Three-Move Blind Signatures in Pairing-Free Groups

Yanbo Chen*

June 7, 2026

Abstract

We propose the first blind signature scheme that simultaneously achieves the following properties:

- It uses a pairing-free group and random oracles in a black-box manner;
- It provably achieves concurrent security based on standard assumptions (DDH) without the algebraic group model (AGM);
- It requires only three moves.

Moreover, the public key, signature, and communication of our scheme all consist of only a constant number of group/field elements.

Prior to our work, black-box, three-move pairing-free schemes were only known in the AGM. A recent line of work proposed and optimized schemes without the AGM, but they all require at least four moves.

1 Introduction

Blind signature schemes [Cha82] are interactive protocols for joint signature generation between a signer holding the secret key and a user holding the message. The main security properties are *one-more unforgeability* and *blindness*. One-more unforgeability guarantees that a malicious user cannot produce more valid signatures than the number of completed signing sessions. Blindness ensures that the user's message is hidden from the signer. When the signer later sees the message-signature pair, it cannot link the pair to the corresponding invocation of the protocol. Blind signatures have found many applications, such as in e-cash [Cha82, CFN90, OO92], anonymous credentials [Bra94, CL01], and e-voting [Cha88, FOO92].

Blind signatures in pairing-free groups. This work focuses on blind signatures in pairing-free groups. Such groups offer significant advantages in terms of computational efficiency and broad compatibility with existing cryptographic libraries.

Pairing-free blind signatures have been studied for over three decades. Early constructions, such as blind Okamoto-Schnorr signatures [Oka94], have only been proved under limited concurrency [PS00, HKL19] in the random oracle model [BR93]. The blind Schnorr signature scheme [CP93] also achieves only bounded concurrency, and its security proof [FPS20, KLX22] additionally assumes the algebraic group model (AGM) [FKL18]. Bounded concurrency is an inherent limitation, as efficient concurrent attacks on these schemes are known [BLL+21].

An exception is Abe's blind signature scheme [Abe01]. The scheme was originally proved concurrently secure only assuming the ROM, but Ohkubo and Abe [OA03] later found the proof flawed and fixed it in the generic group model (GGM) [Sho97, Mau05]. Recently, Abe's scheme was proved secure in the AGM [KLX22]. Subsequent work [TZ22, CKM+23] proposed improved constructions

*University of Ottawa. Email: ychen918@uottawa.ca.

Scheme	Moves	Signature	Communication	Assumption
Clause Schnorr [FPS20]	3	$1\mathbb{G} + 1\mathbb{Z}_p$	$2\mathbb{G} + 3\mathbb{Z}_p$	OMDL, mROS
Abe [Abe01, KLX22]	3	$2\mathbb{G} + 6\mathbb{Z}_p$	$\lambda + 3\mathbb{G} + 6\mathbb{Z}_p$	DL
TZ [TZ22]	3	$4\mathbb{Z}_p$	$2\mathbb{G} + 4\mathbb{Z}_p$	DL
Snowblind [CKM+23]	3	$1\mathbb{G} + 2\mathbb{Z}_p$	$2\mathbb{G} + 4\mathbb{Z}_p$	DL
CTZ-1 [CATZ24]	4	$1\mathbb{G} + 4\mathbb{Z}_p$	$5\mathbb{G} + 5\mathbb{Z}_p$	CT-OMCDH
CTZ-2 [CATZ24]	5	$1\mathbb{G} + 4\mathbb{Z}_p$	$5\mathbb{G} + 5\mathbb{Z}_p$	CT-OMCDH
CTZ-3 [CATZ24]	4	$\Theta(\lambda)(\lambda + \mathbb{G} + \mathbb{Z}_p)$	$\Theta(\lambda)(\lambda + \mathbb{G} + \mathbb{Z}_p)$	CDH
KRW [KRW24]	4	$2\mathbb{G} + 5\mathbb{Z}_p$	$\Theta(\lambda)(\lambda + \mathbb{G} + \mathbb{Z}_p)$	DDH
KR-1 [KR25]	4	$1\mathbb{G} + 5\mathbb{Z}_p$	$10\mathbb{G} + 9\mathbb{Z}_p$	DDH
KR-2 [KR25]	5	$1\mathbb{G} + 6\mathbb{Z}_p$	$10\mathbb{G} + 9\mathbb{Z}_p$	DDH
BHKR [BHKR25]	4	$10\mathbb{G} + 29\mathbb{Z}_p$	$37\mathbb{G} + 40\mathbb{Z}_p$	DDH
RR [RR25]	4	$2\mathbb{G} + 8\mathbb{Z}_p$	$\Theta(\lambda)(\lambda + \mathbb{G} + \mathbb{Z}_p)$	CDH
KLR [KLR25]	4	$6\mathbb{Z}_p$	$\Theta(\lambda)(\lambda + \mathbb{G} + \mathbb{Z}_p)$	DL
Our work	**3**	$21\mathbb{G} + 32\mathbb{Z}_p$	$27\mathbb{G} + 29\mathbb{Z}_p$	DDH

Table 1: Comparison between pairing-free blind signatures. The schemes above the line are proved to be secure in the ROM and the AGM. Those below the line only rely on the ROM.

in the AGM. More recently, some proposals achieve non-AGM, unbounded-concurrency security [CATZ24, KRW24, KR25, BHKR25, KLR25, RR26].

We focus on schemes that make black-box use of groups and random oracles. With non-black-box techniques [Fis06], one can obtain round-optimal (i.e. two-move) pairing-free blind signatures. However, such constructions are significantly less efficient than black-box ones. Non-black-box use of hash functions also calls into question the security analysis when the underlying scheme relies on the random oracle model.

Performance gap between AGM and non-AGM schemes. The most efficient AGM-based construction [CKM+23] has a signature/communication size of three/six group/field elements, and the protocol takes only three moves. According to a recent work by Dietz, Kastner, and Tessaro [DKT26], three moves is optimal for fully black-box pairing-free constructions. While slightly less efficient, the aforementioned series of non-AGM constructions has largely closed these gaps: the most efficient one [KRW24] has a signature size of six elements and constant communication. However, none of them has bridged the gap in the round complexity! Specifically, all the non-AGM constructions require at least four moves, although a three-move construction relying only on the ROM is a natural goal, with attempts dating back to over two decades ago [Abe01].

With preprocessing, a three-move blind signature scheme with unbounded concurrent security allows the signer to pre-generate and send the first message to the user. This reduces the online phase to a single round-trip (1-RTT). In contrast, four-move schemes require 2-RTT during their online execution. In practical applications such as e-cash, the round complexity of blind signatures directly dictates the user-perceived latency.

1.1 Contribution

We construct a blind signature scheme making black-box use of a pairing-free group and random oracles that, for the first time, achieves non-AGM, unbounded concurrent security with only three moves. Signatures, parameters, public keys, and communication all consist of a constant number of group/field elements. The security is based on the standard DDH assumption. We compare our scheme with previous constructions in Table 1.

1.2 Our Techniques

Online-offline paradigm. At a high level, we apply the online-offline paradigm [EGM96] to reduce the round complexity of blind signature schemes. Forget about blindness and consider a four-move interactive signing protocol, where in the first move the user sends the message μ to the signer in the clear. Our idea is to let the signer generate a public key pk of a one-time signature scheme, and initiate the four-move protocol to sign pk. Consequently, the first move of the protocol can be omitted. The actual message μ is sent to the signer in the second move, and the signer generates a one-time signature σ_{ot} on μ. The final signature $(\mathsf{pk}, \sigma_{\mathrm{ot}}, \sigma_{\mathrm{is}})$ consists of pk, the one-time signature σ_{ot} on μ, and the signature σ_{is} on pk generated by the protocol.

Blindness. Things become much more complicated when considering blindness. First, pk should not be included in the signature in the clear, as it allows the signer to link the signature with its signing session. Next, if we do not include pk in the signature, verification of σ_{is} and σ_{ot} also becomes a problem.

As a solution, we base our three-move construction on a special four-move protocol, which we call an *encrypted-message blind signature* (EMBS) scheme. The protocol generates a signature that verifies on *an encryption of the message*.

We let the user blind the one-time key pk and signature σ_{ot} into $\bar{\mathsf{pk}}$ and $\bar{\sigma}_{\mathrm{ot}}$. Meanwhile, the EMBS protocol generates σ_{is} that verifies on an encryption $\mathsf{Enc}(\mathsf{pk})$ of the public key. Furthermore, the user generates a non-interactive proof of knowledge π of the blinding factors, connecting $\bar{\mathsf{pk}}$ and $\mathsf{Enc}(\mathsf{pk})$. The signature consists of $(\mathsf{Enc}(\mathsf{pk}), \sigma_{\mathrm{is}}, \bar{\mathsf{pk}}, \bar{\sigma}_{\mathrm{ot}}, \pi)$. We note that the encryption here essentially plays the role of an extractable commitment, with the encryption key being part of the public parameters and the decryption key unknown to any party.

For security, given a forgery in this form, the reduction either obtains a signature σ_{is} on a new encrypted message, or uses a previously seen encrypted pk. In the latter case, the reduction extracts the blinding factors from π and unblinds $\bar{\sigma}_{\mathrm{ot}}$ to break the one-time security.

Instantiation. We can instantiate the one-time signature scheme with the one-time version of Schnorr signatures [BS07]. For the EMBS scheme, we observe that a variant of the tightly secure blind signature scheme BHKR [BHKR25] suffices for our purpose. The key point is that BHKR is based on a structure-preserving signature and inherits this property to some extent. In particular, the protocol directly signs the original group-element message, unlike other pairing-free constructions which require hashing the message first.

Additional details. To let signatures consist of a constant number of elements, we rely on rewinding to extract the proof of knowledge. Consequently, we actually use a two-time signature scheme, which supports two signing queries across the rewound executions.

We can achieve security under the one-more discrete logarithm assumption based on the two-time version of Schnorr signatures. To further eliminate the use of interactive assumptions, we actually use the two-time version of the Okamoto signature scheme [Oka93], which is secure under the discrete logarithm assumption.

1.3 Concurrent Work

Concurrent to our work, Chairattana-Apirom, Reichle, Tessaro [CART26] solve the same question: they also construct a fully black-box three-move pairing-free blind signature scheme under DDH without the AGM. They also achieve signature size and communication consisting of constant numbers of elements.

Their construction combines the Okamoto-Schnorr blind signature [Oka93] with an algebraic MAC. In our understanding, the underlying idea is similar to ours. The role of the Okamoto-Schnorr

blind signature is conceptually (though not identically) similar to a partially one-time signature, where the nonce is used as the one-time key. They use an algebraic MAC to authenticate the one-time key, rather than an encrypted-message blind signature as in our construction.

Compared to our work, their construction achieves a more compact instantiation (37-element signatures and 31-element communication for computational blindness). They also achieve extensions such as statistical blindness, partial blindness, and strong unforgeability, while we only focus on achieving the basic security notions.

We believe that the different technical path taken in our work has independent value and is meaningful for future work, for example for improving efficiency and instantiation based on different algebraic assumptions. Our construction idea stems from the online-offline paradigm of [EGM96] and uses two-time signatures as a building block. While their use of Okamoto-Schnorr blind signature gives a more efficient instantiation, our use of two-time signatures may provide insight for different instantiations.

2 Preliminaries

Notations.

For any integer n, we use $[n]$ to denote $\{0, \ldots, n-1\}$ and $[n]^+$ to denote $\{1, \ldots, n\}$. For a randomized algorithm A, $y := \mathsf{A}(x; \rho)$ denotes the output of A on input x with predetermined randomness ρ; $y \leftarrow \mathsf{A}(x; \rho)$ denotes the same but with ρ freshly sampled.

We consider concrete security in this work. Cryptographic schemes are not parameterized by a security parameter. Throughout this work, we define all the schemes with respect to a fixed group $\mathbb{G}$ with prime order p and generator G. The security parameter is implicitly used to choose a group at a sufficient security level.

Throughout this paper, we adopt additive notation for all group operations. For a group element $G \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}_p$, we denote scalar multiplication as xG, placing the scalar on the left. However, for a matrix $\mathbf{M} \in \mathbb{G}^{m \times n}$ and a column vector $\mathbf{v} \in \mathbb{Z}_p^n$, we write the product as $\mathbf{M}\mathbf{v}$, placing the scalar vector on the right. We use O to denote the identity element.

Assumptions.

We recall the DDH and DL assumptions.

Definition 1 (Decisional Diffie-Hellman DDH)**.** We say that DDH is (τ, ε)-hard over group $(\mathbb{G}, p, G)$ if for every τ-time algorithm A,

$$\begin{aligned}&|\Pr[\mathsf{A}(H, aG, aH) = 1 : H \leftarrow \mathbb{G}, a \leftarrow \mathbb{Z}_p] \\ &\quad - \Pr[\mathsf{A}(H, aG, bH) = 1 : H \leftarrow \mathbb{G}, a, b \leftarrow \mathbb{Z}_p]| \leq \varepsilon.\end{aligned}$$

Definition 2 (Discrete logarithm DL)**.** We say that DL is (τ, ε)-hard over group $(\mathbb{G}, p, G)$ if for every τ-time algorithm A,

$$\Pr[\mathsf{A}(aG) = a : a \leftarrow \mathbb{Z}_p] \leq \varepsilon.$$

Public-Key Encryption.

We will use the following extension of the ElGamal encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ over $(\mathbb{G}, p, G)$ with message space $\mathbb{G}^l$. The scheme consists of the following algorithms:

- $\mathsf{KGen}() \to (\mathsf{ek}, \mathsf{dk})$: The key generation algorithm KGen samples the decryption key $\mathsf{dk} := (d_1, \ldots, d_l) \leftarrow \mathbb{Z}_p^l$ and computes the encryption key $\mathsf{ek} := (1, d_1, \ldots, d_l) \cdot G \in \mathbb{G}^{l+1}$.
- $\mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho) \to \mathsf{ct}$: Given the encryption key ek and a message $\mathbf{M} \in \mathbb{G}^l$, the encryption algorithm Enc samples $\rho \leftarrow \mathbb{Z}_p$ and computes the ciphertext $\mathsf{ct} := (O, \mathbf{M}) + \rho \cdot \mathsf{ek} \in \mathbb{G}^{l+1}$.

- $\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}) \to \mathbf{M}$: Given the decryption key dk and a ciphertext $\mathsf{ct} = (C_0, C_1, \ldots, C_l)$, the decryption algorithm Dec computes the plaintext as $(C_1, \ldots, C_l) - \mathsf{dk} \cdot C_0$.

A public-key encryption scheme is *rerandomizable* if there exists an algorithm Rerand that, given the encryption key ek and a ciphertext ct, outputs a fresh-looking encryption ct' of the same message. The above scheme is rerandomizable: $\mathsf{Rerand}(\mathsf{ek}, \mathsf{ct})$ samples $r \leftarrow \mathbb{Z}_p$ and outputs $\mathsf{ct}' := \mathsf{ct} + r \cdot \mathsf{ek}$.

The scheme is parameterized by the message length l. Different parts of the construction may use different values of l. To encrypt a shorter message $\mathbf{M} \in \mathbb{G}^{l'}$ with $l' \leq l$, we use the first $l' + 1$ components of ek and write $\mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho)$ by slight abuse of notation.

In the CPA security game CPA^b, the adversary A is given an encryption key ek. Then A outputs two messages $(\mathbf{M}_0, \mathbf{M}_1)$ and receives $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{M}_b)$. We say the scheme is (τ, ε)-CPA secure if for any τ-time A,

$$\left|\Pr[\mathsf{CPA}^0(\mathsf{A}) = 1] - \Pr[\mathsf{CPA}^1(\mathsf{A}) = 1]\right| \leq \varepsilon.$$

By the self-reduction of DDH, PKE is $(\tau, \varepsilon_{\mathrm{ddh}})$-CPA secure if DDH is $(\tau', \varepsilon_{\mathrm{ddh}})$-hard for $\tau' \approx \tau$.

2.1 Σ-Protocols and Linear Preimage Relation

We define Σ-protocols below.

Definition 3 (Σ-protocols)**.** A Σ-protocol for a binary statement-witness relation $\mathcal{R} = \mathcal{X} \times \mathcal{W}$ consists of a challenge set $\mathcal{C}$, a randomness set $\Omega = \Omega(\mathsf{stm})$ (as a function of the statement stm), and the following algorithms:

- $\mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho) \to \mathsf{cm}$: On inputs a statement stm and a witness wit, Com outputs a commitment cm.
- $\mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho) \to \mathsf{rs}$: On inputs a witness wit, a challenge ch, and randomness ρ, the deterministic algorithm Resp outputs a response rs.
- $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) \to 0/1$: On inputs a statement stm, a commitment cm, a challenge ch, and a response rs, the deterministic algorithm Vf outputs 0 or 1.

The following properties are required to hold:

- (Perfect) Correctness: For every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$, every $\rho \in \Omega(\mathsf{stm})$, and every $\mathsf{ch} \in \mathcal{C}$, if we let $\mathsf{cm} := \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)$, $\mathsf{rs} := \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)$, then it must hold that $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) = 1$.
- (Perfect) special honest-verifier zero-knowledge (SHVZK): There exists an efficient simulator algorithm Sim such that, for every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$ and $\mathsf{ch} \in \mathcal{C}$, the distribution of $(\mathsf{cm}, \mathsf{rs}) \leftarrow \mathsf{Sim}(\mathsf{stm}, \mathsf{ch})$ is identical to that of $(\mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho), \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho))$ where $\rho \leftarrow \Omega(\mathsf{stm})$.
- Special soundness: There exists an efficient deterministic extractor algorithm Ext such that, for every $\mathsf{stm} \in \mathcal{X}$ and $\mathsf{cm}, \mathsf{ch}, \mathsf{ch}', \mathsf{rs}, \mathsf{rs}'$ satisfying $\mathsf{ch} \neq \mathsf{ch}'$, $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) = 1$, and $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}', \mathsf{rs}') = 1$, if we let $\mathsf{wit} := \mathsf{Ext}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{ch}', \mathsf{rs}, \mathsf{rs}')$, then it must hold that $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$.

Linear Preimage Relation.

We define the linear preimage relation $\mathcal{R}_{\mathrm{lin}}$ below. Note that we do not explicitly restrict the dimensions of matrices and vectors, but $((\mathbf{M}, \mathbf{V}), \mathbf{w}) \in \mathcal{R}_{\mathrm{lin}}$ implicitly requires that the dimensions of $\mathbf{M}$, $\mathbf{V}$, and $\mathbf{w}$ are compatible.

Definition 4 (Linear preimage relation)**.** Let $\mathcal{X}$ consist of all pairs $(\mathbf{M}, \mathbf{V})$ where $\mathbf{M}$ is a matrix and $\mathbf{V}$ is a column vector over $\mathbb{G}$. Let $\mathcal{W}$ consist of all column vectors $\mathbf{w}$ over $\mathbb{Z}_p$. The linear preimage relation $\mathcal{R}_{\mathrm{lin}} \subseteq \mathcal{X} \times \mathcal{W}$ over group $(\mathbb{G}, p, G)$ is defined as $\{((\mathbf{M}, \mathbf{V}), \mathbf{w}) : \mathbf{M}\mathbf{w} = \mathbf{V}\}$.

In Fig. 1, we describe a Σ-protocol for $\mathcal{R}_{\mathrm{lin}}$ over $(\mathbb{G}, p, G)$, which can be viewed as a generalization of the Schnorr protocol. We also present the SHVZK simulator Sim and the special-soundness extractor Ext. It is straightforward to prove that the protocol satisfies correctness, SHVZK, and special soundness. We also describe the non-interactive prover and verifier algorithms NIP, NIV defined with respect to a hash function H. They are obtained by applying the Fiat-Shamir transformation [FS87] to the Σ-protocol (in fact, we just describe the generic transformation).

$\underline{\mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)}$
$(\mathbf{M}, \mathbf{V}) := \mathsf{stm}$
$\mathbf{r} := \rho$
Return $\mathbf{Mr}$

$\underline{\mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)}$
$\mathbf{w} := \mathsf{wit}$
$\mathbf{r} := \rho$
Return $\mathbf{r} + \mathsf{ch} \cdot \mathbf{w}$

$\underline{\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs})}$
$(\mathbf{M}, \mathbf{V}) := \mathsf{stm}$
Return $[\![\mathbf{M} \cdot \mathsf{rs} = \mathsf{cm} + \mathsf{ch} \cdot \mathbf{V}]\!]$

$\underline{\mathsf{Sim}(\mathsf{stm}, \mathsf{ch})}$
$(\mathbf{M}, \mathbf{V}) := \mathsf{stm}$
$\mathsf{rs} \leftarrow \mathbb{Z}_p^n$
$\mathsf{cm} := \mathbf{M} \cdot \mathsf{rs} - \mathsf{ch} \cdot \mathbf{V}$
Return $(\mathsf{cm}, \mathsf{rs})$

$\underline{\mathsf{Ext}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{ch}', \mathsf{rs}, \mathsf{rs}')}$
Return $(\mathsf{rs}' - \mathsf{rs})/(\mathsf{ch} - \mathsf{ch}')$

$\underline{\mathsf{NIP}^{\mathsf{H}}(\mathsf{stm}, \mathsf{wit})}$
$\mathsf{cm} \leftarrow \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)$
$\mathsf{ch} := \mathsf{H}(\mathsf{stm}, \mathsf{cm})$
$\mathsf{rs} := \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)$
Return $(\mathsf{cm}, \mathsf{rs})$

$\underline{\mathsf{NIV}^{\mathsf{H}}(\mathsf{stm}, (\mathsf{cm}, \mathsf{rs}))}$
$\mathsf{ch} := \mathsf{H}(\mathsf{stm}, \mathsf{cm})$
Return $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs})$

Figure 1: Σ-protocol for $\mathcal{R}_{\text{lin}}$. We have $\mathcal{C} = \mathbb{Z}_p$. For $\mathsf{stm} = (\mathbf{M}, \mathbf{V})$, if $\mathbf{M}$ is an $m \times n$ matrix, then $\Omega(\mathsf{stm}) = \mathbb{Z}_p^n$.

2.2 Three-Move Blind Signatures

We define three-move blind signature schemes below. The signing protocol consists of four algorithms. Sign_1, User_1, Sign_2 generate the protocol messages for the three moves, respectively. The last algorithm User_2 generates the final signature.

Definition 5 (Three-Move Blind Signature Schemes)**.** A three-move blind signature scheme BS consists of the following algorithms:

- $\mathsf{Setup}() \to \mathsf{pp}$: The setup algorithm Setup outputs global parameters pp. All algorithms related to BS implicitly take pp as input.
- $\mathsf{KGen}() \to (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm KGen outputs a public key pk and a secret key sk.
- The interaction between the signer and the user consists of the following algorithms:
  - $\mathsf{Sign}_1(\mathsf{sk}) \to (\mathsf{sst}, \mathsf{smsg}_1)$: On input a secret key sk, the first-round signing algorithm Sign_1 outputs a signer state sst and a first-round signer message smsg_1.
  - $\mathsf{User}_1(\mathsf{pk}, \mu, \mathsf{smsg}_1) \to (\mathsf{ust}, \mathsf{umsg})$: On inputs a public key pk, a message μ, and a first-round signer message smsg_1, the first-round user algorithm User_1 outputs a user state ust and a user message umsg.
  - $\mathsf{Sign}_2(\mathsf{sk}, \mathsf{sst}, \mathsf{umsg}) \to \mathsf{smsg}_2$: On inputs a secret key sk, a signer state sst, and a user message umsg, the second-round signing algorithm Sign_2 outputs a second-round signer message smsg_2.
  - $\mathsf{User}_2(\mathsf{pk}, \mu, \mathsf{ust}, \mathsf{smsg}_2) \to \sigma$: On inputs a public key pk, a message μ, a user state ust, and a second-round signer message smsg_2, the second-round user algorithm User_2 outputs a signature σ.
- $\mathsf{Vf}(\mathsf{pk}, \mu, \sigma) \to 0/1$: On inputs a public key pk, a message μ, and a signature σ, the deterministic verification algorithm Vf outputs 0 or 1.

We say BS is correct if for every $\mathsf{pp} \in \mathsf{Setup}()$, every $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}()$, and every message μ, for signature σ returned by the following procedure, we have $\Pr[\mathsf{Vf}(\mathsf{pk}, \mu, \sigma) = 1] = 1$.

$$(\mathsf{sst}, \mathsf{smsg}_1) \leftarrow \mathsf{Sign}_1(\mathsf{sk}); \quad (\mathsf{ust}, \mathsf{umsg}) \leftarrow \mathsf{User}_1(\mathsf{pk}, \mu, \mathsf{smsg}_1);$$
$$\mathsf{smsg}_2 \leftarrow \mathsf{Sign}_2(\mathsf{sk}, \mathsf{sst}, \mathsf{umsg}); \quad \sigma \leftarrow \mathsf{User}_2(\mathsf{pk}, \mu, \mathsf{ust}, \mathsf{smsg}_2).$$

Blindness is defined by the BLIND security game described in Fig. 2. The game secretly samples $\beta_0 \leftarrow \{0, 1\}$ and lets $\beta_1 := 1 - \beta_0$. The two-stage adversary A first (possibly maliciously) generates a public key pk and two messages μ_0, μ_1. Then it can open two arbitrarily interleaved signing sessions with the user oracles $\textsc{User}_1$, $\textsc{User}_2$. For $j \in \{0, 1\}$, the user runs the protocol on message

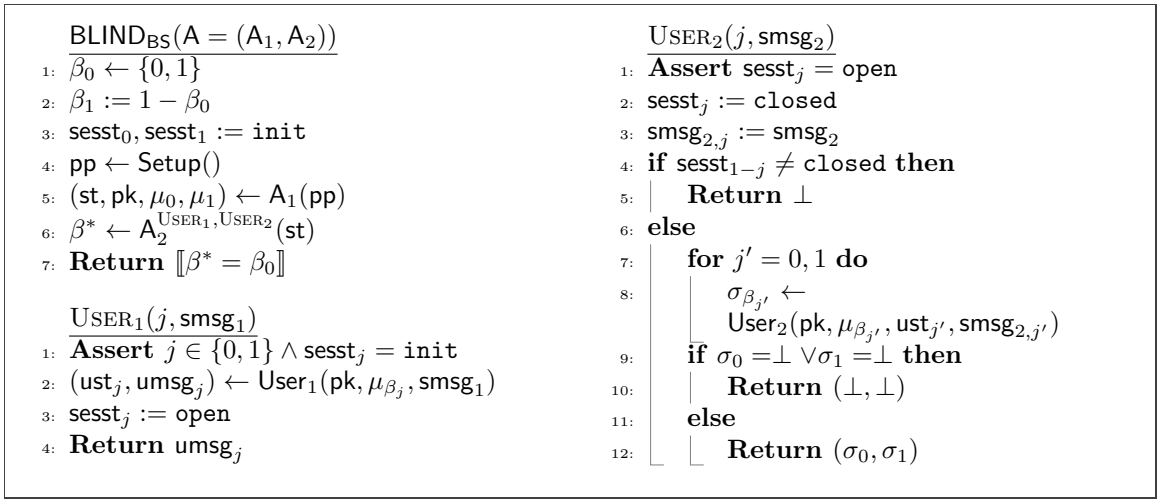

Figure 2: The BLIND security game.

