Two-Round 2PC ECDSA at the Cost of 1 OLE: Applications to Embedded Cryptocurrency Wallets

Michael Adjedj[*] Constantin Blokh[*] Geoffroy Couteau[†‡] Arik Galansky[§¶]
Antoine Joux[‖] Nikolaos Makriyannis[*]

March 6, 2026

Abstract

We present a novel protocol for two-party ECDSA that achieves two rounds (a single back-and-forth communication) at the cost of a single oblivious linear function evaluation (OLE). In comparison, the previous work of Boneh, Haitner, Lindell, and Segev (EUROCRYPT 2025) achieves two rounds but requires expensive zero-knowledge proofs on top of the OLE. We demonstrate this by proving that in the generic group model, any adversary capable of generating forgeries for our protocol can be transformed into an adversary that finds preimages for the ECDSA message digest function (e.g., the SHA family). Interestingly, our analysis is closely related to, and has ramifications for, the 'presignatures' mode of operation—Canetti, Gennaro, Goldfeder, Makriyannis, and Peled (CCS 2020), Groth and Shoup (EUROCRYPT 2022).

Motivated by applications to embedded cryptocurrency wallets, where a single server maintains distinct, shared public keys with separate clients (i.e., a star-shaped topology), and with the goal of minimizing communication, we instantiate our protocol using Paillier encryption and suitable zero-knowledge proofs. To reduce computational overhead, we thoroughly optimize all components of our protocol under sound cryptographic assumptions, specifically small-exponent variants of RSA-style assumptions.

Finally, we implement our protocol and provide benchmarks. At the 128-bit security level, the signing phase requires approximately 50ms of computation time on a standard linux machine, and 2KB of bandwidth.

[*]Fireblocks. {madjedj, costy, nikos}@fireblocks.com
[†]CNRS, IRIF, Université Paris Cité. couteau@irif.fr
[‡]Work done while the author was working as a consultant for Fireblocks.
[§]ZAMA. arik.galansky@gmail.com
[¶]Work done while the author was working for Fireblocks.
[‖]CISPA Helmholtz Center for Information Security. joux@cispa.de

Contents

1 Introduction **1**
- 1.1 Efficiency requirements for 2-party ECDSA . . . 1
- 1.2 Our Contribution . . . 2
  - 1.2.1 Technical contributions. . . . 3
- 1.3 Comparison to the state-of-the-art . . . 5
  - 1.3.1 Comparison to [BHLS24]. . . . 6
  - 1.3.2 On the original (2-round) version of DKLs18 [DKLs18b]. . . . 8
- 1.4 Organization . . . 8

2 Technical Overview **8**
- 2.1 ECDSA at the Cost of 1 OLE . . . 9
  - 2.1.1 Sketch security Analysis . . . 10
- 2.2 Doubly-Enhanced Existential Unforgeability of ECDSA . . . 12
  - 2.2.1 The generic group model . . . 13
  - 2.2.2 Sketch security proof . . . 13
- 2.3 Implementation using Paillier Encryption and Integer Commitments . . . 15
  - 2.3.1 Efficiency considerations . . . 16
  - 2.3.2 Tough primes optimization . . . 16
  - 2.3.3 Fast and compact variant of Damgård-Fujisaki parameters proof . . . 18
  - 2.3.4 Systematic use of small-exponents . . . 19
  - 2.3.5 Dealing with the cheating subgroup problem . . . 20
  - 2.3.6 Applications to embedded cryptocurrency wallets . . . 21

3 Preliminaries **21**
- 3.1 Notation . . . 21
- 3.2 Standard Lemmas . . . 22
- 3.3 Security Assumptions . . . 23
  - 3.3.1 DCR Variants . . . 24
  - 3.3.2 Damgård-Fujisaki to Strong-RSA reduction . . . 25
- 3.4 Paillier Encryption, Damgård-Fujisaki & Pedersen Commitments . . . 26
- 3.5 Zero-Knowledge Proofs . . . 27
  - 3.5.1 NIZK and the Fiat-Shamir Transform . . . 27
  - 3.5.2 NP Relations . . . 27
- 3.6 The EC-Generic Group Model . . . 28

