.r.t. commitments), if for one-left-one-right MIM adversary $\mathcal{A}$ there exists an efficient algorithm E such that for sufficiently large n, any value $v \in \{0,1\}^n$ and any string $z \in \{0,1\}^*$, if the right-session is successful and $\tilde{c} \neq c$, then with overwhelming probability $E(1^n, z, \rho, c, \tilde{c})$ outputs $(\tilde{v}, \tilde{r}_C, r_R)$ such that $\tilde{c} = \langle C(\tilde{v}, \tilde{r}_C), R(r_R)\rangle$, where ρ is the random coins used by $\mathcal{A}$.*[21]

Traditional CNM commitment formulation [110] roughly says that, for any many-left-many-right CMIM adversary $\mathcal{A}$ there exists a simulator, such that the view of an CMIM adversary, together with the values $\mathcal{A}$ commits to in successful right sessions of transcript that is different from those of all left sessions, is indistinguishable from the output of S. Below, we show that the formulation of (one-left-one-right) non-malleable extractable commitment implies the traditional CNM commitment formulation.

Proposition 11.1 *(One-left-one-right) non-malleable extractable commitment implies concurrent non-malleable commitment.*

Proof (outline). For non-malleable commitments w.r.t. commitments, it is shown in [85] that any one-left-many-right non-malleable commitment scheme is also many-left-many-right non-malleable. Thus, we only need to show that (one-left-one-right) non-malleable extractable commitment implies one-left-many-right non-malleable commitment.

[21] A weaker version of non-malleable extractable commitments is to only require $E(1^n, z, \rho, c, \tilde{c})$ output the committed value $\tilde{v}$.

We consider a simulator S who runs a one-left-many-right MIM $\mathcal{A}$ as a subroutine and works as follows: S commits to 0 in the left-session, and just run the honest receiver in all $m = poly(n)$ right sessions. By the hiding property of the underlying commitment scheme, the simulated view output by S and the real view of $\mathcal{A}$ are indistinguishable.

Note that the simulated view of S actually accounts to the output of the following algorithm S' who runs $\mathcal{A}$ as a subroutine and works as follows: S' externally interacts with the committer who commits to 0 in the left session, and for an $i \in \{1, \cdots, m\}$ S' simulates the honest receiver in all right sessions other than the i-th right session. For the i-th right-session, S' externally interacts with an honest receiver. Denote by $\tilde{c}_i$ the transcript of the i-th right session. Note that S' is actually a one-left-one-right MIM, and thus by the definition of non-malleable extractable commitments, there exists an efficient extractor E_i such that if the i-th right session is successful and $\tilde{c} \neq c$, then with overwhelming probability $E(1^n, z, c, \tilde{c}_i, \rho_{S'})$ outputs the value $\tilde{v}_i$ that S' commits to in the i-th right session, where $\rho_{S'}$ is the randomness of S'. Thus, the simulator S can output the values $\mathcal{A}$ successfully commits to in right sessions of transcripts different from that of the left-session, by applying E_i for each i,$1 \leq i \leq m$. Similarly, from the real view of $\mathcal{A}$, the values committed by $\mathcal{A}$ to successful right sessions of different transcript can also be efficiently extracted. As the real view and the simulated view are indistinguishable and the extractors are polynomial-time, we have the values extracted from the real view, together with the real view, is indistinguishable from the values extracted from the simulated view, together with the simulated view. □

Theorem 11.2 *Under NME-JOWF, there exists a 2-round public-coin statistically-binding non-malleable extractable commitment (NMEC) scheme.*

Proof (outline). On a security parameter n and using a NME-JOWF f, the NMEC protocol $\langle C, R \rangle$ is as follows:

Commitment stage:

Round-1. The receiver R sends a random challenge $r \in \{0,1\}^n$.

Round-2. To commit to a message $v \in \{0,1\}^t$, the committer C randomly selects $x_1, \cdots, x_t$, where each $x_i \in \{0,1\}^n, 1 \leq i \leq t$. Denote by B a hard-core predicate for f, C then computes $c = v \oplus B(x_1) \cdots B(x_t)$. Let $m = c||r$ where "||" stands for string concatenation. Then, C computes $(m, \sigma) = f(m, x_1, \cdots, x_t)$, and sends σ to R. After receiving σ, R checks whether (m, σ) is a valid value in the range of f, by the pre-image public verifiability of f. If (m, σ) is not valid, R aborts.

Decommitment Stage: The committer reveals the value v and the randomness $(x_1, \cdots, x_t)$. The receiver R computes $c = v \oplus B(x_1) \cdots B(x_t)$, $m = c||r$ and checks whether $(m, \sigma) = f(m, x_1, \cdots, x_t)$.

Security analysis. For any one-left-one-right MIM $\mathcal{A}$ of arbitrary auxiliary input z, denote by $(\tilde{r}, c, \sigma)$ the transcript of the left-session, and by $(r, \tilde{c}, \tilde{\sigma})$ the transcript of the right session. Here, we use m (resp., $\tilde{m}$) to denote a message sent by an honest player (resp., the MIM adversary $\mathcal{A}$). Note that r is always independent of z (i.e., the value $r||\tilde{c}$ includes a substring r that is independent of z). If $(\tilde{c}, r, \tilde{\sigma})$ is valid and $(r, \tilde{c}, \tilde{\sigma}) \neq (\tilde{r}, c, \sigma)$, then it appears that we can just relay upon the non-malleable extractability of f to extract the values $(\tilde{x}_1, \cdots, \tilde{x}_t)$, and thus the value $\tilde{v}$ committed to the right session.

But, there are some subtleties regarding the above reasoning. On the one hand, the value $m = r||c$ are generated interactively, but in the formulation of NME-JOWF the value m queried to its oracle is chosen by the adversary itself. This means that we cannot directly apply the knowledge extractor ensured by the formulation of NME-JOWF here. On the other hand, in the formulation

of NME-JOWF, the values $(x_1, \cdots, x_t)$ are chosen independently from the public query m. But, here m includes c that is a function of $(x_1, \cdots, x_t)$. To solve these subtlety, we rely upon the pseudorandomness of hard-cores.

We consider an algorithm S, who gets oracle access to $f(\cdot, x_1, \cdots, x_t)$ and runs the MIM adversary $\mathcal{A}$ as a subroutine, works as follows: In the right session, S simply send a random challenge r to $\mathcal{A}$; In the left session, after receive the challenge $\tilde{r}$ sent by $\mathcal{A}$, S randomly choose a value $c \in \{0,1\}^t$, and then queries its oracle f with $m = \tilde{r}||c$. After getting $(m, \sigma) = f(m, x_1, \cdots, x_t)$ from the oracle, S returns back σ to $\mathcal{A}$. After $\mathcal{A}$ outputs a valid second-round message $(\tilde{c}, \tilde{\sigma})$ of the right-session, S outputs $(r||\tilde{c}, \tilde{\sigma})$. According to the non-malleable extractability of f, there exists a polynomial-time algorithm E that, given the random tape of S and the two session transcripts, extracts the preimages values $(\tilde{x}_1, \cdots, \tilde{x}_t)$, and thus the value $\tilde{v}$ committed by $\mathcal{A}$ to the right session.

As the real view of $\mathcal{A}$ and the view of A under the run of S are identical, and the knowledge-extractor E is polynomial-time, we can now have that when applying E on the real view of $\mathcal{A}$, the values $(\tilde{x}_1, \cdots, \tilde{x}_t)$, and thus the value $\tilde{v}$ committed by $\mathcal{A}$ to the right session in the real view, can also be extracted.

Here, we remark that for the above NMEC scheme, it is critical to put c into the input of f. Otherwise, the protocol is not non-malleable. In particular, if only r is put into the input of f, then it could be the case that: the left-session transcript is (r, c, σ) and the right-session transcript is $(r, \tilde{c}, \sigma)$, where $\tilde{c} \neq c$. In this case, we extract nothing from $\{(r, c, \sigma), (r, \tilde{c}, \sigma)\}$ (i.e., the real view of $\mathcal{A}$). □

Note: In general, m can be set to be $c||r||aux_C||aux_R$, where aux_C (resp., aux_R) is some other information sent by C (resp., R). The non-malleable extractability of the above scheme still holds. As we shall see, this allows easier modular uses of NMEC in achieving more complex systems.

11.2.2 Three-Round CNMZK with Both Straight-Line Simulation and Extraction

The protocol construction is based upon the CNMZK protocol presented in Section 11.1.3, and uses the 2-round non-malleable extractable commitment scheme.

Common input: $x \in L$, where L is an $\mathcal{NP}$-language.

Witness: $w \in \{0,1\}^n$

Round-1: Prover P generates (pk_{ots}, sk_{ots}), and sends $\{a, pk_{ots}\}$, where pk_{ots} (resp., sk_{ots}) is the public-key (resp., secret-key) of a one-time signature scheme, and a is the first-round message of LS-protocol.

Round-2: Verifier V randomly chooses $(x_1, \cdots, x_t) \in (\{0,1\}^n)^t$ and $e \in \{0,1\}^n$, computes $v = f(pk_{ots}, x_1, \cdots, x_t)$, and sends (v, e) to P.

Round-3: P checks the validity of v (by the preimage public verifiability of f), and aborts if v is not valid (i.e., not in the range of f). Otherwise, denote $\hat{t} = nt$, P randomly choose $(\hat{x}_1, \cdots, \hat{x}_{\hat{t}}) \in (\{0,1\}^n)^{\hat{t}}$, and computes $c = w||pad \oplus B(\hat{x}_1)||...||B(\hat{x}_{\hat{t}})$, where pad is any padding string such that $|w||pad| = |x_1|| \cdots ||x_t|$. Then, P computes $\sigma = f(pk_{ots}||c||e, \hat{x}_1, \cdots, \hat{x}_{\hat{t}})$, and the third-round message z of LS-protocol, and $sig = sig_{sk_{ots}}(x, a, pk_{ots}, v, e, z, c, \sigma)$. Finally, P sends (z, c, σ, sig) to V.

Note that (pk_{ots}, c, e, σ) is the transcript of the NMEC commitment scheme with e serving as the random challenge and m being defined to be $pk_{ots}||c||e$. (a, e, z) is the transcript of executing LS-protocol for proving the knowledge of $\hat{w}$ and $(\hat{x}_1, \cdots, \hat{x}_{\hat{t}})$ such that (pk_{ots}, e, c, σ) commits to $\hat{w}$ using randomness $(\hat{x}_1, \cdots, \hat{x}_{\hat{t}})$, and either the first n-bit prefix of $\hat{w}$ is a valid

witness for $x \in L$ or $\hat{w} = x_1||\cdots||x_t$ such that $v = f(pk_{ots}, x_1, \cdots, x_t)$. Here the actual statement to LS-protocol is got by $\mathcal{NP}$-reduction.

Verifier decision. V accepts if (1) the one-time signature sid is valid, (2) the NMEC commitment is valid, and (3) the LS-proof (a, e, z) is accepting.

Analysis outline. Straight-line simulation is from the non-malleable extractability of NME-JOWF collections. Note the CMIM adversary actually gets access to NME-JOWF w.r.t. two different parameters (n, t) and $(n, \hat{t})$. If the value v sent by $\mathcal{A}$ in one left session is identical to that sent by the simulator S in one right-sessions, then S already possesses the preimages to v to successfully finish this left-session. If the value v sent by $\mathcal{A}$ in one left session is valid and is different from those sent by the simulator S in all right-sessions, the corresponding preimages can be straight-line extracted (by the definition of NME-JOWF collections). Note that pk_{ots} is always independent of the auxiliary input z of $\mathcal{A}$, and with overwhelming probability, v is also different from σ sent by S in all left-sessions. In actual analysis, we can show that the simulated view of the simulator S can be emulated by an algorithm $\hat{S}$ with oracle access to $f_{n,t}$ and $f_{n,\hat{t}}$. Then the view of $\hat{S}$ is extractable implies the view of S is also extractable.

For straight-line extraction, for each successful right-session that is of distinct transcript, we know the one-time public-key sent in the first-round of this right-session must be different from those sent by S in all left-sessions. This implies the NMEC transcript in the right-session is also different from those in all left-sessions, by the non-malleable extractability of NMEC the value committed to NMEC can be extracted. Again, by the joint one-wayness of f, with overwhelming probability the extracted value is the witness w for $x \in L$. Note that, though the NMEC scheme is executed with other protocols (say, the LS-protocol), all other messages can be simulated by a polynomial-time algorithm with oracle access to $f_{n,t}$ and $f_{n,\hat{t}}$.

11.3 Constant-Round Concurrent Non-Malleable Coin-Tossing

In this section, using non-malleable extractable commitments and CNMZK with straight-line simulation, we present a constant-round concurrent non-malleable coin-tossing (CNMCT) protocol, with straight-line simulation, which can also be shown to be universal composable coin-tossing.

We first define CNMCT in accordance with the definition of CNMCT in the bare public-key model [122]. The reader is referred to [122] for more details about the definition of CNMCT.

Legitimate CRS-simulating algorithm $\mathcal{M}_{CRS}$. Let $(r, \tau_r) \longleftarrow \mathcal{M}_{CRS}(1^n)$, where $\mathcal{M}_{CRS}$ is a PPT algorithm. The PPT algorithm $\mathcal{M}_{CRS}$ is called a legitimate CRS-simulating algorithm with respect to a polynomial-time computable CRS-trapdoorness validating relation $\mathcal{R}_{CRS}$, if the distribution of its first output, i.e., r, is computationally indistinguishable from U_n (the uniform distribution over strings of length n), and $\mathcal{R}_{CRS}(r, \tau_r) = 1$ for all outputs of $\mathcal{M}_{CRS}$ (typically, τ_r is some trapdoor information about r). For a positive polynomial $s(\cdot)$, we denote by $(\{r_1, r_2, \cdot, r_{s(n)}\}, \{\tau_{r_1}, \tau_{r_2}, \cdots, \tau_{r_{s(n)}}\}) \longleftarrow \mathcal{M}_{CRS}^{s(n)}(1^n)$ the output of the experiment of running $\mathcal{M}_{CRS}(1^n)$ *independently* $s(n)$ times, where for any i, $1 \leq i \leq s(n)$, (r_i, τ_{r_i}) denotes the output of the i-th independent execution of $\mathcal{M}_{CRS}$.

$\mathcal{M}_{CRS}$ trivially achievable distribution. Let G be a set of pairs of integers $\{(i_1, j_1), (i_2, j_2), \cdots, (i_t, j_t)\}$, where $1 \leq i_1 < i_2 < \cdots < i_t \leq s(n)$ and $1 \leq j_1, j_2, \cdots, j_t \leq s(n)$ are distinct integers, and $0 \leq t \leq s(n)$ such that G is defined to be the empty set when $t = 0$. Let $\mathcal{M}_{s,n,G}$ be the probability distribution over $(\{0,1\}^n)^{2s(n)}$, obtained by first generating $2s(n) - t$ n-bit strings $\{x_m, y_k | m \in \{1, 2, \cdots, s(n)\}, k \in \{1, 2, \cdots, s(n)\} - \{j_1, j_2, \cdots, j_t\}\}$, by running $\mathcal{M}(1^n)$ independently $2s(n) - t$ times, and then defining $y_{j_d} = x_{i_d}$ for $1 \leq d \leq t$ and taking $(x_1, x_2, \cdots, x_{s(n)}, y_1, y_2, \cdots, y_{s(n)})$ as the output. A probability distribution over $(\{0,1\}^n)^{2s(n)}$ is called *$\mathcal{M}$-trivially achievable*, if it is a convex combination of $U_{s,n,G}$ over all G's.