μ_{β_j} to generate a signature σ_{β_j} in session j. When both sessions are successfully completed, the adversary is given both signatures σ_0, σ_1. The goal of A is to guess the secret bit β_0, i.e., to link the message-signature pairs with the corresponding signing sessions.

Definition 6 (Blindness)**.** We say a three-move blind signature scheme BS is (τ, ε)-blind if for every τ-time algorithm A, $\Pr[\mathsf{BLIND}_{\mathsf{BS}}(\mathsf{A}) = 1] \leq 1/2 + \varepsilon$, where the blindness security game BLIND is defined in Fig. 2.

One-more unforgeability (OMUF) is defined by the OMUF security game described in Fig. 3. The adversary A can interact with the signer oracles $\textsc{Sign}_1$, $\textsc{Sign}_2$ in an arbitrarily concurrent fashion. The goal of A is to forge signatures on $k+1$ distinct messages, where k is the number of completed signing sessions. We stress that it is allowed to have more opened yet incomplete sessions than the number of forged signatures.

Definition 7 (OMUF)**.** We say a three-move blind signature scheme BS is (τ, ε)-one-more unforgeable (OMUF) if for every τ-time algorithm A,

$$\Pr[\mathsf{OMUF}_{\mathsf{BS}}(\mathsf{A}) = 1] \leq \varepsilon,$$

where the OMUF security game OMUF is defined in Fig. 3.

3 Encrypted-Message Blind Signatures

In this section, we define four-move encrypted-message blind signature (EMBS) schemes as our main building block. An EMBS scheme is defined over a group $(\mathbb{G}, p, G)$ and associated with a PKE scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathbb{G}^l$. An encryption key ek is known to both the signer and the user. The signing protocol generates a signature that, instead of the message $\mathbf{M}$, verifies on an encryption of the message $\mathsf{ct} := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho)$. We require that ρ is generated by the user together with the third-move protocol message. We also restrict that the first move is simply sending the message to the signer in the clear. Hence, we only need four algorithms to specify the protocol: Sign_1 and Sign_2 generate the signer's messages for the second and fourth moves, respectively; User_1 generates the user's message for the third move as well as the encryption randomness ρ; and User_2 outputs the final signature. One can also view the protocol as three-move, with the message $\mathbf{M}$ publicly known by both parties in advance.

$\underline{\mathsf{OMUF}_{\mathsf{BS}}(\mathsf{A})}$

1: $k := 0, \mathsf{ctr} := 0$
2: $\mathsf{pp} \leftarrow \mathsf{Setup}()$
3: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$
4: $\{(\mu_i^*, \sigma_i^*)\}_{i \in [k+1]^+} \leftarrow \mathsf{A}^{\textsc{Sign}_1, \textsc{Sign}_2}(\mathsf{pp}, \mathsf{pk})$
5: **Return** $[\![\forall i \neq j \in [k+1]^+ \colon \mu_i^* \neq \mu_j^* \wedge \forall i \in [k+1]^+ \colon \mathsf{Vf}(\mathsf{pk}, \mu_i^*, \sigma_i^*) = 1]\!]$

$\underline{\textsc{Sign}_1()}$

1: $\mathsf{ctr} := \mathsf{ctr} + 1$
2: $j := \mathsf{ctr}$
3: $(\mathsf{sst}_j, \mathsf{smsg}_{1,j}) \leftarrow \mathsf{Sign}_1(\mathsf{sk})$
4: $\mathsf{sesst}_j := \mathtt{open}$
5: **Return** $\mathsf{smsg}_{1,j}$

$\underline{\textsc{Sign}_2(j, \mathsf{umsg})}$

1: **Assert** $\mathsf{sesst}_j = \mathtt{open}$
2: $\mathsf{smsg}_{2,j} \leftarrow \mathsf{Sign}_2(\mathsf{sk}, \mathsf{sst}_j, \mathsf{umsg})$
3: $k := k + 1$
4: $\mathsf{sesst}_j := \mathtt{closed}$
5: **Return** $\mathsf{smsg}_{2,j}$

Figure 3: The OMUF security game.

Definition 8 (Four-Move EMBS)**.** Let $(\mathbb{G}, p, G)$ be a description of a prime-order cyclic group and $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with message space $\mathbb{G}^l$ and randomness space $\mathcal{R}$. A four-move encrypted-message blind signature scheme EMBS over $(\mathbb{G}, p, G)$ associated with PKE consists of the following algorithms.

- $\mathsf{Setup}() \to \mathsf{pp}$: The setup algorithm Setup outputs global parameters pp. All algorithms related to EMBS implicitly take pp as input.
- $\mathsf{KGen}() \to (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm KGen outputs a public key pk and a secret key sk.
- The interaction between the signer and the user consists of the following algorithms:
  - $\mathsf{Sign}_1(\mathbf{M}, \mathsf{ek}, \mathsf{sk}) \to (\mathsf{sst}, \mathsf{smsg}_1)$: On inputs a message $\mathbf{M} \in \mathbb{G}^l$, an encryption key ek, and a secret key sk, the first-round signing algorithm Sign_1 outputs a signer state sst and a first-round signer message smsg_1.
  - $\mathsf{User}_1(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1) \to (\mathsf{ust}, \mathsf{umsg}, \rho)$: On inputs a message $\mathbf{M} \in \mathbb{G}^l$, an encryption key ek, a public key pk, and a first-round signer message smsg_1, the first-round user algorithm User_1 outputs a user state ust, a user message umsg, and an encryption randomness $\rho \in \mathcal{R}$.
  - $\mathsf{Sign}_2(\mathbf{M}, \mathsf{ek}, \mathsf{sk}, \mathsf{sst}, \mathsf{umsg}) \to \mathsf{smsg}_2$: On inputs a message $\mathbf{M} \in \mathbb{G}^l$, an encryption key ek, a secret key sk, a signer state sst, and a user message umsg, the second-round signing algorithm Sign_2 outputs a second-round signer message smsg_2.
  - $\mathsf{User}_2(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{ust}, \mathsf{smsg}_2) \to \sigma$: On inputs a message $\mathbf{M} \in \mathbb{G}^l$, an encryption key ek, a public key pk, a user state ust, and a second-round signer message smsg_2, the second-round user algorithm User_2 outputs a signature σ.
- $\mathsf{Vf}(\mathsf{pk}, \mathsf{ct}, \sigma) \to 0/1$: On inputs a public key pk, a ciphertext ct, and a signature σ, the deterministic verification algorithm Vf outputs 0 or 1.

We say EMBS is correct if for every $\mathsf{EMBS.pp} \in \mathsf{EMBS.Setup}()$, every $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{EMBS.KGen}()$, every $(\mathsf{ek}, \mathsf{dk}) \in \mathsf{PKE.KGen}()$, and every message $\mathbf{M} \in \mathbb{G}^l$, for signature σ and randomness ρ returned by the following procedure, we have $\Pr[\mathsf{Vf}(\mathsf{pk}, \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho), \sigma) = 1] = 1$.

$$(\mathsf{sst}, \mathsf{smsg}_1) \leftarrow \mathsf{Sign}_1(\mathbf{M}, \mathsf{ek}, \mathsf{sk}); \quad (\mathsf{ust}, \mathsf{umsg}, \rho) \leftarrow \mathsf{User}_1(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1);$$
$$\mathsf{smsg}_2 \leftarrow \mathsf{Sign}_2(\mathbf{M}, \mathsf{ek}, \mathsf{sk}, \mathsf{sst}, \mathsf{umsg}); \quad \sigma \leftarrow \mathsf{User}_2(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{ust}, \mathsf{smsg}_2).$$

Blindness of EMBS schemes is notably different from that for blind signature schemes, since the message $\mathbf{M}$ is now known to the signer. We define *encrypted-message blindness* as a joint property of an EMBS scheme and the associated PKE scheme. To set up the security game, a PKE key pair $(\mathsf{ek}, \mathsf{dk})$ is generated, and ek is given to A. The two-stage adversary A first (possibly maliciously) generates a public key pk. Compared to the blindness for blind signature schemes, here A does

$\underline{\mathsf{EMBLIND}_{\mathsf{EMBS}}(\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2))}$

$\beta_0 \leftarrow \{0,1\}$
$\beta_1 := 1 - \beta_0$
$\mathsf{sesst}_0, \mathsf{sesst}_1 := \mathtt{init}$
$\mathsf{EMBS.pp} \leftarrow \mathsf{EMBS.Setup}()$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KGen}()$
$(\mathsf{st}, \mathsf{pk}) \leftarrow \mathsf{A}_1(\mathsf{EMBS.pp}, \mathsf{ek})$
$\beta^* \leftarrow \mathsf{A}_2^{\textsc{User}_1, \textsc{User}_2}(\mathsf{st})$
Return $[\![\beta^* = \beta_0]\!]$

$\underline{\textsc{User}_1(j, \mathbf{M}, \mathsf{smsg}_1)}$

Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \mathtt{init}$
$\mathbf{M}_j := \mathbf{M}$
$(\mathsf{ust}_j, \mathsf{umsg}_j, \rho_j) \leftarrow \mathsf{User}_1(\mathbf{M}_j, \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1)$
$\mathsf{sesst}_j := \mathtt{open}$
Return umsg_j

$\underline{\textsc{User}_2(j, \mathsf{smsg}_2)}$

Assert $\mathsf{sesst}_j = \mathtt{open}$
$\mathsf{sesst}_j := \mathtt{closed}$
$\mathsf{smsg}_{2,j} := \mathsf{smsg}_2$
if $\mathsf{sesst}_{1-j} \neq \mathtt{closed}$ **then**
  Return $\perp$
else
  for $j' = 0, 1$ **do**
    $\sigma_{j'} \leftarrow \mathsf{User}_2(\mathbf{M}_{j'}, \mathsf{ek}, \mathsf{pk}, \mathsf{ust}_{j'}, \mathsf{smsg}_{2,j'})$
    $\mathsf{ct}_{j'} := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}_{j'}, \rho_{j'})$
  if $\sigma_0 = \perp \vee \sigma_1 = \perp$ **then**
    Return $(\perp, \perp)$
  else
    Return $((\mathsf{ct}_{\beta_0}, \sigma_{\beta_0}), (\mathsf{ct}_{\beta_1}, \sigma_{\beta_1}))$

Figure 4: The EMBLIND security game.

not need to choose two messages in advance. Then A can open two arbitrarily interleaved signing sessions with the user oracles $\textsc{User}_1$, $\textsc{User}_2$, with a message $\mathbf{M}$ as the input of $\textsc{User}_1$. When both sessions are completed, the adversary is given two ciphertext-signature pairs $(\mathsf{Enc}(\mathsf{ek}, \mathbf{M}_0; \rho_0), \sigma_0)$ and $(\mathsf{Enc}(\mathsf{ek}, \mathbf{M}_1; \rho_1), \sigma_1)$ that are swapped with probability $1/2$, where the encryption randomness ρ_j and signature σ_j are generated by session j. The goal of A is to guess whether the two pairs are swapped.

Definition 9 (Encrypted-Message Blindness)**.** We say a four-move EMBS scheme EMBS is (τ, ε)-encrypted-message blind if for every τ-time algorithm A, $\Pr[\mathsf{EMBLIND}_{\mathsf{EMBS}}(\mathsf{A}) = 1] \leq 1/2 + \varepsilon$, where the security game EMBLIND is defined in Fig. 4.

Since the message $\mathbf{M}$ is known to the signer, unforgeability of EMBS schemes can be defined similarly to that of standard signature schemes. The adversary A is given the public key and a PKE key pair $(\mathsf{ek}, \mathsf{dk})$. It can interact with the signer oracles $\textsc{Sign}_1$, $\textsc{Sign}_2$ in an arbitrarily concurrent fashion. The goal of A is to forge a signature on a message $\mathbf{M}^*$ on which it has never completed a signing session. We stress that A is allowed to have incomplete sessions on $\mathbf{M}^*$.

We define the unforgeability against two types of attacks, the distinct-message attack (DMA) and the extended-random-message attack (xRMA) [ACD+16]. The distinct-message attack allows A to choose the message $\mathbf{M}$ to sign, with each message signed at most once. In the random-message attack, the message $\mathbf{M}$ is instead uniformly chosen by oracle $\textsc{Sign}_1$. The extended-random-message attack provides some extra information about $\mathbf{M}$. Specifically, in this work we let the extra information be the discrete logarithms of $\mathbf{M}$. It is easy to see that UF-DMA implies UF-xRMA under an additive loss of $O(q_s^2)/p^l$. UF-xRMA security is sufficient for our construction of three-move blind signature scheme, while we will give an instantiation of EMBS with UF-DMA security.

Definition 10 (UFDMA, UFxRMA)**.** A four-move EMBS scheme EMBS is (τ, ε)-unforgeable under distinct-message (resp. extended random-message) attack (UF-DMA (resp. UF-xRMA)) if for every τ-time algorithm A,

$$\Pr[\mathsf{UF\text{-}DMA}_{\mathsf{EMBS}}(\mathsf{A}) = 1] \leq \varepsilon \text{ (resp. } \Pr[\mathsf{UF\text{-}xRMA}_{\mathsf{EMBS}}(\mathsf{A}) = 1] \leq \varepsilon),$$

where the security games UF-DMA/UF-xRMA are defined in Fig. 5.

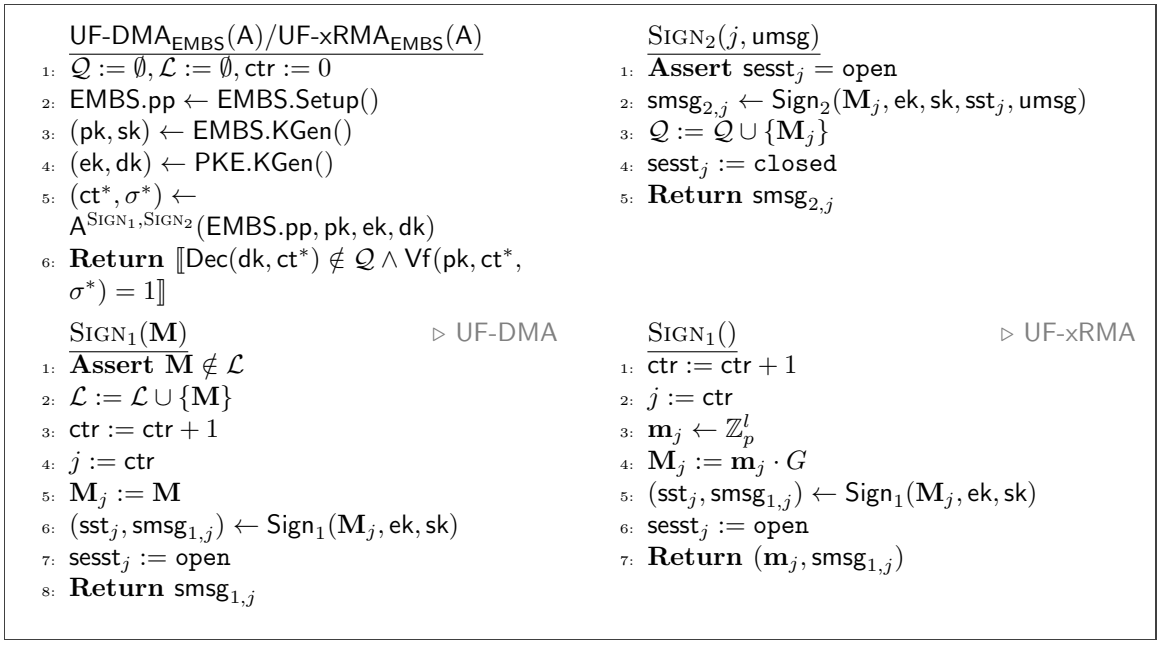

Figure 5: The UF-DMA and UF-xRMA security games.

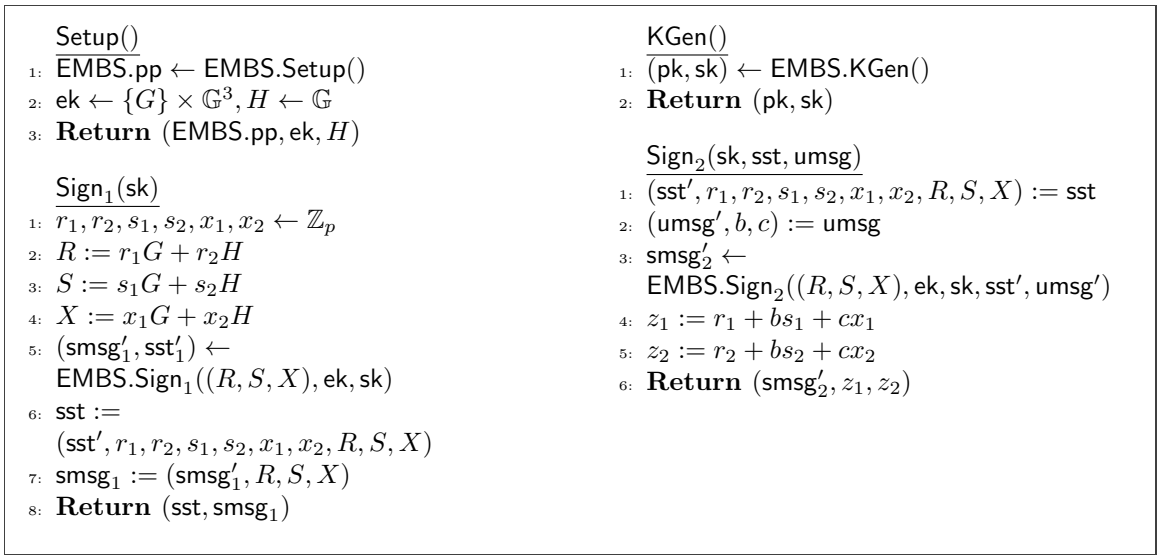

Figure 6: Our construction of three-move blind signature scheme.

4 Our Three-Move Blind Signature Scheme

4.1 Construction

Let PKE be the extended ElGamal encryption scheme specified in Section 2. Let EMBS be a four-move EMBS scheme over $(\mathbb{G}, p, G)$ associated with PKE. Let (NIP, NIV) be the non-interactive proof system for $\mathcal{R}_{\text{lin}}$, transformed from the Σ-protocol Σ in Fig. 1. We present the pseudocode description of our construction of three-move blind signature scheme BS[EMBS] in Figs. 6 and 7. Our construction may be considered as semi-black-box: We require the EMBS scheme to be associated with a specific encryption scheme. Beyond that, we put no restriction on the EMBS scheme and

$\underline{\mathsf{User}_1(\mathsf{pk}, \mu, \mathsf{smsg}_1)}$

$(\mathsf{smsg}'_1, R, S, X) := \mathsf{smsg}_1$
$(\mathsf{ust}', \mathsf{umsg}', \rho_C) \leftarrow \mathsf{EMBS.User}_1((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}'_1)$
$\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x \leftarrow \mathbb{Z}_p$
$\alpha_b, \alpha_c, \gamma \leftarrow \mathbb{Z}_p^*$
$\bar{S} := \alpha_b^{-1}(S + \delta_s G)$
$\bar{X} := \alpha_c^{-1}(X + \delta_x G)$
$\bar{R} := R + \delta_{z_1} G + \delta_{z_2} H$
$\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$
$\mathbf{D} := (D, D_{\alpha_b}, D_{\alpha_c}) \leftarrow \mathsf{Enc}(\mathsf{ek}, (\alpha_b G, \alpha_c G); \rho_D)$
$M^\dagger := \gamma G$
$\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$
$\mathsf{wit} := \mathsf{FormatW}(\rho_C, \rho_D, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c, \gamma)$
$\pi \leftarrow \mathsf{NIP}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm}, \mathsf{wit})$
$(\bar{b}, \bar{c}) := \mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$
$b := \bar{b}/\alpha_b$
$c := \bar{c}/\alpha_c$
$\mathsf{ust} := (\mathsf{ust}', R, S, X, b, c, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi)$
Return $(\mathsf{ust}, (\mathsf{umsg}', b, c))$

$\underline{\mathsf{User}_2(\mathsf{pk}, \mu, \mathsf{ust}, \mathsf{smsg}_2)}$

$(\mathsf{smsg}'_2, z_1, z_2) := \mathsf{smsg}_2$
$(\mathsf{ust}', R, S, X, b, c, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi) := \mathsf{ust}$
$\sigma' \leftarrow \mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$
$\bar{z}_1 := z_1 + \delta_{z_1} + b\delta_s + c\delta_x$
$\bar{z}_2 := z_2 + \delta_{z_2}$
Return $(\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2)$

$\underline{\mathsf{Vf}(\mathsf{pk}, \mu, \sigma)}$

$(\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2) := \sigma$
$(\bar{b}, \bar{c}) := \mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$
$\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$
Return $[\![\mathsf{EMBS.Vf}(\mathsf{pk}, \mathbf{C}, \sigma') \wedge \mathsf{NIV}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm}, \pi) \wedge M^\dagger \neq O \wedge \bar{z}_1 G + \bar{z}_2 H = \bar{R} + \bar{b}\bar{S} + \bar{c}\bar{X}]\!]$

Figure 7: Our construction of three-move blind signature scheme. Random oracle H has range $\mathbb{Z}_p^{*2}$.

uses it in a black-box way.

Setup and key generation. The setup algorithm Setup generates the public parameters, consisting of the EMBS parameters $\mathsf{EMBS.pp} \leftarrow \mathsf{EMBS.Setup}()$, an encryption key $\mathsf{ek} \leftarrow \{G\} \times \mathbb{G}^3$, and a random element $H \leftarrow \mathbb{G}$. The key generation KGen generates an EMBS key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{EMBS.KGen}()$.

Signing protocol. The signing protocol is as follows:

For the first move, the signer samples the two-time secret key $r_1, r_2, s_1, s_2, x_1, x_2 \leftarrow \mathbb{Z}_p$ and computes the two-time public key $R := r_1 G + r_2 H$, $S := s_1 G + s_2 H$, $X := x_1 G + x_2 H$. It runs $\mathsf{EMBS.Sign}_1((R, S, X), \mathsf{ek}, \mathsf{sk})$ to obtain the EMBS protocol message and state $(\mathsf{smsg}'_1, \mathsf{sst}'_1)$. It outputs $(\mathsf{smsg}'_1, R, S, X)$ as the protocol message.

For the second move, the user receives $(\mathsf{smsg}'_1, R, S, X)$. Then the user runs $\mathsf{EMBS.User}_1((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}'_1)$ to obtain the protocol message, state, and the encryption randomness $(\mathsf{ust}', \mathsf{umsg}', \rho_C)$. It uses ρ_C to generate an encryption of the two-time key $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$.

The user samples the blinding factors $\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x \leftarrow \mathbb{Z}_p$ and $\alpha_b, \alpha_c \leftarrow \mathbb{Z}_p^*$. It blinds the two-time public key into

$$\bar{S} := \alpha_b^{-1}(S + \delta_s G),\ \ \bar{X} := \alpha_c^{-1}(X + \delta_x G),\ \ \bar{R} := R + \delta_{z_1} G + \delta_{z_2} H.$$

It commits the blinding factors α_b, α_c by $\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (\alpha_b G, \alpha_c G); \rho_D)$, where the encryption uses the first three components of ek as described in Section 2. It generates a proof of knowledge of the blinding factors that connect the blinded key $(\bar{R}, \bar{S}, \bar{X})$ and the plain key (R, S, X) encrypted in $\mathbf{C}$, where α_b, α_c are bound in $\mathbf{D}$ and are non-zero. For the proof of non-zeros, the user samples $\gamma \leftarrow \mathbb{Z}_p^*$ and lets $M^\dagger := \gamma G$. Then it uses NIP to generate a non-interactive proof of knowledge π of $(\rho_C, \rho_D, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c, \eta_b, \zeta_b, \eta_c, \zeta_c)$ satisfying

$$\rho_C(G, E_1, E_2, E_3) + (O,\ \bar{R} - \delta_{z_1} G - \delta_{z_2} H,\ \alpha_b \bar{S} - \delta_s G,\ \alpha_c \bar{X} - \delta_x G) = \mathbf{C}, \tag{1}$$

$$\rho_D(G, E_1, E_2) + (O, \alpha_b G, \alpha_c G) = \mathbf{D}, \tag{2}$$

$$\eta_b(D, D_{\alpha_b}) + \zeta_b(G, E_1) = (O, M^\dagger), \tag{3}$$

$$\eta_c(D, D_{\alpha_c}) + \zeta_c(G, E_2) = (O, M^\dagger), \tag{4}$$

where ek is parsed as (G, E_1, E_2, E_3) and $\mathbf{D}$ as $(D, D_{\alpha_b}, D_{\alpha_c})$. Eq. (1) states that $\mathbf{C}$ encrypts the unblinded two-time key. Eq. (2) states that $\mathbf{D}$ encrypts $(\alpha_b G, \alpha_c G)$. Eqs. (3) and (4) state that there is a way to rerandomize (D, D_{α_b}) and (D, D_{α_c}) into an encryption of $M^\dagger$, respectively. Note that $M^\dagger \neq O$. Thus, Eqs. (3) and (4) guarantee that $\mathbf{D}$ does not encrypt O. The user uses $\eta_b := \gamma/\alpha_b$, $\zeta_b := \eta_b \rho_D$, $\eta_c := \gamma/\alpha_c$, $\zeta_c := \eta_c \rho_D$ in the witness. We assume helper algorithms $\mathsf{FormatS}$ and $\mathsf{FormatW}$ that compute and format the statement and witness into the form of the $\mathcal{R}_{\text{lin}}$ relation. We omit their description, which is straightforward.