4 2-Party Threshold ECDSA **28**
- 4.1 Sampling the RSA modulus . . . 29
- 4.2 Non-Interactive Setup . . . 30
- 4.3 Interactive Key-Generation . . . 31
- 4.4 Distributed Signing . . . 33

5 Underlying ZK Protocols **33**
5.1 Small Factors Proof (Π_{fac}) . . . 35
5.1.1 Special Soundness of Π_{fac} . . . 36
5.2 Well-Formed Modulus (Π_{mod}) . . . 36
5.3 Well-Formed Signature (Π_{sig}) . . . 38
5.3.1 Special Soundness of Π_{sig} . . . 38
5.3.2 Partial Straight-Line Extractability of Π_{sig} in the GGM . . . 39
5.4 Discrete Logarithm (Π_{dlog}) . . . 40
5.5 Encrypted Discrete Log (Π_{dlenc}) . . . 41
5.6 Damgård-Fujisaki Parameters (Π_{df}) . . . 43
5.7 Damgård-Fujisaki-*star* Parameters ($\Pi_{\mathsf{df}*}$) . . . 44

6 Security Analysis **44**
6.1 Lazy symbolic interfacing with the GGM and the signing oracle . . . 45
6.1.1 Unique representations . . . 46
6.1.2 Knowledge of exponents . . . 47
6.2 The client is corrupted, single key-generation . . . 47
6.3 The client is corrupted, multiple key-generation . . . 54
6.4 The server is corrupted . . . 55

7 Unforgeability in the Generic Group Model **60**
7.1 The Generic Group Model . . . 61
7.1.1 Symbolic lazy evaluation . . . 62
7.2 Doubly-Enhanced Unforgeability in the GGM . . . 63
7.2.1 Preliminary Facts . . . 64
7.2.2 Simulation . . . 64
7.2.3 Analysis . . . 66
7.2.4 Proof of Proposition 7.10 . . . 68

8 Benchmarks **70**
8.1 Concrete Parameters . . . 70
8.2 Experimental Benchmarks . . . 72

A Analysis of the Assumptions **80**
A.1 The short-exponent indistinguishability assumption . . . 80
A.1.1 Background on the short-exponent discrete logarithm assumption . . . 80
A.1.2 Reduction to SEDL . . . 80
A.2 Factoring tough-RSA numbers compared standard RSA numbers . . . 82

B Symbolic Evaluation to Lazy Evaluation Reduction **83**
B.1 Lazy Evaluation 1 . . . 83
B.2 Lazy Evaluation 2 . . . 84
B.3 Symbolic Evaluation 1 . . . 86
B.4 Symbolic Evaluation 2 . . . 88

C 'Verifiable OLE' from OLE in the ROM **89**

1 Introduction

Threshold signatures enable a group of participants to collectively sign a document such that a subset of the participants can produce a valid signature if and only if it meets the threshold requirement. Threshold cryptographic primitives, of which threshold signatures are perhaps the most prominent example, enhance traditional cryptographic primitives with an additional layer of security by distributing the ability to execute a cryptographic task and sharing the secret data used by the scheme among the participants.

Threshold signatures enjoy several attractive benefits over their non-threshold counterpart: they are more resilient to failures or loss of data, add a layer of decentralization, and in many scenarios, help guarantee that signatures are only issued when a pre-specified policy is met. In the past decades, an important and widespread research effort has been devoted to the design and analysis of efficient threshold signature schemes, and a call for standardizing threshold schemes has recently been announced by the National Institute of Standards and Technology (NIST) [BP23].