Definition 11.3 (concurrently non-malleable coin-tossing CNMCT) *Let $\Pi = \langle L, R \rangle$ be a two-party protocol. We say that Π is a concurrently non-malleable coin-tossing protocol, if for any PPT $s(n)$-CMIM adversary $\mathcal{A}$ there exists a probabilistic (expected) polynomial-time algorithm S such that, for any sufficiently large n, any auxiliary input $z \in \{0,1\}^*$, any PPT CRS-simulating algorithm $\mathcal{M}_{CRS}$ and any polynomial-time computable (CRS-trapdoor validating) relation $\mathcal{R}_{CRS}$, the following hold, in accordance with the experiment $\mathbf{Expt}_{\mathbf{CNMCT}}(1^n, z)$ described below (page 115):*

$\mathbf{Expt}_{\mathbf{CNMCT}}(1^n, z)$

The simulator S:

$(str, sta) \longleftarrow S^{\mathcal{A}(1^n,z)}_{PROOF}(1^n, z)$. That is, on inputs $(1^n, z)$ and with oracle access to $\mathcal{A}(1^n, z)$ (by providing random coins to $\mathcal{A}$ and running $\mathcal{A}$ as a subroutine), the simulator S outputs a simulated transcript str and some state information sta. Denote by $R_L = \{R_L^{(1)}, R_L^{(2)}, \cdots, R_L^{(s(n))}\}$ the set of outputs of the $s(n)$ left sessions in str and by $R_R = \{R_R^{(1)}, R_R^{(2)}, \cdots, R_R^{(s(n))}\}$ the set of outputs of the $s(n)$ right sessions in str. The state information sta consists, among others, of two sub-sets (of $s(n)$ components each): $sta_L = \{sta_L^{(1)}, sta_L^{(2)}, \cdots, sta_L^{(s(n))}\}$ and $sta_R = \{sta_R^{(1)}, sta_R^{(2)}, \cdots, sta_R^{(s(n))})\}$.

- **Simulatability.** *The following ensembles are computationally indistinguishable: $\{S(1^n, z)\}_{1^n, z \in \{0,1\}^*}$ and $\{view_{\mathcal{A}}(1^n, z)\}_{1^n, z \in \{0,1\}^*}$.*
- **Strategy-restricted and predefinable randomness.** *With overwhelming probability, both the distribution of (R_L, sta_L) and that of (R_R, sta_R) are identical to the distribution of $\mathcal{M}_{CRS}^{s(n)}(1^n)$; furthermore, the distribution of (R_L, R_R) is $\mathcal{M}$-trivially achievable.*

The probabilities are taken over the randomness of S, the randomness of $\mathcal{M}_{CRS}$, and the randomness of $\mathcal{A}$.

As clarified in [122], the property of strategy-restricted and predefinable randomness is a formal definition of the following informal statements: *the coin-tossing output of each left (resp., right) session is either independent of the outputs of all other sessions OR copied from the output of one right (resp., left) session on the opposite CMIM part; furthermore, the output of each session in one CMIM part can be copied into the opposite CMIM part at most once. Moreover, the simulator S sets and controls, at the same time* in an online fashion*, the coin-tossing outputs of* all *left and right sessions in the simulated transcript (in the sense that S knows the corresponding trapdoor information of* all *the coin-tossing outputs appearing in the simulated transcript).*

Protocol construction: The CNMCT protocol $\langle L, R \rangle$ is presented below:

Stage-1: L commits to a random string $r_L \in \{0,1\}^n$, by running a non-malleable extractable commitment scheme with R.

Stage-2: R sends a random string r_R to L.

Stage-3: L sends $r = r_L \oplus r_R$, and proves the knowledge (r_L, ran) to R that the NMEC commitment of Stage-1 commits to $r_L = r \oplus r_R$ using randomness ran, by running a straight-line simulatable CNMZK protocol.

The security of the CNMCT protocol can be easily derived from the non-malleable extractability of NMEC and the non-malleable extractability of the function f used in the straight-line simulatable CNMZK in Stage-3.

11.4 Applications to Complete Non-Malleable Encryption and Signature

In the security formulation of public-key encryption and digital signature, the adversary possesses no auxiliary input. We note that resettable NME-OWF directly implies a *deterministic* digital signature scheme, and resettable universal NME-OWF collection implies a *deterministic* complete non-malleable signature scheme [55].

For achieving public-key encryption schemes, the key point is to set up a non-malleable extractable trapdoor one-way functions (NME-TOWF). We use the idea from [28] to derive extractable trapdoor function from any extractable function. In [28], given a extractable one-way function f and a trapdoor OWF π, the extractable trapdoor OWF is defined to be $g(x) = (f(x), \pi(x))$, where it is assumed that f and π remain one-way with respect to each other. Note that the extractable trapdoor OWF above is not publicly pre-image verifiable and is not necessarily non-malleable. To achieve a PKE scheme, the work [28] treats the public-key of π as the public-key, and the trapdoor of π as the secret-key. Then, to encrypt a bit b, the sender randomly choose x and computes the ciphertext as $(f(x), \pi(x), \hat{c} = m \oplus B(x))$ where B is the hard core for the function g. It is shown in [28] that such a PKE scheme is extractable (or, plaintext-aware), *Conditioned on the adversary does not see another ciphertext.* It is also easy to see that such a PKE scheme is not non-malleable. For example, after seeing a ciphertext $(f(x), \pi(x), \hat{c}$, an adversary can simply output $(f(x), \pi(x), 1 - \hat{c}$ that encrypts $1 - m$.

We show that NME-JOWF can be used to achieve plain-text aware PKE schemes against adaptive chosen message attacks, and universal NME-JOWF collections can be used to achieve (plain-text aware) complete non-malleable encryption.

Given an NME-JOWF f and trapdoor function π, construct a non-malleable extractor trapdoor joint OWF (NME-TJOWF) as follows: $g(m, x_1, \cdots, x_t) = f(m', x_1, \cdots, x_t)$, where $m' = m||\pi(x_1)|| \cdots ||\pi(x_t)$. To encrypt a message $m \in \{0,1\}^t$, the sender randomly selects $(x_1, \cdots, x_t)$ and sets the ciphertext to be $f(\hat{m}, x_1, \cdots, x_t)$, where $\hat{m} = pk||\pi(x_1)|| \cdots ||\pi(x_t)||c$ and $c = m \oplus B(x_1)|| \cdots ||B(x_t)$. The non-malleability of such a PKE scheme is from the following observations: after seeing many ciphertexts, suppose the adversary outputs a different ciphertext that is valid in accordance the public verifiability of f, then the values $(x_1, \cdots, x_t)$ (and thus the encrypted message m) can be efficiently extracted. Note that in the traditional security formulation of PKE, the underlying adversary possesses no auxiliary input.

12 Candidates of (Resettable) NME-JOWF

In this section, we seek for candidates of a special case of NME-JOWF of the form $f(m, x_1, \cdots, x_t) = (m, f'(m, x_1, \cdots, x_t))$, where $f'(m, x_1, \cdots, x_t) = (\hat{f}(x_1), \cdots, \hat{f}(x_t), \pi(m, x_1, \cdots, x_t))$ and $\hat{f} : \mathcal{D} \cap \{0,1\}^n \to \{0,1\}^l$ is a OWF, where l is polynomial in n. Furthermore, we simply assume that for any m and for any permutation $(x'_1, \cdots, x'_t)$ of $(x_1, \cdots, x_t)$, the range of $f'(m, x_1, \cdots, x_t)$ and the range of $f'(m, x'_1, \cdots, x'_t)$ are viewed to be the same. (Some candidates proposed below also satisfy the stronger requirement that: if $(x'_1, \cdots, x'_t) \neq (x_1, \cdots, x_t)$, then the range of $f'(m, x_1, \cdots, x_t)$ and the range of $f'(m, x'_1, \cdots, x'_t)$ are viewed to be also different.)

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic multiplicative groups with the same prime order q, g_1 (resp., g_2 be a generator of $\mathbb{G}_1$ (resp., $\mathbb{G}_2$) and $h : \{0,1\}^* \to Z_q^*$ be a hash function. We assume the group element presentation in $\mathbb{G}_1$ is shorter than that in $\mathbb{G}_2$. Let e be a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \forall a, b \in \mathbb{Z}_p$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degeneracy: There exist $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$;
- Computability: There exists an efficient algorithm to compute $e(g_1, g_2)$ for $\forall g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.

12.1 General Candidates of (Resettable) NME-JOWF Based on Pairings

Let $b \in \{0,1\}$ and $\bar{b} = 1 - b$, $f(m, x_1, \cdots, x_t) = (m, X_1 = g_b^{x_1}, \cdots, X_t = g_b^{x_t}, \tau = g_{\bar{b}}^{\frac{1}{f_0 + f_1 x_1 + \cdots + f_t x_t}})$, where m denotes a set of public messages, g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $X_i \in \mathbb{G}/1_{\mathbb{G}}, 1 \leq i \leq t$ (where some of DH-components are fixed ones, e.g., public-keys, and some are ephemeral ones). We allow one of g_1, g_2 can be a hidden value (that is only known and possessed by the knowledge prover). When both g_1, g_2 are public values, g_1, g_2 can also be identical (for symmetric pairings). In case of $f_0 + f_1 x_1 + \cdots + f_t x_t = 0$, we say that the function f is not defined. The input of each function $f_i, 0 \leq i \leq t$ is a (maybe empty) subset of $\{\hat{m}, X_1 = g^{x_1}, \cdots, X_t = g^{x_t}\}$ where $\hat{m} \in m$ denotes the message to be authenticated, and the deterministic functions of $f_0, f_1, \cdots, f_t$ *is polynomial-time computable* and satisfy the following conditions:

Public (authenticated) message coverage: Denote by $\mathcal{S}$ all the inputs to the functions $f_0, f_1, \cdots, f_t$, we have $\{\hat{m}, X_1, \cdots, X_t\} \subseteq \mathcal{S}$, where $\hat{m}$ denotes the public messages to be authenticated.

Public (authenticated) message commitment (assuming the outputs of $f_0, f_1, \cdots, f_t$ are random elements in Z_q^*): Given $\{\hat{m}, X_1, \cdots, X_t\}$ no probabilistic polynomial-time algorithm can find $\{\hat{m}', X_1' = g_b^{x_1'}, \cdots, X_t' = g_b^{x_t'}\} \neq \{\hat{m}, X_1, \cdots, X_t\}$, where $X_k' \in \mathbb{G}$, $1 \leq k \leq t$, such that:

- The outputs of the functions $f_0, f_1, \cdots, f_t$ are identical on $\{\hat{m}', X_1', \cdots, X_t'\}$ and $\{\hat{m}, X_1, \cdots, X_t\}$.
- Or, the values of $\{v_1 x_1, \cdots, v_t x_t\}$ are identical to the values $\{v_1' x_1', \cdots, v_t' x_t'\}$, where $v_1, \cdots, v_t$ are the outputs of $f_1, \cdots, f_t$ on inputs $(\hat{m}, X_1, \cdots, X_t)$ and $v_1', \cdots, v_t'$ are the outputs of $f_1, \cdots, f_t$ on inputs $\{\bar{m}', X_1', \cdots, X_t'\} \neq \{\hat{m}, X_1, \cdots, X_t\}$.

Mutual independence: Assuming the outputs of $f_0, f_1, \cdots, f_t$ are random elements in Z_q^*, on input $(p, q, \mathbb{G}, \mathbb{G}_T, e)$, no matter how a malicious PPT algorithm does (particularly on choosing $X_1, \cdots, X_t$), for any value $v \in Z_q^*$ it holds that:

- In case $t = 1$, the probability $\Pr[v_1 x_1 = v]$ is negligible (in $|q|$), where v_1 is the output of f_1 on $(\hat{m}, X_1)$ chosen by the adversary itself, and $X_1 = g_b^{x_1} \in \mathbb{G}/1_{\mathbb{G}}$.
- In case $t \geq 2$, there is at most one $k \in \{1, \cdots, t\}$ such that the probability $\Pr[v_i x_i = v]$ is negligible (in $|q|$) for any $i, 1 \leq i \neq k \leq t$, where v_i is the output of f_i on $(\hat{m}, X_1, \cdots, X_t)$ chosen by the adversary itself, $X_j = g_b^{x_j} \in \mathbb{G}/1_{\mathbb{G}}$, $1 \leq j \leq t$, and X_k is a fixed DH-component.

The validity of $f(\bar{m}, x_1, \cdots, x_t) = (\hat{m}, X_1 = g_b^{x_1}, \cdots, X_t = g_b^{x_t}, \tau = g_{\bar{b}}^{\frac{1}{f_0 + f_1 x_1 + \cdots + f_t x_t}})$ can be publicly verified by checking whether $X_i \in G_b/1_{G_b}$, $1 \leq i \leq t$ and $e(g_b^{f_0} X_1^{f_1} \cdots X_t^{f_t}, \tau) = e(g_b, g_{\bar{b}})$.

12.2 Some Concrete Instantiations

Next, we present some (preferable) concrete instantiations of the general NME-JOWF candidate based on pairings. Needless to say, more concrete candidates can be instantiated from the general pairing-based candidate.[22] In the following description, $h(\hat{m})$ is defined to be 0 if $\hat{m}$ is an empty string. For most cases (except for the case of 1(b) when $f_0 = h(\hat{m})$), if $\hat{m} \in Z_q^*$ $h(\hat{m})$ can also be replaced by $\hat{m}$. In the following description, $c \in Z_p^*$ is a random value that is a part of the function parameter (when defining self-composable (R)NME-JOWF collections), or $c = x_{X_1}$ or $c = X_1 \mod q$, where x_{X_1} is the x-coordinate of X_1.

[22] More variants of the candidates can be found in the patent document.

1. Case $t = 1$. For the case of $t = 1$, we present the following preferable instantiations:

   (a) $f_1 = h(\hat{m}, X_1)$, $f_0 = c$.

   (b) $f_1 = h(X_1)$, $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$.

   (c) $f_1 = x_{X_1} \mod q$, and $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, where x_{X_1} denotes the x-coordinate of X_1.

2. Case $t = 2$. For the case of $t = 2$, we present the following preferable instantiations:

   (a) For $\delta \in \{0, 1\}$, $f_\delta = h(\hat{m}, X_1, X_2)$, $f_{1-\delta} = 1$, $f_0 = 0$ or $f_0 = c$.

   (b) $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, $f_1 = h(X_1)$ or $f_1 = h(1, X_1)$, $f_2 = h(X_2)$ or $f_2 = h(2, X_2)$. Note that when $f_1 = h(1, X_1)$ and $f_2 = h(2, X_2)$, the order of X_1, X_2 in the output of f is also fixed.

   (c) $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, $f_1 = h(X_1)$ or $f_1 = h(1, X_1)$, $f_2 = h(X_2)$ or $f_2 = h(2, X_2)$. Note that when $f_1 = h(1, X_1)$ and $f_2 = h(2, X_2)$, the order of X_1, X_2 in the output of f is also fixed.

   (d) $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, $f_1 = x_{X_1} \mod q$ and $f_2 = x_{X_2} \mod q$.

3. Case $t > 2$. For the case of $t > 2$, we present the following preferable instantiations:[23]

   (a) $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, $f_i = h(X_i)$ or $f_i = h(i, X_i)$ for each $i, 1 \leq i \leq t$. Note that for the case $f_i = h(i, X_i)$ for all $i, 1 \leq i \leq t$, the order of $X_1, \cdots, X_t$ in the output of f is also fixed.

   (b) $f_0 = h(\hat{m})$ or $f_0 = h(\hat{m}) + c$, $f_i = x_{X_i)} \mod q$ for each $i, 1 \leq i \leq t$.

   (c) $f_0 = 0$, $f_k = h(\bar{m}, X_k)$ for some $k, 1 \leq k \leq t$, and $f_j = h(X_j)$ for $1 \leq j \neq k \leq t$.

Discussion: In a way, the above paring-based NME-JOWF can be viewed as non-interactive non-malleable proof of the joint knowledge of $(x_1, \cdots, x_t)$ (together authentication of $\hat{m}$). Note that some of the above (R)NME-JOWF candidates can be without hashing (when $\hat{m} \in Z_q^*$, and $f_i = x_{X_i} \mod q$). When using hash function for some candidates, we actually do not assume the hash function used to be a random oracle. We only need the following properties of the underlying hash function: (1) collision-resistance and (2) given a value v, no PPT algorithm can output $X = g^x$ such that $\Pr[xh(X) = v]$ is non-negligible. We also note most candidates, except 1(a) when c is a random value and 2(b) when c is a random value, can also be (resettable) universal non-malleable extractable joint OWF collections.