Then the user computes $(\bar{b}, \bar{c}) := \mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$ for a random oracle H with range $\mathbb{Z}_p^{*2}$. It unblinds them into $b := \bar{b}/\alpha_b$ and $c := \bar{c}/\alpha_c$. It returns (umsg', b, c) as the protocol message.

For the third move, the signer receives (umsg', b, c). It runs $\mathsf{EMBS.Sign}_2((R, S, X), \mathsf{ek}, \mathsf{sk}, \mathsf{sst}', \mathsf{umsg}')$ to obtain the EMBS protocol message smsg'_2, and it computes the two-time signature $(z_1, z_2) := (r_1, r_2) + b(s_1, s_2) + c(x_1, x_2)$. It sends $(\mathsf{smsg}'_2, z_1, z_2)$.

Finally, the user receives $(\mathsf{smsg}'_2, z_1, z_2)$. It runs $\mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$ to obtain the EMBS signature σ'. Then it blinds the two-time signature into $\bar{z}_1 := z_1 + \delta_{z_1} + b\delta_s + c\delta_x$, $\bar{z}_2 := z_2 + \delta_{z_2}$. It returns $(\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2)$ as the signature.

Verification. To verify the signature, the verifier computes $(\bar{b}, \bar{c}) := \mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$ and verifies that: 1) σ' is a valid EMBS signature on $\mathbf{C}$, 2) π is valid, 3) $M^\dagger \neq O$, 4) $(\bar{z}_1, \bar{z}_2)$ is a valid two-time signature, namely $\bar{z}_1 G + \bar{z}_2 H = \bar{R} + \bar{b}\bar{S} + \bar{c}\bar{X}$.

Correctness. For correctness, note that $\bar{b}\bar{S} = \alpha_b b \cdot \alpha_b^{-1}(S + \delta_s G) = b(S + \delta_s G)$, and similarly $\bar{c}\bar{X} = c(X + \delta_x G)$. Hence, we have

$$\begin{aligned}
&\bar{z}_1 G + \bar{z}_2 H - \bar{b}\bar{S} - \bar{c}\bar{X} \\
&= (z_1 + \delta_{z_1} + b\delta_s + c\delta_x)G + (z_2 + \delta_{z_2})H - b(S + \delta_s G) - c(X + \delta_x G) \\
&= z_1 G + z_2 H - bS - cX + \delta_{z_1} G + \delta_{z_2} H \\
&= R + \delta_{z_1} G + \delta_{z_2} H = \bar{R}.
\end{aligned}$$

4.2 Blindness

Theorem 1 (Blindness). *If* DDH *is* $(\tau', \varepsilon_{\text{ddh}})$*-hard over* $(\mathbb{G}, p, G)$*, and* EMBS *is* $(\tau', \varepsilon_{\text{embs}})$*-encrypted-message blind, then* BS[EMBS] *is* $(\tau, \varepsilon_{\text{bs}})$*-blind, where essentially* $\tau' \approx \tau$*, and*

$$\varepsilon_{\text{bs}} \leq \varepsilon_{\text{embs}} + 4\varepsilon_{\text{ddh}} + \frac{O(q_h)}{p}.$$

We defer the proof to Section A and provide the idea here. We will build a series of game hops and finally arrive at a game for the reduction B who plays the $\mathsf{EMBLIND}_{\mathsf{EMBS}}$ game to simulate. In the game hops, we can let User_1 simulate the non-interactive proofs. Then, we can let User_1 generate $\mathbf{D}$ as an encryption of an unrelated message. Next, we can let User_1 only generate umsg' and uniformly sample (b, c). All the other operations are deferred to User_2.

Now in User_2, we can reorder the generation of variables. Essentially, the game samples the blinded components uniformly and independently, and computes the other variables only if needed. As a result, in the signature, all the components are "session-independent", except $\mathbf{C}, \pi, \sigma'$. The reduction generates the session-independent part and uses the two encryption-signature pairs that it receives from the EMBLIND game to generate the session-dependent $\mathbf{C}, \pi, \sigma'$. By doing so, it simulates the modified BLIND game such that $\beta_0 = 1$ if and only if the encryption-signature pairs are swapped.

4.3 One-More Unforgeability

Theorem 2 (OMUF). *If* DL *is* $(\tau', \varepsilon_{\text{dl}})$*-hard over* $(\mathbb{G}, p, G)$*, and* EMBS *is* $(q_s, \tau', \varepsilon_{\text{embs}})$*-*UF-xRMA*, then* BS[EMBS] *is* $(q_s, \tau, \varepsilon_{\text{bs}})$*-*OMUF *for any adversary that makes at most* q_h *queries to each random oracle, where essentially* $\tau' \approx \tau$*,* $q = O(q_s + q_h)$*, and*

$$\varepsilon_{\text{bs}} \leq \varepsilon_{\text{embs}} + \sqrt{2q\varepsilon_{\text{dl}} + O(q^2)/p}.$$

Proof. Let (R_j, S_j, X_j) denote the (R, S, X) returned by Sign_1 in session j. Assume A is an adversary that $(\tau, \varepsilon_{\text{bs}})$-breaks the OMUF of BS[EMBS]. Let dk be the decryption key corresponding to ek. For each $i \in [k+1]^+$, parse σ_i^* into $(\sigma_i'^*, \mathbf{C}_i^*, \mathbf{D}_i^*, M_i^{\dagger *}, \bar{R}_i^*, \bar{S}_i^*, \bar{X}_i^*, \pi_i^*, \bar{z}_{1,i}^*, \bar{z}_{2,i}^*)$. Let $(R_i^*, S_i^*, X_i^*) = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}_i^*)$. There are two cases:

- Case 1: There exists $i^* \in [k+1]^+$ such that for every complete session j (i.e. $\mathsf{sesst}_j = \texttt{closed}$), $(R_{i^*}^*, S_{i^*}^*, X_{i^*}^*) \neq (R_j, S_j, X_j)$.
- Case 2: For every $i \in [k+1]^+$, there exists $j \in [q_s]^+$ such that session j is complete, and $(R_i^*, S_i^*, X_i^*) = (R_j, S_j, X_j)$.

The theorem follows from the two lemmas below, each stating the existence of a reduction working in one case. □

Lemma 3 (Case 1). *Let* $\varepsilon_{\text{bs},1}$ *be the probability that the* τ*-time* A *wins the* OMUF *game, and Case 1 holds. Then there exists a reduction* B *that* $(q_s, \tau', \varepsilon_{\text{embs}})$*-breaks the* UF-xRMA *of* EMBS*, where essentially* $\tau' \approx \tau$*, and* $\varepsilon_{\text{embs}} = \varepsilon_{\text{bs},1}$*.*

Lemma 4 (Case 2). *Let* $\varepsilon_{\text{bs},2}$ *be the probability that the* τ*-time* A *wins the* OMUF *game, and Case 2 holds. Then there exists a reduction* B *that* $(\tau', \varepsilon_{\text{dl}})$*-breaks* DL*, where* $\tau' = 2\tau + O(q)$*,* $q = 2q_h + q_s + 1$*, and*

$$\varepsilon_{\text{dl}} \geq \frac{\varepsilon_{\text{bs},2}^2}{2q} - \frac{O(q)}{p}.$$

4.4 Case 1

Proof of Lemma 3. We construct a reduction B breaking the UF-xRMA security.

Simulate the OMUF game. Given its own signing oracles SIGN'_1, SIGN'_2, we show how B simulates the OMUF game for A.

- Setup: B is given EMBS.pp, pk, ek, dk as inputs. It samples $H := hG$ with known discrete logarithm $h \leftarrow \mathbb{Z}_p$ and sets pp as $(\mathsf{EMBS.pp}, \mathsf{ek}, H)$.
- SIGN_1: Upon the j-th SIGN_1 query, B queries its own oracle SIGN'_1 also for the j-th time, and it receives $(\mathbf{m}_j, \mathsf{smsg}'_{1,j})$, where $\mathbf{m}_j$ is parsed into (r, s, x). It computes (R_j, S_j, X_j), which is $\mathbf{M}_j := (rG, sG, xG)$. It uniformly samples $r_{2,j}, s_{2,j}, x_{2,j} \leftarrow \mathbb{Z}_p$ and computes $r_{1,j} := r - hr_{2,j}$, $s_{1,j} := s - hs_{2,j}$, $x_{1,j} := x - hx_{2,j}$. It returns $(\mathsf{smsg}'_{1,j}, R_j, S_j, X_j)$.
- SIGN_2: On inputs (j, umsg), B parses umsg into $(\mathsf{umsg}', b_j, c_j)$. Then it queries its own oracle $\text{SIGN}_2(j, \mathsf{umsg}')$ and receives smsg'_2. It faithfully runs the remaining of SIGN_2, i.e., it computes $z_{1,j} := r_{1,j} + b_j s_{1,j} + c_j x_{1,j}$ and $z_{2,j} := r_{2,j} + b_j s_{2,j} + c_j x_{2,j}$. It returns $(\mathsf{smsg}'_2, z_{1,j}, z_{2,j})$.

One can verify that B perfectly simulates the OMUF game.

UF-xRMA forgery. We now show how B obtains its own forgery if A wins the OMUF game and Case 1 holds. B computes $(R^*_i, S^*_i, X^*_i) := \mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*_i)$ for every $i \in [k+1]^+$. Since Case 1 holds, there exists an $i^* \in [k+1]^+$ such that $(R^*_{i^*}, S^*_{i^*}, X^*_{i^*}) \neq (R_j, S_j, X_j)$ for every completed session j. This means that $(R^*_{i^*}, S^*_{i^*}, X^*_{i^*})$ is not recorded in the set $\mathcal{Q}$ of the UF-xRMA game. Moreover, the validity of A's forgery guarantees that $\sigma'^*_{i^*}$ is a valid EMBS signature on $\mathbf{C}^*_{i^*}$. Therefore, B wins the UF-xRMA game by returning $(\mathbf{C}^*_{i^*}, \sigma'^*_{i^*})$. □

4.5 Case 2

Algebraic one-more preimage game. In Fig. 8, we define the algebraic one-more preimage game $\mathsf{AOMP}[q]$ for function $\mathsf{F}_H(y_1, y_2) = y_1 G + y_2 H$ over $(\mathbb{G}, p, G)$. The game uniformly samples H and q preimages $\{(y_{1,j}, y_{2,j})\}_{j \in [q]^+}$. Given the q images $\{Y_j\}_{j \in [q]^+}$, which we also call the "challenge elements", the goal of A is to find out the q preimages with the help of $q-1$ calls of a preimage oracle PREIMG. The oracle takes as input a linear combination $\sum_{j \in [q]^+} \beta_j Y_j$ of the challenge elements and returns $\sum_{j \in [q]^+} \beta_j \cdot (y_{1,j}, y_{2,j})$.

The following lemma from [TZ23] reduces AOMP to DL. The proof idea is as follows. C executes $\mathsf{AOMP}[q](\mathsf{B})$ with its input element embedded in H. The key is to show that, conditioned on any possible view in the game, there remain at least two equally possible values of $\{(y_{1,j}, y_{2,j})\}_{j \in [q]^+}$. Hence, with probability at least 1/2, B provides a second preimage $(y^*_{1,j}, y^*_{2,j})$ of some Y_j, which allows C to compute the discrete logarithm of H.

Lemma 5 ([TZ23]). *If there exists an algorithm* B *that* $(\tau, \varepsilon_{\text{aomp}})$*-breaks the* $\mathsf{AOMP}[q]$ *game, then there exists a reduction* C *that* $(\tau', \varepsilon_{\text{dl}})$*-breaks* DL*, where* $\tau' = \tau + O(q)$ *and* $\varepsilon_{\text{dl}} \geq \varepsilon_{\text{aomp}}/2$.

To prove Lemma 4, we will build a reduction that solves the $\mathsf{AOMP}[3q_s]$ game. The proof of Lemma 4 is deferred to Section B. We give the proof idea below.

Recall that in Case 2, the encrypted two-time key $(R^*_i, S^*_i, X^*_i) = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*_i)$ in each forged signature σ^*_i is copied from a complete signing session, i.e., there exists a complete session $j(i)$ (as a function of i) such that $(R^*_i, S^*_i, X^*_i) = (R_{j(i)}, S_{j(i)}, X_{j(i)})$. Let $(b^*_i, c^*_i) = (\bar{b}^*_i/\alpha_b, \bar{c}^*_i/\alpha_c)$, where (α_b, α_c) are the blinding factors committed in $\mathbf{D}^*_i$, and

$$(\bar{b}^*_i, \bar{c}^*_i) = \mathsf{H}(\mathbf{C}^*_i, \mathbf{D}^*_i, M^{\dagger *}_i, \bar{R}^*_i, \bar{S}^*_i, \bar{X}^*_i, \pi^*_i, \mu^*_i).$$

We claim that there exists a "special forgery" $(\mu^*_{i^*}, \sigma^*_{i^*})$ such that $(b^*_{i^*}, c^*_{i^*}) \neq (b_{j(i^*)}, c_{j(i^*)})$, where $(b_{j(i^*)}, c_{j(i^*)})$ is part of the user-message in session $j(i^*)$. To see the claim, note that the winning condition of the OMUF requires the adversary to output more forgeries than the number of complete sessions. Hence, there must exist $i \neq i'$ such that $j(i) = j(i')$. On the other hand, $(b^*_i, c^*_i) \neq (b^*_{i'}, c^*_{i'})$ with overwhelming probability: Since $(b^*_i, c^*_i) = (\bar{b}^*_i/\alpha_b, \bar{c}^*_i/\alpha_c)$, where $(\bar{b}^*_i, \bar{c}^*_i)$ is the output of the random oracle with (α_b, α_c) having been committed in the random oracle input, $(b^*_i, c^*_i) = (b^*_{i'}, c^*_{i'})$

$\underline{\mathsf{AOMP}[q](\mathsf{A})}$
1: $\mathsf{ctr} := 0$
2: $H \leftarrow \mathbb{G}$
3: **for** $j \in [q]^+$ **do**
4: $\quad y_{1,j}, y_{2,j} \leftarrow \mathbb{Z}_p$
5: $\quad Y_j := y_{1,j}G + y_{2,j}H$
6: $\{(y^*_{1,j}, y^*_{2,j})\}_{j\in[q]^+} \leftarrow \mathsf{A}^{\textsc{PreImg}}(H, \{Y_j\}_{j\in[q]^+})$
7: **Return** $[\![\forall j \in [q]^+ : y^*_{1,j}G + y^*_{2,j}H = Y_j]\!]$

$\underline{\textsc{PreImg}(Z, \{\beta_j\}_{j\in[q]^+})}$
1: **Assert** $\mathsf{ctr} \le q - 1$
2: **Assert** $Z = \sum_{j\in[q]^+} \beta_j Y_j$
3: $\mathsf{ctr} := \mathsf{ctr} + 1$
4: **Return** $\sum_{j\in[q]^+} \beta_j \cdot (y_{1,j}, y_{2,j})$

Figure 8: The AOMP game. In our proof, we write $\textsc{PreImg}(Z)$ where Z is expressed as a linear combination of the challenge elements $\{Y_j\}_{j\in[q]^+}$, which implicitly specifies the input $\{\beta_j\}_{j\in[q]^+}$.

would essentially imply a random oracle collision. Consequently, at least one of (μ^*_i, σ^*_i) and $(\mu^*_{i'}, \sigma^*_{i'})$ should be the special forgery.

Omit the subscript i^* and denote the special forgery as (μ^*, σ^*), where $\sigma^* = (\sigma'^*, \mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \bar{z}^*_1, \bar{z}^*_2)$. Also let j^* denote $j(i^*)$. Note that the witness of π^* contains the blinding factors $(\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c)$, which allow unblinding the signature $(\bar{z}^*_1, \bar{z}^*_2)$ under the blinded two-time key $(\bar{R}^*, \bar{S}^*, \bar{X}^*)$ into a signature (z^*_1, z^*_2) under the encrypted two-time key $(R^*, S^*, X^*) = (R_{j^*}, S_{j^*}, X_{j^*})$, satisfying

$$z^*_1 G + z^*_2 G = R_{j^*} + b^* S_{j^*} + c^* X_{j^*},$$

where

$$b^* = \bar{b}^*/\alpha_b, \quad c^* = \bar{c}^*/\alpha_c.$$

The definition of the special forgery guarantees that $(b^*, c^*) \neq (b_{j^*}, c_{j^*})$. That is, (z^*_1, z^*_2) is a new signature under $(R_{j^*}, S_{j^*}, X_{j^*})$, distinct from what has been issued in the signing session.

We build a reduction to $\mathsf{AOMP}[3q_s]$. The reduction uses its input elements as the two-time keys used in the signing sessions and uses its $\textsc{PreImg}$ oracle to issue the signatures. It rewinds the adversary to extract the witness for π^*, unblind the signature, and learn a new signature under $(R_{j^*}, S_{j^*}, X_{j^*})$. Consequently, it learns one more linear equation of the input elements, which allows it to solve for the preimages with fewer than $3q_s$ $\textsc{PreImg}$ queries.

We need to be careful to ensure that the reduction learns a new equation that is linearly independent of those it has learned from the $\textsc{PreImg}$ queries for issuing signatures. In the second (i.e., rewound) execution, the adversary must submit π^{**}, the rewound non-interactive proof that proves the same statement as π^* and allows for extracting the witness, to the random oracle at some point in the security game. Depending on when such a π^{**} appears for the first time, the reduction behaves differently:

- If π^{**} appears *before* the reduction issues a signature under $(R_{j^*}, S_{j^*}, X_{j^*})$ in the second execution, then the reduction immediately unblinds the special forgery in the first execution. It learns the following equation from the unblinded signature:

  $$z^*_1 G + z^*_2 G = R_{j^*} + b^* S_{j^*} + c^* X_{j^*}.$$

  Given that $(b^*, c^*) \neq (b_{j^*}, c_{j^*})$, the above equation is guaranteed to be independent of the equation it has learned when issuing the signature in the first execution:

  $$z_1 G + z_2 G = R_{j^*} + b_{j^*} S_{j^*} + c_{j^*} X_{j^*}.$$

- If π^{**} appears *after* the reduction issues a signature under $(R_{j^*}, S_{j^*}, X_{j^*})$ in the second execution,

the reduction has learned two equations when issuing the signatures:

$$z_1 G + z_2 G = R_{j^*} + b_{j^*} S_{j^*} + c_{j^*} X_{j^*},$$
$$z'_1 G + z'_2 G = R_{j^*} + b'_{j^*} S_{j^*} + c'_{j^*} X_{j^*}.$$

Unblinding the special forgery in the first execution may not give an independent equation: Although $(b^*, c^*) \neq (b_{j^*}, c_{j^*})$, it may be the case that $(1, b^*, c^*) \in \mathsf{Span}\{(1, b_{j^*}, c_{j^*}), (1, b'_{j^*}, c'_{j^*})\}$. Our solution is to instead let the reduction unblind the special forgery in the second execution, which gives an equation

$$z_1^{**} G + z_2^{**} G = R_{j^*} + b^{**} S_{j^*} + c^{**} X_{j^*},$$

where

$$b^{**} = \bar{b}^{**}/\alpha_b, \quad \bar{c}^{**} = \bar{c}^{**}/\alpha_c$$

with $(\bar{b}^{**}, \bar{c}^{**})$ being a random oracle output that is also determined *after* the reduction has learned the above two equations. The randomness of $(\bar{b}^{**}, \bar{c}^{**})$ guarantees that $(1, b^{**}, c^{**}) \in \mathsf{Span}\{(1, b_{j^*}, c_{j^*}), (1, b'_{j^*}, c'_{j^*})\}$ is unlikely.

5 Preliminaries for Instantiation

5.1 Blindable Σ-Protocols

We define blindable Σ-protocols following [KR25].

Definition 11 (Blindable Σ-protocols)**.** A blindable Σ-protocol for relation $\mathcal{R} = \mathcal{X} \times \mathcal{W}$ with challenge set $\mathcal{C}$, randomness set Ω, and blind randomness set $\bar{\Omega}$ consists of the following algorithms:

- $\mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho) \to \mathsf{cm}$: On inputs a statement stm and a witness wit, Com outputs a commitment cm.
- $\mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho) \to \mathsf{rs}$: On inputs a witness wit, a challenge ch, and randomness ρ, Resp outputs a response rs.
- $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) \to 0/1$: On inputs a statement stm, a commitment cm, a challenge ch, and a response rs, Vf outputs 0 or 1.
- $\mathsf{BCom}(\mathsf{stm}, \mathsf{cm}; \bar{\rho}) \to \bar{\mathsf{cm}}$: On inputs a statement stm and a commitment cm, BCom outputs a commitment $\bar{\mathsf{cm}}$.
- $\mathsf{BChal}(\mathsf{ch}, \bar{\rho}) \to \bar{\mathsf{ch}}$: On inputs a challenge ch and blind randomness $\bar{\rho}$, BChal outputs a challenge $\bar{\mathsf{ch}}$.
- $\mathsf{BChal}^{-1}(\bar{\mathsf{ch}}, \bar{\rho}) \to \mathsf{ch}$: On inputs a challenge $\bar{\mathsf{ch}}$ and blind randomness $\bar{\rho}$, BChal^{-1} outputs a challenge ch.
- $\mathsf{BResp}(\mathsf{rs}, \bar{\rho}) \to \bar{\mathsf{rs}}$: On inputs a response rs and blind randomness $\bar{\rho}$, BResp outputs a response $\bar{\mathsf{rs}}$.

We require that $(\mathsf{Com}, \mathsf{Resp}, \mathsf{Vf})$ is a (normal) Σ-protocol for $\mathcal{R}$ with challenge set $\mathcal{C}$ and randomness set Ω, for which we require correctness, SHVZK, and special soundness. Moreover, we require a blindable Σ-protocol to be correct, blindable, and translatable, as defined below.

Definition 12 (Correctness)**.** A blindable Σ-protocol $\mathbf{\Sigma}$ is correct if the following holds:

- (Perfect) correctness: For every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$, $\rho \in \Omega$, and $\mathsf{ch} \in \mathcal{C}$, if we let $\mathsf{cm} := \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)$, $\mathsf{rs} := \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)$, then it must hold that $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) = 1$.
- (Perfect) blind correctness: For every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$, $(\mathsf{cm}, \mathsf{ch}, \mathsf{rs})$ such that $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) = 1$, and $\bar{\rho} \in \bar{\Omega}$, if we let $\bar{\mathsf{cm}} := \mathsf{BCom}(\mathsf{stm}, \mathsf{cm}; \bar{\rho})$, $\bar{\mathsf{ch}} := \mathsf{BChal}(\mathsf{ch}, \bar{\rho})$, and $\bar{\mathsf{rs}} := \mathsf{BResp}(\mathsf{rs}, \bar{\rho})$, then it must hold that $\mathsf{Vf}(\mathsf{stm}, \bar{\mathsf{cm}}, \bar{\mathsf{ch}}, \bar{\mathsf{rs}}) = 1$.
- Bijectivity of BChal: For every $\bar{\rho} \in \bar{\Omega}$, $\mathsf{BChal}(\cdot, \bar{\rho})$ is a permutation over $\mathcal{C}$ with $\mathsf{BChal}^{-1}(\cdot, \bar{\rho})$ as its inverse.

Definition 13 (Blindability)**.** A blindable Σ-protocol $\mathbf{\Sigma}$ is (perfectly) blindable if for every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}$, $(\mathsf{cm}, \mathsf{ch}, \mathsf{rs})$ such that $\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs}) = 1$, the following two procedures have identically distributed outputs:

- Blind the given transcript: Sample $\bar{\rho} \leftarrow \bar{\Omega}$; Compute $\bar{\mathsf{cm}} := \mathsf{BCom}(\mathsf{stm}, \mathsf{cm}; \bar{\rho})$, $\bar{\mathsf{ch}} := \mathsf{BChal}(\mathsf{ch}, \bar{\rho})$, and $\bar{\mathsf{rs}} := \mathsf{BResp}(\mathsf{rs}, \bar{\rho})$; Output $(\bar{\mathsf{cm}}, \bar{\mathsf{ch}}, \bar{\mathsf{rs}})$.
- Sample a new transcript: Sample $\rho \leftarrow \Omega$, $\bar{\mathsf{ch}} \leftarrow \mathcal{C}$; Compute $\bar{\mathsf{cm}} := \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)$ and $\bar{\mathsf{rs}} := \mathsf{Resp}(\mathsf{wit}, \bar{\mathsf{ch}}, \rho)$; Output $(\bar{\mathsf{cm}}, \bar{\mathsf{ch}}, \bar{\mathsf{rs}})$.