The Elliptic Curve Digital Signature Algorithm (ECDSA) is by far the most widely deployed signature scheme and among the most commonly deployed cryptographic primitives of any kind. It is a core component of the Internet infrastructure, and the main cryptographic tool used in distributed ledgers to execute transactions. This makes it a natural target for threshold mechanisms. Unfortunately, unlike "threshold-friendly" signature schemes such as Schnorr [Sch91] and BBS [BBS04], ECDSA has long proven to be challenging to run in a threshold setting. Concretely, given a secret key $x \leftarrow \mathbb{Z}_q$ and a public key $X = g^x \in \mathbb{G}$ over an elliptic curve $\mathbb{G}$ of order q generated by g, an ECDSA signature on a message m is computed as (R, σ) where $R = g^k$ is a public nonce, and $\sigma = (\mathsf{F}(m) + x \cdot r) \cdot k^{-1} \bmod q$, where r denotes the first coordinate of the point R on the curve, and F denotes the ECDSA message digest function (concretely implemented with the SHA family of hash functions). The non-linear structure of σ (particularly, the inversion of k) makes it non-trivial to securely generate in a distributed fashion.

1.1 Efficiency requirements for 2-party ECDSA

ECDSA is used widely, and several companies are deploying commercial solutions for 2-party threshold ECDSA (e.g. Fireblocks [Fir], Fordefi [For], Sodot [Sod], Utila [Uti], Zengo [Zen], to name a few). When designing a protocol aimed for deployment in the real world, it is instructive to evaluate the relevant efficiency metrics. Ideally, a threshold ECDSA protocol should achieve the following efficiency features:

- **Optimal round complexity** for the signing protocol and for distributed key generation. The minimal number of rounds of threshold ECDSA is 2, and any larger round complexity has a significant impact on latency.
- **Low computational overhead**, to allow clients with low-end devices to run the protocol, or servers to run efficiently a large number of instances of the protocol.
- **Low communication overhead**, to minimize bandwidth usage in a WAN setting, but also to enable useful features such as offline signing [BMP22].

We elaborate briefly on these aspects, and refer the reader to Section 1.3 for a more detailed discussion on the relevant efficiency metrics. When deploying a 2-party ECDSA protocol at a large

scale, the bottleneck efficiency of all state-of-the-art solutions is the roundtrip time for serving distant clients, since a single round trip can add a latency ranging from 50-100ms (distant machines in the same country) to more than 300ms (machines in distant countries). An implication of this is that the computational overhead (measured as the runtime ignoring bandwidth costs and latency) matters only insofar as it exceeds significantly the latency of a round-optimal protocol: a computational overhead of 500ms versus 50ms makes a significant difference, but an overhead of 50ms versus 5ms matters much less. Regarding bandwidth, with the large-scale deployment of 2-party ECDSA protocols comes a diversification of use cases and communication scenarios. Clients might want to execute the protocol via an Android or Apple device, where the natural communication API goes through push notifications; or use offline signing, that requires scanning a QR code; or possibly making transactions in large batches, or support other deployment scenarios. Each use case comes with its own constraints (a QR code is 3kB, push notifications are 4-5kB, etc), and achieving a very low communication overhead (few kilobytes) is the most versatile strategy to avoid dismissing possible future communication scenarios ahead-of-time.

Despite the large number of works tackling this challenge, e.g., [Lin17; LN18; GG18; CGGMP20; DKLs24; XAXYC21], to the best of our knowledge, all end-to-end threshold ECDSA protocols currently compromise on at least one of the above efficiency criteria, even in the simplest scenario of two-party threshold ECDSA.

1.2 Our Contribution

In this work, we introduce a new threshold ECDSA protocol in the two-party setting, that we thoroughly optimize to provide the best-possible efficiency features for the relevant efficiency metrics outlined above. Our protocol achieves a desirable sweet spot and strikes a good balance among all efficiency measures, making it an ideal choice for real world deployment. Concretely, for the 128-bit security level, our protocol is

- **Round optimal.** Signing runs in two rounds.
- **Fast.** Our end-to-end implementation generates a signature in 50ms.
- **Communication-efficient.** Our signing protocol communicates only 2KB.