13 Applications of the Concrete Pairing-Based RNME-OWF Candidates

13.1 (Deterministic) Complete Non-Malleable Signatures of Knowledge

Roughly speaking, by complete non-malleable signature of knowledge (CNMSOK), we mean that, on a security parameter n, after adaptively getting a list $\mathcal{Q} = \{(pk, m_1, \sigma_1), \cdots, (pk, m_q, \sigma_q)\}$ from a signing oracle of public-key pk, where q is polynomial in n, if the forger $\mathcal{A}$ outputs $(pk', m, \sigma) \notin \mathcal{Q}$ such that σ is a valid signature of m under the public-key pk', then $\mathcal{A}$ has to know the secret-key of pk' (in the sense that such secret-key can be efficiently extracted from the random coins of $\mathcal{A}$ and

[23] In case of some X_i's are fixed public-keys (and known to all players in the system), the corresponding f_i could be set to be 1 in certain scenarios.

the oracle answers $\mathcal{Q}$). We remark CNMSOK are useful when using signatures as a building tool in develop more complex non-malleable systems (e.g., as we shall see, *pnon-malleable key-exchange over Internet* and *non-malleable range proof*, etc) running in concurrent settings like the Internet.

Most of the above pairing-based RNME-JOWF candidates (except 1(a) and 2(a)) can be used for achieving CNMSOK. Below, we only present several preferable solutions.[24]

The first solution:

Public-key: $\{g_1, g_2, , \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, h, X = g_2^x, e(g_1, g_2)\}$, where x is taken uniformly at random from Z_q^*.

Signature generation: On a message m to be signed, the signer computes $\tau = g_1^{\frac{1}{xh(m,X)+h(m)}}$ (or, simply $\tau = g_1^{\frac{1}{xf(X)+h(m)}}$, where $f(X) = x_X \mod q$ or $f(X) = h(X)$). The reason is to use g_1 in signature generation is that group element representation in $\mathbb{G}_1$ is shorter than that in $\mathbb{G}_2$, and thus we have shorter signatures.

Dealing with the case of $xf(X) + h(m) = 0$: Though it is unlikely that $xf(X) + h(m) = 0$ (note that there is only one m such that $xf(X)+m = 0$), a general method is to simply define $\frac{1}{0} = 0$ (i.e., τ is defined to be $1_{\mathbb{G}_1}$ in this case); Another method to deal with this undefined case is to concatenate a random string r to m, i.e., letting $m' = m||r$, and put r into the signature. With this general method, in case of $xf(X) + h(m, r) = 0$, the signer re-selects a new r until $xf(X) + h(m, r) \neq 0$. Such general methods apply also to other solutions.

Signature verification: On (m, τ), the verifier checks whether ($\tau \in \mathbb{G}_1$[25] and) $e(\tau, X^{h(m,X)} g^{h(m)}) = e(g_1, g_2)$ (or, $e(\tau, X^{f(X)} g^{h(m)}) = e(g, g)$). If $\tau = 1_{\mathbb{G}_1}$, the verifier checks that $X^{f(X)} = g^{-h(m)}$. To improve online efficiency, $X^{f(X)}$ can be offline pre-computed.

The second solution:

Public-key: $\{g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, h, X = g_2^x, C = g_2^c, e(g_1, g_2)\}$, where x and c are taken uniformly at random from Z_q^*.

Signature generation: On a message m to be signed, the signer computes $\tau = g_1^{\frac{1}{xh(m,X,C)+c}}$ (or, simply $\tau = g_1^{\frac{1}{xh(m,X)+c}}$ for standard signature scheme that however is not CNMSOK).[26]

Signature verification: On (m, τ), the verifier checks whether $e(\tau, X^{h(m,X,C)} C) = e(g_1, g_2)$. *The advantage of the second solution over the first solution is the more efficient signature verification.*

The third solution:

Public-key: $\{g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, h, X = g_2^x, e(g_1, g_2)\}$, where x is taken uniformly at random from Z_q^*.

Signature generation: On a message m to be signed, compute $\tau = g_1^{\frac{1}{xh(m,X)+x_X}}$, where x_X is the x-coordinate of X (or $X \mod q$).

Signature verification: On (m, τ), the verifier checks whether $e(\tau, X^{h(m,X)} g^{x_X}) = e(g_1, g_2)$, where g^{x_X} can be pre-computed for improving online verification efficiency.

[24] In all the proposed solutions, we can also put signer *id* into the input of h, e.g., by replaying m to be $id||m$, which are particularly useful for achieving non-malleable key-exchange using CNMSOK.

[25] For presentation simplicity, we do not explicitly describe this kind of group membership check in the following description.

[26] In accordance with the general candidates of RNME-JOWF based on pairing, some other possible solutions are: $\tau = g^{\frac{1}{xf(X)+ch(C,m)}}$, $\tau = g^{\frac{1}{x+ch(X,C)+h(m)}}$, $\tau = g^{\frac{1}{xf(X)+cf(C)+h(m)}}$, etc.

13.2 Non-Malleable Batch (Two-Party or Triparty) DHKE

Let player $\hat{A}$ be of public-key $A = g_1^a \in \mathbb{G}$ and secret-key $a \in Z_q^*$, player $\hat{B}$ be of public-key $B = g_1^b \in \mathbb{G}$ and secret-key $b \in Z_q^*$, player $\hat{C}$ be of public-key $C = g_1^c \in \mathbb{G}$ and secret-key $c \in Z_q^*$. For presentation simplicity, below we describe a batch Diffile-Hellman key-exchange (BDHKE) protocol (with mutual authentication) between $\hat{A}$ and $\hat{B}$.

The case of sharing a single session-key between two parties:

- $\hat{A}$ generates and sends $\{sid, \hat{A}, X = g_1^x, \tau_A = g_2^{\frac{1}{ah(\hat{A},A,X,sid)+x}}\}$, where x is taken uniformly at random from Z_q^* and sid is a unique session identifier.[27]
- Upon receiving (sid, X, τ_A), $\hat{B}$ checks that $X \in \mathbb{G}_1$ and $e(A^{h(sid,\hat{A},A,X)}X, \tau_A) = e(g_1, g_2)$. If any verification fails, $\hat{B}$ aborts; Otherwise, $\hat{B}$ generates and sends back $\{sid, \hat{B}, Y = g_1^y, \tau_B = g_2^{\frac{1}{ah(\hat{B},B,Y,sid)+y}}\}$, where y is taken uniformly at random from Z_q^*. $\hat{B}$ derives the session-key as $K = KDF(X^y, \hat{A}, \hat{B})$, where KDF is a (deterministic) key-derivation function.[28]
- Upon receiving $\{sid, Y, \tau_B\}$, $\hat{A}$ checks whether $Y \in \mathbb{G}_1$ and $e(B^{h(\hat{B},B,Y,sid)}Y, \tau_B) = e(g_1, g_2)$. If any verification fails, $\hat{A}$ aborts; Otherwise, $\hat{A}$ computes the session-key as $K = KDF(Y^x, \hat{A}, \hat{B})$.

The security of the above DHKE scheme assumes the CDH assumption, even if the authentication values $\tau_{\hat{A}}$ and $\tau_{\hat{B}}$ are given to the adversary. One way to weaken the underlying assumption is to encrypt one's authentication value using its peer's public-key (a CPA-secure PKE scheme, e.g., ElGamal encryption [51], is sufficient for this purpose). In particular, using ElGamal encryption, the value X_1 (resp., Y_1) can be served as the group element in the ciphertext. That is, $\tau_{\hat{A}}$ (resp., $\tau_{\hat{B}}$) is changed to be $B^{x_1}\tau_{\hat{A}}$ (resp., $A^{y_1}\tau_{\hat{B}}$).

The case of sharing a single session-key among three parties: In this case, we assume $\mathbb{G}_1 = \mathbb{G}_2$ and $g = g_1 = g_2$.

- $\hat{A}$ generates and sends $\{sid, \hat{A}, X = g^x, \tau_A = g^{\frac{1}{ah(\hat{A},A,X,sid)+x}}\}$; $\hat{B}$ generates and sends $\{sid, \hat{B}, Y = g^y, \tau_B = g^{\frac{1}{ah(\hat{B},B,Y,sid)+y}}\}$; $\hat{C}$ generates and sends back $\{sid, \hat{C}, Z = g^z, \tau_C = g^{\frac{1}{ah(\hat{C},C,Z,sid)+z}}\}$.
- After according validity verifications, the session-key is derived as $K = KDF(e(g,g)^{xyz}, \hat{A}, \hat{B}, \hat{C})$. Note that $e(g,g)^{xyz} = e(Y,Z)^x = e(X,Z)^y = e(Y,Z)^x$.

The case of sharing multiple session-keys between two parties:

- Player $\hat{A}$ generates and sends $\{sid, \hat{A}, X_1 = g_1^{x_1}, \cdots, X_t = g_1^{x_t}, \tau_A\}$, where for any $i, 1 \le i \le t, x_i \leftarrow Z_q^*$, $\tau_A = g_2^{\frac{1}{ah(sid,\hat{A},A)+x_1 f(X_1)+x_2 f(X_2)+\cdots+x_t f(X_t)}}$, where $f(X_i) = x_{X_i}$ or $f(X_i) = h(X_i)$.[29]
- Upon receiving $\{sid, \hat{A}, X_1, \cdots, X_t, \tau_A\}$, player $\hat{B}$ checks $X_i \in \mathbb{G}_1$ for each $i, 1 \le i \le t$, and $e(A^{h(\hat{A},A,sid)}X_1^{f(X_1)} \cdots X_t^{f(X_t)}, \tau_A) = e(g_1, g_2)$. *Note the computation of* $A^{h(\hat{A},A,sid)}X_1^{f(X_1)} \cdots X_t^{f(X_t)}$ *can be much more efficient than computing* $t+1$ *separate exponentiations.* If any verification fails, $\hat{B}$ aborts; Otherwise, $\hat{B}$ generates and sends $\{sid, \hat{B}, Y_1 = g_1^{y_1}, \cdots, Y_t = g_1^{y_t}, \tau_B\}$, where $\tau_B = g_2^{\frac{1}{bh(sid,\hat{B},B)+y_1 f(Y_1)+y_2 f(Y_2)+\cdots+y_t f(Y_t)}}$.

[27] In practice, *sid* could be the concatenation of two random nonces (or just the ephemeral DH-components) exchanged between $\hat{A}$ and $\hat{B}$; In some applications, *sid* could also be a time-stamp.

[28] In all non-malleable DHKE protocols based on (standard, ID-based, certificateless) NMSOK, players' identities can also be removed from the input of KDF.

[29] Other preferable computations of τ_A include: $\tau_A = g_2^{\frac{1}{af(A)+x_1 f(X_1)+x_2 f(X_2)+\cdots+x_t f(X_t)+h(\hat{A},sid)}}$, $\tau_A = g_2^{\frac{1}{a+x_1 f(X_1)+x_2 f(X_2)+\cdots+x_t f(X_t)+h(\hat{A},A,sid)}}$, etc.

- Session-key derivation: Each session run can generate t session-keys $K_i = KDF(g_1^{x_i y_i}, \hat{A}, \hat{B})$, $1 \leq i \leq t$. Actually, up to t^2 session-keys can be generated (suppose ephemeral DH-exponents are well protected).

The case of sharing multiple session-keys among three parties: In this case, again we assume $g = g_1 = g_2$ and $\mathbb{G}_1 = \mathbb{G}_2$.

- $\hat{A}$ sends $\{sid, \hat{A}, X_1 = g^{x_1}, \cdots, X_t = g^{x_t}, \tau_A\}$, where $\tau_A = g^{\frac{1}{ah(sid,\hat{A},A)+x_1 f(X_1)+x_2 f(X_2)+\cdots+x_t f(X_t)}}$;
  $\hat{B}$ generates and sends $\{sid, \hat{B}, Y_1, \cdots, Y_t, \tau_B\}$, where $\tau_B = g^{\frac{1}{bh(sid,\hat{B},B)+y_1 f(Y_1)+y_2 f(Y_2)+\cdots+y_t f(Y_t)}}$;
  $\hat{C}$ generates and sends $\{sid, \hat{C}, Z_1 = g^{z_1}, \cdots, Z_t = g^{z_t}, \tau_B\}$, where $\tau_C = g^{\frac{1}{ch(sid,\hat{C},C)+z_1 f(Z_1)+z_2 f(Z_2)+\cdots+z_t f(Z_t)}}$.
- Session-key derivation: After according verifications, each player can derive t-session keys $K_i = KDF(e(g,g)^{x_i y_i z_i}, \hat{A}, \hat{B}, \hat{C})$, $1 \leq i \leq t$. Actually, up to t^3 session-keys can be derived from one session run (assuming ephemeral DH-exponents are well protected).

13.3 Identity-Based (Certificateless) Non-Malleable Signature of Knowledge

In this section, for presentation simplicity, we use symmetric pairings, i.e., $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and $g_1 = g_2 = g$. The proposed schemes can also be trivially extended to the asymmetric pairing case. Generally speaking, for ID-based signatures, DH-components can be generated with g_1 as the base. But, for certificateless schemes, it is better to use g_2 as the base for forming public-keys in order to get shorter signatures.

Setup: The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. PKG also chooses two hash functions: $H : \{0,1\}^* \rightarrow \mathbb{G}/1_{\mathbb{G}}$ and $h : \{0,1\}* \rightarrow Z_q^*$. The public parameters are: $(G, G_T, \hat{e}, g, S, H, h)$.

User secret-key extract: For a user with identity $\hat{A}$, its public-key is set to be $A = H(\hat{A})$, and the PKG generates the associated secret-key of the user as $S_A = A^s$. Similarly, a user of identity $\hat{B}$ is of public-key $B = H(\hat{B})$ and secret-key $S_B = B^s$. And, a user of identity $\hat{C}$ is of public-key $C = H(\hat{C})$ and secret-key $S_C = C^s$.

In the following preferred solutions of ID-based (certificateless) non-malleable signature of knowledge (NMSOK), we assume $\hat{A}$ is the signer.

The first solution:

Signature generation: On a message $m \in \{0,1\}^*$ to be signed, the signer $\hat{A}$ computes $X = g^x$ where $x \leftarrow Z_q^*$, and $\tau = (S_A)^{\frac{1}{rh(m,X)+c}}$ where c can be one of the following:

1. $c = x_X$, where x_X is the x-coordinate of X. In case X is fixed (particularly for certificateless version, the value g^{x_X} can also be a part of the public-key of $\hat{A}$ for more efficient signature verification.
2. $c = x_S$, where x_S is the x-coordinate of S. In this case, g^{x_S} can also be treated as a part of the public-key of PKG.
3. $c \in Z_q^*$ is a random value, and c (together with g^c) is treated as a part of the public-key of PKG.
4. $c = h(m)$

The signature is (X, τ). (In case X, together with A, is treated as $\hat{A}$'s public-key, this scheme becomes a certificateless one.)

Signature verification: On (m, X, τ), the verifier checks whether $e(X^{h(m,X)}g^c, \tau)(= e(g, A^s)) = e(S, A)$, where $e(S, A)$ can be offline pre-computed.

The second solution:

Signature generation: On a message $m \in \{0,1\}^*$ to be signed, the signer $\hat{A}$ computes $X = g^x$ where $x \leftarrow Z_q^*$, and $\tau = (S_A)^{\frac{1}{xf(X)+h(m)}}$ where $f(X) = h(X)$ or $f(X) = x_X$. The signature is (X, τ). (In case X, together with A, is treated as $\hat{A}$'s public-key, this scheme becomes a certificateless one.)