Definition 14 (Translatability)**.** Let $\mathbf{\Sigma}$ be a blindable Σ-protocol with respect to $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, $\mathcal{C}$, Ω, $\bar{\Omega}$. Let $\mathcal{V}$ be a set and $\mathsf{TStm} : \mathcal{X} \times \mathcal{V} \to \mathcal{X}$ an efficiently computable function, such that for any $\mathsf{stm} \in \mathcal{X}_{\mathcal{R}}$ and $v \in \mathcal{V}$, $\mathsf{TStm}(\mathsf{stm}, v) \in \mathcal{X}_{\mathcal{R}}$. We say $\mathbf{\Sigma}$ is (perfectly) translatable with respect to $\mathcal{V}$ and TStm, if there exists an efficiently computable function $\mathsf{TResp} : \mathcal{V} \times \mathcal{C} \times \{0,1\}^* \to \{0,1\}^*$ such that for every $(\mathsf{stm}, \mathsf{wit}) \in \mathcal{R}, (\mathsf{stm}', \mathsf{wit}') \in \mathcal{R}, v \in \mathcal{V}$ satisfying $\mathsf{stm}' = \mathsf{TStm}(\mathsf{stm}, v)$, the following two procedures have identically distributed outputs:

- Translate a transcript for $(\mathsf{stm}, \mathsf{wit})$: Sample $\rho \leftarrow \Omega$, $\mathsf{ch} \leftarrow \mathcal{C}$; Compute $\mathsf{cm} := \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)$, $\mathsf{rs} := \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)$, and $\mathsf{rs}' := \mathsf{TResp}(v, \mathsf{ch}, \mathsf{rs})$; Output $(\mathsf{cm}, \mathsf{ch}, \mathsf{rs}')$.
- Sample a transcript for $(\mathsf{stm}', \mathsf{wit}')$: Sample $\rho \leftarrow \Omega$, $\mathsf{ch} \leftarrow \mathcal{C}$; Compute $\mathsf{cm} := \mathsf{Com}(\mathsf{stm}', \mathsf{wit}'; \rho)$ and $\mathsf{rs}' := \mathsf{Resp}(\mathsf{wit}', \mathsf{ch}, \rho)$; Output $(\mathsf{cm}, \mathsf{ch}, \mathsf{rs}')$.

5.2 Blindable Σ-Protocol for OR Composition of Linear Preimage Relations

We define the conjunction of OR-composed linear preimage relations below.

Definition 15 (Conjunction of OR-composed linear preimage relations)**.** Relation $\mathcal{R}_{\mathrm{or},l} \subseteq \mathcal{X}^{2l} \times \{\{0,1\} \times \mathcal{W}\}$ is defined as

$$\left\{ \left(\{(\mathbf{M}_{i,j}, \mathbf{V}_{i,j})\}_{i\in[l],j\in[2]}, \{(b_i, \mathbf{w}_i)\}_{i\in[l]} \right) \middle| \forall i \in [l] : \mathbf{M}_{i,b_i}\mathbf{w}_i = \mathbf{V}_{i,b_i} \right\}.$$

In Fig. 9, we present a blindable Σ-protocol $\mathbf{\Sigma}$ for this relation. The protocol applies the OR composition [CDS94] to the Σ-protocol for $\mathcal{R}_{\mathrm{lin}}$, and proves l such OR statements in parallel. It follows the well-known technique (e.g. in [AO00]) to blind the OR proofs. It is straightforward to verify that $\mathbf{\Sigma}$ is correct, blindable, and translatable. For translatability, we provide the corresponding algorithms TStm and TResp in Fig. 9. Essentially, the translatability allows any additive shift to the statement and witness.

5.3 Dual-Mode Commitment Schemes

We will use the following dual-mode commitment scheme CM over $(\mathbb{G}, p, G)$ with message space $\mathbb{G}^l$. CM (implicitly) takes a common reference string $\mathsf{crs} \in \mathbb{G}^{(l+1)\times(l+1)}$ as an input. Given a message $\mathbf{M} \in \mathbb{G}^l$ and randomness $\vec{\eta} \leftarrow \mathbb{Z}_p^{l+1}$, $\mathsf{CM.Com}(\mathbf{M}; \vec{\eta})$ outputs the commitment as $\mathsf{cm} := \mathsf{crs} \cdot \vec{\eta} + \binom{O}{\mathbf{M}} \in \mathbb{G}^{l+1}$. The scheme is parameterized by the message length l. Different parts of the construction may use different values of l and correspondingly sized crs.

CM has the following properties:

- Hiding mode: If crs is rank-$(l+1)$, then CM is in the hiding mode. In the hiding mode, for any message $\mathbf{M}$, $\mathsf{CM.Com}(\mathbf{M})$ is uniformly distributed over $\mathbb{G}^{l+1}$ (with $\vec{\eta}$ uniformly chosen from $\mathbb{Z}_p^{l+1}$).
- Binding (extractable) mode: If crs is rank-1, then for any $\mathsf{cm} \in \mathbb{G}^{l+1}$, there exists a unique message $\mathbf{M}$ such that cm is a possible commitment to $\mathbf{M}$. Moreover, given $\mathsf{td} = (d_1, \ldots, d_l)$ such that

$$\mathsf{crs} = \begin{pmatrix} \mathbf{A} \\ d_1\mathbf{A} \\ \vdots \\ d_l\mathbf{A} \end{pmatrix}$$

$\underline{\mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho)}$
$\{(\mathbf{M}_{i,j}, \mathbf{V}_{i,j})\}_{i\in[l],j=0,1} := \mathsf{stm}$
$\{(b_i, \mathbf{w}_i)\}_{i\in[l]} := \mathsf{wit}$
$\{(\mathbf{r}_i, \mathsf{ch}_{i,\bar{b}_i}, \mathsf{rs}_{i,\bar{b}_i})\}_{i\in[l]} := \rho$
for $i \in [l]$ **do**
  $\mathsf{cm}_{i,b_i} := \mathbf{M}_{i,b_i}\mathbf{r}_i$
  $\mathsf{cm}_{i,\bar{b}_i} := \mathbf{M}_{i,\bar{b}_i} \cdot \mathsf{rs}_{i,\bar{b}_i} - \mathsf{ch}_{i,\bar{b}_i} \cdot \mathbf{V}_{i,\bar{b}_i}$
Return $\{\mathsf{cm}_{i,j}\}_{i\in[l],j\in[2]}$

$\underline{\mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho)}$
$\{(b_i, \mathbf{w}_i)\}_{i\in[l]} := \mathsf{wit}$
$\{(\mathbf{r}_i, \mathsf{ch}_{i,\bar{b}_i}, \mathsf{rs}_{i,\bar{b}_i})\}_{i\in[l]} := \rho$
for $i \in [l]$ **do**
  $\mathsf{ch}_{i,b_i} := \mathsf{ch} - \mathsf{ch}_{i,\bar{b}_i}$
  $\mathsf{rs}_{i,b_i} := \mathbf{r}_i + \mathsf{ch}_{i,b_i} \cdot \mathbf{w}_i$
Return $\{(\mathsf{ch}_{i,j}, \mathsf{rs}_{i,j})\}_{i\in[l],j\in[2]}$

$\underline{\mathsf{Vf}(\mathsf{stm}, \mathsf{cm}, \mathsf{ch}, \mathsf{rs})}$
$\{(\mathbf{M}_{i,j}, \mathbf{V}_{i,j})\}_{i\in[l],j=0,1} := \mathsf{stm}$
$\{\mathsf{cm}_{i,j}\}_{i\in[l],j\in[2]} := \mathsf{cm}$
$\{\mathsf{rs}_{i,j}\}_{i\in[l],j\in[2]} := \mathsf{rs}$
for $i \in [l]$ **do**
  if $\mathsf{ch}_{i,0} + \mathsf{ch}_{i,1} \neq \mathsf{ch}$ **then**
    Return 0
  for $j \in [2]$ **do**
    if $\mathsf{Vf}((\mathbf{M}_{i,j}, \mathbf{V}_{i,j}), \mathsf{cm}_{i,j}, \mathsf{ch}_{i,j}, \mathsf{rs}_{i,j}) = 0$ **then**
      Return 0
Return 1

$\underline{\mathsf{TStm}(\mathsf{stm}, v)}$
$\{(\mathbf{M}_{i,j}, \mathbf{V}_{i,j})\}_{i\in[l],j\in[2]} := \mathsf{stm}$
$\{\delta_{\mathbf{w},i,j}\}_{i\in[l],j\in[2]} := v$
for $i \in [l], j \in [2]$ **do**
  $\mathbf{V}'_{i,j} := \mathbf{V}_{i,j} + \mathbf{M}_{i,j}\delta_{\mathbf{w},i,j}$
Return $\{(\mathbf{M}_{i,j}, \mathbf{V}'_{i,j})\}_{i\in[l],j\in[2]}$

$\underline{\mathsf{BCom}(\mathsf{stm}, \mathsf{cm}; \bar{\rho})}$
$\{(\mathbf{M}_{i,j}, \mathbf{V}_{i,j})\}_{i\in[l],j=0,1} := \mathsf{stm}$
$\{\mathsf{cm}_{i,j}\}_{i\in[l],j\in[2]} := \mathsf{cm}$
$(\delta_{\mathsf{ch}}, \{\delta_{\mathsf{ch},i,0}\}_{i\in[l]}, \{\delta_{\mathsf{rs},i,j}\}_{i\in[l],j\in[2]}) := \bar{\rho}$
for $i \in [l]$ **do**
  $\delta_{\mathsf{ch},i,1} := \delta_{\mathsf{ch}} - \delta_{\mathsf{ch},i,0}$
  for $j \in [2]$ **do**
    $\bar{\mathsf{cm}}_{i,j} := \mathsf{cm}_{i,j} + \mathbf{M}_{i,j} \cdot \delta_{\mathsf{rs},i,j} - \delta_{\mathsf{ch},i,j} \cdot \mathbf{V}_{i,j}$
Return $\{\bar{\mathsf{cm}}_{i,j}\}_{i\in[l],j\in[2]}$

$\underline{\mathsf{BChal}(\mathsf{ch}, \bar{\rho})}$
$(\delta_{\mathsf{ch}}, \{\delta_{\mathsf{ch},i,0}\}_{i\in[l]}, \{\delta_{\mathsf{rs},i,j}\}_{i\in[l],j\in[2]}) := \bar{\rho}$
$\bar{\mathsf{ch}} := \mathsf{ch} + \delta_{\mathsf{ch}}$

$\underline{\mathsf{BChal}^{-1}(\bar{\mathsf{ch}}, \bar{\rho})}$
$(\delta_{\mathsf{ch}}, \{\delta_{\mathsf{ch},i,0}\}_{i\in[l]}, \{\delta_{\mathsf{rs},i,j}\}_{i\in[l],j\in[2]}) := \bar{\rho}$
$\mathsf{ch} := \bar{\mathsf{ch}} - \delta_{\mathsf{ch}}$

$\underline{\mathsf{BResp}(\mathsf{rs}, \bar{\rho})}$
$\{(\mathsf{ch}_{i,j}, \mathsf{rs}_{i,j})\}_{i\in[l],j\in[2]} := \mathsf{rs}$
$(\delta_{\mathsf{ch}}, \{\delta_{\mathsf{ch},i,0}\}_{i\in[l]}, \{\delta_{\mathsf{rs},i,j}\}_{i\in[l],j\in[2]}) := \bar{\rho}$
for $i \in [l]$ **do**
  $\delta_{\mathsf{ch},i,1} := \delta_{\mathsf{ch}} - \delta_{\mathsf{ch},i,0}$
  for $j \in [2]$ **do**
    $\bar{\mathsf{ch}}_{i,j} := \mathsf{ch}_{i,j} + \delta_{\mathsf{ch},i,j}$
    $\bar{\mathsf{rs}}_{i,j} := \mathsf{rs}_{i,j} + \delta_{\mathsf{rs},i,j}$
Return $\{(\bar{\mathsf{ch}}_{i,j}, \bar{\mathsf{rs}}_{i,j})\}_{i\in[l],j\in[2]}$

$\underline{\mathsf{TResp}(v, \mathsf{ch}, \mathsf{rs})}$
$\{\delta_{\mathbf{w},i,j}\}_{i\in[l],j\in[2]} := v$
$\{(\mathsf{ch}_{i,j}, \mathsf{rs}_{i,j})\}_{i\in[l],j\in[2]} := \mathsf{rs}$
for $i \in [l], j \in [2]$ **do**
  $\mathsf{rs}'_{i,j} := \mathsf{rs}_{i,j} + \mathsf{ch}_{i,j}\delta_{\mathbf{w},i,j}$
Return $\{(\mathsf{ch}_{i,j}, \mathsf{rs}'_{i,j})\}_{i\in[l],j\in[2]}$

Figure 9: Blindable Σ-protocol for $\mathcal{R}_{\mathrm{or},l}$. For any $b \in \{0,1\}$, we use $\bar{b}$ to denote $1-b$.

for a row vector $\mathbf{A} \in \mathbb{G}^{l+1}$, the committed message can be efficiently extracted from cm.

- Distributions of crs: A uniformly random rank-$(l+1)$ crs (hiding mode) is of statistical distance $O(1/p)$ from the uniform distribution over $\mathbb{G}^{(l+1)\times(l+1)}$.
  If DDH is $(\tau', \varepsilon_{\mathrm{ddh}})$-hard, then a uniformly random rank-1 crs (binding mode) is $(\tau, l \cdot \varepsilon_{\mathrm{ddh}})$-computationally indistinguishable from the uniform distribution over $\mathbb{G}^{(l+1)\times(l+1)}$.

6 Instantiation of Encrypted-Message Blind Signatures

6.1 Construction

Let PKE be the public-key encryption scheme specified in Section 2. Let RE also be that scheme, but used as a rerandomizable encryption scheme. Let CM be the dual-mode commitment scheme

specified in Section 5.3. Let Σ be the blindable Σ-protocol for $\mathcal{R}_{\mathrm{or},l}$ specified in Fig. 9. Let (NIP, NIV) be the non-interactive proof system for $\mathcal{R}_{\mathrm{lin}}$ transformed from the protocol Σ' specified in Fig. 1. We present the pseudocode description of our instantiation of an EMBS scheme associated with PKE in Fig. 10. Our construction closely follows [BHKR25] but is simpler. As we do not insist on tight security, we replace a commitment step in their construction with a guessing argument in the proof of UF-DMA.

Setup and key generation. The setup algorithm Setup generates random elements $H, U, V \leftarrow \mathbb{G}$, three encryption keys $\mathsf{ek}_{\mathsf{RE},0}, \mathsf{ek}_{\mathsf{RE},1}, \mathsf{ek}_{\mathsf{RE},2} \leftarrow \{G\} \times \mathbb{G}^3$, and common reference strings $\mathsf{crs} \leftarrow \mathbb{G}^{4\times 4}$ and $\mathsf{crs}' \leftarrow \mathbb{G}^{2\times 2}$ for CM. Here crs is used to commit messages in $\mathbb{G}^3$, and crs' is used to commit messages in $\mathbb{G}$.

The key generation algorithm KGen samples $\mathbf{X} \leftarrow \mathbb{G}^3$ and sets $w := 0$, $y := 0$. It generates commitments $\mathsf{cmx} \leftarrow \mathsf{CM.Com}(\mathbf{X}; \vec{\eta}_x)$ under crs, $\mathsf{cmw} \leftarrow \mathsf{CM.Com}(wG; \vec{\eta}_w)$ under crs', and $\mathsf{cmy} \leftarrow \mathsf{CM.Com}(yG; \vec{\eta}_y)$ under crs'. It returns $(\mathsf{cmx}, \mathsf{cmw}, \mathsf{cmy})$ as the public key and $(\mathbf{x}, y, \vec{\eta}_x, \vec{\eta}_w, \vec{\eta}_y)$ as the secret key.

Signing protocol. In the first move, the user sends message $\mathbf{M}$ to the signer in the clear.

For the second move, the signer computes an encryption $\mathbf{C}_0 := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; 0)$ of $\mathbf{M}$ using randomness 0. It lets $\mathbf{z}_0 := \mathbf{z}_1 := \mathbf{x}$ and $z_2 := 0$, and it computes $\mathbf{Z}_0 := \mathbf{z}_0 G$, $\mathbf{Z}_1 := \mathbf{z}_1 G$, $Z_2 := z_2 G$. They are encrypted into $\mathsf{ct}_0 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},0}, \mathbf{Z}_0; \zeta_0)$, $\mathsf{ct}_1 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},1}, \mathbf{Z}_1; \zeta_1)$, $\mathsf{ct}_2 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},2}, Z_2; \zeta_2)$ under $\mathsf{ek}_{\mathsf{RE},0}, \mathsf{ek}_{\mathsf{RE},1}, \mathsf{ek}_{\mathsf{RE},2}$, respectively.

The signer initiates Σ for the following statements to generate cm_Σ. Below, consider encryption keys as column vectors.

- $\mathcal{S}_{\mathsf{ct}_0}$ OR $\mathcal{S}_{\mathrm{dh}}$:
  - $\mathcal{S}_{\mathsf{ct}_0}$ states that ct_0 encrypts $\mathbf{X} + w\mathbf{M}$, where $\mathbf{X}$ is the message committed in cmx, and wG is the message committed in cmw. Formally, it states the existence of $\vec{\eta}_x$, ζ_0, $\vec{\eta}_w$, w such that
  $$\mathsf{ek}_{\mathsf{RE},0} \cdot \zeta_0 - \mathsf{crs} \cdot \vec{\eta}_x = \mathsf{ct}_0 - \mathsf{cmx} - w\mathbf{C}_0$$
  and
  $$\mathsf{crs}' \cdot \vec{\eta}_w = \mathsf{cmw} - (O, wG).$$
  - $\mathcal{S}_{\mathrm{dh}}$ states that (G, H, U, V) is a DH-tuple. Formally, it states the existence of u such that $(U, V) = u(G, H)$.
- $\mathcal{S}_{\mathsf{ct}_2}$ OR $\mathcal{S}_{=}$:
  - $\mathcal{S}_{\mathsf{ct}_2}$ states that ct_2 encrypts the message committed in cmy. Formally, it states the existence of $\vec{\eta}_y$, ζ_2 such that
  $$\mathsf{ek}_{\mathsf{RE},2} \cdot \zeta_2 - \mathsf{crs}' \cdot \vec{\eta}_y = \mathsf{ct}_2 - \mathsf{cmy}.$$
  - $\mathcal{S}_{=}$ states that ct_0 and ct_1 encrypt the same message. Formally, it states the existence of ζ_0, ζ_1 such that
  $$\mathsf{ek}_{\mathsf{RE},0} \cdot \zeta_0 - \mathsf{ek}_{\mathsf{RE},1} \cdot \zeta_1 = \mathsf{ct}_0 - \mathsf{ct}_1.$$

The signer uses the witness for $\mathcal{S}_{\mathsf{ct}_0}$ and $\mathcal{S}_{=}$ to perform Σ. We assume helper algorithms FormatS and FormatW that compute and format the statement and witness into the form of the $\mathcal{R}_{\mathrm{or},l}$ relation. We omit their specific descriptions. The signer sends $\{\mathsf{ct}_i\}_{i\in[3]}$ and cm_Σ.

For the third move, the user samples $\rho \leftarrow \mathbb{Z}_p$ and generates $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho)$. It rerandomizes $\{\mathsf{ct}_i\}_{i\in[3]}$ to $\{\mathsf{ct}'_i\}_{i\in[3]}$ using rerandomization factors $r_0, r_1, r_2 \leftarrow \mathbb{Z}_p$. The user translates the statement stm set up by the signer to the statement stm', where $\mathbf{C}_0$ and $\{\mathsf{ct}_i\}_{i\in[3]}$ are replaced with $\mathbf{C}$ and $\{\mathsf{ct}'_i\}_{i\in[3]}$. We assume a helper algorithm FormatV that generates the value v satisfying the syntax of TStm and TResp from ρ, r_0, r_1, r_2.

$\underline{\mathsf{Setup}()}$
1: $H, U, V \leftarrow \mathbb{G}$
2: $\mathsf{ek}_{\mathsf{RE},0}, \mathsf{ek}_{\mathsf{RE},1}, \mathsf{ek}_{\mathsf{RE},2} \leftarrow \{G\} \times \mathbb{G}^3$
3: $\mathsf{crs} \leftarrow \mathbb{G}^{4\times 4}$, $\mathsf{crs}' \leftarrow \mathbb{G}^{2\times 2}$
4: **Return** $(H, U, V, \mathsf{ek}_{\mathsf{RE},0}, \mathsf{ek}_{\mathsf{RE},1}, \mathsf{ek}_{\mathsf{RE},2}, \mathsf{crs}, \mathsf{crs}')$

$\underline{\mathsf{KGen}()}$
1: $\mathbf{X} \leftarrow \mathbb{G}^3$, $w := 0$, $y := 0$
2: $\mathsf{cmx} \leftarrow \mathsf{CM.Com}(\mathbf{X}; \vec{\eta}_x)$
3: $\mathsf{cmw} \leftarrow \mathsf{CM.Com}(wG; \vec{\eta}_w)$
4: $\mathsf{cmy} \leftarrow \mathsf{CM.Com}(yG; \vec{\eta}_y)$
5: **Return** $(\mathsf{cmx}, \mathsf{cmw}, \mathsf{cmy}), (\mathbf{x}, y, \vec{\eta}_x, \vec{\eta}_w, \vec{\eta}_y)$

$\underline{\mathsf{Sign}_1(\mathbf{M}, \mathsf{ek}, \mathsf{sk})}$
1: $(\mathbf{X}, y, \vec{\eta}_x, \vec{\eta}_w, \vec{\eta}_y) := \mathsf{sk}$
2: $\mathbf{C}_0 := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; 0)$
3: $\mathbf{Z}_0 := \mathbf{Z}_1 := \mathbf{X}$
4: $z_2 := 0$, $Z_2 := z_2 G$
5: $\mathsf{ct}_0 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},0}, \mathbf{Z}_0; \zeta_0)$
6: $\mathsf{ct}_1 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},1}, \mathbf{Z}_1; \zeta_1)$
7: $\mathsf{ct}_2 \leftarrow \mathsf{RE.Enc}(\mathsf{ek}_{\mathsf{RE},2}, Z_2; \zeta_2)$
8: $\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}_0, \{\mathsf{ct}_i\}_{i\in[3]})$
9: $\mathsf{wit} := \mathsf{FormatW}(\mathsf{sk}, \{\zeta_i\}_{i\in[3]})$
10: $\mathsf{cm}_\Sigma \leftarrow \mathsf{Com}(\mathsf{stm}, \mathsf{wit}; \rho_\Sigma)$
11: $\mathsf{sst} := (\mathsf{wit}, \rho_\Sigma)$
12: **Return** $(\mathsf{sst}, \{\mathsf{ct}_i\}_{i\in[3]}, \mathsf{cm}_\Sigma)$

$\underline{\mathsf{Sign}_2(\mathbf{M}, \mathsf{ek}, \mathsf{sk}, \mathsf{sst}, \mathsf{umsg})}$
1: $(\mathsf{wit}, \rho_\Sigma) := \mathsf{sst}$
2: $\mathsf{ch} := \mathsf{umsg}$
3: $\mathsf{rs} := \mathsf{Resp}(\mathsf{wit}, \mathsf{ch}, \rho_\Sigma)$
4: $\mathsf{stm}^\dagger := \mathsf{FormatS}^\dagger(\mathsf{pk}, \{\mathsf{ct}_i\}_{i\in[3]})$
5: $\mathsf{wit}^\dagger := \mathsf{FormatW}^\dagger(\mathsf{sk}, \{\zeta_i\}_{i\in[3]})$
6: $\pi \leftarrow \mathsf{NIP}^{\mathsf{H}_\pi(\mathsf{pk}, \{\mathsf{ct}_i\}_{i\in[3]}, \cdot)}(\mathsf{stm}^\dagger, \mathsf{wit}^\dagger)$
7: **Return** (rs, π)

$\underline{\mathsf{User}_1(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1)}$
1: $(\{\mathsf{ct}_i\}_{i\in[3]}, \mathsf{cm}_\Sigma) := \mathsf{smsg}_1$
2: $\rho \leftarrow \mathbb{Z}_p$
3: $\mathbf{C}_0 := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; 0)$
4: $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, \mathbf{M}; \rho)$
5: $\mathsf{stm} := \mathsf{FormatS}(\mathsf{pk}, \mathbf{C}_0, \{\mathsf{ct}_i\}_{i\in[3]})$
6: **for** $i \in [3]$ **do**
7: $\quad r_i \leftarrow \mathbb{Z}_p$
8: $\quad \mathsf{ct}'_i := \mathsf{RE.Rerand}(\mathsf{ek}_{\mathsf{RE},i}, \mathsf{ct}_i; r_i)$
9: $v := \mathsf{FormatV}(\rho, \{r_i\}_{i\in[3]})$
10: $\mathsf{stm}' := \Sigma.\mathsf{TStm}(\mathsf{stm}, v)$
11: $\bar{\mathsf{cm}}_\Sigma \leftarrow \mathsf{BCom}(\mathsf{stm}, \mathsf{cm}_\Sigma; \bar{\rho})$
12: $\bar{\mathsf{ch}} := \mathsf{H}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma)$
13: $\mathsf{ch} := \mathsf{BChal}^{-1}(\bar{\mathsf{ch}}, \bar{\rho}_\Sigma)$
14: $\mathsf{ust} := (v, \bar{\rho}_\Sigma, \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{ch}})$
15: **Return** $(\mathsf{ust}, \mathsf{ch}, \rho)$

$\underline{\mathsf{User}_2(\mathbf{M}, \mathsf{ek}, \mathsf{pk}, \mathsf{ust}, \mathsf{smsg}_2)}$
1: $(v, \bar{\rho}_\Sigma, \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{ch}}) := \mathsf{ust}$
2: $(\mathsf{rs}, \pi) := \mathsf{smsg}_2$
3: $\mathsf{stm}^\dagger := \mathsf{FormatS}^\dagger(\mathsf{pk}, \{\mathsf{ct}_i\}_{i\in[3]})$
4: **if** $\mathsf{NIV}^{\mathsf{H}_\pi(\mathsf{pk}, \{\mathsf{ct}_i\}_{i\in[3]}, \cdot)}(\mathsf{stm}^\dagger, \pi) = 0$ **then**
5: $\quad$ **Return** $\perp$
6: $\bar{\mathsf{rs}} := \mathsf{BResp}(\mathsf{rs}, \bar{\rho}_\Sigma)$
7: $\bar{\mathsf{rs}}' := \mathsf{TResp}(v, \bar{\mathsf{ch}}, \bar{\mathsf{rs}})$
8: **Return** $(\{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{rs}}')$

$\underline{\mathsf{Vf}(\mathsf{pk}, \mathbf{C}, \sigma)}$
1: $(\{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{ch}}, \bar{\mathsf{rs}}') := \sigma$
2: $\mathsf{stm}' := \mathsf{FormatS}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]})$
3: $\bar{\mathsf{ch}} := \mathsf{H}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma)$
4: **Return** $\mathsf{Vf}(\mathsf{stm}', \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{ch}}, \bar{\mathsf{rs}}')$

Figure 10: Our instantiation of four-move encrypted-message blind signature scheme EMBS.