To achieve these unmatched efficiency features, we adopt a strategy that deviates from the approach of using only the most conservative assumptions. Instead, we seek to thoroughly optimize all components of our protocols whenever we can do so under sound cryptographic assumptions backed up by solid arguments (analysis in the generic group model, or security against all known cryptanalytic methods). In particular, our protocols rely on the following assumptions:

- The *doubly-enhanced unforgeability* of ECDSA, a strengthening of the security of ECDSA that has been introduced recently in [CGGMP20; GS22]. We include a formal proof of the doubly-enhanced unforgeability of ECDSA in the generic group model, expanding upon previous security analyses of ECDSA in the generic group model [Bro05; GS22].
- The hardness of factoring *tough RSA moduli* of the form $N = p \cdot q$ where $(p-1)/2, (q-1)/2$ are products of distinct random 256-bit primes.

- The *small-exponent indistinguishability assumption* [Gen00; KK04; CC18], which states that the distributions $\{u^x \bmod N \ : \ x \leftarrow \{0,1\}^{2\ell}\}$ and $\{u^x \bmod N \ : \ x \leftarrow \mathbb{Z}_{2^\nu N}\}$ are indistinguishable, where ℓ is a security parameter (typically $\ell = 128$), ν is a statistical security parameter (we use $\nu = 64$) and N is an RSA modulus defined as above.
- The strong-RSA and DCR assumptions over $\mathbb{Z}_N$ and $\mathbb{Z}_{N^2}$ respectively, where N is defined as above.
- The *straight-line extractability* of the NIZK obtained by compiling Schnorr's proof of knowledge of a discrete logarithm via the Fiat-Shamir transform, and that of our (Fiat-Shamir compiled) zero-knowledge proof (ZKP) of well-formedness of the ECDSA signature. We prove the straight-line extractability of both protocols in the generic group and random oracle models, respectively unconditionally and under the strong-RSA assumption

1.2.1 Technical contributions.

Our two-party threshold ECDSA protocol combines a large number of technical innovations, guided by our aim to track down all efficiency bottlenecks in the protocol. We believe that several of our techniques are of independent interest. In particular, we expect that some of these techniques will become part of the toolbox for designing efficient threshold ECDSA schemes, and can have broader implications. We overview some of our main techniques and their implications below:

Generalizing 'Rerandomized presignatures' [GS22]. The security of our protocol relies on an additional assumption regarding the unforgeability of ECDSA, closely related to the 'presignatures' mode of operation. To elaborate, this is a mode of operation where the random part of the signature—the *nonce*—is computed ahead of time, before the message to be signed is known. In [GS22], the authors show that when the nonce is suitably rerandomized, presignatures do not incur any security loss in the generic group model (GGM) compared to standard ECDSA. In this paper, we strengthen this result by showing that even if the attacker tries multiple rerandomizations (e.g., when rerandomization depends on the output of a random oracle) before deciding which message and nonce to sign, the security loss in the GGM is minimal compared to standard ECDSA—essentially a small multiplicative constant $< 2^5$. This is a key aspect of our analysis, allowing us to achieve a two-round signing protocol without incurring a penalty in computation.

This result is of independent interest and already enjoyed another application: it was used in a subsequent work at CCS 2025 [LLZD25] to solve the long-standing open problem of building a two-round multiparty threshold ECDSA protocol.