Signature verification: On (m, X, τ), the verifier checks whether $e(X^{f(X)}g^{h(m)}, \tau)(= e(g, A^s)) = e(S, A)$, where $e(S, A)$ and $X^{f(X)}$ (in case X is fixed) can be offline pre-computed (for certificateless version, $X^{f(X)}$ can also be included in the public-key of $\hat{A}$.

The third solution:

Signature generation: On a message $m \in \{0,1\}^*$ to be signed, the signer $\hat{A}$ computes $X_1 = S_A^{x_1} = (A^s)^{x_1}$ and $X_2 = S_A^{x_2} = (A^s)^{x_2}$ where $x_1, x_2 \leftarrow Z_q^*$, and $\tau = g^{\frac{1}{x_1 h(m,X_1,X_2)+x_2}}$. The signature is (X_1, X_2, τ).

Signature verification: On (m, X_1, X_2, τ), the verifier checks whether $e(\tau, X_1^{h(m,X_1,X_2)}X_2)(= e(g, A^s)) = e(S, A)$, where $e(S, A)$ can be offline pre-computed.

Remark: One advantage of the third solution is that: the signature (X_1, X_2, τ) well hides the values x_1, x_2 (and thus S_A), as the generator S_A is private. Also putting S_A as the generator in computing X_1, X_2 has the advantage that signer's secret-key is only involved in offline pre-computation, and thus can be much better protected. Another advantage is the more efficient signature verification. By resettability of RNME-JOWF, the values X_1, X_2 can be reused for multiple signatures. Note that in case X_1, X_2 (of one of them) are fixed once and for all and are treated (together with $A = H(\hat{A})$)as the public-key of $\hat{A}$, the above signature is actually a certificateless one.

The fourth solution:

Signature generation: On a message $m \in \{0,1\}^*$ to be signed, the signer $\hat{A}$ computes $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$ where $x_1, x_2 \leftarrow Z_q^*$, and $\tau = (S_A)^{\frac{1}{x_1 h(m,X_1,X_2)+x_2}}$. The signature is (X_1, X_2, τ).

Signature verification: On (m, X_1, X_2, τ), the verifier checks whether $e(X_1^{h(m,X_1,X_2)}X_2, \tau)(= e(g, A^s)) = e(S, A)$, where $e(S, A)$ can be offline pre-computed.

Again, in case X_1, X_2 (or one of them, say X_1) are fixed once and for all, and are treated (together with $A = H(\hat{A})$)as the public-key of $\hat{A}$, the above signature becomes a certificateless one.

13.4 ID-Based (Certificateless) Non-Malleable DHKE

We note that the first and second solutions of ID-based non-malleable signature of knowledge can be directly used to achieve ID-based non-malleable DHKE, and the fourth solution can be used to achieve ID-based NMDHKE (with 2 session-keys generated per session run) and certificateless NMDHKE (with a single session-key generated per session run) between two parties or among three parties.

The general idea is: each player signs the session-identifier and its own identity, e.g, $(sid, \hat{A})$ for the player of $\hat{A}$. That is, in the above NMSOK solutions, the message m to be signed is now sid and player's own identity.[30]. We assume $(Y = g^y, \tau_B)$ (resp., $(Z = g^z, \tau_C)$) the values generated by $\hat{B}$ (resp., $\hat{C}$) with the first or second solution of NMSOK, and assume $(Y_1 = g^{y-1}, Y_2 = g^{y_2}, \tau_B)$ (resp., $(Z_1 = g^{z_1}, Z_2 = g^{z_2}, \tau_C)$) the values generated by $\hat{B}$ (resp., $\hat{C}$) with the fourth solution of NMSOK. In particular, for certificateless implementation based on the fourth solution of NMSOK, we assume Y_1 (resp., Z_1) is fixed and is a part of the public-key of $\hat{B}$ (resp., $\hat{C}$).

After according verifications (group membership of peer's DH-components, and the validity of the authenticated value τ), the session-key is derived from $KDF(g^{xy}, \hat{A}, \hat{B})$ for a two-party run, or $KDF(e(g,g)^{xyz}, \hat{A}, \hat{B}, \hat{C})$ for a three-party run for ID-based non-malleable DHKE based on the first and second solutions of NMSOK. For ID-based non-malleable DHKE based on the fourth solution, two session-keys $K_i = KDF(g^{x_i y_i}, \hat{A}, \hat{B})$ (resp, $K_i = KDF(e(g,g)^{x_i y_i z_i}, \hat{A}, \hat{B}, \hat{C})$), $1 \leq i \leq 2$ can be derived for ID-based non-malleable DHKE between two parties $\hat{A}, \hat{B}$ (resp, three parties $\hat{A}, \hat{B}, \hat{C}$). Actually, up to 4 (resp., 8) session-keys $K_{i,j} = KDF(g^{x_i y_j}, \hat{A}, \hat{B})$, $1 \leq i, j \leq 2$ (resp, $K_{i,j,k} = KDF(e(g,g)^{x_i y_j z_k}, \hat{A}, \hat{B}, \hat{C})$), $1 \leq i, j, k \leq 2$) can be derived for ID-based non-malleable DHKE between two parties (resp., three parties). For certificateless implementation based on the fourth solution of NMSOK, a session-key $KDF(g^{x_2 y_2}, \hat{A}, \hat{B})$ (resp., $KDF(e(g,g)^{x_2 y_2 z_2}, \hat{A}, \hat{B}, \hat{C})$) can be generated per session run between two parties (resp., three parties).

Solutions for ID-based (certificateless) batch two-party or triparty NMDHKE:

- $\hat{A}$ sends $\{sid, X_1 = g^{x_1}, \cdots, X_t = g^{x_t}, \tau_A\}$, where for certificateless version X_1 is fixed and is a part of the public-key of $\hat{A}$, τ_A can have the following preferable computations:

  1. $\tau_A = S_A^{\frac{1}{x_1 f_1(X_1) + \cdots + x_t f_t(X_t) + h(sid, \hat{A}) + \hat{c}}}$, where $f_i(X_i) = h(X_i)$ or $f_i(X_i) = x_{X_i}$, $1 \leq i \leq t$, $\hat{c} = 0$ or $\hat{c}$ is a random value that is published (together with $g^{\hat{c}}$) as a part of the public-key of PKG. Note that the validity of τ_A can be publicly checked, by checking whether $e(X_1^{f_1(X_1)} \cdots X_t^{f_t(X_t)} g^{h(sid,\hat{A})} g^{\hat{c}}, \tau_A)(= e(g, S_A)) = e(S, A)$.
  2. $\tau_A = S_A^{\frac{1}{x_1 h(sid, \hat{A}, X_1) + x_2 f_2(X_2) \cdots + x_t f_t(X_t) + \hat{c}}}$. The validity of τ_A can be publicly checked, by checking whether $e(X_1^{h(sid,\hat{A},X_1)} \cdots X_t^{f_t(X_t)} g^{\hat{c}}, \tau_A) = e(S, A)$.

- $\hat{B}$ sends $\{sid, Y_1 = g^{y_1}, \cdots, Y_t = g^{y_t}, \tau_B\}$, where for certificateless version Y_1 is fixed and is a part of the public-key of $\hat{B}$, the computation of τ_B has the following preferable computations:

  1. $\tau_B = S_B^{\frac{1}{y_1 f(Y_1) + \cdots + y_t f(Y_t) + h(sid, \hat{B}) + \hat{c}}}$.
  2. $\tau_B = S_B^{\frac{1}{y_1 h(sid, \hat{B}, Y_1) + y_2 f_2(Y_2) \cdots + y_t f(Y_t) + \hat{c}}}$.

- $\hat{C}$ sends $\{sid, Z_1 = g^{z_1}, \cdots, Z_t = g^{z_t}, \tau_C\}$, where for certificateless version Z_1 is fixed and is a part of the public-key of $\hat{C}$, the computation of τ_C has the following preferable computations:

  1. $\tau_C = S_C^{\frac{1}{z_1 f(Z_1) + \cdots + z_t f(Z_t) + h(sid, \hat{C}) + \hat{c}}}$.
  2. $\tau_C = S_C^{\frac{1}{z_1 h(sid, \hat{C}, Y_1) + z_2 f_2(Z_2) \cdots + z_t f(Z_t) + \hat{c}}}$.

[30] In case of no session-identifier, each player needs also to at least sign its peer's identity or both its peer's identity and DH-components.

- Mutual authentication and key derivation: For DHKE between two parties $\hat{A}$ and $\hat{B}$, $\hat{A}$ checks the group membership of $\mathbb{G}$ of its peer's DH-components and the validity of τ_B. If all the verifications are successful, $\hat{A}$ computes t (resp., $t-1$ for certificateless implementation) session-keys $K_i = KDF(g^{x_i y_i}, \hat{A}, \hat{B})$ for $1 \leq i \leq t$ (resp., $2 \leq i \leq t$). Actually, $\hat{A}$ can compute out up to t^2 $K_{i,j} = KDF(g^{x_i y_j}, \hat{A}, \hat{B})$, $1 \leq i, j \leq t$ (resp., $(t-1)^2$ $K_{i,j} = KDF(g^{x_i y_j}, \hat{A}, \hat{B})$, $2 \leq i, j \leq t$, for certificateless implementation) session-keys per session run (assuming all the ephemeral DH-exponents are well protected). For DHKE among three parties $\hat{A}, \hat{B}, \hat{C}$, $\hat{A}$ checks the group membership of $\mathbb{G}$ of DH-components from $\hat{B}, \hat{C}$, and the validity of τ_B and τ_C. If all the verifications are successful, $\hat{A}$ computes t (resp., $t-1$ for certificateless implementation) session-keys $K_i = KDF(e(g,g)^{x_i y_i z_i}, \hat{A}, \hat{B}, \hat{C})$ for $1 \leq i \leq t$ (resp., $2 \leq i \leq t$). Actually, $\hat{A}$ can compute out up to t^3 session-keys $K_{i,j,k} = KDF(e(g,g)^{x_i y_j z_k}, \hat{A}, \hat{B}, \hat{C})$ for $1 \leq i, j, k \leq t$ (resp., $(t-1)^3$ session-keys $K_{i,j,k} = KDF(e(g,g)^{x_i y_j z_k}, \hat{A}, \hat{B}, \hat{C})$ for $2 \leq i, j, k \leq t$ for certificateless implementation) per session run (assuming all the ephemeral DH-exponents are well protected).[31]

13.5 Non-Malleable ID-Based Encryption

For presentation simplicity, we propose the preferable schemes based on symmetric pairings (the extension to asymmetric pairings is direct).

The first (preferable) solution:

Setup: The PKG picks $r_0, r_1, x \leftarrow Z_q^*$, and computes $X = g^x$. The master public-key is $mpk = (g, R_0 = g^{r_0}, r_1, X, e, e(g,g))$, where $e : G \times G \rightarrow \mathbb{G}_T$, and the master secret-key is x.[32]

Key Generation: To create the secret-key for $ID \in \{0,1\}^*$, the PKG computes $sk_{ID} = g^{\frac{1}{xh(X,ID)+r_0}} \in G/1_G$.[33] Note that the validity of sk_{ID} can be publicly verified by checking $e(T_{ID}, sk_{ID}) = e(g,g)$, where $T_{ID} = X^{h(X,ID)} R_0$.

For improving encryption efficiency, the value T_{ID} can be pre-computed and be a part of the public-key of the user ID. To improve decryption efficiency, the values $T_{ID}^{r_1}$ and $e(T,T)$ can serve as a part of the secret-key of user ID.

Encryption: To encrypt a message $m \in G$ for user ID, picks $r \leftarrow Z_q^*$, computes $R = T_{ID}^r (= X^{h(X,ID)r} \cdot g^{r_0 r})$, and $C = e(g,g)^r \cdot m, \tau = T_{ID}^{\frac{1}{rh(R,C)+r_1}} (= X^{h(X,ID)\frac{1}{rh(R,C)+r_1}} \cdot g^{r_0 \frac{1}{rh(R,C)+r_1}})$. The ciphertext is (R, C, τ).[34]

Decryption: On (R, C, τ), user ID first checks whether $R \in G/1_G$ and $e(R^{h(R,C)} \cdot T_{ID}^{r_1}, \tau) = e(T,T)$. If the check fails, it outputs $\perp$ indicating invalid ciphertext; otherwise, it output $m = C/e(R, sk_{ID})$.

[31] In practice, $g^{x_i y_i}$ or $e(g,g)^{x_i y_i z_i}$ in the input of KDF can simply be $x_{g^{x_i y_i}}$ or $x_{e(g,g)^{x_i y_i z_i}}$, i.e., their corresponding x-coordinates. Such a treatment applies also to other solutions.

[32] For shorter and simpler mpk, we can set $r_0 = r_1$, and even simply $r_0 = r_1 = x_X \mod q$.

[33] In case $xh(X,ID) + r_0 = 0$, we can concatenate a random string to ID to force $xh(X,ID) + r_0 \neq 0$. Another solution (for certificateless implementation) is to let mpk include a list of r_i, and to find an r_i such that $xh(X,ID) + r_i \neq 0$, in this case the r_i should be included into user's public-key.

[34] There is another alternative encryption: we derive two keys (K_1, K_2) from $e(g,g)^r$ (e.g, $(K_1, K_2) = KDF(x_{e(g,g)^r}, R)$), and then use K_1 to encrypt m (e.g., by a symmetric encryption algorithm) to get C and authenticate R, C using a MAC with K_2 serving as the MAC-key to get τ. Such alternative solution applies also to all non-malleable encryption schemes proposed in this section.

Notes: The encrypter can pre-compute and store T_{ID} (that is fixed for user ID). Also, user ID (i.e., the decryptor) can also pre-compute $e(T,T)$ and T^{r_1}. Thus, the decryptor only performs about 1 exponentiation and two pairings.

The second (preferable) solution:

Setup: The PKG computes $X = g^x$ and $Y = g^y$, where $x, y \leftarrow Z_q^*$. The master public-key is $mpk = (g, X, Y, e)$, where $e : G \times G \rightarrow \mathbb{G}_T$, and the master secret-key is (x, y).

Key Generation: To create the secret-key for $ID \in \{0,1\}^*$, the PKG computes $sk_{ID} = g^{\frac{1}{x+yh(X,Y,ID)}}$. For improving online efficiency, the value $T = XY^{h(X,Y,ID)}$ can be pre-computed and can serve as a part of user's public-key, the value $e(T,T)$ can serve as a part of user's secret-key.

Encryption: To encrypt a message $m \in G$ for user ID, picks $r, r' \leftarrow Z_q^*$, computes $R = T^r$ and $R' = T^{r'}$, $C = e(g,g)^r \cdot m$ and $\tau = T^{\frac{1}{rh(R,R',C)+r'}}$. The ciphertext is (R, R', C, τ).

Decryption: On (R, R', C, τ), user ID first checks whether $R, R' \in G/1_G$ and $e(R^{h(R,R',C)} \cdot R', \tau) = e(T,T)$. If the check fails, it outputs $\perp$ indicating invalid ciphertext; otherwise, it output $m = C/e(R, sk_{ID})$.

The third (preferable) solution:

Setup: The PKG computes $X = g^x$ and $Y = g^y$, where $x, y \leftarrow Z_q^*$. The master public-key is $mpk = (g, X, Y, e)$, where $e : G \times G \rightarrow \mathbb{G}_T$, and the master secret-key is (x, y).

Key Generation: To create the secret-key for $ID \in \{0,1\}^*$, the PKG computes $sk_{ID} = g^{\frac{1}{x+yh(X,Y,ID)}}$. For improving online efficiency, the value $T = XY^{h(X,Y,ID)}$ can be pre-computed and can serve as a part of user's public-key, the value $e(T,T)$ can serve as a part of user's secret-key.