Under the original statement stm, the user blinds cm_Σ into $\bar{\mathsf{cm}}_\Sigma$ using BCom with randomness $\bar{\rho}$. It computes $\bar{\mathsf{ch}} := \mathsf{H}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma)$. Then the user unblinds the challenge using BChal^{-1} to get ch. It sends ch and also fixes the encryption randomness ρ.

For the fourth move, the signer responds to ch with rs. Then it generates a non-interactive proof π using NIP which proves that ct_0, ct_1, ct_2 encrypt the message committed in cmx, the message committed in cmx, and O, respectively. Formally, it proves the existence of $\vec{\eta}_x$, ζ_0, ζ_1, ζ_2 such that

$$\begin{aligned} \mathsf{ek}_{\mathsf{RE},0} \cdot \zeta_0 - \mathsf{crs} \cdot \vec{\eta}_x &= \mathsf{ct}_0 - \mathsf{cmx}, \\ \mathsf{ek}_{\mathsf{RE},1} \cdot \zeta_1 - \mathsf{crs} \cdot \vec{\eta}_x &= \mathsf{ct}_1 - \mathsf{cmx}, \\ \mathsf{ek}_{\mathsf{RE},2} \cdot \zeta_2 &= \mathsf{ct}_2. \end{aligned}$$

We assume helper algorithms $\mathsf{FormatS}^\dagger$ and $\mathsf{FormatW}^\dagger$ for this statement. The signer sends (rs, π).

Finally, the user verifies the non-interactive proof. If it verifies, then the user blinds rs into $\bar{\mathsf{rs}}$ under stm, and then translates into $\bar{\mathsf{rs}}'$ under stm'. The signature consists of $(\{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{rs}}')$.

Verification. To verify the signature, one computes $\bar{\mathsf{ch}} := \mathsf{H}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_\Sigma)$ and runs $\Sigma.\mathsf{Vf}(\mathsf{stm}', \bar{\mathsf{cm}}_\Sigma, \bar{\mathsf{ch}}, \bar{\mathsf{rs}}')$. The correctness follows from the properties of Σ.

Optimization and Efficiency.

It is well-known that for $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{or},l}$, we can let the non-interactive proof and the EMBS signature contain $(\mathsf{ch}, \mathsf{rs})$ instead of $(\mathsf{cm}, \mathsf{rs})$. Also, we can let cmy, $\vec{\eta}_y$, and ct_2 consist of two elements. With this optimization, the signature size and communication of EMBS are $10\mathbb{G} + 17\mathbb{Z}_p$ and $24\mathbb{G} + 25\mathbb{Z}_p$, respectively.

Also note that, for both our EMBS and BS[EMBS], the public parameters are uniformly distributed, so they can be generated by a hash function without any trusted setup.

6.2 Encrypted-Message Blindness

Theorem 6 (Encrypted-message blindness)**.** *If* DDH *is* $(\tau', \varepsilon_{\text{ddh}})$*-hard over* $(\mathbb{G}, p, G)$*, then* EMBS *is* $(\tau, \varepsilon_{\text{embs}})$*-encrypted-message blind, where essentially* $\tau' \approx \tau$*, and*

$$\varepsilon_{\text{embs}} \leq 5\varepsilon_{\text{ddh}} + \frac{O(q_h)}{p}.$$

We defer the proof to Section C.

6.3 UF-DMA

Theorem 7 (UF-DMA)**.** *If* DDH *is* $(\tau', \varepsilon_{\text{ddh}})$*-hard over* $(\mathbb{G}, p, G)$ *then* EMBS *is* $(\tau, \varepsilon_{\text{embs}})$-UF-DMA*, where essentially* $\tau' \approx \tau$*,* $q = O(q_s + q_h)$*, and*

$$\varepsilon_{\text{embs}} \leq O(q_s \log q_s)(\varepsilon_{\text{ddh}} + \frac{q}{p}) + \frac{O(q^2)}{p}.$$

We defer the proof to Section D. The proof closely follows [BHKR25]. We sketch the main idea here. For simplicity, assume that A always returns $\mathbf{C}^*$ that encrypts $\mathbf{M}^* \notin \{\mathbf{M}_j\}_{j\in[q_s]^+}$. We introduce an additional forgery check: we require $\mathsf{pt}_0^* \in \mathcal{P}$, where $\mathsf{pt}_0^* = \mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},0}, \mathsf{ct}_0^*)$, and $\mathcal{P}$ is defined as $\{\mathbf{X}\}_{j\in[q_s]^+} = \{\mathbf{x}G\}$. This does not decrease the winning probability much.

We also consider a "dual game". In the original game, we have $w := 0$. In the dual game, we set $w := 1$. The signing oracle generates ct_0, ct_1 as encryptions of $\mathbf{X} + \mathbf{M}$ instead of $\mathbf{X}$. The definition of $\mathcal{P}$ is modified to $\{\mathbf{X} + \mathbf{M}_j\}_{j\in[q_s]^+}$.

We note that in the dual game, forgery is unconditionally hard. Since $\mathcal{S}_{\text{dh}}$ is false, A has to make $\mathcal{S}_{\mathsf{ct}_0}$ true, i.e., let $\mathsf{pt}_0^* = \mathbf{X} + \mathbf{M}^*$. The forgery check $\mathsf{pt}_0^* \in \mathcal{P}$ requires $\mathbf{M}^* \in \{\mathbf{M}_j\}_{j\in[q_s]^+}$, which contradicts the assumption $\mathbf{M}^* \notin \{\mathbf{M}_j\}_{j\in[q_s]^+}$.

The heaviest part of the proof is to show that the winning probabilities of the original game and the dual game are close. The key step is to change what ct_0 and ct_1 encrypt in each session. The proof follows the adaptive partitioning technique [Hof17, AHN+17, BHKR25], which performs a partitioning loop. In each step, we break the statement $\mathcal{S}_=$ for half of the sessions while still requiring it on the forgery. The double key structure (encrypting the same message under $\mathsf{ek}_{\mathsf{RE},0}$ and $\mathsf{ek}_{\mathsf{RE},1}$) allows us to change the plaintext pt_0, pt_1 one by one while still being able to perform the forgery check about $\mathcal{P}$. After sufficiently many steps, the plaintext used in each session becomes independently random. The same game hops also apply to the dual game, so we can relate their winning probabilities.

Acknowledgements

We thank the Crypto 2026 reviewers for their helpful comments and suggestions.

References

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Berlin, Heidelberg, May 2001. 1, 2

[ACD+16] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *Journal of Cryptology*, 29(4):833–878, October 2016. 9

[AHN+17] Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 548–580. Springer, Cham, August 2017. 21

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Berlin, Heidelberg, August 2000. 17

[BHKR25] Nicholas Brandt, Dennis Hofheinz, Michael Klooß, and Michael Reichle. Tightly-secure blind signatures in pairing-free groups. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part VI*, volume 16250 of *LNCS*, pages 337–369. Springer, Singapore, December 2025. 2, 3, 19, 21

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Cham, October 2021. 1

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. 29, 30

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 1

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Berlin, Heidelberg, August 1994. 1

[BS07] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, Berlin, Heidelberg, April 2007. 3

[CART26] Rutchathon Chairattana-Apirom, Michael Reichle, and Stefano Tessaro. Playing tag with okamoto-schnorr: Three-move pairing-free blind signatures from DDH. Cryptology ePrint Archive, Paper 2026/571, 2026. 3

[CATZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Pairing-free blind signatures from CDH assumptions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 174–209. Springer, Cham, August 2024. 2

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994. 17

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, New York, August 1990. 1

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. 1

[Cha88] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 177–182. Springer, Berlin, Heidelberg, May 1988. 1

[CKM+23] Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Snowblind: A threshold blind signature in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 710–742. Springer, Cham, August 2023. 1, 2

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001. 1

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993. 1

[DKT26] Marian Dietz, Julia Kastner, and Stefano Tessaro. On the impossibility of round-optimal pairing-free blind signatures in the ROM. Cryptology ePrint Archive, Paper 2026/090, 2026. 2

[EGM96] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, March 1996. 3, 4

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Berlin, Heidelberg, August 2006. 2

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. 1

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 244–251. Springer, 1992. 1

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020. 1, 2

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. 5

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Cham, May 2019. 1

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 489–518. Springer, Cham, April / May 2017. 21

[KLR25] Michael Klooß, Russell W. F. Lai, and Michael Reichle. Blind signatures from arguments of inequality. Cryptology ePrint Archive, Report 2025/1886, 2025. 2

[KLX22] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Cham, March 2022. 1, 2

[KR25] Michael Klooß and Michael Reichle. Blind signatures from proofs of inequality. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 157–189. Springer, Cham, August 2025. 2, 16

[KRW24] Michael Klooß, Michael Reichle, and Benedikt Wagner. Practical blind signatures in pairing-free groups. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 363–395. Springer, Singapore, December 2024. 2

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. 1

[OA03] Miyako Ohkubo and Masayuki Abe. Security of three-move blind signature schemes reconsidered. The 2003 Symposium on Cryptography and Information Security (SCIS), 2003. 1

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993. 3

[Oka94] Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 61–74. Springer, Berlin, Heidelberg, August 1994. 1

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Berlin, Heidelberg, August 1992. 1

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 1

[RR25] Michael Reichle and Zoé Reinke. Threshold blind signatures from CDH. Cryptology ePrint Archive, Report 2025/1798, 2025. 2

[RR26] Michael Reichle and Zoé Reinke. Threshold blind signatures from CDH. In Shi Bai and Edoardo Persichetti, editors, *Public-Key Cryptography - PKC 2026 - 29th IACR International Conference on Practice and Theory of Public-Key Cryptography, West Palm Beach, FL, USA, May 25-28, 2026, Proceedings, Part I*, Lecture Notes in Computer Science, pages 269–301. Springer, 2026. 2

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. 1

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Cham, May / June 2022. 1, 2

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023. 14

A Proof of Blindness

Proof. Assume A is an algorithm that $(\tau, \varepsilon_{\mathrm{bs}})$-breaks the blindness of $\mathsf{BS}[\mathsf{EMBS}]$. We begin by a series of game hops. Then we give a reduction B that breaks the encrypted-message blindness of EMBS. Let G_0 be the original $\mathsf{BLIND}_{\mathsf{BS}}[\mathsf{EMBS}](\mathsf{A})$ game. Random oracles H and H_π are implemented by lazy sampling. We use tables $\mathsf{H}[\cdot]$ and $\mathsf{H}_\pi[\cdot]$ to store the previous queries and the corresponding answers. We give pseudocode descriptions of the game hops in Figs. 11 to 13. Verbal explanations are as follows. We let ε_i denote the probability that A wins the game G_i (i.e., G_i returns 1). Consequently, we have $\varepsilon_{\mathrm{bs}} = \varepsilon_0$.

G_1: Simulate non-interactive proofs. In this game, $\textsc{User}_1$ simulates the non-interactive proof by $\pi \leftarrow \mathsf{NISim}^{\mathsf{H}_\pi(\mathbf{C},\mathbf{D},M^\dagger,\bar{R},\bar{S},\bar{X},\cdot)}(\mathsf{stm})$ instead of honestly generating it. Specifically, $\mathsf{NISim}^{\mathsf{H}_\pi(\mathbf{C},\mathbf{D},M^\dagger,\bar{R},\bar{S},\bar{X},\cdot)}$ uniformly samples $\mathsf{ch} \leftarrow \mathbb{Z}_p$ and $(\mathsf{cm}, \mathsf{rs}) \leftarrow \mathsf{Sim}(\mathsf{stm}, \mathsf{ch})$ where Sim is the SHVZK simulator of Σ. Then it programs the random oracle $\mathsf{H}_\pi[\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \mathsf{cm}] := \mathsf{ch}$ and returns the proof $\pi := (\mathsf{cm}, \mathsf{rs})$.

By the perfect SHVZK, the view of A in this game differs from that in the preceding game only if NISim overwrites the answer to an existing query. Since $\mathsf{stm} = (\mathbf{M}, \mathbf{V})$ has a non-trivial $\mathbf{M} \neq O$, the probability of any possible value of cm is bounded by $1/p$. The cm in the simulated proof hits an existing H_π query with probability at most $(q_h + 1)/p$. Since $\textsc{User}_1$ is called at most twice in total, we have $\varepsilon_1 \geq \varepsilon_0 - 2(q_h + 1)/p$.

G_2: Encrypt unrelated messages in $\mathbf{D}$. In this game, $\textsc{User}_1$ generates $\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (G, G); \rho_D)$ as an encryption of (G, G) instead of $(\alpha_b G, \alpha_c G)$. In the preceding game, since wit is not used to generate π anymore, ρ_D is only used to generate $\mathbf{D}$. Thus, we can reduce the distinguishing of this game and the preceding game to the CPA security of PKE. Consequently, the difference of winning probabilities $\varepsilon_1 - \varepsilon_2$ is bounded by the CPA security bound, so $\varepsilon_2 \geq \varepsilon_1 - 4\varepsilon_{\mathrm{ddh}}$.

G_3: Program fresh H values in $\textsc{User}_1$. In this game, instead of making the query $\mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$, $\textsc{User}_1$ samples $(\bar{b}, \bar{c}) \leftarrow \mathbb{Z}_p^{*2}$ and programs $\mathsf{H}[\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu] := (\bar{b}, \bar{c})$. The game differs from the preceding game only if this programming overwrites the answer to an existing

$\underline{\mathsf{BLIND}_{\mathsf{BS}}[\mathsf{EMBS}](\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2))}$

$\beta_0 \leftarrow \{0,1\}$; $\beta_1 := 1 - \beta_0$
$\mathsf{sesst}_0, \mathsf{sesst}_1 := \mathtt{init}$
$\mathsf{EMBS.pp} \leftarrow \mathsf{EMBS.Setup}()$
$\mathsf{ek} \leftarrow \{G\} \times \mathbb{G}^3, H \leftarrow \mathbb{G}$
$(\mathsf{st}, \mathsf{pk}, \mu_0, \mu_1) \leftarrow \mathsf{A}_1(\mathsf{pp})$
$\beta^* \leftarrow \mathsf{A}_2^{\textsc{User}_1, \textsc{User}_2, \mathsf{H}, \mathsf{H}_\pi}(\mathsf{st})$
Return $[\![\beta^* = \beta_0]\!]$

$\underline{\textsc{User}_1(j, \mathsf{smsg}_1)}$ ▷ G_0–G_4
Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \mathtt{init}$
$(\mathsf{smsg}_1', R, S, X) := \mathsf{smsg}_1$
$(\mathsf{ust}', \mathsf{umsg}', \rho_C) \leftarrow \mathsf{EMBS.User}_1((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1')$
$\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x \leftarrow \mathbb{Z}_p$
$\alpha_b, \alpha_c, \gamma \leftarrow \mathbb{Z}_p^*$
$\bar{S} := \alpha_b^{-1}(S + \delta_s G)$
$\bar{X} := \alpha_c^{-1}(X + \delta_x G)$
$\bar{R} := R + \delta_{z_1} G + \delta_{z_2} H$
$\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$
$\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (\alpha_b G, \alpha_c G); \rho_D)$ ▷ G_0–G_1
$\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (G, G); \rho_D)$ ▷ G_2–G_4
$M^\dagger := \gamma G$
$\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$
$\mathsf{wit} := \mathsf{FormatW}(\rho_C, \rho_D, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c, \gamma)$ ▷ G_0
$\pi \leftarrow \mathsf{NIP}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm}, \mathsf{wit})$ ▷ G_0
$\pi \leftarrow \mathsf{NISim}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm})$ ▷ G_1–G_4
$(\bar{b}, \bar{c}) := \mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$ ▷ G_0–G_2
$(\bar{b}, \bar{c}) \leftarrow \mathbb{Z}_p^{*2}$ ▷ G_3
$(b, c) \leftarrow \mathbb{Z}_p^{*2}$ ▷ G_4
$\bar{b} := \alpha_b b, \quad \bar{c} := \alpha_c c$ ▷ G_4
$\mathsf{H}[\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu] := (\bar{b}, \bar{c})$ ▷ G_3–G_4
$b := \bar{b}/\alpha_b$ ▷ G_0–G_3
$c := \bar{c}/\alpha_c$ ▷ G_0–G_3
$\mathsf{sesst}_j := \mathtt{open}$
$\mathsf{ust}_j := (\mathsf{ust}', R, S, X, b, c, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi)$
Return (umsg', b, c)

$\underline{\textsc{User}_1(j, \mathsf{smsg}_1)}$ ▷ G_5–G_8
Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \mathtt{init}$
$(\mathsf{smsg}_1', R, S, X) := \mathsf{smsg}_1$
$(\mathsf{ust}', \mathsf{umsg}', \rho_C) \leftarrow \mathsf{EMBS.User}_1((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{smsg}_1')$
$(b, c) \leftarrow \mathbb{Z}_p^{*2}$
$\mathsf{sesst}_j := \mathtt{open}$
$\mathsf{ust}_j := (\mathsf{ust}', R, S, X, \rho_C, b, c)$
Return (umsg', b, c)

Figure 11: Setup and $\textsc{User}_1$ in G_0–G_8 for the proof of Theorem 1.

query. Observe that $\bar{R}$ is uniformly random over $\mathbb{G}$. Consequently, the probability of any possible input of H in $\textsc{User}_1$ is bounded by $1/p$. It hits an existing query with probability at most $(q_h+1)/p$.

Since there are two sessions in total, we have $\varepsilon_3 \geq \varepsilon_2 - 2(q_h+1)/p$.

G_4: Uniformly sample (b,c) in $\mathbf{User}_1$. In the preceding game, since $(\bar{b},\bar{c})$ is freshly sampled from $\mathbb{Z}_p^{*2}$, $(b,c) := (\bar{b}/\alpha_b, \bar{c}/\alpha_c)$ is also uniformly distributed over $\mathbb{Z}_p^{*2}$. Therefore, we can let this game sample $(b,c) \leftarrow \mathbb{Z}_p^{*2}$ and then compute $(\bar{b},\bar{c}) := (\alpha_b b, \alpha_c c)$ instead. This does not change the view of A. We have $\varepsilon_4 = \varepsilon_3$.

G_5: Defer most operations to $\mathbf{User}_2$. Note that the output of $\textsc{User}_1$ only consists of the umsg' returned by $\mathsf{EMBS.User}_1((R,S,X),\mathsf{ek},\mathsf{pk},\mathsf{smsg}_1')$ and $(b,c) \leftarrow \mathbb{Z}_p^{*2}$. In this game, all the other computations in $\textsc{User}_1$ are deferred to $\textsc{User}_2$ when both sessions are completed. These include generation of the encryptions $\mathbf{C}$, $\mathbf{D}$, sampling of $\delta_{z_1},\delta_{z_2},\delta_s,\delta_x,\alpha_b,\alpha_c,\gamma$, computation of $\bar{S}$, $\bar{X},\bar{R},M^\dagger,\bar{b},\bar{c}$, generation of the non-interactive proof π, and programming $\mathsf{H}[\mathbf{C},\mathbf{D},M^\dagger,\bar{R},\bar{S},\bar{X},\pi,\mu]$. $\textsc{User}_1$ stores $(\mathsf{ust}',R,S,X,\rho_C,b,c)$ as ust_j.

This game differs from the preceding game only if A makes any one of the two queries $\mathsf{H}(\mathbf{C},\mathbf{D},M^\dagger,\bar{R},\bar{S},\bar{X},\pi,\mu)$ before the two sessions are completed, which should have been programmed in $\textsc{User}_1$ in the preceding game but not yet in this game. Since $\bar{R}$ is uniformly random over $\mathbb{G}$ and independent of the output of $\textsc{User}_1$, each query from A hits these two queries with probability at most $2/p$. By the union bound, we have $\varepsilon_5 \geq \varepsilon_4 - 2q_h/p$.

G_6: Reorder sampling and computation in $\mathbf{User}_2$. In the preceding game, for each session, $\textsc{User}_2$ samples

$$\delta_{z_1},\delta_{z_2},\delta_s,\delta_x \leftarrow \mathbb{Z}_p, \quad \alpha_b,\alpha_c,\gamma \leftarrow \mathbb{Z}_p^*$$

and then computes

$$\bar{S} := \alpha_b^{-1}(S+\delta_s G), \quad \bar{X} := \alpha_c^{-1}(X+\delta_x G), \quad \bar{R} := R + \delta_{z_1}G + \delta_{z_2}H, \quad M^\dagger := \gamma G$$
$$\bar{b} := \alpha_b b, \quad \bar{c} := \alpha_c c, \quad \bar{z}_1 := z_1 + \delta_{z_1} + b\delta_s + c\delta_x, \quad \bar{z}_2 := z_2 + \delta_{z_2}.$$

Observe that, due to the randomness of $(\alpha_b,\alpha_c,\delta_{z_1},\delta_{z_2},\delta_s,\delta_x,\gamma)$, we have $(\bar{b},\bar{c},\bar{z}_1,\bar{z}_2,\bar{S},\bar{X},M^\dagger)$ uniformly distributed over $\mathbb{Z}_p^{*2}\times\mathbb{Z}_p^2\times\mathbb{G}^2\times\mathbb{G}^*$.

Therefore, we can reorder the above sampling and computation in this game. Specifically, this game samples

$$\bar{b},\bar{c} \leftarrow \mathbb{Z}_p^*, \quad \bar{z}_1,\bar{z}_2 \leftarrow \mathbb{Z}_p, \quad \bar{S},\bar{X} \leftarrow \mathbb{G}, \quad M^\dagger \leftarrow \mathbb{G}^*$$

and then computes

$$\alpha_b := \bar{b}/b, \quad \alpha_c := \bar{c}/c, \quad \delta_{z_1} := \bar{z}_1 - z_1 - b\delta_s - c\delta_x, \quad \delta_{z_2} := \bar{z}_2 - z_2,$$
$$\delta_s := \mathsf{dlog}(\alpha_b\bar{S} - S), \quad \delta_x := \mathsf{dlog}(\alpha_c\bar{X} - X), \quad \bar{R} := R + \delta_{z_1}G + \delta_{z_2}H.$$

Note that this game is inefficient as it computes δ_s, δ_x as the discrete logarithms of $\alpha_b\bar{S}-S$ and $\alpha_c\bar{X}-X$, respectively. The reordering does not affect the view of A. We have $\varepsilon_6 = \varepsilon_5$.

G_7: Omit computation in $\mathbf{User}_2$. In the preceding game, it can be verified that $\bar{R} = \bar{z}_1 G + \bar{z}_2 H - \bar{b}\bar{S} - \bar{c}\bar{X}$ (as in the correctness proof of the scheme). We let this game compute $\bar{R}$ in this way. Consequently, observe that $\alpha_b,\alpha_c,\delta_{z_1},\delta_{z_2},\delta_s,\delta_x$ are not referenced after being computed. We can let this game omit these computations. Note that this game becomes efficient. This change does not affect the view of A. We have $\varepsilon_7 = \varepsilon_6$.