Blazing fast RSA modulus generation. We introduce and use a fast algorithm that generates an RSA modulus $N = pq$ where $(p-1)/2$ and $(q-1)/2$ are products of random independent 256-bit primes (we call N a *tough-primes* RSA modulus). In contrast, the standard solution for generating an RSA modulus adopted in most previous works on threshold ECDSA [GG18; CGGMP20] (as well as in other settings such as range proofs [CPP17; CKLR21; CGKR22], blind signatures [KNR24], aggregate signatures [HW18], and many more) is to generate $N = pq$ as a product of *strong primes*, where $(p-1)/2$ and $(q-1)/2$ are also primes. The main conceptual innovation of our algorithm stems from two observations:

- In light of the state-of-the-art cryptanalysis against factoring and discrete logarithms, it appears that the use of a tough-primes RSA modulus provides *identical security guarantees* (compared to a product of strong primes) in a wide variety of contexts (indeed, every context we could think of), including the use of Damgård-Fujisaki strong-RSA-based integer commitments, Paillier encryption, and small-exponent discrete logarithm assumptions.

- Tough-prime RSA moduli are *considerably faster* to sample compared to products of strong primes, with a quadratic heuristic runtime gap. Specifically, in our experimental benchmarks, our implementation samples 1024- and 1536-bit tough-prime moduli in an average time of 74 ms and 198 ms, respectively, compared to 1174 ms and 7200 ms for a product of strong primes. (As a baseline, sampling regular RSA moduli with 1024 and 1536 bits takes 46ms and 141ms in our benchmarks).

We believe that the use of products of strong primes was justified in previous works by the assumption that they are typically generated during a one-time setup. However, in real-world applications, this generation happens whenever two parties execute a distributed key generation protocol. In practice, this can happen quite often over a large-scale system, and the use of strong primes incurs a considerable slowdown. We expect our faster alternative sampler for RSA moduli to provide a secure and highly competitive alternative.

Compact and Fast ZKP for Damgård-Fujisaki parameters. Many threshold ECDSA protocols typically emulate homomorphic computations over the integers using Paillier encryption [Lin17; GG18; CGGMP20; XAXYC21] to compute a signature σ. To avoid devastating overflow attacks [MYG23], it is common to rely on zero-knowledge proofs of knowledge of integer relations, using the standard Damgård-Fujisaki integer commitment scheme [DF02]. However, proving zero-knowledge requires asking the verifier (who samples the Damgård-Fujisaki parameters) to first demonstrate that the parameters are honest. Unfortunately, the best-known zero-knowledge proofs for this statement have soundness $1/2$, and achieving negligible soundness error requires a large number of parallel repetitions, using a lot of communication.

In this work, we explore an alternative approach. The approach is tailored to our setting, but we expect it to be usable in many similar scenarios. At a high level, we use the 'large challenge' variant from [HLM25] of the baseline protocol for proving honest generation of the Damgård-Fujisaki parameters, which allows us to achieve negligible soundness error in a few repetitions. This approach introduces certain technical challenges, which are explained in detail in the technical overview (specifically, the 'cheating subgroup problem'—see Section 2.3.5). Additionally, we improve upon the protocol of [HLM25] by using *small exponents*. Concretely, when computing $t^x \bmod \hat{N}$, where $\hat{N}$ is the RSA modulus and $t \in \mathbb{Z}^*_{\hat{N}}$ is a public base, we select the random exponent x uniformly from a much smaller domain of size $2^{2\ell}$ (for security parameter ℓ), rather than from the full exponent range of bit-length roughly $\log(\hat{N})$. This results in a manyfold improvement in both computation and communication, as detailed in the technical overview.

Systematic use of small exponents. In fact, our use of the small-exponent optimization is ubiquitous throughout the protocol. Almost every exponentiation in an RSA or Paillier group uses a small exponent. Since the computational load scales linearly with the size of the exponent, this approach yields a significant performance improvement, contributing to the highly competitive running times presented in the experimental benchmarks. However, this optimization introduces an

additional security assumption—the small exponent indistinguishability assumption—as mentioned at the beginning of the introduction. In Appendix A, we additionally prove that the small exponent indistinguishability assumption is *equivalent* to a much more well-studied assumption: the hardness of computing discrete logarithms when the exponents are sampled from a small interval (of size $2^{2\ell}$). Our reduction works over arbitrary groups, including hidden-order groups, improving over the result of [KK04] that is restricted to prime-order group.