Encryption: To encrypt a message $m \in G$ for user ID, picks $r \leftarrow Z_q^*$, computes $R = T^r$, $C = e(g,g)^r \cdot m, \tau = T^{\frac{1}{rf(R)+h(C)}}$, where $f(R) = h(R)$ or $f(R) = x_R$. [35] The ciphertext is (R, C, τ). The advantage of this solution is of shorter ciphertext and more efficient encryption.

Decryption: On (R, C, τ), user ID first checks whether $R \in G/1_G$ and $e(R^{f(R)} \cdot T^{h(C)}, \tau) = e(T,T)$. If the check fails, it outputs $\perp$ indicating invalid ciphertext; otherwise, it output $m = C/e(R, sk_{ID})$.

13.6 (Universal) Non-Malleable Verifiable Random Functions

By (universal) non-malleable verifiable random function (NMVRF), we mean that for any PPT adversary $\mathcal{A}$ there exists another PPT algorithm E such that, after seeing many VRF input-output pairs, denoted $\mathcal{Q}$, obtained adaptively by any PPT adversary $\mathcal{A}$ under a public-key PK, if $\mathcal{A}$ is able to output a valid VRF output, denoted σ, under public-key PK' (and parameters) chosen by $\mathcal{A}$ itself, then on input $(PK, \mathcal{Q}, r_{\mathcal{A}}, PK', \sigma)$ the algorithm E outputs the secret-key corresponding to PK' with overwhelming probability, where $r_{\mathcal{A}}$ is the random coins of $\mathcal{A}$.

Based on the non-malleable signature of knowledge and the VRF of [48], we present two (preferable) solutions of NMVRF.

The first solution:

[35] Other alternatives of τ include: $\tau = T^{\frac{1}{rh(R,C)+\hat{c}}}$, where $\hat{c}$ can be one of the following: a random value $\hat{c} \in Z_q^*$ that is included into mpk (in this case, $T^{\hat{c}}$ can also be a part of user's public-key for certificateless implementation), x_R, x_X, x_Y, x_C, etc.

Public-key: $\{g, \mathbb{G}, \mathbb{G}_T, e, h, X = g^x, C = g^c\}$, where x and c are taken uniformly at random from Z_q^*.

Secret-key: x, c

PRF evaluation: On input m, computes $\{R = e(g,g)^{\frac{1}{xh(m,X,C)+c}}, \tau = g^{\frac{1}{xh(m,X,C)+c}}\}$

Public verification: On (m, R, τ), the verifier checks whether $e(\tau, X^{h(m,X,C)}C) = e(g,g)$ and $e(g,\tau) = R$.

The second solution:

- Public-key $PK = X = g^x$ and secret-key $SK = x \in Z_q^*$.
- PRF evaluation: for $m \in Z_q$, $PRF_{SK}(x) = \{R = e(g,g)^{\frac{1}{xh(X,m)+m}}, \tau = g^{\frac{1}{xh(X,m)+m}}\}$
- Public verification: Given (R, τ) checks whether the following two equations both hold: $e(g^m X^{h(X,m)}, \tau) = e(g,g)$ and $e(g,\tau) = R$.

13.7 Non-Malleable Set Membership and Range Proofs

Set membership and range proofs are a special kind of zero-knowledge proofs. With a membership proof, the prover is able to convince the verifier of the fact that: it possesses a secret value (or, digitally committed value) that is a member of a given public set. Range proofs are in turn a special kind of range proofs, where the given public set is a specified integer range.

Set membership and range proofs are fundamental building tools in various cryptographic applications, e.g, anonymous credential, e-cash, e-voting, etc.

13.7.1 Motivation for *Non-Malleable* Set Membership and Range Proofs

Consider the following motivating example for range proofs (which is a variant of that given in [124]): Suppose there is a firewall which grants the access of some private network only to users from a specific range of IP address, say with IP address of the form "10.x.y.z" such that $1 \leq x, y, z \leq 100$. Each user when accessing the private network has to prove to the firewall that he has a valid credential corresponding to his IP address, while he does not want to reveal his actual IP address due to some privacy concern. Suppose the user's IP address is 10.68.0.1 and he holds an anonymous credential for his IP value $172228609 = 10 \times 256^3 + 68 \times 256^2 + 1$. The user can prove to the firewall, using the range proof, that his IP value lies in the range $[10 \times 256^3, 10 \times 256^3 + 100 \times 256^2 + 100 \times 256 + 100]$, without revealing exactly his IP address is.

However, suppose there is another firewall which grants the access of some private network only to users, with IP addresses of the form "$20.x'.y'.y'$" where $0 \leq x', y', z' \leq 200$. Now, a man-in-the-middle (MIM) adversary, seeing range proofs for the range $[10 \times 256^3, 10 \times 256^3 + 100 \times 256^2 + 100 \times 256 + 100]$, can potentially malleate the range proof for $[10 \times 256^3, 10 \times 256^3 + 100 \times 256^2 + 100 \times 256 + 100]$ into a valid range proof for $[20 \times 256^3, 20 \times 256^3 + 200 \times 256^2 + 200 \times 256 + 200]$, while the adversary does not hold a credential for the IP address "$20.x'.y'.y'$". This shows the importance of achieving non-malleability for set membership and range proofs.

Roughly speaking, by a non-malleable set membership (NMSM) or (NMR) range proof, we mean that whenever the adversary successfully finishes a set membership or range proof *w.r.t. a set or range (possibly with digital commitment to the private set or range vable) that is different from what being proved by the honest provers*, the adversary does indeed "know" the corresponding the (private) set or range member. We note that most exiting set membership or range proofs in the

literature do not consider or achieve non-malleability. In general, non-malleable set membership or range proof can be defined and achieved with some general paradigms in the literature of non-malleable zero-knowledge, but the resultant schemes with the general paradigms are usefully not efficient enough to be employed in practice.

13.7.2 Analysis and Achieving NMR Proof Based on YHMSWY scheme

The YHMSWY scheme, which is non-interactive in the common reference string (CRS) model, is based on [87, 63], and is particularly based on the following Lagrange's theorem from the year 1770.

Theorem 13.1 (Lagrange, 1770) *For an integer μ, $\mu \geq 0$ if and only if there exist (efficiently computable) integers $\omega_1, \omega_2, \omega_3, \omega_4$ such that $\mu = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$.*

Based on Lagrange's theorem, given a range $[a, b]$, to prove the knowledge of a private value $\mu \in [a, b]$, one just needs to prove that both $\mu_1 = \mu - a$ and $\mu_2 = b - \mu$ are non-negative. This in turn corresponds to prove the knowledge of $(\mu, \omega_{i,j}, i \in \{0, 1\}, j \in \{1, 2, 3, 4\})$ such that $\mu_i = \omega_{i,1}^2 + \omega_{i,2}^2 + \omega_{i,3}^2 + \omega_{i,4}^2$ for $i \in \{1, 2\}$.

The YHMSWY is based on the non-interactive witness indistinguishable proof presented in [63], which is in turn based on the subgroup decision problem. Let $\mathbb{G}, \mathbb{G}_T$ be groups of order $n = pq$, where p and q are prime. Let g be the generator $\mathbb{G}$, and denote by $\mathbb{G}_q$ the subgroup of $\mathbb{G}$ of order q. The subgroup decision problem roughly says that given an element $u \in \mathbb{G}$ it is hard to distinguish whether $u \in G$ or $u \in \mathbb{G}_q$.

The YHMSWY construction. On a security parameter k, the YHMSWY construction is recalled below:

Setup $\mathcal{G}(1^k)$. $\mathcal{G}$ outputs $gk = (n, \mathbb{G}, \mathbb{G}_T, e)$ and $sk = (p, q)$.

CRS generation $K(gk, sk)$. K randomly selects a generator g of $\mathbb{G}$ and a generator u of $\mathbb{G}_q$, and outputs the CRS $\sigma = (g, u)$.

Range proof $P(gk, \sigma, (a, b), \mu, \omega_{i,j}, i \in \{1, 2\}, j \in \{1, 2, 3, 4\})$. The prover P randomly chooses $r_{i,j} \in_R \mathcal{Z}_n$ and computes $T_{i,j} = g^{\omega_{i,j}} u^{r_{i,j}}$ (i.e., commitment to $\omega_{i,j}$ using randomness $r_{i,j}$). Then, P randomly picks $r_\mu \in_R \mathcal{Z}_n$ and computes $T_\mu = g^\mu u^{r_\mu}$. P then calculates:

$$\phi_1 = g^{-r_\mu + 2\sum_{j=1}^4 r_{1,j}\omega_{1,j}} u^{\sum_{j=1}^4 r_{1,j}^2},$$

$$\phi_2 = g^{r_\mu + 2\sum_{j=1}^4 r_{2,j}\omega_{2,j}} u^{\sum_{j=1}^4 r_{2,j}^2}$$

P sends the proof $\pi = (\{T_{1,j}, T_{2,j}\}_{j\in[4]}, T_\mu, \phi_1, \phi_2)$ to the verifier V.

Proof verification. V outputs 1 if the following equations hold, otherwise it outputs 0.

$$e(g^a T_\mu^{-1}, g) \cdot \prod_{j=1}^4 e(T_{1,j}, T_{1,j}) = e(u, \phi_1)$$

$$e(T_\mu g^{-b}, g) \cdot \prod_{j=1}^4 e(T_{2,j}, T_{2,j}) = e(u, \phi_2).$$

A MIM attack against YHMSWY scheme: Given a proof $\pi = (\{T_{1,j}, T_{2,j}\}_{j\in[4]}, T_\mu, \phi_1, \phi_2)$ for the knowledge $\mu \in [a, b]$, a MIM adversary can construct a proof $\pi' = (\{T'_{1,j}, T'_{2,j}\}_{j\in[4]}, T'_\mu, \phi'_1, \phi'_2)$ for proving the knowledge of $\mu' \in [a', b'] = [4a, 4b]$, by setting:

$$\phi'_1 = \phi_1^4 (= g^{-4r_\mu + 8\sum_{j=1}^4 r_{1,j}\omega_{1,j}} u^{4(r_{1,j})^2}),$$
$$\phi'_2 = \phi_2^4 (= g^{4r_\mu + 8\sum_{j=1}^4 r_{2,j}\omega_{2,j}} u^{4(r_{2,j})^2}),$$
$$T'_{i,j} = T_{i,j}^2 (= g^{2\omega_{i,j}} u^{2r_{i,j}}),$$
$$T'_\mu = T_\mu^4 (= g^{4\mu} u^{4r_\mu}).$$

The above MIM attack is valid, by observing that:

$$\begin{aligned}
& e(g^{a'}(T'_\mu)^{-1}, g) \cdot \prod_{j=1}^4 e(T'_{1,j}, T'_{1,j}) \\
= \; & e(g^{4a}(T_\mu^4)^{-1}, g) \cdot \prod_{j=1}^4 e(g^{2\omega_{1,j}} u^{2r_{1,j}}, g^{2\omega_{1,j}} u^{2r_{1,j}}) \\
= \; & e(g^{4a-4\mu} u^{-4r_\mu}, g) \cdot \prod_{j=1}^4 \Big(e(g,g)^{4\omega_{1,j}^2} \cdot e(u,g)^{8r_{1,j}\omega_{1,j}} \cdot e(u, u^{4r_{1,j}^2}) \Big) \\
= \; & e(g,g)^{4(\sum_{j=1}^4 \omega_{1,j}^2 + a - \mu)} \cdot e(u, g^{-4r_\mu + 8\sum_{j=1}^4 r_{1,j}\omega_{1,j}} u^{4r_{1,j}^2}) \\
= \; & e(u, \phi'_1).
\end{aligned}$$

$$\begin{aligned}
& e(T'_\mu g^{-b'}, g) \cdot \prod_{j=1}^4 e(T'_{2,j}, T'_{2,j}) \\
= \; & e(g^{4\mu - 4b} u^{4r_\mu}, g) \cdot \prod_{j=1}^4 e(g^{2\omega_{2,j}} u^{2r_{2,j}}, g^{2\omega_{1,j}} u^{2r_{2,j}}) \\
= \; & e(g^{4\mu - 4b} u^{4r_\mu}, g) \cdot \prod_{j=1}^4 \Big(e(g,g)^{4\omega_{2,j}^2} \cdot e(u,g)^{8r_{2,j}\omega_{2,j}} \cdot e(u, u^{4r_{2,j}^2}) \Big) \\
= \; & e(g,g)^{4(\sum_{j=1}^4 \omega_{2,j}^2 + \mu - b)} \cdot e(u, g^{4r_\mu + 8\sum_{j=1}^4 r_{2,j}\omega_{2,j}} u^{4r_{2,j}^2}) \\
= \; & e(u, \phi'_2).
\end{aligned}$$

Note: Needless to say, other similar attacks against the YHMSWY scheme can also be demonstrated. Here, we remark that such MIM attacks do not violate the security proof presented in [124], as the security proof conducted in [124] does not take non-malleability into account.

Practical approach for achieving non-malleable range proof. Let $H : \{0,1\}^* \to Z_n^*$ be a hash function, and define $h_\mu = H(T_{1,1}, \cdots, T_{2,4})$. That is, h_μ essentially commits to $T_{i,j}$'s, $i \in \{1,2\}$ and $j \in [4]$. The non-malleable version of the YHMSWY scheme is described below:

Setup $\mathcal{G}(1^k)$. $\mathcal{G}$ outputs $gk = (n, \mathbb{G}, \mathbb{G}_T, e)$ and $sk = (p, q)$.

CRS generation $K(gk, sk)$. K randomly selects a generator g of $\mathbb{G}$ and a generator u of $\mathbb{G}_q$, and outputs the CRS $\sigma = (g, u)$.

Range proof. The prover P randomly chooses $r_{i,j} \in_R \mathcal{Z}_n$ and $r_\mu \in_R \mathcal{Z}_n$, and computes $T_{i,j} = g^{\omega_{i,j}} u^{r_{i,j} h_{i,j}}$. Then, P randomly picks $r_\mu \in_R \mathcal{Z}_n$ and computes $T_\mu = g^\mu u^{r_\mu h_\mu}$, where $h_\mu = H(T_{1,1}, \cdots, T_{2,4})$. P then calculates:

$$\phi_1 = g^{-r_\mu h_\mu + 2\sum_{j=1}^4 r_{1,j}\omega_{1,j}} u^{\sum_{j=1}^4 r_{1,j}^2},$$

$$\phi_2 = g^{r_\mu h_\mu + 2\sum_{j=1}^{4} r_{2,j}\omega_{2,j}} u^{\sum_{j=1}^{4} r_{2,j}^2}$$

P sends the proof $\pi = (\{T_{1,j}, T_{2,j}\}_{j\in[4]}, T_\mu, \phi_1, \phi_2)$ to the verifier V.

Proof verification. V outputs 1 if the following equations hold, otherwise it outputs 0.

$$e(g^a T_\mu^{-1}, g) \cdot \prod_{j=1}^{4} e(T_{1,j}, T_{1,j}) = e(u, \phi_1)$$

$$e(T_\mu g^{-b}, g) \cdot \prod_{j=1}^{4} e(T_{2,j}, T_{2,j}) = e(u, \phi_2).$$

Completeness of non-malleable range proof.

$$\begin{aligned}
& e(g^a (T_\mu)^{-1}, g) \cdot \prod_{j=1}^{4} e(T_{1,j}, T_{1,j}) \\
=\ & e(g^{a-\mu} u^{-r_\mu h_\mu}, g) \cdot \prod_{j=1}^{4} e(g^{\omega_{1,j}} u^{r_{1,j}}, g^{\omega_{1,j}} u^{r_{1,j}}) \\
=\ & e(g^{a-\mu} u^{-r_\mu h_\mu}, g) \cdot \prod_{j=1}^{4} \left(e(g,g)^{\omega_{1,j}^2} \cdot e(u,g)^{2r_{1,j}\omega_{1,j}} \cdot e(u, u^{r_{1,j}^2}) \right) \\
=\ & e(g,g)^{\sum_{j=1}^{4} \omega_{1,j}^2 + a - \mu} \cdot e(u, g^{-r_\mu h \mu + 2\sum_{j=1}^{4} r_{1,j}\omega_{1,j}} u^{r_{1,j}^2}) \\
=\ & e(u, \phi_1).
\end{aligned}$$

$$\begin{aligned}
& e(T_\mu g^{-b}, g) \cdot \prod_{j=1}^{4} e(T_{2,j}, T_{2,j}) \\
=\ & e(g^{\mu-b} u^{r_\mu h_\mu}, g) \cdot \prod_{j=1}^{4} e(g^{\omega_{2,j}} u^{r_{2,j}}, g^{\omega_{1,j}} u^{r_{2,j}}) \\
=\ & e(g^{\mu-b} u^{r_\mu h_\mu}, g) \cdot \prod_{j=1}^{4} \left(e(g,g)^{\omega_{2,j}^2} \cdot e(u,g)^{2r_{2,j}\omega_{2,j}} \cdot e(u, u^{2r_{2,j}^2}) \right) \\
=\ & e(g,g)^{\sum_{j=1}^{4} \omega_{2,j}^2 + \mu - b} \cdot e(u, g^{r_\mu h_\mu + 2\sum_{j=1}^{4} r_{2,j}\omega_{2,j}} u^{r_{2,j}^2}) \\
=\ & e(u, \phi_2).
\end{aligned}$$

Perfect soundness of the non-malleable range proof is derived from the perfectly-binding property of the underlying commitments (with u is of order q).