$\textsc{User}_2(j, \mathsf{smsg}_2)$ ▷ G_0–G_4

Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \texttt{open}$
$\mathsf{sesst}_j := \texttt{closed}, \quad \mathsf{smsg}_{2,j} := \mathsf{smsg}_2$
if $\mathsf{sesst}_{1-j} \neq \texttt{closed}$ **then**
  Return $\bot$
else
  for $j' = 0, 1$ **do**
    $(\mathsf{smsg}'_2, z_1, z_2) := \mathsf{smsg}_{2,j'}$
    $(\mathsf{ust}', R, S, X, b, c, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi) := \mathsf{ust}_{j'}$
    $\sigma' \leftarrow \mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$
    $\bar{z}_1 := z_1 + \delta_{z_1} + b\delta_s + c\delta_x$
    $\bar{z}_2 := z_2 + \delta_{z_2}$
    $\sigma_{\beta_{j'}} := (\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2)$
  Return (σ_0, σ_1)

$\textsc{User}_2(j, \mathsf{smsg}_2)$ ▷ G_5–G_7

Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \texttt{open}$
$\mathsf{sesst}_j := \texttt{closed}, \quad \mathsf{smsg}_{2,j} := \mathsf{smsg}_2$
if $\mathsf{sesst}_{1-j} \neq \texttt{closed}$ **then**
  Return $\bot$
else
  for $j' = 0, 1$ **do**
    $(\mathsf{smsg}'_2, z_1, z_2) := \mathsf{smsg}_{2,j'}$
    $(\mathsf{ust}', R, S, X, \rho_C, b, c) := \mathsf{ust}_{j'}$
    $\sigma' \leftarrow \mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$
    $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$
    $\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x \leftarrow \mathbb{Z}_p, \quad \alpha_b, \alpha_c, \gamma \leftarrow \mathbb{Z}_p^*$ ▷ G_5
    $\bar{S} := \alpha_b^{-1}(S + \delta_s G), \quad \bar{X} := \alpha_c^{-1}(X + \delta_x G)$ ▷ G_5
    $\bar{R} := R + \delta_{z_1} G + \delta_{z_2} H$ ▷ G_5
    $M^\dagger := \gamma G$ ▷ G_5
    $\bar{b} := \alpha_b b, \quad \bar{c} := \alpha_c c$ ▷ G_5
    $\bar{z}_1 := z_1 + \delta_{z_1} + b\delta_s + c\delta_x, \quad \bar{z}_2 := z_2 + \delta_{z_2}$ ▷ G_5
    $\bar{b}, \bar{c} \leftarrow \mathbb{Z}_p^*, \quad \bar{z}_1, \bar{z}_2 \leftarrow \mathbb{Z}_p, \quad \bar{S}, \bar{X} \leftarrow \mathbb{G}, \quad M^\dagger \leftarrow \mathbb{G}^*$ ▷ G_6–G_7
    $\alpha_b := \bar{b}/b, \quad \alpha_c := \bar{c}/c$ ▷ G_6
    $\delta_s := \mathsf{dlog}(\alpha_b \bar{S} - S), \quad \delta_x := \mathsf{dlog}(\alpha_c \bar{X} - X)$ ▷ G_6
    $\delta_{z_1} := \bar{z}_1 - z_1 - b\delta_s - c\delta_x, \quad \delta_{z_2} := \bar{z}_2 - z_2$ ▷ G_6
    $\bar{R} := R + \delta_{z_1} G + \delta_{z_2} H$ ▷ G_6
    $\bar{R} := \bar{z}_1 G + \bar{z}_2 H - \bar{b}\bar{S} - \bar{c}\bar{X}$ ▷ G_7
    $\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (G, G); \rho_D)$
    $\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$
    $\pi \leftarrow \mathsf{NISim}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm})$
    $\mathsf{H}[\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu_{\beta_{j'}}] := (\bar{b}, \bar{c})$
    $\sigma_{\beta_{j'}} := (\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2)$
  Return (σ_0, σ_1)

Figure 12: $\textsc{User}_2$ in G_0–G_7 for the proof of Theorem 1.

G_8: Generate σ_0 and σ_1 in order. When both sessions are completed, instead of iterating over sessions $j' = 0, 1$, we let this game iterate over signatures $i = 0, 1$ and generate σ_0 and σ_1 in

$\underline{\textsc{User}_2(j, \mathsf{smsg}_2)}$ ▷ G_8

Assert $j \in \{0,1\} \wedge \mathsf{sesst}_j = \texttt{open}$
$\mathsf{sesst}_j := \texttt{closed}, \quad \mathsf{smsg}_{2,j} := \mathsf{smsg}_2$
if $\mathsf{sesst}_{1-j} \neq \texttt{closed}$ **then**
  Return $\perp$
else
  $j_0 := \beta_0, \quad j_1 := \beta_1$
  for $i = 0, 1$ **do**
    $\bar{b}, \bar{c} \leftarrow \mathbb{Z}_p^*, \quad \bar{z}_1, \bar{z}_2 \leftarrow \mathbb{Z}_p, \quad \bar{S}, \bar{X} \leftarrow \mathbb{G}, \quad M^\dagger \leftarrow \mathbb{G}^*$
    $\bar{R} := \bar{z}_1 G + \bar{z}_2 H - \bar{b}\bar{S} - \bar{c}\bar{X}$
    $\mathbf{D} \leftarrow \mathsf{Enc}(\mathsf{ek}, (G, G); \rho_D)$
    $(\mathsf{smsg}'_2, z_1, z_2) := \mathsf{smsg}_{2,j_i}$ ▷ *z_1, z_2 are not needed*
    $(\mathsf{ust}', R, S, X, b, c) := \mathsf{ust}_{j_i}$ ▷ *b, c are not needed*
    $\sigma' \leftarrow \mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$
    $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$
    $\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$
    $\pi \leftarrow \mathsf{NISim}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm})$
    $\mathsf{H}[\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu_i] := (\bar{b}, \bar{c})$
    $\sigma_i := (\sigma', \mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \bar{z}_1, \bar{z}_2)$
  Return (σ_0, σ_1)

Figure 13: $\textsc{User}_2$ in G_8 for the proof of Theorem 1.

order. For $i \in \{0,1\}$, let j_i be the session corresponding to message μ_i. Equivalently, let $j_0 = \beta_0$ and $j_1 = \beta_1$. $\textsc{User}_2$ generates σ_i as follows. First, it samples $\bar{b}, \bar{c}, \bar{z}_1, \bar{z}_2, \bar{S}, \bar{X}, M^\dagger$, computes $\bar{R}$, and generates $\mathbf{D}$ (recall that it is an encryption of (G, G)). This part is independent of j_i. The remaining steps depend on j_i. From smsg_{2,j_i} and ust_{j_i}, $\textsc{User}_2$ retrieves $\mathsf{smsg}'_2, \mathsf{ust}', R, S, X$ from session j_i. Then it generates $\sigma' \leftarrow \mathsf{EMBS.User}_2((R, S, X), \mathsf{ek}, \mathsf{pk}, \mathsf{ust}', \mathsf{smsg}'_2)$, $\mathbf{C} := \mathsf{Enc}(\mathsf{ek}, (R, S, X); \rho_C)$, the non-interactive proof π, and programs H. The changes in this game are only conceptual. We have $\varepsilon_8 = \varepsilon_7$.

Reduction. Now, we are ready to construct the reduction B who plays the $\mathsf{EMBLIND}_{\mathsf{EMBS}}$ game. With its own oracles $\textsc{User}'_1$, $\textsc{User}'_2$, and secret bits β'_0, β'_1, B simulates G_8 as follows, such that the secret bits (β_0, β_1) of G_8 equal the secret bits (β'_0, β'_1) of the $\mathsf{EMBLIND}_{\mathsf{EMBS}}$ game. In $\textsc{User}_1$, B queries $\textsc{User}'_1(j, (R, S, X), \mathsf{smsg}'_1)$ to obtain umsg'. The encryption randomness ρ_C for session j is generated but unknown to B.

B simulates $\textsc{User}_2$ as follows. When both sessions are completed, B also completes its two sessions. For $j \in \{0,1\}$, B retrieves smsg'_2 from $\mathsf{smsg}_{2,j}$ and queries $\textsc{User}'_2(j, \mathsf{smsg}_{2,j})$. It then receives $(\sigma'_{\beta'_0}, \mathbf{C}_{\beta'_0}), (\sigma'_{\beta'_1}, \mathbf{C}_{\beta'_1})$. For $i \in \{0,1\}$, B uses $(\sigma'_{\beta'_i}, \mathbf{C}_{\beta'_i})$ as the $(\sigma', \mathbf{C})$ when computing σ_i.

If $\beta'_0 = 0$ (and $\beta'_1 = 1$), then $(\sigma'_{\beta'_i}, \mathbf{C}_{\beta'_i})$ is the signature-encryption pair generated by session i. Thus, the simulated G_8 has $j_0 = 0$ and $j_1 = 1$, which means $\beta_0 = 0$, $\beta_1 = 1$. Conversely, if $\beta'_0 = 1$ (and $\beta'_1 = 0$), then $(\sigma'_{\beta'_i}, \mathbf{C}_{\beta'_i})$ is the signature-encryption pair generated by session $1 - i$. This corresponds to the behavior of G_8 when $\beta_0 = 1$ and $\beta_1 = 0$. In conclusion, B perfectly simulates G_8. We let B copy A's output, and hence $\varepsilon_{\text{embs}} = \varepsilon_8$. This completes the proof. □

B Proof of OMUF

General Forking Lemma. We will use the general forking lemma by Bellare and Neven [BN06].

$\underline{\mathsf{fork}^{\mathsf{W}}(X)}$

$\rho \leftarrow \mathcal{R}$

$h_1, \ldots, h_q \leftarrow \mathcal{H}$

$(I, Y) \leftarrow \mathsf{W}(X, h_1, \ldots, h_q; \rho)$

if $I = 0$ **then**

  Return $\perp$

$h'_I, \ldots, h'_q \leftarrow \mathcal{H}$

$(I', Y') \leftarrow \mathsf{W}(X, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$

if $I \neq I' \vee h_I = h'_I$ **then**

  Return $\perp$

Return (I, Y, Y')

Figure 14: Algorithm $\mathsf{fork}^{\mathsf{W}}(X)$ in Lemma 8.

Lemma 8 (General forking lemma [BN06])**.** *Fix an integer $q \geq 1$, a set $\mathcal{H}$ of size $|\mathcal{H}| \geq 2$. Let W be a randomized algorithm that takes as inputs X, h_1, ..., h_q, takes as random coin tosses from set $\mathcal{R}$, and outputs a tuple (I, Y) where $I \in \{0, \ldots, q\}$, and auxiliary output Y. Let $\mathcal{D}_X$ be an unspecified distribution. Let*

$$\varepsilon = \Pr[I \geq 1 : X \leftarrow \mathcal{D}_X; h_1, \ldots, h_q \leftarrow \mathcal{H}; (I, Y) \leftarrow \mathsf{W}(X, h_1, \ldots, h_q)].$$

Define the forking algorithm $\mathsf{fork}^{\mathsf{W}}$ with respect to W as in Fig. 14. Let

$$\varepsilon' = \Pr[\mathsf{fork}^{\mathsf{W}}(X) \neq \perp : X \leftarrow \mathcal{D}_X].$$

Then $\varepsilon' \geq \varepsilon(\varepsilon/q - 1/|\mathcal{H}|)$. Alternatively, $\varepsilon \leq \sqrt{q\varepsilon'} + q/|\mathcal{H}|$.

Proof of Lemma 4. We construct the reduction B that plays the AOMP game.

Define functions F_{bc}. We define a function F_{bc} that takes inputs in the form of $(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$, the same as H. We restrict that it is only defined for $(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$ that have been queried to H. In other words, each function value of F_{bc} is associated with a H query. Informally, $\mathsf{F}_{bc}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$ is the value of (b, c) returned by User_1 if User_1 makes the query $\mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$. Looking ahead, it also means the value of (b, c) that B can obtain, when it extracts the blinding factors (as the witness for π) and uses them to unblind $\bar{b}$ and $\bar{c}$.

The specific definitions are as follows. Let dk be the decryption key corresponding to ek. Let $(\alpha_b G, \alpha_c G)$ be $\mathsf{Dec}(\mathsf{dk}, \mathbf{D})$ and $(\bar{b}, \bar{c})$ be $\mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$. We define $\mathsf{F}_{bc}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu) = (\bar{b}/\alpha_b, \bar{c}/\alpha_c)$. If $\alpha_b = 0$ or $\alpha_c = 0$, then we define $\mathsf{F}_{bc}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu) = \perp$.

Modify security game. We modify the original OMUF game as follows to obtain G_1:

- Generate dk: In the setup phase, the game generates $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.KGen}()$ and lets ek be part of the public parameters.
- Verify π upon H queries: Upon any H query $\mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$, the game executes $\mathsf{NIV}^{\mathsf{H}_\pi(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \cdot)}(\mathsf{stm}, \pi)$ with $\mathsf{stm} := \mathsf{FormatS}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X})$.
- Rule out (R_j, S_j, X_j) collisions: The game aborts when there exist $j \neq j' \in [q_s]^+$ such that $(R_j, S_j, X_j) = (R_{j'}, S_{j'}, X_{j'})$.
- Ensure $\mathbf{D}^*_i$ does not decrypt to O: For each $i \in [k+1]^+$, the game aborts if any element of $\mathsf{Dec}(\mathsf{dk}, \mathbf{D}^*_i)$ equals O.

- Rule out non-$\perp$ F_{bc} collisions: The game aborts when there exist two distinct H queries such that the corresponding F_{bc} values are equal and are non-$\perp$.

For the last change, we note that F_{bc} is not efficiently computable. Nevertheless, the game can still efficiently detect collisions as follows. For each fresh H query $\mathsf{H}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$, the game decrypts $\mathbf{D}$ to obtain $(\alpha_b G, \alpha_c G)$. If $\alpha_b G$ or $\alpha_c G$ equals O, then $\mathsf{F}_{bc}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu) = \perp$. Otherwise, let $(b, c) = \mathsf{F}_{bc}(\mathbf{C}, \mathbf{D}, M^\dagger, \bar{R}, \bar{S}, \bar{X}, \pi, \mu)$. After sampling the answer $(\bar{b}, \bar{c})$, the game computes and records $(\bar{b}^{-1}(\alpha_b G), \bar{c}^{-1}(\alpha_c G))$, which is equal to $(b^{-1}G, c^{-1}G)$. The game detects F_{bc} collisions by detecting collisions between these $(b^{-1}G, c^{-1}G)$ values.

Now we analyze how these changes affect the probability that A wins and Case 2 holds.

- The first change does not affect the view of A.
- The second change introduces q_h artificial H_π queries. Including the $k+1 \leq q_s+1$ H_π queries for forgery verification, the total number of H_π queries is bounded by $q = 2q_h + q_s + 1$.
- The game aborts with probability at most $q_s^2/p^2 < q^2/p$ due to the third change.
- Recall that for every $i \in [k+1]^+$, the validity of the forgery requires that $M_i^{\dagger *} \neq O$ and π_i^* verifies, where π_i^* proves the existence of η_b, η_c such that $\eta_b(D, D_{\alpha_b})$ and $\eta_c(D, D_{\alpha_c})$ both decrypt to $M_i^{\dagger *}$, where $\mathbf{D}_i^*$ is parsed into $(D, D_{\alpha_b}, D_{\alpha_c})$. If any element that $\mathbf{D}_i^*$ encrypts is O, such (η_b, η_c) does not exist. By the special soundness of Σ, for a statement where a witness does not exist, there exists at most one challenge $\mathsf{ch} \in \mathbb{Z}_p$ such that a valid response exists. By the union bound, except with probability q/p, there exists no H_π query such that the corresponding statement has no witness while the answer hits this special challenge ch. Consequently, a false statement cannot be proved, so $\mathbf{D}_i^*$ cannot encrypt any O. In conclusion, the game aborts with probability at most q/p due to the fourth change.
- Consider the last change. Since $(\bar{b}, \bar{c})$ is uniformly chosen after α_b and α_c have been fixed in $\mathbf{D}$, the F_{bc} value associated with each H query is uniform over $\mathbb{Z}_p^{*2}$ if it is non-$\perp$. Thus, the collision probability is at most $q^2/(p-1)^2$.

Let ε_1 be the probability that A wins G_1 with Case 2 holding. We have $\varepsilon_1 \geq \varepsilon_{\mathrm{bs},2} - (q^2+q)/p - q^2/(p-1)^2 = \varepsilon_{\mathrm{bs},2} - O(q^2)/p$.

Define the special forgery. Among the $k+1$ forgeries, we define the "special forgery". For every $i \in [k+1]^+$, parse σ_i^* into $(\sigma_i'^*, \mathbf{C}_i^*, \mathbf{D}_i^*, M_i^{\dagger *}, \bar{R}_i^*, \bar{S}_i^*, \bar{X}_i^*, \pi_i^*, \bar{z}_{1,i}^*, \bar{z}_{2,i}^*)$, and let $(R_i^*, S_i^*, X_i^*) = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}_i^*)$. Since Case 2 holds, for every $i \in [k+1]^+$, there exists a completed session j such that $(R_i^*, S_i^*, X_i^*) = (R_j, S_j, X_j)$. Moreover, since we have ruled out (R_j, S_j, X_j) collisions, each i corresponds to a unique completed session j in this way. We denote this session by a function $j(i)$. Let i^* be the smallest index such that $\mathsf{F}_{bc}(\mathbf{C}_{i^*}^*, \mathbf{D}_{i^*}^*, M_{i^*}^{\dagger *}, \bar{R}_{i^*}^*, \bar{S}_{i^*}^*, \bar{X}_{i^*}^*, \pi_{i^*}^*, \mu_{i^*}^*) \neq (b_{j(i^*)}, c_{j(i^*)})$, where (b_j, c_j) denotes the input (b, c) to $\textsc{Sign}_2$ in session j. We define the special forgery as $(\mu_{i^*}^*, \sigma_{i^*}^*)$. For simplicity, we omit the subscript i^* and denote the special forgery using (μ^*, σ^*), where $\sigma^* = (\sigma'^*, \mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \bar{z}_1^*, \bar{z}_2^*)$.

We argue that the special forgery must exist. Namely, there must exist an i^* such that

$$\mathsf{F}_{bc}(\mathbf{C}_{i^*}^*, \mathbf{D}_{i^*}^*, M_{i^*}^{\dagger *}, \bar{R}_{i^*}^*, \bar{S}_{i^*}^*, \bar{X}_{i^*}^*, \pi_{i^*}^*, \mu_{i^*}^*) \neq (b_{j(i^*)}, c_{j(i^*)}).$$

Since there are at most k completed sessions, there must exist $i \neq i' \in [k+1]^+$ such that $j(i) = j(i')$. Since we have ruled out F_{bc} collisions, we have

$$\mathsf{F}_{bc}(\mathbf{C}_i^*, \mathbf{D}_i^*, M_i^{\dagger *}, \bar{R}_i^*, \bar{S}_i^*, \bar{X}_i^*, \pi_i^*, \mu_i^*) \neq \mathsf{F}_{bc}(\mathbf{C}_{i'}^*, \mathbf{D}_{i'}^*, M_{i'}^{\dagger *}, \bar{R}_{i'}^*, \bar{S}_{i'}^*, \bar{X}_{i'}^*, \pi_{i'}^*, \mu_{i'}^*).$$

Therefore, at least one of i and i' satisfies the condition for i^*.

We note that the special forgery can be efficiently identified. Let

$$(b, c) = \mathsf{F}_{bc}(\mathbf{C}_i^*, \mathbf{D}_i^*, M_i^{\dagger *}, \bar{R}_i^*, \bar{S}_i^*, \bar{X}_i^*, \pi_i^*, \mu_i^*).$$

Although (b, c) is not efficiently computable, note that $(b^{-1}G, c^{-1}G)$ can be efficiently computed, as we have shown when ruling out F_{bc} collisions. Therefore, the special forgery can be identified by comparing this value with $(b_{j(i)}^{-1}G, c_{j(i)}^{-1}G)$.

Wrapper. In preparation for constructing our rewinding reduction, we first specify a wrapper W that satisfies the syntax of the forking lemma (Lemma 8). We first specify its inputs:

- W takes $H \leftarrow \mathbb{G}$ and $3q_s$ challenge elements $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ as inputs. The challenge elements are generated as $R_j := r_{1,j}G + r_{2,j}H$, $S_j := s_{1,j}G + s_{2,j}H$, and $X_j := x_{1,j}G + x_{2,j}H$ for $j \in [q_s]^+$, where $r_{1,j}, r_{2,j}, s_{1,j}, s_{2,j}, x_{1,j}, x_{2,j}$ are all uniformly chosen from $\mathbb{Z}_p$.
  For notational convenience, we identify the aforementioned inputs of W with the same variable names used in the security game. Looking ahead, W will assign to each variable in the security game the value of the corresponding input of W with the same name (if it exists).
- W also takes $h_1, \ldots, h_q$ as inputs, each uniformly distributed over $\mathbb{Z}_p$.
- Moreover, it has access to the preimage oracle PreImg. Given a linear combination of the challenge elements $\sum_{j\in[q_s]^+}(\beta_{R_j}R_j + \beta_{S_j}S_j + \beta_{X_j}X_j)$, with the combination coefficients $\{(\beta_{R_j}, \beta_{S_j}, \beta_{X_j})\}_{j\in[q_s]^+}$ (implicitly) as the inputs, PreImg returns $\sum_{j\in[q_s]^+}(\beta_{R_j}(r_{1,j}, r_{2,j}) + \beta_{S_j}(s_{1,j}, s_{2,j}) + \beta_{X_j}(x_{1,j}, x_{2,j}))$.

W randomly chooses the random coin tosses for A using its own coin tosses. It runs A and simulates G_1 as follows:

- Setup: W sets the public parameter H to be its input of the same name. It executes the remaining part of the setup phase faithfully, namely generating the encryption key ek and the EMBS key pair $(\mathsf{pk}, \mathsf{sk})$. We recall that ek is generated with a known corresponding decryption key dk.
- Sign$_1$: Upon the j-th Sign$_1$ query, W sets R_j, S_j, X_j to be its inputs of the same name. It executes $\mathsf{EMBS.Sign}_1((R_j, S_j, X_j), \mathsf{ek}, \mathsf{sk})$ faithfully, to obtain $(\mathsf{smsg}'_1, \mathsf{sst}')$. W returns $(\mathsf{smsg}'_1, R_j, S_j, X_j)$.
- Sign$_2$: On inputs (j, umsg), W parses umsg into $(\mathsf{umsg}', b_j, c_j)$ and retrieves sst' from the signer state of session j. W runs $\mathsf{EMBS.Sign}_2((R_j, S_j, X_j), \mathsf{ek}, \mathsf{sk}, \mathsf{sst}', \mathsf{umsg}')$ faithfully to obtain smsg'_2. It queries PreImg$(R_j + b_jS_j + c_jX_j)$ to obtain $(z_{1,j}, z_{2,j}) := (r_{1,j}, r_{2,j}) + b_j(s_{1,j}, s_{2,j}) + c_j(x_{1,j}, x_{2,j})$. Finally, W returns $(\mathsf{smsg}'_2, z_{1,j}, z_{2,j})$.
- H_π: During the game, W answers each distinct H_π query with $h_1, \ldots, h_q$ in order.

W verifies whether A wins G_1, i.e., A returns $k+1$ valid signatures $\{\sigma_i^*\}_{i\in[k+1]^+}$ on distinct $\{\mu_i^*\}_{i\in[k+1]^+}$. For $i \in [k+1]^+$, parse σ_i^* into $(\sigma_i'^*, \mathbf{C}_i^*, \mathbf{D}_i^*, M_i^{\dagger *}, \bar{R}_i^*, \bar{S}_i^*, \bar{X}_i^*, \pi_i^*, \bar{z}_{1,i}^*, \bar{z}_{2,i}^*)$, and let $(R_i^*, S_i^*, X_i^*) = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}_i^*)$. W also verifies whether Case 2 holds, i.e., for every $i \in [k+1]^+$, there exists $j \in [q_s]^+$ such that session j is complete, and $(R_i^*, S_i^*, X_i^*) = (R_j, S_j, X_j)$.

If A wins and Case 2 holds, then we let W identify the special forgery μ^*, $(\sigma'^*, \mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \bar{z}_1^*, \bar{z}_2^*)$. It returns I such that h_I was assigned to the H_π query corresponding to π^*, namely $\mathsf{H}_\pi(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \mathsf{cm}^*)$ where $(\mathsf{cm}^*, \mathsf{rs}^*) = \pi^*$. W sets the auxiliary output (the Y in Lemma 8) to be the whole history of the game, including all variables used in the game and all interactions with A. If A does not win or Case 2 does not hold, then W returns $I = 0$.

We will apply the forking lemma (Lemma 8) to W', the combination of W and the PreImg oracle considered as a single algorithm. Specifically, W' takes $\{(r_{1,j}, r_{2,j}, s_{1,j}, s_{2,j}, x_{1,j}, x_{2,j})\}_{j\in[q_s]^+}$ and $h_1, \ldots, h_q$ as inputs. Then W' uses $\{(r_{1,j}, r_{2,j}, s_{1,j}, s_{2,j}, x_{1,j}, x_{2,j})\}_{j\in[q_s]^+}$ to compute the inputs $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ for W. It runs W on inputs $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ and $h_1, \ldots, h_q$ and answers the PreImg queries from W. At the end, W' forwards the output of W.

Rewinding. Now, we describe how our reduction B, who plays the $\mathsf{AOMP}[3q_s]$ game, rewinds A. B uses W as a subroutine, which takes black-box access of A. It takes H and the challenge elements $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ as inputs and has access to the PreImg oracle. B runs $\mathsf{fork}^{\mathsf{W}'}$ as defined in Lemma 8 on inputs $\{(r_{1,j}, r_{2,j}, s_{1,j}, s_{2,j}, x_{1,j}, x_{2,j})\}_{j\in[q_s]^+}$, the secret preimages in the AOMP game which are unknown to B. Inside W', B implements the W-part by running W as a subroutine, while it performs the other part, i.e., computing inputs for W and answering PreImg queries, using its own inputs and PreImg oracle.

In the following, we describe the whole procedure how B runs $\mathsf{fork}^{\mathsf{W}'}$. Under certain conditions,

B does not have to complete the whole procedure to win the $\mathsf{AOMP}[3q_s]$ game.[1] We will come to this part later.