Additional contributions. As an additional contribution, we show that our protocol is particularly well suited for the *embedded cryptocurrency wallet* use case, where a single server maintains distinct, shared public keys with multiple clients (i.e., a star-shaped topology). The goals in this scenario are: (i) minimizing the server's computational load in both key generation and signing, as the server must perform these tasks repeatedly with many clients; (ii) minimizing the protocol's bandwidth usage, which is crucial when the server is communicating with multiple clients simultaneously; and (iii) minimize the client's running time to enhance the user experience. Our protocol effectively achieves these goals, making it particularly well suited for this use case.

1.3 Comparison to the state-of-the-art

On Table 1, we compare our new protocol with the most efficient threshold ECDSA from the literature. Our protocol compares very favorably to the state of the art and achieves a sweet spot, combining optimal round complexity, low communication complexity, and low computational overhead. The only round-optimal protocol protocol [BHLS24] requires nearly an order of magnitude more computation and incurs almost 2× the communication. Among three-round protocols, those that are less computationally intensive than ours still incur over 57× more communication. Conversely, protocols that are more computationally intensive run 4-5× slower and incur 2-3× more communication. Lastly, the protocol of [Lin17] runs in 4 rounds and achieves 2× smaller communication and 4× faster computation. However, this comes at the cost of a weaker security guarantee, where security is not preserved under arbitrary concurrency (it can be attacked in this model [MYG23]).

Impact of latency on runtime. It is important to note that in deployments of multi-party ECDSA involving geographically dispersed machines, network latency—not computation—dominates overall runtime. For instance, *a single round trip* between relatively close machines (e.g., inside the US) adds 50–100ms [DKLs19], whereas for distant machines, latency can range from 100–260ms—and in some cases, a round trip can take up to 0.5 seconds [DKLs19]. For protocols in the 0-50ms runtime range (which captures the most computation-efficient protocols we are discussing here), network latency from each additional round becomes the dominant cost—adding a single round can exceed the entire computation time. Indicatively, [DKLs19] reports roundtrip times of 90ms for East Coast US-Western Europe and 348ms for India-Belgium

Impact of communication. Communication costs can adversely impact devices with limited communication or bandwidth capabilities. For example, QR codes are limited to 3KB [Iso]; devices relying on QR codes for communication would need to split messages into multiple QR codes for most protocols in Table 1 (the only exception, apart from our protocol, is the non-concurrently

	OLEs	Rounds	Comm. (KB)	Comp. (ms)	Concurrent security	Code repo
[Lin17]	1	4	0.9	12	✗	[Unb19]
[DKLs18a]	3	3	135	3	✓	[Neu18]
[XAXYC21] (Paillier)	1	3	6.3†	226†	✓	N/A
[XALCCXYZ23]	1	3	4.1†	209†	✓	N/A
[DKLs24]	2	3	115	29	✓	[Sil23]
[BHLS24] (EC)	1‡	2	3.8‡	416∼1012‡	✓	N/A
[BHLS24] (Paillier)	1‡	2	17.3‡	1168‡	✓	N/A
This Work	**1**	**2**	**2**	**48**	✓	*art. planned*

Table 1: Comparison of our protocol with the most round-, communication-, and computation-efficient protocols in the literature for ECDSA signatures on `secp256k1` for the 128-bit security level. Values are color-coded in blue if they match or outperform our work, and in red otherwise. We stress that "Comp." refers to pure computation time and ignores latency and bandwidth. All benchmarks were conducted on the same machine (an Intel Core i7-1365U CPU) and run on a single thread, except for those marked with '†' and '‡' because no public implementation was available. For entries marked with '†', we report values from the corresponding paper; for '‡', see Section 1.3.1.