Zero-knowledge property of the non-malleable range proof is derived from the perfectly-hiding trapdoorness of the underlying commitments *when u is set to be order n.*

Non-malleability arguments: The only differences between the non-malleable version and the original YHMSWY scheme lie in the change of T_μ from $g^\mu u^{r_\mu}$ is changed to be $g^\mu u^{r_\mu h_\mu}$, and the corresponding change of ϕ_1, ϕ_2. Note that, by the perfectly-binding property of the underlying commitment, $g^\mu u^{r_\mu h_\mu}$ commits to h_μ, which in turn commits to $T_{i,j}$'s.

With the non-malleable version of YHMSWY scheme, to successfully malleate a proof for the range $[a, b]$ into a proof for a different range $[a', b']$, an MIM adversary has to modify some $T_{i,j}$'s. This

makes $T_\mu = g^\mu u^{r_\mu} h_\mu$ in the original proof for the range $[a, b]$ useless for generating $T'_\mu = g^{\mu'} u^{r'_\mu} h'_\mu$. In particular, in the random oracle model (assuming H as a random oracle) T'_μ and T_μ are independent. Thus, to successfully pass the verification procedures, the MIM adversary has to honestly generate $T'_\mu = g^{\mu'} u^{r'_\mu} h'_\mu$ (particularly with the knowledge of μ'.

Note: Another (more inefficient) alternative fixing approach is: defining $h_\mu = H(T_\mu)$, and $T_{i,j}$ is changed to be $g^{\omega_{i,j}} u r_{i,j} h_\mu$, with according changes in ϕ_1 and ϕ_2.

Note: As the YHMSWY scheme is based on the paring-based non-interactive protocols developed in [63], the above NM fixing approach can also be applied to most protocols developed in [63] to obtain the corresponding practical non-malleable versions (which will be presented in the full formal version of this work).

13.7.3 Analysis and Achieving NMSM argument Based on CCS scheme

The CCS set membership proof [23] is based upon Boneh-Boyen short signatures, which is recalled below:

Common input: g, h, a commitment C, and a set Φ, where g, h are generators of a group $\mathbb{G}$ of order p.

Prover input: σ,r such that $C = g^\sigma h^r$ and $\sigma \in \Phi$.

- Verifier randomly picks $x \in_R Z_p$ and sends $y = g^x$ and $\{A_i = g^{\frac{1}{x+i}}\}_{i \in \Phi}$.
- Prover randomly picks $v \in_R Z_p$ and sends $V = A_\sigma^v$. Then prover proves to the verifier of the knowledge (σ, r, v) such that $C = g^\sigma h^r$ and $V = g^{\frac{v}{x+\sigma}}$, via the following Σ-protocol:
  - Prover randomly picks $s, t, m \in_R Z_p$ and sends $a = e(V, g)^{-s} e(g, g)^t$ and $D = g^s h^m$.
  - Verifier sends a random challenge $c \in_R Z_p$.
  - Prover sends $z_\sigma = s - \sigma c$, $z_v = t - vc$ and $z_r = m - rc$. Verifier outputs "accept" if and only if: $D = C^c h^{z_r} g^{z_\sigma}$ and $a = e(V, y)^c \cdot e(V, g)^{-z_\sigma} \cdot e(g, g)^{z_{z_v}}$.

An MIM attack. We show an MIM attack, in which an MIM adversary $\mathcal{A}$, by interacting with an honest prover on the left on common inputs $C' = g^{\sigma'} h^{r'}$ and Φ', can successfully prove to an honest verifier on the right on the inputs $C = g^\sigma h^r$ and Φ, where $\sigma' = 2\sigma$, $r' = 2r$ and $\Phi' = 2\Phi$.

- After receiving $y = g^x$ and $\{A_i = g^{\frac{1}{x+i}}\}_{i \in \Phi}$ from verifier on the right, the MIM adversary $\mathcal{A}$ computes and sends $y' = y^2$ and $\{A_{i'} = A_i^{\frac{1}{2}} (= g^{\frac{1}{2x+2i}})\}_{i'=2i \in \Phi'}$ to the prover.
- After receiving $V' = A_{\sigma'}^{v'} (= (A_\sigma^{\frac{1}{2}v'}))$ from prover on the left, $\mathcal{A}$ sends $V = V'^2 (= A_\sigma^{v'} = A_\sigma^v)$ to verifier on the right. Note that $v = v'$.
- After receiving $a' = e(V', g)^{-s'} e(g, g)^{t'}$ and $D' = g^{s'} h^{m'}$ from prover on the left, $\mathcal{A}$ computes and sends $a = a'^2$ and $D = D'$ to verifier on the right.

  Note that $a' = e(A_\sigma^{v'}, g)^{-s'/2} e(g, g)^{t'} = e(V, g)^{-s'/2} e(g, g)^{t'}$, $a = a'^2 = e(V, g)^{-s'} e(g, g)^{2t'}$. Define $s = s', t = 2t'$ and $m = m'$, we have a and D on the right are in the correct form. Specifically, $a = e(V, g)^{-s'} e(g, g)^{2t'} = e(V, g)^{-s} e(g, g)^t$ and $D = D' = g^s h^m$.
- After receiving c from verifier on the right, $\mathcal{A}$ sends $c' = c/2$ to prover on the right.
- After receiving $z_{\sigma'}, z_{v'}, z_{r'}$ from prover on the left, $\mathcal{A}$ sends $z_\sigma = z_{\sigma'}, z_v = 2z_{v'}, z_r = z_{r'}$ to verifier on the right.

Recall $\sigma' = 2\sigma, r' = 2r, v = v', s = s', t = 2t'$ and $m = m'$, we have that: $z_\sigma = z_{\sigma'} = s' - \sigma' c' = s' - 2\sigma\frac{c}{2} = s - \sigma c$, $z_v = 2z_{v'} = 2(t' - v'c') = 2 \cdot \frac{1}{2}(t - vc) = t - vc$, $z_r = z_{r'} = m' - r'c' = m - 2r \cdot \frac{1}{2}c = m - rc$, and $D = D' = g^s h^m$. This shows that all the messages sent by $\mathcal{A}$ on the right are of correct form, and thus the honest verifier will accept on the right.

Note: The above attack assumes members of Φ are elements of Z_p. When Φ is an arbitrary set, the standard trick (as employed by the underlying Boneh-Boyen signatures) is to employ a hash function $H : \{0,1\}^* \rightarrow Z_p$. In particularly, the underlying Boneh-Boyen signature is changed to be $\tau = g^{\frac{1}{x+H(i)}}$. But, the variant of the CCS scheme [23] with this general Boneh-Boyen signature scheme is still not non-malleable (similar MIM attack still can be shown). One partial reason is that the general Boneh-Boyen signature scheme ($y = g^x, \tau = g^{\frac{1}{x+H(i)}}$) is not completely non-malleable. Specifically, given a public-key $y = g^x$ and a signature $\tau = g^{\frac{1}{x+H(i)}}$ on i an MIM adversary $\mathcal{A}$ can generate a new signature y' and a valid signature τ' on $i' \neq i$ under y' as follows: suppose $H(i') = tH(i)$, then $\mathcal{A}$ can generate $y' = y^t$ and $\tau' = \tau^t = g^{\frac{1}{tx+tH(i)}} = g^{\frac{1}{x'+H(i')}}$. This also serves as another good motivation for the importance and desirability of complete non-malleable signatures (for achieving more complex non-malleable cryptographic systems).

Practical fixing approach for non-malleable set membership argument: The key idea is to employ our complete non-malleable signature scheme: $y = g^x, \tau = g^{\frac{1}{xf(y)+H(i)}}$, where f is either a $\{0,1\}^* \rightarrow Z_p^*$ hash function or simply just the x-coordinate value of y, and i is an arbitrary value. The following non-malleable set membership argument scheme is for arbitrary set Φ.

Common input: g, h, a commitment C, an arbitrary set Φ, and a collision-resistant hash function $H : \{0,1\}^* \rightarrow Z_p^*$, where g, h are generators of a group $\mathbb{G}$ of order p.

Prover input: σ,r such that $C = g^{H(\sigma)}h^r$ and $\sigma \in \Phi$.

- Verifier randomly picks $x \in_R Z_p$ and sends $y = g^x$ and $\{A_i = g^{\frac{1}{xf(y)+H(i)}}\}_{i\in\Phi}$.
- Prover randomly picks $v \in_R Z_p$ and sends $V = A_\sigma^v$. Then prover proves to the verifier of the knowledge (σ, r, v) such that $C = g^{H(\sigma)}h^r$ and $V = g^{\frac{v}{xf(y)+H(\sigma)}}$, via the following Σ-protocol:
  - Prover randomly picks $s, t, m \in_R Z_p$ and sends $a = e(V,g)^{-s}e(g,g)^t$ and $D = g^s h^m$.
  - Verifier sends a random challenge $c \in_R Z_p$.
  - Prover sends $z_\sigma = s - H(\sigma)c$, $z_v = t - vc$ and $z_r = m - rc$.
  - Verifier outputs "accept" if and only if: $D = C^c h^{z_r} g^{z_\sigma}$ and $a = e(V, y^{f(y)})^c \cdot e(V,g)^{-z_\sigma} \cdot e(g,g)^{z_v}$.

The completeness of the above non-malleable version of CCS-scheme is from the following observations: $C^c h^{z_r} g^{z_\sigma} = (g^{H(\sigma)}h^r)^c h^m h^{-rc} g^s g^{-H(\sigma)c} = g^s h^m = D$, and $e(V, y^{f(y)})^c \cdot e(V,g)^{-z_\sigma} \cdot e(g,g)^{z_v} = e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{vcxf(y)} \cdot e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{vH(\sigma)c} \cdot e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{-vs} \cdot e(g,g)^t \cdot e(g,g)^{-vc} = e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{vcxf(y)+vH(\sigma)c} \cdot e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{-vs} \cdot e(g,g)^t \cdot e(g,g)^{-vc} = e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{vc(xf(y)+H(\sigma))} \cdot e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{-vs} \cdot e(g,g)^t \cdot e(g,g)^{-vc} = e(g,g)^{vc} \cdot e(g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{-vs} \cdot e(g,g)^t \cdot e(g,g)^{-vc} = (g^{\frac{1}{xf(y)+H(\sigma)}}, g)^{-vs} \cdot e(g,g)^t = e(V,g)^{-s}e(g,g)^t = a$.

Computational soundness (resp., honest-verifier zero-knowledge HVZK) is derived from the special soundness (resp., HVZK) of the underlying Σ-protocol.

14 Conclusion and Future Investigations

In this work, we investigate highly practical approaches for achieving non-malleable cryptosystems secure against concurrent man-in-the-middles. We start our study with the Diffie-Hellman key-exchange (DHKE) protocol, which is at the root of public-key cryptography and is one of the main pillars of both theory and practice of cryptography. We develop the mechanisms of non-malleable joint proof-of-knowledge (NMJPOK) and self-sealed joint proof-of-knowledge (SSJPOK), which are of independent values. In particular, using NMJPOK and SSJPOK as the key building tools, we present a new family of DHKE protocols, with remarkable performance among security, privacy, efficiency and easy deployment. Particularly important to applied crypto engineering, the newly developed DHKE protocols add novelties and values to a range of key industry standards for ensuring network security (e.g., IKE, (H)MQV, SSH, etc). Along the way, we also investigate the security definition frameworks for DHKE, and clarify various subtleties surrounding the design and analysis of non-malleable DHKE protocols.

Then, motivated by the building tools, NMJPOK and SSJPOK, proposed and justified in this work, we formulate non-malleable extractable joint one-way function (NME-JOWF), and demonstrate general applications of NME-JOWF (including 3-round CNMZK and UCZK in the plain model). Then, we propose candidates of NME-JOWF based upon bilinear pairings, and show various concrete applications of the pairing-based NME-JOWF candidates.

An important future research direction is to investigate and develop more NME-OWF candidates (particularly without bilinear pairings).

Acknowledgments. We thank Hugo Krawczyk, Frances F. Yao and Bin Zhu for helpful discussions and suggestions. Yunlei Zhao thanks Huijia Lin for kindly sending an electronic copy of [86], and Zongyang Zhang for helpful discussions on CNMZK with adaptive corruption.

References

[1] M. Abdalla, M. Bellare and P. Rogaway. The Oracle Diffie-Hellman Assumptions and An Analysis of DHIES. Topics in Cryptology-CT-RSA 2001, LNCS 2020: 143-158, 2001.

[2] M. Abdalla, E. Bresson, O. Chevassut and D. Pointcheval. Password-based Group Key Exchange in a Constant Number of Rounds. International Conference on Theory and Practice in Public-Key Cryptography (PKC 2006), pages 427-442, LNCS 3958, Springer-Verlag, 2006.

[3] S. Al-Riyami and K. Paterson. Certificateless Public-Key Cryptography. In Advances in Cryptology-Proceedings of ASIACRYPT 2003, LNCS 2894, pages 452-473, Springer-Verlag, 2003.

[4] American National Standard (ANSI) X9.42-2001. Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.

[5] American National Standard (ANSI) X9.42-2001. Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Elliptic Curve Cryptography.

[6] J. Baek, R. Steinfeld and Y. Zheng. Formal Proofs of the Security of Signcryption. *Journal of Cryptology*, 20(2): 203-235, 2007.

[7] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *N. Koblitz (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, Springer-Verlag, 1996.

[8] M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *ACM Symposium on Theory of Computing*, pages 419-428, 1998.

[9] M. Bellare and A. Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *M. Franklin (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2004, LNCS 3152*, pages 273-289, Springer-Verlag, 2004.

[10] M. Bellare and A. Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In *P. J. Lee (Ed.): Advances in Cryptology-Proceedings of Asiacrypt 2004, LNCS 3329*, pages 48-62, Springer-Verlag, 2004.

[11] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *D. Stinson (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1993, LNCS 773*, pages 273-289, Springer-Verlag, 1993.

[12] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In*ACM Conference on Computer and Communications Security*, pages 62-73, 1993.

[13] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology-Proceedings of EUROCRYPT 1994, LNCS 950*, Springer-Verlag, 1994.

[14] M. Blum. Coin Flipping by Telephone. In *proc. IEEE Spring COMPCOM*, pages 133-137, 1982.

[15] D. Boneh and M. Franklin. Identity-Based Encryption From the Weil Pairing. In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 213-229, Springer-Verlag, 2001.

[16] D. Boneh, B. Lynn and H. Shacham. Short Signatures from the Weil Pairing. ASIACRYPT 2001, LNCS 2248, pages 514-532.