To be specific, B does the following:

- Forward PreImg queries throughout the procedure: Recall that each PreImg query is a linear combination of the challenge elements. We identify each query using the vector $\vec{\beta}$ of the combination coefficients. Suppose that B has queried its own PreImg oracle with $\vec{\beta}_1, \ldots, \vec{\beta}_l$ and received the answers $\mathbf{y}_1, \ldots, \mathbf{y}_l$. When W makes a PreImg query with vector $\vec{\beta}$, B checks if $\vec{\beta}$ is in the span of $\{\vec{\beta}_1, \ldots, \vec{\beta}_l\}$. If it is, then B can express $\vec{\beta}$ as a linear combination $\sum_{i\in[l]^+} \theta_i \vec{\beta}_i$ and answers W's PreImg query with $\sum_{i\in[l]^+} \theta_i \mathbf{y}_i$. If $\vec{\beta}$ is not in the span, then B queries $\vec{\beta}$ to its own PreImg oracle and forwards the answer to W. In this way, B ensures that it only makes linearly independent queries to its PreImg oracle.
- First execution of W: B runs W for the first time. B forwards its own inputs H, $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ as the corresponding inputs of W with the same variable names. B uniformly samples the inputs $h_1, \ldots, h_q \leftarrow \mathbb{Z}_p$ of W. In addition, B uniformly samples the random coin tosses for W. B answers W's PreImg queries in the aforementioned way. At the end, if W returns $I = 0$, then B aborts. If $I \geq 1$, then B runs W for the second time.
- Second execution of W: B runs W for the second time. It still forwards its own inputs H, $\{(R_j, S_j, X_j)\}_{j\in[q_s]^+}$ as the corresponding inputs of W with the same variable names. This time, B provides W with inputs $h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q$, where $h_1, \ldots, h_{I-1}$ are the same as the first execution while $h'_I, \ldots, h'_q \leftarrow \mathbb{Z}_p$ are resampled uniformly. It sets the random coin tosses to be the same as in the first time. Consequently, A (inside W) also uses the same coin tosses as in the first execution. B answers W's PreImg queries in the aforementioned way. At the end, if W returns $I' = I$, and it holds that $h'_I \neq h_I$, then this rewinding procedure succeeds. Otherwise, B aborts.

Note that B makes at most q_s queries to its PreImg oracle for each execution of W. In total, B makes at most $2q_s$ queries for this rewinding procedure.

The crucial query. We define the *crucial query* to be the first H query in the second execution of W satisfying the following conditions:

- It is in the form of $\mathsf{H}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi, \mu)$;
- $\mathsf{NIV}^{\mathsf{H}_\pi(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \cdot)}(\mathsf{stm}, \pi) = 1$, where $\mathsf{stm} = \mathsf{FormatS}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*)$; and
- $\mathsf{H}_\pi(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \mathsf{cm})$ is assigned with h'_I (i.e., it is the I-th distinct H_π query, also the rewinding point), where we parse π into $(\mathsf{cm}, \mathsf{rs})$.

Here, $(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*)$ and π^* are those from the special forgery in the first execution of W. Recall that when we modified the security game, we have made G_1 verify π for each H query.

We argue that if the rewinding procedure succeeds, then the crucial query must exist. Indeed, if the rewinding succeeds, the special forgery in the second execution yields a query

$$\mathsf{H}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$$

that is made when verifying the forgery. Since $I = I'$, and by the definition of I in W, the verification of π^* and π^{**} makes the same H_π query (i.e., the input to H_π is the same). This implies that

$$(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*) = (\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}).$$

Moreover, π^{**} is a valid proof for the same statement, and the corresponding H_π query is assigned with h'_I. Therefore, this H query satisfies all three conditions above, so the crucial query exists. We stress that the crucial query may occur before A terminates. Lemma 8 lower-bounds the probability that the rewinding procedure succeeds, which also bounds the probability that the crucial query exists. We have that the crucial query exists with probability at least $\varepsilon_1(\varepsilon_1/q - 1/p)$.

[1] In particular, W is not used in a black-box way. B monitors certain conditions in W and may terminate midway.

Extract the blinding factors. Once the crucial query $\mathsf{H}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi, \mu)$ occurs in the second execution, B can perform the extraction by combining π^* from the first execution with π from the crucial query, together with the distinct challenges h_I and h'_I.

Specifically, parse π^* and π into $(\mathsf{cm}^*, \mathsf{rs}^*)$ and $(\mathsf{cm}, \mathsf{rs})$, respectively. Since the crucial query and π^* correspond to the same H_π query, we have $\mathsf{cm}^* = \mathsf{cm}$. Moreover, we have $\mathsf{ch}^* = h_I \neq h'_I = \mathsf{ch}$, where ch^* and ch are the challenges used in π^* and π, respectively. Therefore, B has obtained two valid Σ-transcripts $(\mathsf{cm}^*, \mathsf{ch}^*, \mathsf{rs}^*)$ and $(\mathsf{cm}, \mathsf{ch}, \mathsf{rs})$ under the same statement $\mathsf{FormatS}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*)$, with $\mathsf{cm}^* = \mathsf{cm}$ and $\mathsf{ch}^* \neq \mathsf{ch}$. By the special soundness of Σ, B can extract a witness for the statement.

Specifically, B can extract $(\rho_C, \rho_D, \delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c, \eta_b, \zeta_b, \eta_c, \zeta_c)$ satisfying

$$\rho_C(G, E_1, E_2, E_3) + (O,\ \bar{R}^* - \delta_{z_1} G - \delta_{z_2} H,\ \alpha_b \bar{S}^* - \delta_s G,\ \alpha_c \bar{X}^* - \delta_x G) = \mathbf{C}^*, \tag{5}$$

$$\rho_D(G, E_1, E_2) + (O, \alpha_b G, \alpha_c G) = \mathbf{D}^*, \tag{6}$$

$$\eta_b(D^*, D^*_{\alpha_b}) + \zeta_b(G, E_1) = (O, M^{\dagger *}), \tag{7}$$

$$\eta_c(D^*, D^*_{\alpha_c}) + \zeta_c(G, E_2) = (O, M^{\dagger *}), \tag{8}$$

where ek is parsed as (G, E_1, E_2, E_3) and $\mathbf{D}^*$ as $(D^*, D^*_{\alpha_b}, D^*_{\alpha_c})$. Eq. (5) implies that

$$\mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*) = (\bar{R}^* - \delta_{z_1} G - \delta_{z_2} H, \alpha_b \bar{S}^* - \delta_s G, \alpha_c \bar{X}^* - \delta_x G).$$

Define j^* as $j(i^*)$ where i^* is the index of the special forgery (in the first execution of W). By the definition of the function $j(\cdot)$ (used to define the special forgery), we have $\mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*) = (R_{j^*}, S_{j^*}, X_{j^*})$. Therefore, we have

$$R_{j^*} = \bar{R}^* - \delta_{z_1} G - \delta_{z_2} H, \tag{9}$$

$$S_{j^*} = \alpha_b \bar{S}^* - \delta_s G, \tag{10}$$

$$X_{j^*} = \alpha_c \bar{X}^* - \delta_x G. \tag{11}$$

Eq. (6) implies that

$$\mathsf{Dec}(\mathsf{dk}, \mathbf{D}^*) = (\alpha_b G, \alpha_c G).$$

Moreover, the existence of $(\eta_b, \zeta_b, \eta_c, \zeta_c)$ satisfying Eqs. (7) and (8) implies that

$$\alpha_b \neq 0,\ \alpha_c \neq 0 \tag{12}$$

Solve AOMP. We are ready to show how B wins the $\mathsf{AOMP}[3q_s]$ game. B can solve for a sufficient solution $\{(r^*_{1,j}, r^*_{2,j}, s^*_{1,j}, s^*_{2,j}, x^*_{1,j}, x^*_{2,j})\}_{j \in [q_s]^+}$ if it can collect $3q_s$ equations in the form of

$$y_1 G + y_2 H = \sum_{j \in [q_s]^+} (\beta_{R_j} R_j + \beta_{S_j} S_j + \beta_{X_j} X_j),$$

such that the coefficient vectors for the right-hand side of these equations are linearly independent.

From each PreImg query, B can obtain one such equation. In particular, to execute the rewinding procedure, each PreImg query gives an equation in the form of

$$z_{1,j} G + z_{2,j} H = R_j + b_j S_j + c_j X_j,$$

where $(\beta_{R_j}, \beta_{S_j}, \beta_{X_j}) \neq (0, 0, 0)$ only holds for one j. We only need to show that, at some point during the rewinding procedure, B can extract one equation that is linearly independent of those it has obtained from the PreImg queries. Then B terminates the rewinding procedure and uses all its remaining PreImg queries to obtain more independent equations. Eventually, it obtains one more equation than the number of its PreImg queries, that is $3q_s$ equations, and can solve for a solution.

Let j^* be the unique completed session such that $\mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*) = (R_{j^*}, S_{j^*}, X_{j^*})$, i.e., $j^* = j(i^*)$ where i^* is the index of the special forgery. Depending on the moment when the crucial query occurs, there are two cases:

- Case (i): The crucial query occurs before session j^* is completed in the second execution of W.
- Case (ii): The crucial query occurs after session j^* is completed in the second execution of W.

Extract one more equation: Case (i). In Case (i), B runs the extraction once the crucial query occurs. It obtains $\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c$ such that Eqs. (9) to (12) hold, where Eq. (12) states that $\alpha_b \neq 0$ and $\alpha_c \neq 0$. B computes

$$b^* := \bar{b}^*/\alpha_b, \quad c^* := \bar{c}^*/\alpha_c, \quad z_1^* := \bar{z}_1^* - \delta_{z_1} - b^*\delta_s - c^*\delta_x, \quad z_2^* := \bar{z}_2^* - \delta_{z_2}, \tag{13}$$

where $(\bar{b}^*, \bar{c}^*) = \mathsf{H}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \mu^*)$ in the first execution of W. The validity of the special forgery in the first execution guarantees that

$$\bar{z}_1^* G + \bar{z}_2^* H = \bar{R}^* + \bar{b}^* \bar{S}^* + \bar{c}^* \bar{X}^*. \tag{14}$$

Combining Eqs. (9) to (11), (13) and (14), we have

$$z_1^* G + z_2^* H = R_{j^*} + b^* S_{j^*} + c^* X_{j^*}. \tag{15}$$

This is the new equation that B extracts.

Since $(\alpha_b G, \alpha_c G) = \mathsf{Dec}(\mathsf{dk}, \mathbf{D}^*)$ (implied by Eq. (6)), and by the definition of F_{bc}, we have $(b^*, c^*) = \mathsf{F}_{bc}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \mu^*)$. Recall that the special forgery is defined such that $\mathsf{F}_{bc}(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*, \pi^*, \mu^*) \neq (b_{j^*}, c_{j^*})$, and hence $(b^*, c^*) \neq (b_{j^*}, c_{j^*})$. As Case (i) holds, the only PreImg query that B has made about $(R_{j^*}, S_{j^*}, X_{j^*})$ is to complete session j^*, obtaining (z_{1,j^*}, z_{2,j^*}) satisfying

$$z_{1,j^*} G + z_{2,j^*} H = R_{j^*} + b_{j^*} S_{j^*} + c_{j^*} X_{j^*}. \tag{16}$$

Since $(b^*, c^*) \neq (b_{j^*}, c_{j^*})$, the newly extracted equation Eq. (15) must be linearly independent of Eq. (16). Thus, B successfully obtains one more equation.

Extract one more equation: Case (ii). In Case (ii), B runs the extraction after the rewinding procedure is finished. Let μ^{**}, $(\sigma'^{**}, \mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \bar{z}_1^{**}, \bar{z}_2^{**})$ be the special forgery in the second execution of W, where

$$(\mathbf{C}^*, \mathbf{D}^*, M^{\dagger *}, \bar{R}^*, \bar{S}^*, \bar{X}^*) = (\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}).$$

As in Case (i), B extracts $\delta_{z_1}, \delta_{z_2}, \delta_s, \delta_x, \alpha_b, \alpha_c$ such that Eqs. (9) to (12) hold, where Eq. (12) states that $\alpha_b \neq 0$ and $\alpha_c \neq 0$. Differently from Case (i), B computes

$$b^{**} := \bar{b}^{**}/\alpha_b, \quad c^{**} := \bar{c}^{**}/\alpha_c, \quad z_1^{**} := \bar{z}_1^{**} - \delta_{z_1} - b^{**}\delta_s - c^{**}\delta_x, \quad z_2^{**} := \bar{z}_2^{**} - \delta_{z_2}, \tag{17}$$

where $(\bar{b}^{**}, \bar{c}^{**}) = \mathsf{H}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$. That is, B unblinds the special forgery from the second execution. The validity of the forgery guarantees that

$$\bar{z}_1^{**} G + \bar{z}_2^{**} H = \bar{R}^{**} + \bar{b}^{**} \bar{S}^{**} + \bar{c}^{**} \bar{X}^{**}. \tag{18}$$

Combining Eqs. (9) to (11), (17) and (18), we have

$$z_1^{**} G + z_2^{**} H = R_{j^*} + b^{**} S_{j^*} + c^{**} X_{j^*}. \tag{19}$$

This is the new equation that B extracts.

As Case (ii) holds, B has completed session j^* in both executions of W. Let (b_{j^*}, c_{j^*}) and (b'_{j^*}, c'_{j^*}) be the inputs to $\text{Sign}_2(j^*, \cdot)$ in the first and the second execution, respectively. If $(b_{j^*}, c_{j^*}) = (b'_{j^*}, c'_{j^*})$, then B has only made one PreImg query about $(R_{j^*}, S_{j^*}, X_{j^*})$. By the definition of special forgery (in the second execution), we have $(b^{**}, c^{**}) = \mathsf{F}_{bc}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**}) \neq (b'_{j^*}, c'_{j^*})$. Therefore, Eq. (19) is a linearly independent new equation as in Case (i).

We need to be careful about the case that $(b_{j^*}, c_{j^*}) \neq (b'_{j^*}, c'_{j^*})$. In this case, B has made two PreImg queries about $(R_{j^*}, S_{j^*}, X_{j^*})$ and obtained two linearly independent equations

$$z_{1,j^*}G + z_{2,j^*}H = R_{j^*} + b_{j^*}S_{j^*} + c_{j^*}X_{j^*}, \tag{20}$$
$$z'_{1,j^*}G + z'_{2,j^*}H = R_{j^*} + b'_{j^*}S_{j^*} + c'_{j^*}X_{j^*}, \tag{21}$$

in the two executions, respectively. B wins the $\mathsf{AOMP}[3q_s]$ game if the newly extracted equation Eq. (19) is independent of Eqs. (20) and (21), namely $(1, b^{**}, c^{**}) \notin \mathsf{Span}\{(1, b_{j^*}, c_{j^*}), (1, b'_{j^*}, c'_{j^*})\}$.

We bound the probability that, conversely, $(1, b^{**}, c^{**}) \in \mathsf{Span}\{(1, b_{j^*}, c_{j^*}), (1, b'_{j^*}, c'_{j^*})\}$. Recall the following facts:

- $(b^{**}, c^{**}) = \mathsf{F}_{bc}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$.
- The value of $\mathsf{F}_{bc}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$ is uniformly distributed over $\mathbb{Z}_p^{*2}$ upon the query $\mathsf{H}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$.
- $\mathsf{H}(\mathbf{C}^{**}, \mathbf{D}^{**}, M^{\dagger **}, \bar{R}^{**}, \bar{S}^{**}, \bar{X}^{**}, \pi^{**}, \mu^{**})$ is a crucial query.
- In Case (ii), the first crucial query occurs after (b'_{j^*}, c'_{j^*}) has been fixed.

Since $(b_{j^*}, c_{j^*}) \neq (b'_{j^*}, c'_{j^*})$, the two vectors $(1, b_{j^*}, c_{j^*})$ and $(1, b'_{j^*}, c'_{j^*})$ span a 2-dimensional subspace of $\mathbb{Z}_p^3$. For a uniformly random $(b^{**}, c^{**}) \in \mathbb{Z}_p^{*2}$, the vector $(1, b^{**}, c^{**})$ lies in this subspace with probability at most $1/(p-1)$. By the union bound over all possible crucial query positions, the probability that $(1, b^{**}, c^{**}) \in \mathsf{Span}\{(1, b_{j^*}, c_{j^*}), (1, b'_{j^*}, c'_{j^*})\}$ is at most $q/(p-1)$. Thus, in Case (ii) B successfully obtains one more equation except with probability at most $q/(p-1)$.

Let us put the previous bounds together. B wins the $\mathsf{AOMP}[3q_s]$ game with probability at least $\varepsilon_1(\varepsilon_1/q - 1/p) - q/(p-1)$. Recall that $\varepsilon_1 \geq \varepsilon_{\mathrm{bs},2} - O(q^2)/p$. Consequently, B wins with probability at least $\varepsilon_{\mathrm{bs},2}^2/q - O(q)/p$. Combining with Lemma 5 completes the proof. □

C Proof of Encrypted-Message Blindness

Proof of Theorem 6. Assume A is an algorithm that $(\tau, \varepsilon_{\mathrm{embs}})$-breaks the encrypted-message blindness of EMBS. We will build a series of game hops. Let G_0 be the original $\mathsf{EMBLIND}_{\mathsf{EMBS}}(\mathsf{A})$ game. Random oracles are implemented by lazy sampling. Let ε_i denote the probability that A wins G_i. We have $\varepsilon_{\mathrm{embs}} = \varepsilon_0$.

G_1: Switch to binding mode. In this game, we switch crs (for cmx) to the binding mode, together with an extraction key td. By the distribution of crs (Section 5.3), we have $\varepsilon_1 \geq \varepsilon_0 - 3\varepsilon_{\mathrm{ddh}}$.

G_2: Program fresh H values in User$_1$. Instead of querying $\mathsf{H}(\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_{\mathbf{\Sigma}})$, this game uniformly samples $\bar{\mathsf{ch}} \leftarrow \mathbb{Z}_p$ and programs $\mathsf{H}[\mathsf{pk}, \mathbf{C}, \{\mathsf{ct}'_i\}_{i\in[3]}, \bar{\mathsf{cm}}_{\mathbf{\Sigma}}] := \bar{\mathsf{ch}}$. This game differs from the preceding game only if this programming overwrites an existing query. Observe that the probability of any possible input of H in User$_1$ is bounded by $1/p$. It hits an existing query with probability at most $(q_h+1)/p$. Since there are two sessions, we have $\varepsilon_2 \geq \varepsilon_1 - 2(q_h+1)/p$.

G_3: Uniformly sample ch in User$_1$. In the preceding game, since $\bar{\mathsf{ch}}$ is freshly sampled from $\mathbb{Z}_p$, $\mathsf{ch} := \mathsf{BChal}^{-1}(\bar{\mathsf{ch}}, \bar{\rho}_{\mathbf{\Sigma}})$ is also uniformly distributed over $\mathbb{Z}_p$. We can let this game sample $\mathsf{ch} \leftarrow \mathbb{Z}_p$ and compute $\bar{\mathsf{ch}} := \mathsf{BChal}(\mathsf{ch}, \bar{\rho}_{\mathbf{\Sigma}})$. We have $\varepsilon_3 = \varepsilon_2$.

G_4: Defer most operations to User$_2$. Note that generating $\mathsf{ch} \leftarrow \mathbb{Z}_p$ is enough for User$_1$ to return. We let this game defer all the other operations to User$_2$. This game differs from the preceding game only if A makes any one of the two H queries made by the user oracles before the two sessions are completed. Since the inputs of these queries are distributed uniformly over a set of cardinality at least p and independently of ch, we have $\varepsilon_4 \geq \varepsilon_3 - 2q_h/p$.

G_5: **Compute fresh transcript for stm.** By the perfect blindability of Σ, in $\textsc{User}_2$, the game brute-forces a witness of stm and directly computes a valid transcript $(\bar{\mathsf{cm}}, \bar{\mathsf{ch}}, \bar{\mathsf{rs}})$ under stm. This game differs from the previous game only if stm does not have a witness. If so, then the statement $\mathsf{stm}^\dagger$ proven by the signer does not have a witness, either. By the soundness of Σ', if $\mathsf{stm}^\dagger$ does not have a witness, then A can generate a valid proof with probability at most q_h/p. Therefore, $\varepsilon_5 \geq \varepsilon_4 - q_h/p$. Note that this game becomes inefficient.

G_6: **Compute fresh transcript for stm'.** By the perfect translatability of Σ, in $\textsc{User}_2$, the game brute-forces a witness of stm' and directly computes a valid transcript $(\bar{\mathsf{cm}}, \bar{\mathsf{ch}}, \bar{\mathsf{rs}}')$ under stm'. Note that by Definition 14, TStm is required to preserve the validity of statements. If stm has a witness, then stm' must also have a witness. Therefore, this game is identical to the preceding game, and $\varepsilon_6 = \varepsilon_5$.

G_7: **Use SHVZK simulator.** The game now computes $(\bar{\mathsf{cm}}, \bar{\mathsf{rs}}') \leftarrow \mathsf{Sim}(\mathsf{stm}', \bar{\mathsf{ch}})$. By the perfect SHVZK, this game is identical to the preceding game, and $\varepsilon_7 = \varepsilon_6$. Note that this game becomes efficient again.

G_8: **Compute $\{\mathsf{ct}'_i\}_{i\in[3]}$ as fresh encryptions.** The game extracts $\mathbf{X}$ using td from cmx and computes $\{\mathsf{ct}'_i\}_{i\in[3]}$ as fresh encryptions of $\mathbf{X}$, $\mathbf{X}$, O, respectively. Thanks to G_6, $\textsc{User}_2$ does not reference the rerandomization factors $\{r_i\}_{i\in[3]}$. By the rerandomizability of RE, the game differs from the previous one only if $\{\mathsf{ct}_i\}_{i\in[3]}$ are not encryptions of $\mathbf{X}$, $\mathbf{X}$, O, respectively. If so, $\mathsf{stm}^\dagger$ does not have a witness. By the soundness of Σ', A can generate a valid proof π with probability at most q_h/p. Therefore, $\varepsilon_8 \geq \varepsilon_7 - q_h/p$.

G_9: **Compute $\mathbf{C}$ as encryption of O.** For each session j, this game computes $\mathbf{C}_j \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{O}; \rho)$ as an encryption of $\mathbf{O}$ instead of $\mathbf{M}_j$, where $\mathbf{O}$ denotes the zero vector in $\mathbb{G}^3$. Note that $\textsc{User}_2$ uses the SHVZK simulator rather than running TResp. Consequently, the game does not reference the randomness ρ except for generating $\mathbf{C}_j$. Therefore, we can reduce the distinguishing of this game and the preceding game to the CPA security of PKE. Consequently, $\varepsilon_9 \geq \varepsilon_8 - 2\varepsilon_{\mathrm{ddh}}$.

Finally, we note that for each session j, $\textsc{User}_2$ generates $\mathbf{C}_j$ as an encryption of $\mathbf{O}$, $\{\mathsf{ct}'_i\}_{i\in[3]}$ as fresh encryptions of $\mathbf{X}$, $\mathbf{X}$, O, respectively, and $(\bar{\mathsf{cm}}, \bar{\mathsf{rs}}')$ using the SHVZK simulator. Whether the encryption-signature pairs are swapped does not affect the view of A, so $\varepsilon_9 = 1/2$. □

D Proof of UF-DMA

We will use the mCPA security of RE. In the mCPA security game mCPA^b, A instead outputs two lists of messages $(\{\mathbf{M}_{0,i}\}_{i\in[q]^+}, \{\mathbf{M}_{1,i}\}_{i\in[q]^+})$. By the self-reduction of DDH, RE is $(\tau, 3\varepsilon_{\mathrm{ddh}})$-mCPA secure if DDH is $(\tau', \varepsilon_{\mathrm{ddh}})$-hard for $\tau' = \tau + O(q)$.

Proof. Assume A is an adversary that $(\tau, \varepsilon_{\mathrm{embs}})$-breaks the $\mathsf{UF\text{-}DMA}$ of EMBS. Without loss of generality, we assume that $q_s \geq 2$ and A uses all the q_s queries. This can be ensured by a wrapper which, after A terminates, uses the remaining $\textsc{Sign}_1$ queries on distinct messages. Without loss of generality, we assume that $\mathbf{M}^* := \mathsf{Dec}(\mathsf{dk}, \mathsf{ct}^*)$ is the message of some incomplete signing session. Given any winning adversary A, this can be ensured by a wrapper: after A outputs its forgery $(\mathsf{ct}^*, \sigma^*)$, if $\mathbf{M}^* := \mathsf{Dec}(\mathsf{dk}, \mathsf{ct}^*)$ has not appeared as the message of any $\textsc{Sign}_1$ query, the wrapper issues one additional $\textsc{Sign}_1(\mathbf{M}^*)$ query before returning the forgery. This is feasible since A receives dk as input in $\mathsf{UF\text{-}DMA}$. The extra query does not complete any session and leaves $\mathcal{Q}$ unchanged, so the forgery remains valid. This introduces at most one additional $\textsc{Sign}_1$ query, which is absorbed into the $O(\cdot)$ bounds. Let ε_i denote the probability that A wins G_i. Let G_0 be the original $\mathsf{UF\text{-}DMA}_{\mathsf{EMBS}}(\mathsf{A})$ game, so $\varepsilon_0 = \varepsilon_{\mathrm{embs}}$.

G_1. We modify G_0 to obtain G_1 as follows:
- The game samples each $\mathsf{ek}_{\mathsf{RE},i}$ with the corresponding decryption key $\mathsf{dk}_{\mathsf{RE},i}$.
- The game samples (U, V) that is linearly independent of (G, H).
- $\textsc{Sign}_2$ simulates the non-interactive proof π using NISim, which samples ch, runs the SHVZK simulator $\mathsf{Sim}(\mathsf{stm}^\dagger, \mathsf{ch})$, and programs H_π.