secure protocol of [Lin17], which would require multiple round trips). Second, push notifications[1] are limited to 4–5KB on Android and Apple devices [Inc16]. Consequently, [DKLs18b; DKLs24] cannot rely on push notifications to handle message delivery and must use a different, likely slower network, or add additional communication rounds. Finally, another example is the use case of batch signing, where users want to sign a batch of signatures (which occurs in many settings, e.g. a server interacting with multiple clients possibly at the same time, as in the embedded cryptocurrency wallet use case that we target), high communication costs can cause significant slowdowns when one of parties is, say, a mobile phone. Concretely, assuming an upload speed of 50Mbps, signing 100 messages with DKLs18 (where each party transmits $135/2 = 67.5$KB of data) will add at least $100 \cdot 67.5\text{KB}/(50 \cdot 10^6 \text{ bits/s}) = 1.08$s, solely due to the bandwidth limitation. Using our protocol, the cost goes down to $100 \cdot 2\text{KB}/(50 \cdot 10^6 \text{ bits/s}) = 0.032$s.

Security assumptions and guarantees. All protocols above rely on the random oracle model (ROM). All but [DKLs24] (which is in the pure OT-hybrid model and ROM) rely on some EC-group-theoretic ([DKLs18b]) or/and number theoretic assumptions like security of Paillier encryption ([XAXYC21; Lin17; BHLS24]), strong-RSA ([XAXYC21; XALCCXYZ23; BHLS24]) and security of Joye-Libert encryption ([XALCCXYZ23]).

1.3.1 Comparison to [BHLS24].

Boneh, Haitner, Lindell, and Segev propose a two-round protocol for ECDSA where the baseline protocol is similar to the non-optimized version of the protocol in this paper, further augmented with an *exponent verifiable random function* (eVRF), where eVRF is a pseudorandom primitive that

[1]The push notification network is a highly optimized, distributed cloud infrastructure that delivers lightweight messages to mobile devices with low latency using persistent connections and efficient protocols. It is engineered to handle millions of notifications concurrently.

returns $k \in \mathbb{F}_q$ to P_1 and $R = g^k \in \mathbb{G}$ to P_2. They propose two different implementations for this primitive: one using Paillier encryption and integer commitments, and one using general-purpose SNARKS (namely *bulletproofs* [BBBPWM18]). The values reported in Table 1 are estimated for both bulletproofs (EC-based) and the Paillier-based construction.

Communication cost of eVRF. For the EC construction, Boneh, Haitner, Lindell, and Segev [BHLS24] estimate each proof to consist of 28 group elements. This amounts to $28 \cdot 256 = 0.9$KB per proof. For the Paillier construction, each proof to consist of 32 group elements and 8 Paillier ciphertexts [BHLS25b]. For a 3072-bit Paillier modulus (corresponding to NIST's 128-bit security level [Bar20]), this amounts to $32 \cdot 256 + 8 \cdot 6144 = 7.2$KB.

Computation cost of EC-based eVRF. Regarding the runtime of the EC construction, in the Eurocrypt'25 presentation of their work, Boneh, Haitner, Lindell, and Segev report runtimes based on a preliminary implementation of 800ms/400ms for prover/verifier, and estimate that these numbers could be improved to 300ms/150ms. The latter estimate translates to around 900ms of computation solely for the two eVRF invocations required in their protocol.

To err on the safe side, we also compute a lower bound on the cost based on the estimation in [BHLS24] that the dominant cost of the protocol is a multiexponentiation of approximately 4000 base elements (our experiments in OpenSSL indicate that this takes 12ms on average). For proving, we use the corresponding cost of of twice as many *individual* exponentiation, which is the equivalent prover cost according to the original Bulletproof paper [BBBPWM18]. Our experiments in OpenSSL indicate that each exponentiation takes $8 \cdot 10^{-3}$ms for the secp256r1 curve on average—which yields a total proving cost of $2 \cdot 4000 \cdot 8 \cdot 10^{-3} = 64$ms. As there are two PRF evaluations per eVRF invocation (so twice as many constraints), this amounts to a lower bound $2 \cdot 12 + 2 \cdot 64 = 152$ms of pure computation time per eVRF invocation, or around 300ms in total for the two invocations.