[17] E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval. Mutual Authentication and Group Key-Agreement for Lower-Power Mobile Devices. Journal of Computer Communications, 27(127), 1730-1737, 2004. (Some minor revisions are made to deal with some attacks observed by Nam et al in Cryptology ePrint Archive Report No. 2004/251.)

[18] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *Advances in Cryptology-Proceedings of EUROCRYPT 1994*, pages 279-290, Springer-Verlag, 1994.

[19] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge In *E. F. Brickell (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1992, LNCS 740*, pages 390-420, Springer-Verlag, 1992.

[20] M. Bellare and O. Goldreich. On Probabilistic versus Deterministic Provers in the Definition of Proofs Of Knowledge. Electronic Colloquium on Computational Complexity, 13(136), 2006.

[21] A. Boldyreva, D. Cash, M. Fischlin and B. Warinschi. Foundations of Non-Malleable Hash and One-Way Functions. ASIACRYPT 2009, papges 524-541, LNCS 5912.

[22] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology-Proceedings of EUROCRYPT 2004, LNCS 3027*, pages 223-238, Springer-Verlag, 2004.

[23] J. Camenisch, R. Chaabouni AND A. Shelat. Efficient Protocols for Set Membership and Range Proofs AsiaCrypt 2008, pages 234-252.

[24] R. Canetti. Towards Realizing Random Oracles: Hash Functions that Hide All Partial Information. CRYPTO 1997, LNCS 1294, pages 455-469.

[25] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136-145, 2001.

[26] R. Canetti. Security and Composition of Cryptographic Protocols: A Tutorial. SIGACT News, 37(3,4), 2006.

[27] R. Canetti and R. R. Dakdouk. Extractable perfectly one-way functions. ICALP, Track C, 2008.

[28] R. Canetti and R. R. Dakdouk. Towards a Theory of Extractable Functions. TCC, 2009.

[29] R. Canetti, Y. Dodis, R. Pass and S. Walfish. Universal Composable Security with Global Setup. In *S. Vadhan (Ed.): Theory of Cryptography (TCC) 2007, LNCS 4392*, pages 61-85, Springer-Verlag, 2007.

[30] R. Canetti, U. Feige, O. Goldreich and M. Naor. Adaptively Secure Multi-Party Computation. In *ACM Symposium on Theory of Computing*, pages 639-648, 1996.

[31] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 235-244, 2000.

[32] R. Canetti, D. Micciancio and O. Reingold. Perfectly one-way probabilistic hash functions. STOC'98, pages 131C 140.

[33] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology-Proceedings of EUROCRYPT 2001, LNCS 2045*, Springer-Verlag, 2001. Available also from Cryptology ePrint Archive, Report No. 2001/040.

[34] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Advances in Cryptology-Proceedings of EUROCRYPT 2002, LNCS* , pages 143-161, Springer-Verlag, 2002.

[35] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In *M. Yung (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 143-161, Springer-Verlag, 2002.

[36] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(log\ n)$ Rounds. In *ACM Symposium on Theory of Computing*, pages 570-579, 2001.

[37] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *ACM Symposium on Theory of Computing*, pages 494-503, 2002.

[38] L. Chen and C. Kudla. Identity Based Key Agreement Protocols From Pairings. In *IEEE Computer Security Foundations Workshop*, pages 219-233, 2002. Full version available at: Cryptology ePrint Archive, Report 2002/184.

[39] R. Cramer. Modular Design of Secure, yet Practical Cryptographic Protocols, PhD Thesis, University of Amsterdam, 1996.

[40] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33: 167-226, 2003.

[41] I. Damgård. Towards Practical Public-Key Systems Secure Against Chosen Ciphertext Attacks. In *J. Feigenbaum (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1991, LNCS 576*, pages 445-456. Springer-Verlag, 1991.

[42] A. Dent. Cramer-Shoup Encryption Scheme is Plantext Aware in the Standard Model. In *Advances in Cryptology-Proceedings of EUROCRYPT 2006, LNCS 4004*, pages 289-307. Springer-Verlag, 2006.

[43] T. Dierks and C. Allen. The TLS Protocol, Version 1.0. Request for Comments: 2246, January 1999.

[44] M. Di Raimondo and R. Gennaro. New Approaches for Deniable Authentication. In proc. of 12nd ACM Conference on Computer and Communications Security (ACM CCS'05), ACM Press, pages 112-121, 2005.

[45] M. Di Raimondo, R. Gennaro and H. Krawczyk. Deniable Authentication and Key Exchange. ACM CCS'06, pages 466-475. Full version appears in Cryptology ePrint Archive Report No. 2006/280.

[46] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transaction on Information Theory*, 22(6): 644-654, 1976.

[47] V. S. Dimitrov, G. A. Jullien and W. C. Miller. Complexity and Fast Algorithms for Multiexponentiations. *IEEE Transactions on Computers*, 49(2): 141-147.

[48] Y. Dodis and A. Yampolskiy. A Verifiable Random Function With Short Proofs and Keys. PKC 2005, pages 416-431.

[49] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2): 391-437, 2000. Preliminary version in *ACM Symposium on Theory of Computing*, pages 542-552, 1991.

[50] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. In *ACM Symposium on Theory of Computing*, pages 409-418, 1998.

[51] T. El Gamal. A Public-Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31: 469-472, 1985.

[52] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital Sigantures. In Crypto'89, pages 263-277.

[53] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *A. Odlyzko (Ed.): Advances in Cryptology-Proceedings of CRYPTO'86, LNCS 263*, pages 186-194. Springer-Verlag, 1986.

[54] FIPS Pub 186-2, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2, US Department of Commerce/National Institute of Standard and Technology, Githersburg, Maryland, USA, January 27, 2000. (Chance notice is made on October 5 2001.)

[55] M. Fischlin. Completely Non-Malleable Schemes. In ICALP 2005.

[56] A. Freier, P. Karlton and P. Kocher. The SSL Protocol, Version 3.0. INTERNET-DRAFT, draft-freier-ssl-version3-02.txt, November 1996.

[57] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Advances in Cryptology-Proceedings of CRYPTO'99, LNCS 1666, pages 537-554, Springer-Verlag, 1999.

[58] O. Goldreich, S. Micali and A. Wigderson. How to Play Any Mental Game. In *ACM Symposium on Theory of Computing*, pages 218-229, 1987.

[59] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All language in $\mathcal{NP}$ Have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery*, 38(1): 691-729, 1991.

[60] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems In *ACM Symposium on Theory of Computing*, pages 291-304, 1985.

[61] S. Goldwasser, S. Micali and C. Rackoff. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIMA Journal on Computing*, 17(2): 281-308, 1988.

[62] D. M. Gordon. A Survey of Fast Exponentiation Methods. *Journal of Algorithms*, 27(1): 129-146, 1998.

[63] J. Groth and A. Sahai. Efficient Non-Interactive Proof Systems for Bilinear Groups. EuroCrypt 2008, pages 415-432.

[64] L. Guillou and J. J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In *C. G. Gnther (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1988, LNCS 330* , pages 123-128, Springer-Verlag, 1988.

[65] S. Hada and T. Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *H. Krawczyk (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1998, LNCS 1462* , pages 408-423, Springer-Verlag, 1998.

[66] *D. Harkins and D. Carreal (Ed.): The Internet Key-Exchange (IKE)*, RFC 2409, Nov., 1998.

[67] K. Hickman. The SSL Protocol. Online document, Feburary 1995. Available at www.netscape.com/eng/security/SSL-2.html.

[68] IEEE 1363-2000: Standard Specifications for Public Key Cryptography.

[69] ISO/IEC IS 15946-3. Information Technology - Security Techniques - Cryptographic Techniques Based on Elliptic Curves - Part 3: Key Establishment, 2002.

[70] H. Jin, D. S. Wong and Y. Xu. Complexity Analysis of a Fast Modular Multiexponentiation Algorithm. Cryptology ePrint Archive, Report 2008/210.

[71] B. Kaliski. An Unknown Key-Share Attack on the MQV Key Agreement Protocol. *ACM Transactions on Information and System Security (TISSEC)*, 4(3): 275-288, 2001.

[72] P. Karn and W.A. Simpson. The Photuris Session Key Management Protocol, draft-ietf-ipsec-photuris-03.txt, Sept., 1995.

[73] J. Katz and Y. Lindell. Introduction to Modern Cryptography, Chapman and Hall/CRC press, 2007.

[74] J. Katz and J. Shin. Modeling Insider Attackss on Group Key Exchange. ACM CCS 2005, 180-189.

[75] J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 110-125, Springer-Verlag, 2003.

[76] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. The Internet Engineering Task Force: INTERNET-DRAFT, October 2002.

[77] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments 2401, 1998.

[78] H. Krawczyk. SKEME: a Versatile Secure Key Exchange Mechanism for Internet In *proc. of 1996 IEEE Symposium on Network and Distributed System Security (SNDSS'96)*, pages 114-127.

[79] H. Krawczyk. SIGMA: the "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. Invited Talk at *D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 400-425, Springer-Verlag, 2003.

[80] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *V. Shoup (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2005, LNCS 3621*, pages 546-566. Springer-Verlag, 2005.

[81] H. Krawczyk. HMQV in IEEE P1363. July 2006.

[82] S. Kunz-Jacques and D. Pointcheval. A New Key Exchange Protocol Based on MQV Assuming Public Computations. In SCN'06, LNCS 4116, pages 186-200, Springer-Verlag, 2006.

[83] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Designs, Codes and Cryptography*, 28: 119-134, 2003.

[84] D. Lapidot and A. Shamir. Publicly-Verifiable Non-Interactive Zero-Knowledge Proofs. In *A.J. Menezes and S. A. Vanstone (Ed.): Advances in Cryptology-Proceedings of CRYPTO 1990, LNCS 537*, pages 353-365. Springer-Verlag, 1990.

[85] H. Lin and R. Pass. Concurrent Non-Malleable Commitments from Any One-Way Function. TCC 2009.

[86] H. Lin and R. Pass. Concurrent Non-Malleable Zero-Knowledge with Adaptive Inputs. TCC 2011, to appear.

[87] H. Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. AsiaCrypt 2003, pages 398-415.

[88] Y. Lindell. Adaptive Zero-Knowledge Proofs and Adaptively Secure Oblivious Transfer. TCC 2009.

[89] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *IEEE Symposium on Foundations of Computer Science*, pages 394-403, 2003.

[90] Y. Lindell. Lower Bounds and Impossibility Results for Concurrenet Self Composition. *Journal of Cryptology*, 21(2): 200-249, 2008.

[91] http://www.unix.com.ua/orelly/networking_2ndEd/ssh/ch01_05.htm

[92] C.Lonvick. The Secure Shell (SSH) Protocol Architecture. IETF RFC-4251, 2006.

[93] C.Lonvick. The Secure Shell (SSH) Transport Layer Protocol. IETF RFC-4253, 2006.

[94] C.Lonvick. The Secure Shell (SSH) Authentication Protocol. IETF RFC-4252, 2006.

[95] C.Lonvick. The Secure Shell (SSH) Connection Protocol. IETF RFC-4254, 2006.

[96] W. Mao. Modern Cryptography: Theory and Practice. Prentice Hall PTR, 2004.

[97] U. Maurer and S. Wolf. Diffie-Hellman Oracles. In *Advances in Cryptology-Proceedings of CRYPTO 1996, LNCS 1109*, pages 268-282, Springer-Verlag, 1996.

[98] A. Menezes. Another Look at HMQV. Cryptology ePrint Archive, Report No. 2005/205.

[99] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1995

[100] A. Menezes, M. Qu, and S. Vanstone. Some New Key Agreement Protocols Providing Mutual Implicit Authentication. Second Workshop on Selected Areas in Cryptography (SAC'95), 1995.

[101] D. Naccache, D. M'Raihi, S. Vaudenay and D. Raphaeli. Can D.S.A be Improved? Complexity Trade-Offs with the Digital Signature Standard. In *Advances in Cryptology-Proceedings of EUROCRYPT 1994, LNCS 950*, pages 77-85, Springer-Verlag, 1994.

[102] M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 1(2): 231-262 (2004).

[103] J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. In *M. Yung (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2002, LNCS 2442*, pages 111-126, Springer-Verlag, 2002.

[104] NIST Special Publication 800-56 (DRAFT): Recommendation on Key Establishment Schemes. Draft 2, January 2003.

[105] NSAs Elliptic Curve Licensing Agreement. Presentation by Mr. John Stasak (Cryptography Office, National Security Agency) to the IETF's Security Area Advisory Group, November 2004.

[106] T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In PKC'01, LNCS 1992, pages 104-118, Springer-Verlag, 2001.

[107] H. Orman. The OAKLEY Key Determination Protocol. Request for Comments 2412, Nov., 1998.

[108] R. Pass. On Deniabililty in the Common Reference String and Random Oracle Models. In*D. Boneh (Ed.): Advances in Cryptology-Proceedings of CRYPTO 2003, LNCS 2729*, pages 316-337, Springer-Verlag 2003.

[109] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *ACM Symposium on Theory of Computing*, pages 533-542, 2005.

[110] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. In *IEEE Symposium on Foundations of Computer Science*, pages 563-572, 2005.

[111] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13: 361-396, 2000.

[112] C. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3): 161-174, 1991.

[113] A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. In In *Advances in Cryptology-Proceedings of CRYPTO 2001, LNCS 2139*, pages 355-367, Springer-Verlag, 1996.

[114] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. Eurocrypt'97, LNCS 1233, pp. 256-266.

[115] SP 800-56 (DRAFT), Special Publication 800-56, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology, July 2005.

[116] D. R. Stinson and J. Wu. An Efficient and Secure Two-Flow Zero-Knowledge Identification Protocol. Cryptology ePring Archive, Report No. 2006/337, 16 Oct 2006.

[117] D. R. Stinson and J. Wu. A Zero-Knowledge Identification and Key Agreement Protocol. Cryptology ePring Archive, Report No. 2007/116, 30 Mar 2007.

[118] D. R. Stinson and J. Wu. Revision of [117] based on [121, 125]. July 2007.

[119] T. Ylonen. SSH Protocol Architecture. INTERNET-DRAFT, draft-ietf-architecture-15.txt, 2002.

[120] T. Ylonen. SSH Transport Layer Protocol. INTERNET-DRAFT, draft-ietf-architecture-13.txt, 2002.

[121] A. C. C. Yao, F. F. Yao, Y. Zhao and B. Zhu. Deniable Internet Key-Exchange. Cryptology ePrint Archive 2007/191.

[122] A. C. Yao, M. Yung and Y. Zhao. Adaptive Concurrent Non-Malleability with Bare Public-Keys. CoRR, abs/0910.3282, 2009.

[123] M. Yung and Y. Zhao. Generic yet Practical Resettable Zero-Knowledge in the Bare Public-Key Model. Eurocrypt 2007, pages 129-147.

[124] T.H.Yuen, Q.Huang, Y.Mu, W. Susilo, D.S. Wong, and G. Yang. Efficient Non-Interactive Range Proof COCOON 2009, pages 138-147.

[125] Y. Zhao. Personal communications for clarifying the part of [121] related to [117]. June 2007.

[126] M. Yung and Y. Zhao. Interactive Zero-Knowledge with Restricted Random Oracles. Theory of Cryptography Conference (TCC), pages 21-40, 2006.

A The SIGMA Protocol

SIGMA key-exchange protocol. We briefly recall (the basic version of) the SIGMA protocol of [79] (that serves as the underlying basis of the IKEv2 standard), which is depicted in Figure 15 (page 140), where PRF is a pseudorandom function and KDF is a key-derivation function.