The first change does not affect the view of A. The second change introduces a statistical difference of $1/p$. The third change makes the game differ from the preceding only if the programming overwrites an existing query. By the union bound, it introduces an additive loss of $O(q^2)/p$. Hence, we have $\varepsilon_1 \geq \varepsilon_0 - O(q^2)/p$.

G_2: Replace the winning condition about $\mathcal{Q}$. Let $\mathbf{M}_j$ be the message queried to $\textsc{Sign}_1$ in session j. More generally, for each session $j \in [q_s]^+$, we use subscript j to denote the session-specific variables produced by $\textsc{Sign}_1$ in session j: $\mathsf{ct}_{0,j}$, $\mathsf{ct}_{1,j}$, $\mathsf{ct}_{2,j}$ for the ciphertexts, $\mathbf{Z}_{0,j}$, $\mathbf{Z}_{1,j}$ for the encoded values, and $z_{2,j}$ for the scalar satisfying $\mathbf{Z}_{2,j} = z_{2,j}G$. At the beginning, the game samples $j^* \leftarrow [q_s]^+$. Define $\mathcal{J}$ to be $[q_s]^+ \setminus \{j^*\}$. Let $\mathbf{M}^* = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*)$. At the end, the winning condition $\mathbf{M}^* \notin \mathcal{Q}$ is replaced with $\mathbf{M}^* \notin \{\mathbf{M}_j\}_{j\in\mathcal{J}}$.

By the WLOG assumption, $\mathbf{M}^*$ is the message of some incomplete session. Since the $\textsc{Sign}_1$ oracle enforces distinct messages, this session is unique. A wins in this game if and only if $\mathbf{M}^* = \mathbf{M}_{j^*}$, i.e., j^* equals that unique session. Since j^* is sampled uniformly from $[q_s]^+$ independently of the view of A, this holds with probability $1/q_s$. Thus, we have $\varepsilon_2 \geq \varepsilon_1/q_s$.

G_3: Generate ct_{0,j^*}, ct_{1,j^*} as encryptions of O. In this game, $\textsc{Sign}_1$ generates ct_{0,j^*} and ct_{1,j^*} as encryptions of O rather than $\mathbf{X}$ in session j^*.

Due to the winning condition introduced in G_2, the adversary only wins if session j^* is incomplete, so we only need to consider $\textsc{Sign}_1$ (we can let the game abort if session j^* proceeds to $\textsc{Sign}_2$, which does not reduce the winning probability). Although changing ct_{0,j^*} or ct_{1,j^*} may render statements $\mathcal{S}_{\mathsf{ct}_0}$ or $\mathcal{S}_=$ false (in particular, $\mathcal{S}_=$ fails in the intermediate step where only one of the two has been changed), the oracle $\textsc{Sign}_1$ can still faithfully run $\Sigma.\mathsf{BCom}$: by our construction, $\Sigma.\mathsf{BCom}(\mathsf{stm}, \mathsf{wit})$ uses only the bit b from the witness $\mathsf{wit} = (b, \mathbf{w})$, not the vector $\mathbf{w}$. Hence $\Sigma.\mathsf{BCom}$ executes correctly even when no valid $\mathbf{w}$ exists.

By two applications of the CPA security of RE, we have $\varepsilon_3 \geq \varepsilon_2 - 2\varepsilon_{\mathrm{ddh}}$.

G_4: Generate crs and crs' in binding mode. This game samples crs and crs' as uniformly random rank-1 matrices, setting CM to the binding mode. We have $\varepsilon_4 \geq \varepsilon_3 - 4\varepsilon_{\mathrm{ddh}}$.

G_5: Introduce forgery check about $\mathcal{P}$. Define $\mathcal{P} := \{\mathbf{X} + w\mathbf{M}_j\}_{j\in\mathcal{J}} = \{\mathbf{X}\}$. At the end, parse σ^* into $(\{\mathsf{ct}_i^*\}_{i\in[3]}, \bar{\mathsf{cm}}^*, \bar{\mathsf{rs}}^*)$. Let pt_0^* be $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},0}, \mathsf{ct}_0^*)$. This game checks an additional winning condition: it is required that $\mathsf{pt}_0^* \in \mathcal{P}$.

We need to bound the probability that $\mathsf{pt}_0^* \neq \mathbf{X}$. Note that (G, H, U, V) is not a DH-tuple (i.e., statement $\mathcal{S}_{\mathrm{dh}}$ is false), and CM is in the binding mode. If $\mathsf{pt}_0^* \neq \mathbf{X}$, then statement $\mathcal{S}_{\mathsf{ct}_0}$ must be false, and hence $\mathsf{stm}^* = \mathsf{FormatS}(\mathsf{pk}, \mathsf{ek}, \mathbf{C}^*, \{\mathsf{ct}_i^*\}_{i\in[3]})$ has no witness. By the soundness of Σ, A can generate such a forgery with probability at most q/p. Hence, $\varepsilon_5 \geq \varepsilon_4 - q/p$.

G_5^*: The dual game. We modify G_5 into G_5^* as follows:
- In the key generation, the game sets $w := 1$ instead of 0.
- $\textsc{Sign}_1$ still sets $\mathbf{Z}_0 = \mathbf{Z}_1 = \mathbf{X}$, and generates ct_0, ct_1 as encryptions of $\mathbf{Z}_0 + w\mathbf{M} = \mathbf{X} + \mathbf{M}$ and $\mathbf{Z}_1 + w\mathbf{M} = \mathbf{X} + \mathbf{M}$, respectively. Namely, $\textsc{Sign}_1$ faithfully follows algorithm Sign_1 with $w = 1$.
- $\mathcal{P}$ is changed to $\{\mathbf{X} + w\mathbf{M}_j\}_{j\in\mathcal{J}} = \{\mathbf{X} + \mathbf{M}_j\}_{j\in\mathcal{J}}$.

Let $\mathbf{M}^* = \mathsf{Dec}(\mathsf{dk}, \mathbf{C}^*)$. As (G, H, U, V) is not a DH-tuple, statement $\mathcal{S}_{\mathrm{dh}}$ is false. As CM is in the binding mode, if $\mathsf{pt}_0^* \neq \mathbf{X} + \mathbf{M}^*$, then statement $\mathcal{S}_{\mathsf{ct}_0}$ must be false, and the statement $\mathsf{FormatS}(\mathsf{pk},$

$\mathsf{ek}, \mathbf{C}^*, \{\mathsf{ct}_i^*\}_{i\in[3]})$ does not have a witness. By the soundness of Σ, A can generate such a forgery with probability at most q/p.

Otherwise, assume that $\mathsf{pt}_0^* = \mathbf{X}+\mathbf{M}^*$. The forgery check about $\mathcal{P}$ requires that $\mathbf{M}^* \in \{\mathbf{M}_j\}_{j\in\mathcal{J}}$. This contradicts the forgery check introduced in G_2. Therefore, we have $\varepsilon_5^* \le q/p$.

The proof is completed by the following lemma:

Lemma 9. $\varepsilon_5 - \varepsilon_5^* \le O(\log q_s)(\varepsilon_{\mathrm{ddh}} + q/p)$

□

Proof of Lemma 9. We proceed to define a sequence of game hops, such that no matter whether we start from G_5 or G_5^*, these game hops bring us to the same intermediate game G_I. We describe the game hops from G_5.

G_6: Generate (U,V) with witness. This game generates $(U,V) := u(G,H)$ for $u \leftarrow \mathbb{Z}_p$. Now statement $\mathcal{S}_{\mathrm{dh}}$ is true. We have $\varepsilon_6 \ge \varepsilon_5 - \varepsilon_{\mathrm{ddh}} - 1/p$.

G_7: Use witness for $\mathcal{S}_{\mathrm{dh}}$. This game uses the witness for $\mathcal{S}_{\mathrm{dh}}$ (i.e., u) instead of $\mathcal{S}_{\mathsf{ct}_0}$ to perform Σ. By the perfect SHVZK of Σ, we have $\varepsilon_7 = \varepsilon_6$.

G_8: Generate crs in hiding mode. This game samples crs as a uniformly random rank-4 matrix over $\mathbb{G}$. Now CM is in the hiding mode. We have $\varepsilon_8 \ge \varepsilon_7 - 3\varepsilon_{\mathrm{ddh}} - O(1)/p$.

Partitioning.

Let N be the smallest even number such that $2^N \ge q_s$. Every $j \in [q_s]$ is represented by N bits. We will describe a partitioning loop of N steps. Let $k[1] = k \mod 2$. Let $j|_k$ denote the k-bit prefix, i.e. the least significant k bits of j. For any $b \in \{0,1\}$, let $\bar{b}$ denote $1-b$.

At the beginning of the k-th step, we have $\mathbf{Z}_{0,j} = \mathbf{Z}_{1,j} = \mathsf{F}_k(j|_k)$ for $j \in \mathcal{J}$, and $\mathcal{P}$ is defined as $\{\mathsf{F}_k(j|_k) + w\mathsf{M}_j\}_{j\in\mathcal{J}}$ (we have $w = 0$ if the game hops start from G_5, and $w = 1$ if they start from G_5^*), where F_k is a random function on k-bit inputs taking values in $\mathbb{G}^3$. In particular, when $k = 0$, $\mathsf{F}_k(j|_k) = \mathsf{F}_k(\emptyset)$ is the uniformly random secret key $\mathbf{X}$. The forgery check about $\mathcal{P}$ requires $\mathsf{pt}_{k[1]}^* := \mathsf{Dec}(\mathsf{dk}_{\mathsf{RE},k[1]}, \mathsf{ct}_{k[1]}^*) \in \mathcal{P}$. When $k = 0$, $k[1] = 0$ following the previous games.

For $k \in [N]$, we describe the game hops for the k-th step.

$\mathsf{G}_{8.k.0}$: Beginning of the k-th step. This game is identical to G_8 if $k = 0$, and to $\mathsf{G}_{8.(k-1).15}$ if $k \ge 1$.

$\mathsf{G}_{8.k.1}$: Sample β and set $y := \bar{\beta}$. This game samples $\beta \leftarrow \{0,1\}$ and sets $y := \bar{\beta}$. As crs is in the hiding mode, we have $\varepsilon_{8.k.1} = \varepsilon_{8.k.0}$.

$\mathsf{G}_{8.k.2}$: Set $z_{2,j} := j[k+1]$. Recall that the signing oracles use the witness for statement $\mathcal{S}_=$ rather than $\mathcal{S}_{\mathsf{ct}_2}$, so they can perform correctly with $\mathcal{S}_{\mathsf{ct}_2}$ being false. This game sets $z_{2,j} := j[k+1]$ for every session $j \in \mathcal{J}$. Note that $\mathsf{dk}_{\mathsf{RE},2}$ and $\zeta_{2,j}$ are not referenced (except when using $\zeta_{2,j}$ to encrypt Z_2). Hence, the distinguishing of this game and the preceding game can be reduced to the mCPA security of RE. Therefore, we have $\varepsilon_{8.k.2} \ge \varepsilon_{8.k.1} - 3\varepsilon_{\mathrm{ddh}}$.

$\mathsf{G}_{8.k.3}$: Abort if $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},2}, \mathsf{ct}_2^*) \neq \beta G$. This game introduces an additional forgery check: the game aborts if $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},2}, \mathsf{ct}_2^*) \neq \beta G$. Since β is only used to set y, and y is perfectly hidden by the commitment, β is independent of the view of A. Hence, we have $\varepsilon_{8.k.3} = \varepsilon_{8.k.2}/2$.

$\mathsf{G}_{8.k.4}$**: Use witness for $\mathcal{S}_{\mathsf{ct},2}$ if $\beta \neq j[k+1]$.** If $\beta \neq j[k+1]$, then $z_{2,j}$ is set to be $j[k+1] = \bar{\beta} = y$. Therefore, the statement $\mathcal{S}_{\mathsf{ct},2}$ is true. This game uses the witness of $\mathcal{S}_{\mathsf{ct}_2}$, to perform Σ for $j \in \mathcal{J}$ with $j[k+1] \neq \beta$. The witness consists of the randomness $\vec{\eta}_y$ and ζ_2 for the commitment and the encryption, respectively. By the perfect SHVZK of Σ, we have $\varepsilon_{8.k.4} = \varepsilon_{8.k.3}$.

$\mathsf{G}_{8.k.5}$**: Change $\mathbf{Z}_{\overline{k[1]},j}$ if $\beta \neq j[k+1]$.** Let F'_k be a random function on k-bit inputs and independent of F_k. In this game, $\textsc{Sign}_1$ sets $\mathbf{Z}_{\overline{k[1]},j}$ by the following:

$$\mathbf{Z}_{\overline{k[1]},j} := \begin{cases} \mathsf{F}_k(j|_k), & \beta = j[k+1] \\ \mathsf{F}'_k(j|_k), & \beta \neq j[k+1] \end{cases}, \quad j \in \mathcal{J}.$$

Namely, the game changes $\mathbf{Z}_{\overline{k[1]},j}$ for $j \in \mathcal{J}$ with $\beta \neq j[k+1]$, using F'_k rather than F_k.

Note that by the change in the preceding game, $\beta \neq j[k+1]$ means that the signing oracles use the witness of statement $\mathcal{S}_{\mathsf{ct}_2}$, so they can perform correctly with $\mathcal{S}_=$ being false. Also note that the forgery check about $\mathcal{P}$ is on $\mathsf{pt}^*_{k[1]} = \mathsf{Dec}(\mathsf{dk}_{\mathsf{RE},k[1]}, \mathsf{ct}^*_{k[1]})$, and $\mathsf{dk}_{\mathsf{RE},\overline{k[1]}}$ is never referenced. Therefore, by the mCPA security of RE, we have $\varepsilon_{8.k.5} \geq \varepsilon_{8.k.4} - 3\varepsilon_{\mathrm{ddh}}$.

$\mathsf{G}_{8.k.6}$**: Generate crs in binding mode.** This game samples crs as a uniformly random rank-1 matrix. Now CM is in the binding mode. We have $\varepsilon_{8.k.6} \geq \varepsilon_{8.k.5} - 3\varepsilon_{\mathrm{ddh}} - O(1)/p$.

$\mathsf{G}_{8.k.7}$**: Change forgery check about $\mathcal{P}$.** This game checks whether $\mathsf{pt}^*_{\overline{k[1]}} := \mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},\overline{k[1]}}, \mathsf{ct}^*_{\overline{k[1]}}) \in \mathcal{P}$ instead of $\mathsf{pt}^*_{k[1]}$. Since $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},2}, \mathsf{ct}^*_2) \neq \beta G = yG$ (due to the change in $\mathsf{G}_{8.k.3}$), and now CM is in the binding mode, the statement $\mathcal{S}_{\mathsf{ct}_2}$ is false for the forgery. Thus, by the soundness of Σ, the statement $\mathcal{S}_=$ needs to be true except with probability q_h/p. Consequently, we have $\mathsf{pt}^*_0 = \mathsf{pt}^*_1$, and changing the forgery check does not change the result. In conclusion, we have $\varepsilon_{8.k.7} \geq \varepsilon_{8.k.6} - q_h/p$.

$\mathsf{G}_{8.k.8}$**: Generate crs in hiding mode.** This game samples crs as a uniformly rank-4 matrix, which switches CM to the hiding mode. We have $\varepsilon_{8.k.8} \geq \varepsilon_{8.k.7} - 3\varepsilon_{\mathrm{ddh}} - O(1)/p$.

$\mathsf{G}_{8.k.9}$**: Change $\mathbf{Z}_{k[1],j}$ if $\beta \neq j[k+1]$.** In this game, $\textsc{Sign}_1$ sets $\mathbf{Z}_{k[1],j}$ by the following:

$$\mathbf{Z}_{k[1],j} := \begin{cases} \mathsf{F}_k(j|_k), & \beta = j[k+1] \\ \mathsf{F}'_k(j|_k), & \beta \neq j[k+1] \end{cases}, \quad j \in \mathcal{J}.$$

Namely, the game changes $\mathbf{Z}_{k[1],j}$ for $j \in \mathcal{J}$ with $\beta \neq j[k+1]$, using F'_k rather than F_k.

Note that by the change in $\mathsf{G}_{8.k.4}$, $\beta \neq j[k+1]$ means that the signing oracles use the witness of statement $\mathcal{S}_{\mathsf{ct}_2}$, so they can perform correctly with $\mathcal{S}_=$ being false. Also note that the forgery check about $\mathcal{P}$ is on $\mathsf{pt}^*_{\overline{k[1]}} = \mathsf{Dec}(\mathsf{dk}_{\mathsf{RE},\overline{k[1]}}, \mathsf{ct}^*_{\overline{k[1]}})$ (by $\mathsf{G}_{8.k.7}$), and $\mathsf{dk}_{\mathsf{RE},k[1]}$ is never referenced. Therefore, by the mCPA security of RE, we have $\varepsilon_{8.k.9} \geq \varepsilon_{8.k.8} - 3\varepsilon_{\mathrm{ddh}}$.

$\mathsf{G}_{8.k.10}$**: Use witness for $\mathcal{S}_=$ if $\beta \neq j[k+1]$.** By $\mathsf{G}_{8.k.5}$, $\mathsf{G}_{8.k.9}$, both $\mathbf{Z}_{0,j}$ and $\mathbf{Z}_{1,j}$ are set to be $\mathsf{F}'_k(j|_k)$ if $\beta \neq j[k+1], j \in \mathcal{J}$, so the statement $\mathcal{S}_=$ becomes true again. We let the game use the witness for $\mathcal{S}_=$ to perform Σ. By the perfect SHVZK, we have $\varepsilon_{8.k.10} = \varepsilon_{8.k.9}$.

$\mathsf{G}_{8.k.11}$**: Rewrite $\mathbf{Z}_{0,j}$, $\mathbf{Z}_{1,j}$ as $\mathsf{F}_{k+1}(j|_{k+1})$.** Note that for $j \in \mathcal{J}$, $\mathbf{Z}_{0,j}$ and $\mathbf{Z}_{1,j}$ are set to be

$$\begin{cases} \mathsf{F}_k(j|_k), & j[k+1] = \beta \\ \mathsf{F}'_k(j|_k), & j[k+1] = \bar{\beta} \end{cases}$$

This can be rewritten as $\mathsf{F}_{k+1}(j|_{k+1})$, where F_{k+1} is a random function on $(k+1)$-bit inputs taking values in $\mathbb{G}^3$. We have $\varepsilon_{8.k.11} = \varepsilon_{8.k.10}$.

$\mathsf{G}_{8.k.12}$**: Change the definition of $\mathcal{P}$.** In the preceding game, $\mathcal{P}$ is defined as

$$\begin{aligned} \mathcal{P} &= \{\mathsf{F}_k(j|_k) + w\mathsf{M}_j\}_{j\in\mathcal{J}} \\ &= \underbrace{\{\mathsf{F}_k(j|_k) + w\mathsf{M}_j \mid j \in \mathcal{J}, j[k+1] = \beta\}}_{=:\mathcal{S}_{\mathrm{both}}} \cup \underbrace{\{\mathsf{F}_k(j|_k) + w\mathsf{M}_j \mid j \in \mathcal{J}, j[k+1] = \bar{\beta}]\}}_{=:\mathcal{S}_{k.11}} \end{aligned}$$

In this game, $\mathcal{P}$ is defined as

$$\begin{aligned} \mathcal{P} &= \{\mathsf{F}_{k+1}(j|_{k+1}) + w\mathsf{M}_j\}_{j\in\mathcal{J}} \\ &= \underbrace{\{\mathsf{F}_k(j|_k) + w\mathsf{M}_j \mid j \in \mathcal{J}, j[k+1] = \beta\}}_{=\mathcal{S}_{\mathrm{both}}} \cup \underbrace{\{\mathsf{F}'_k(j|_k) + w\mathsf{M}_j \mid j \in \mathcal{J}, j[k+1] = \bar{\beta}]\}}_{=:\mathcal{S}_{k.12}}. \end{aligned}$$

We define an intermediate game $\mathsf{G}_{8.k.11'}$ where $\mathcal{P}$ is defined to be $\mathcal{S}_{\mathrm{both}}$.

$\mathsf{G}_{8.k.11'}$ differs from the preceding game only if A returns a forgery with $\mathsf{pt}^*_{\overline{k[1]}} \in \mathcal{S}_{k.11} \setminus \mathcal{S}_{\mathrm{both}}$, which the preceding game accepts while G' does not. Observe that

$$\begin{aligned} \mathcal{S}_{k.11} \setminus \mathcal{S}_{\mathrm{both}} &= \{\mathsf{F}_k(j|_k) + w\mathsf{M}_j \mid j \in \mathcal{J},\ j[k+1] = \bar{\beta}, \quad \forall j' \in \mathcal{J} : j'|_{k+1} \neq (\beta, j|_k)\} \\ &= \{\mathsf{F}_k(s) + w\mathbf{M}_j \mid (\bar{\beta}, s) \in \mathcal{J}_{k+1}, (\beta, s) \notin \mathcal{J}_{k+1}\}, \end{aligned}$$

where $\mathcal{J}_{k+1}$ is the set of $k+1$-bit prefixes of all strings in $\mathcal{J}$. Note that for any k-bit s, $\mathsf{F}_k(s)$ is only referenced in session $j \in \mathcal{J}$ such that the $(k+1)$-bit prefix of j is (β, s). Thus, the F_k values in $\mathcal{S}_{k.11} \setminus \mathcal{S}_{\mathrm{both}}$ are never referenced and perfectly hidden from A. We thus have $\varepsilon_{8.k.11'} \geq \varepsilon_{8.k.11} - q_s/p$.

$\mathsf{G}_{8.k.12}$ accepts more values of $\mathsf{pt}^*_{\overline{k[1]}}$ than $\mathsf{G}_{8.k.11'}$, so $\varepsilon_{8.k.12} \geq \varepsilon_{8.k.11'}$. Therefore, we have $\varepsilon_{8.k.12} \geq \varepsilon_{8.k.11} - q_s/p$.

$\mathsf{G}_{8.k.13}$**: Do not abort if** $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},2}, \mathsf{ct}^*_2) \neq \beta G$**.** In this game, the aborting condition $\mathsf{RE.Dec}(\mathsf{dk}_{\mathsf{RE},2}, \mathsf{ct}^*_2) \neq \beta G$ is removed. Thanks to $\mathsf{G}_{8.k.10}$, the signing oracles use witness for $\mathcal{S}_=$ no matter $j[k+1] = \beta$ or not. Thanks to $\mathsf{G}_{8.k.11}$, the assignment of $\mathbf{Z}_{0,j}$ and $\mathbf{Z}_{1,j}$ does not need to check whether $j[k+1] = \beta$. Therefore, the game does not reference β except setting y. Since CM is in the hiding mode, β is perfectly hidden. Therefore, we have $\varepsilon_{8.k.13} = 2\varepsilon_{8.k.12}$.

$\mathsf{G}_{8.k.14}$**: Set** $z_{2,j} := 0$**.** This game sets $z_{2,j} := 0$ for every $j \in \mathcal{J}$. Since the signing oracles use the witness for $\mathcal{S}_=$, the randomness for $\mathsf{ct}_{2,j}$ is not referenced. By the mCPA security of RE, we have $\varepsilon_{8.k.14} \geq \varepsilon_{8.k.13} - 3\varepsilon_{\mathrm{ddh}}$.

$\mathsf{G}_{8.k.15}$**: Do not sample** β**.** This game does not sample β and sets $y := 0$. Since CM is in the hiding mode, we have $\varepsilon_{8.k.15} = \varepsilon_{8.k.14}$.

End of the partitioning loop. After the partitioning loop, we have $\mathbf{Z}_{0,j} = \mathbf{Z}_{1,j} = \mathsf{F}(j)$ for $j \in \mathcal{J}$ for a random function F taking values in $\mathbb{G}^3$. $\mathcal{P}$ is defined as $\{\mathsf{F}(j) + w\mathsf{M}_j\}_{j\in\mathcal{J}}$. The forgery check about $\mathcal{P}$ is on pt^*_0.

The intermediate game. Now, we have arrived at the intermediate game G_I. In session j^*, ct_{0,j^*} and ct_{1,j^*} encrypt O. We have $\mathbf{Z}_{0,j} = \mathbf{Z}_{1,j} = \mathsf{F}(j)$ for $j \in \mathcal{J}$ for a random function F. $\mathcal{P}$ is defined as $\{\mathsf{F}(j)\}_{j\in\mathcal{J}}$. The forgery check about $\mathcal{P}$ is on pt_0^*. Equivalently, in this game, $\mathsf{ct}_{0,j}$ and $\mathsf{ct}_{1,j}$ encrypt a uniformly random plaintext pt_j for each $j \in \mathcal{J}$, and $\mathcal{P}$ is defined as $\{\mathsf{pt}_j\}_{j\in\mathcal{J}}$.

The same sequence of game hops can be applied to G_5^*. In session j^*, ct_{0,j^*} and ct_{1,j^*} encrypt O. That will end up with a game where $\mathbf{Z}_{0,j} = \mathbf{Z}_{1,j} = \mathsf{F}(j)$ for $j \in \mathcal{J}$ for a random function F. $\mathcal{P}$ is defined as $\{\mathsf{F}(j) + \mathbf{M}_j\}_{j\in\mathcal{J}}$. Note that $\mathsf{ct}_{0,j}$ and $\mathsf{ct}_{1,j}$ also encrypt a uniformly random plaintext pt_j in this game, and $\mathcal{P}$ also equals $\{\mathsf{pt}_j\}_{j\in\mathcal{J}}$. We have bridged G_5 and G_5^* using this intermediate game. Putting everything together, we have

$$\varepsilon_5 - \varepsilon_5^* \leq O(\log q_s)(\varepsilon_{\mathrm{ddh}} + q/p).$$

□