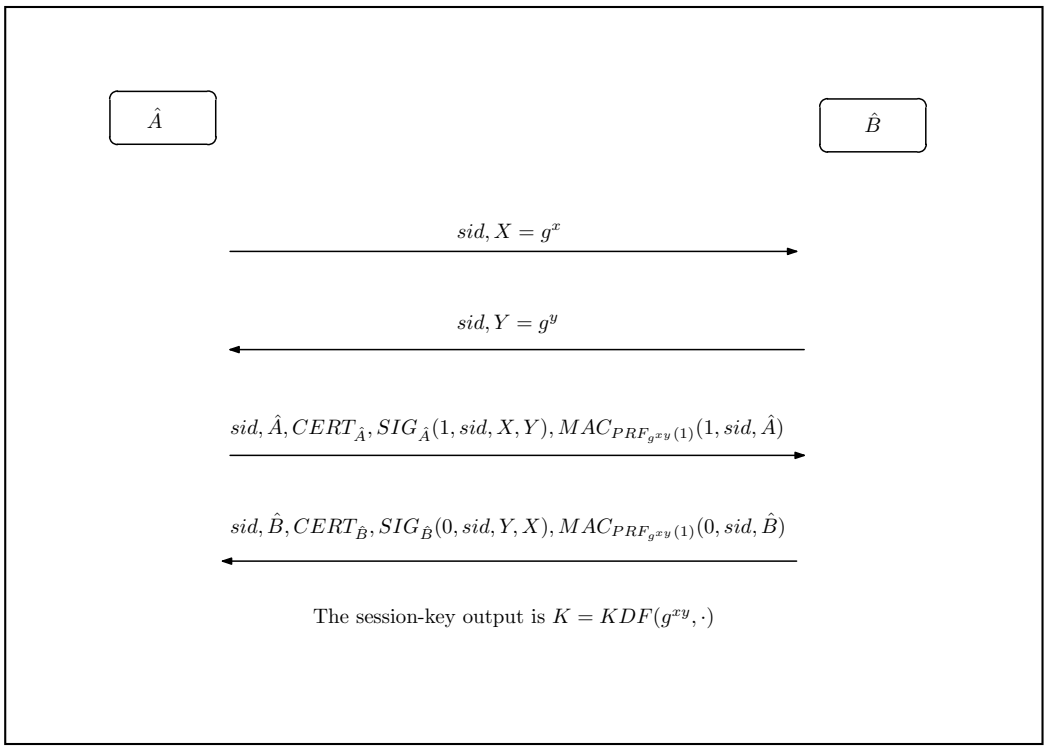


Figure 15: The SIGMA protocol (the main model)

The "cutting-last-message" attack. The "cutting-last-message" attack on the SIGMA protocol works as follows [96]: A man-in-the-middle $\mathscr{A}$ interacts with $\hat{A}$ in the name of $\hat{M}$, while concurrently interacting with $\hat{B}$ in the name of $\hat{A}$. $\hat{M}$ just relays messages between $\hat{A}$ and $\hat{B}$, but aborts after receiving the last message from $\hat{B}$. Such a simple attack results in authentication failure as follow: $\hat{B}$ is perfectly fooled to believe it has shared a session key with $\hat{A}$, while $\hat{A}$ only thinks it ever took part in an aborted session with $\hat{M}$. This attack can be simply prevented by adding an additional fifth-round of "acknowledgement" from $\hat{A}$ to $\hat{B}$ (as suggested in [35]), but increasing the system complexity.

On the role of "0" and "1" in SIGMA. We remark that putting "0" and "1" into the inputs of SIG and MAC in (the basic version of) SIGMA, which explicitly indicate the roles of the protocol initiator and the responder, plays a critical role in preventing reflection attacks. Specifically, for the protocol variant of SIGMA with "0" and "1" removed from the inputs of MAC and SIG, there exists an effective reflection attack: the adversary sets $Y = X$ and rebounds the third-round message back to $\hat{A}$ in the fourth-round in the same name of $\hat{A}$. (The work [35] also proposed some advanced versions of the SIGMA protocol that resist against the reflection attack without bearing the explicit role indicators "0" and "1".) But, putting "0" and "1" into SIG and MAC may violate players' privacy concerning their protocol roles. Specifically, in this case, players' roles are *publicly* verifiable from the signatures.

B Variants of (H)MQV

Three-round HMQV (resp., MQV) adds key confirmation as follows: let $K_m = H_K(K_{\hat{A}}, 0) = H_K(K_{\hat{B}}, 0)$, $\hat{B}$ uses K_m as the MAC key to authenticate 0 (resp., $(2, \hat{B}, \hat{A}, Y, X)$) in the second-round of HMQV (resp., MQV); and $\hat{A}$ uses K_m to authenticate 1 (resp., $(3, \hat{A}, \hat{B}, X, Y)$) in an additional third-round of HMQV (resp., MQV). The session-key is set to be $K = H_K(K_{\hat{A}}, 1) = H_K(K_{\hat{B}}, 1)$.

In one-round HMQV, only $\hat{A}$ sends X, and the session-key is derived as follows: $K_{\hat{A}} = B^{x+da}$, $K_{\hat{B}} = (XA^d)^b$, $d = h(X, \hat{A}, \hat{B})$, $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$.

C More variants of YZ

Some (weak) variants: We present and clarify a list of some other seemingly (maybe unreasonably) weak variants, which further highlight the advantages of YZ, its careful design and sound design rational.

$K_{\hat{A}} = B^{fa+dx}Y^{ca+ex}$ and $K_{\hat{B}} = A^{fb+cy}X^{db+ey}$, where $c = h(\hat{A}, A, Y)$, $d = h(\hat{B}, B, X)$, $e = h(X, Y)$ and f=1. This variant has the following weaknesses: (1) It can lose the TBRNM security when the values A^b or B^a is pre-computed. Specifically, suppose a CMIM adversary $\mathscr{A}$ interacts with $\hat{A}$ as the responder in the name $\hat{M}_1$ in a session (referred to as the test-session), and concurrently interacts with $\hat{B}$ as the initiator in the name of $\hat{M}_2$ in another session (referred to as the second-session). After receiving X from $\hat{A}$ in the first session, $\mathscr{A}$ registers X as the public-key of $\hat{M}_2$ in the second-session. In case *both* of the ephemeral private values y and X^{b+cy} (pre-computed by $\hat{B}$ in the second-session) are composed to $\mathscr{A}$, $\mathscr{A}$ can compute out $X^b = (X^{b+cy}/X^{cy})$, where $c = h(\hat{M}_2, X, Y)$. Now, further suppose $\mathscr{A}$ also learns the static secret-key a of $\hat{A}$, $\mathscr{A}$ can then successfully impersonate the player $\hat{B}$ (i.e., $\hat{M}_1 = \hat{B}$) in the test-session *but actually without knowing the secret-key b* (to this end, $\mathscr{A}$ will generate the DH-component honestly in the test-session). Note that the test-session is still internally-unexposed, and the internally-exposed second-session is not the matching session of the test-session. It is interesting to note that, though the variant provides seemingly stronger security than that of the original YZ, it actually can weaken the security besides complicating the protocol and reducing efficiency. We remark that the underlying reason beneath such attacks is: the leakage of the ephemeral values of *both* y and X^{b+cy} can render a CDH solver oracle, say X^b, to the adversary. Also, the attack can be prevented by explicitly requiring POK/POP during key registration. (2) Another weakness of the variant is: if $\hat{B}$ sets $Y = A$ and $B = (Y^c X^e)^{-1}$, then $K_{\hat{B}} = X^{db}$. This weakness can be prevented by setting $f = h(\hat{A}, A, \hat{B}, B)$, but at the price of sacrificing the advantage of pre-computability and modular computability (specifically, the players have to make sure their roles, initiator or responder, before computing f).

$K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, where $c = h(\hat{A}, A, X, Y)$, $d = h(\hat{B}, B, X, Y)$, $e = 1$ (or, $c = h(\hat{A}, A, X)$, $d = h(\hat{B}, B, Y)$, $e = h(X, Y)$). Besides potential security weaknesses, the main disadvantage of such variants is the loss of pre-computability of A^{cy} and B^{dx}.

$K_{\hat{A}} = B^{dxt}Y^{cat}$ and $K_{\hat{B}} = A^{cyt}X^{dbt}$, where $c = h(\hat{A}, A, Y)$ and $d = h(\hat{B}, B, X)$ (of course, other values of c, d are possible as long as the union of the inputs cover the tag, e.g., $c = 1$ and $d = h(\hat{A}, A, \hat{B}, B, X, Y)$, or, $c = h(\hat{A}, A, \hat{B}, B, Y)$ and $d = h(\hat{B}, B, \hat{A}, A, X)$). The weaknesses of these protocol variants are: (1) If both of the static secret-keys, a and b, are leaked, session-keys of previous sessions can be derived. That is, they lose the perfect forward security; (2) If $\hat{B}$ sets $B = A$, then $K_{\hat{A}} = K_{\hat{B}} = A^{dxt+cyt}$. If the static secret-key a is leaked, the attack can compute out $K_{\hat{A}} = K_{\hat{B}}$ without knowing the DH-exponent y or x. We note this variant can be used to partially fix the flaws of the DHKE protocol (Cryptology ePrint Archive 2007/116), caused by the value $h(g^{ay}g^{xb})$ as clarified in Cryptology ePrint Archive 2007/191.

$K_{\hat{A}} = B^{a+x}Y^{a+ex}$ and $K_{\hat{B}} = A^{b+y}X^{b+ey}$, where $e = h(\hat{A}, A, \hat{B}, B, X, Y)$. This variant has the following (maybe also unreasonable) weaknesses: (1) If the player $\hat{B}$ sets $Y = B^{-1}$, then $K_{\hat{B}} = X^{b+ey} = X^{b(1-e)}$; (2) If the player $\hat{B}$ sets $Y = (BX)^{-1}$ (i.e., $y = -b - x$) and $B = A$, then $K_{\hat{B}} = A^{b+y}X^{b+ey} = X^{ey}$. One remedy for the above weaknesses is: $K_{\hat{A}} = B^{fa+x}Y^{a+ex}$ and $K_{\hat{B}} = A^{fb+y}X^{b+ey}$, where $e = h(\hat{A}, A, \hat{B}, B, X, Y)$ and $f = h(X, Y, \hat{A}, A, \hat{B}, B)$. But, this remedy sacrifices the advantage of on-line efficiency of YZ.

$K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, where $c = h(\hat{B}, B, Y)$, $d = h(\hat{A}, A, X)$, $e = h(X, Y)$. This variant has the following weakness: if $\hat{B}$ sets $B = Y^{-e/d}$ (i.e., $b = -y(e/d)$), then $K_{\hat{B}} = A^{cyt}X^{dbt+eyt} = A^{cyt}$; in case of the leakage of a, $\hat{B}$ can finish the session without knowing either y or b. Though such weakness can be remedied by setting $e = h(\hat{A}, A, \hat{B}, B, X, Y)$, this variant loses the advantage of modular and post-ID computability of YZ. Similarly, we can also strengthen the YZ protocol (depicted in Figure 7) for seemingly (unjustified) stronger robustness by setting $e = h(\hat{A}, A, \hat{B}, B, X, Y)$, but at the price of sacrificing the advantage of modular and post-ID computability of YZ.

$K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, where $c = h(\hat{A}, Y)$, $d = h(\hat{B}, X)$, $e = h(X, Y)$ or $e = h(\hat{A}, A, \hat{B}, B, X, Y)$. This is the variant of YZ, where player's public-key A (resp., B) is removed from the input of c (resp., d). The weaknesses of this variant of YZ are: (1) If $\hat{B}$ sets $Y = X$ and $B = A^{-c/d}$ (i.e., $b = -ac/d$), then $K_{\hat{B}} = A^{cyt}X^{dbt+eyt} = X^{eyt}$. This weakness can be remedied by requiring POK/POP for public-key registrations; (2) If $\hat{B}$ sets $B = g^{d^{-1}}$, then $K_{\hat{B}} = A^{cyt}X^{dbt+eyt} = A^{cyt}X^{t+eyt}$; (3) if $\hat{B}$ sets $Y = X$ and $B = g^{d^{-1}}/A^{c/d}$ (i.e., $b = d^{-1} - a(c/d)$), then $K_{\hat{B}} = A^{cyt}X^{dbt+eyt} = X^{t+eyt}$; furthermore, if $e = h(X, Y)$ (or any other values independent of B), $\hat{B}$ can set $Y = X$ and $B = g^{d^{-1}}/A^{c/d}Y^{e/d}$, and in this case $K_{\hat{B}} = X^t$. The weaknesses (2) and (3) cannot be remedied by POK/POP of key-registration, but can be remedied if the player $\hat{A}$ makes sure that its peer's public-key B is fixed before sending its DH-component X, which however may limit the deployments in certain scenario (e.g., in the post-ID model for preserving player's privacy). We note the variants of rYZ and sYZ, when public-keys are removed from c, d and/or e, suffer from the similar weaknesses, though they are provably secure in the CK-framework.

$K_{\hat{A}} = B^{dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{cyt}X^{dbt+eyt}$, where $c = h(Y)$, $d = h(X)$, $e = h(X, Y)$. This is the variant of YZ, where both player's identity and public-key $(\hat{A}, A)$ (resp., $(\hat{B}, B)$) are removed from the input of c (resp., d). Besides above clarified weaknesses, such variant additionally has the following weaknesses (the variants of rYZ and sYZ, when player's identities are removed from c, d and/or e, suffer from the similar weaknesses): (1) Unknown key share (UKS) attack: with this variant, the session-key is not adhere to the tag of the session. Consider a man-in-the-middle adversary, who impersonates to $\hat{A}$ as $\hat{B}' \neq \hat{B}$ but of same public-key B and simultaneously impersonates to $\hat{B}$ as $\hat{A}' \neq \hat{A}$ but of the same public-key A, then the adversary can set two sessions of different tags to be of the same session-key. This weakness can be remedied by putting players' identities $(\hat{A}, \hat{B})$ into the input of the key-derivation function, and/or requiring POK/POP of key registration. (2) Weakness against secrecy leakage: suppose the ephemeral private value $X^{dbt+eyt}$ (generated and kept by $\hat{B}$, e.g., when $\hat{A}$ works in the post-ID model as in csYZ) and $\hat{A}$'s secret-key a are exposed to the adversary, then the adversary can successfully finish the session with $\hat{A}$ in the name of $\hat{B}$ but without knowing either b or y. This weakness can be remedied by requiring $\hat{B}$ not to pre-compute $X^{dbt+eyt}$, but may limit the protocol deployments.

Some other useful variants are: (1) The function e in YZ is set to be $h(\hat{A}, A, \hat{B}, B, X, Y)$. This variant loses the post-ID and modular computability of YZ; (2) The function e in rYZ is just set to be $H(X, Y)$, and/or, the function c is set to be $h(\hat{A}, A, Y)$ and d is set to be $h(\hat{B}, B, X)$. Such variant may impair the strong TBRNM security of rYZ against powerful secrecy exposure.

General YZ: Let each pub_i, $1 \leq i \leq 4$, be a (maybe empty) subset of other messages, *except for the DH-components* (X, Y), related to the run of Diffie-Hellman key-exchange, which typically

include: players identity information, public-keys, session-identifier, players' network addresses, etc. Then, all the above variants, and YZ and rYZ and sYZ, are all carefully instantiated from the following general protocol structure, called general YZ (needless to say, some instantiations of the general YZ structure may be weak and even insecure):

$K_{\hat{A}} = B^{fa+dxt}Y^{cat+ext}$ and $K_{\hat{B}} = A^{fb+cyt}X^{dbt+eyt}$, the session-key is $K = H_K(K_{\hat{A}}) = H_K(K_{\hat{B}})$, where $c = h(pub_1, Y)$ or $c = 1$, $d = h(pub_2, X)$ or $d = 1$, $e = h(pub_3, X, Y)$ or $e = 0$, $f = 0$ for full deniability, $f = 1$ or $f = h(pub_4)$ or $f = h(X, Y, pub_4)$ for group deniability. It is recommended that the input of c includes $\hat{A}$, the input of d includes $\hat{B}$; when $c = d = 1$, e cannot be 0 and the input of e should include $(\hat{A}, \hat{B})$. In case CA does not perform the subgroup test for registered public-keys, fa and fb should also be multiplied by the cofactor t.