Computation cost of Paillier-based eVRF. For a 2048 bit modulus, Boneh, Haitner, Lindell, and Segev report 80ms of both proving and verification time (so 160ms total) [BHLS25b]. The cost of the proof is dominated by "full" modular exponentiation—where the exponent size is on the order of the modulus—and this operation scales cubically with the modulus length. So, for a 3072-bit Paillier modulus (corresponding to NIST's 128-bit security level [Bar20]), we estimate the total cost per eVRF invocation to be approximately $3.3 \cdot 160 = 528$ms, hence 1056ms for the two eVRF invocation in their protocol.

Total costs including the baseline protocol. The values in Table 1 take into account the aforementioned overhead of the two eVRFs (one in each direction), to which we add the cost of the baseline protocol of [BHLS24]. The latter is estimated as follows: [BHLS24] is essentially a variant of [Lin17] augmented with a ZK proof in the last round of signing. Communication-wise, this proof is estimated as three times the size of a Paillier randomizer (so $3 \cdot 3072 = 1.15$KB). Computation-wise, this proof costs 30ms for a 2048-bit modulus according to [BHLS24] which would give approximately 100ms for a 3072-bit modulus as its cost is dominated by the cost of full modular exponentiation and this operation scales with the cubic of the modulus size. This gives a total of $100 + 12 = 112$ms for the baseline protocol. For the EC version, we report both the number resulting from our conservative lower bound (counting only the runtime of exponentiations

and ignoring all other costs) and the number resulting from the authors' estimate of the runtime of an optimized version of their implementation [BHLS25a].

1.3.2 On the original (2-round) version of DKLs18 [DKLs18b].

In an earlier version of this paper, we included the two-round protocol from [DKLs18b] in our comparison section, but have since removed it because the authors have deprecated this protocol due to security concerns (see section 1.1 of [DKLs18a]), advising against its use. In addition to a few (minor) flaws, the original version relies on a generic group model argument that is very loose (it only provides $\text{poly}(k)/2^k$ security gap with an unspecified poly) and for which only a sketch security analysis is given. Later, [GS22] showed a concrete attack on a related ECDSA functionality where the adversary gets a meaningful advantage; due to this attack (which is not directly applicable to [DKLs18b]) and other flaws, the authors of [DKLs18b] recommend not to use their original protocol. We stress that, in contrast, our scheme has a GGM proof which is tight and provably avoids the [GS22] attack.

The revised protocol [DKLs18a], as well as its follow-up [DKLs24], does not have 2 rounds anymore—they both use three rounds. Hence, state-of-the-art "DKLs" protocols are not round optimal, when our protocol is.

1.4 Organization

In Section 2, we provide a detailed technical overview of our main contributions. Section 3 introduces necessary preliminaries, and Section 4 presents our main construction, a round-optimal 2-party threshold ECDSA protocol. The protocols described in Section 4 rely on multiple zero-knowledge proofs; we provide optimized instantiations of these zero-knowledge proofs together with their security analysis in Section 5.Sections 6 and 7 are dedicated to the security analysis of our protocol, including the analysis for the doubly enhanced unforgeability of ECDSA in the GGM, and Section 8 reports our benchmarks. Eventually, in Appendix A, we analyze some of the assumptions used in our work. This includes a reduction from small exponent indistinguishability to small exponent discrete logarithms, and a discussion on the security of tough RSA moduli. Appendix B provides additional technical lemmas used in our security proofs in the generic group model.

2 Technical Overview

In this section, we present our two-round ECDSA protocol. For ease of read, we abstract out a base version of the protocol (stripped of some of its optimizations) that involves two main components: an *inner protocol* that realizes a (form of) verifiable OLE functionality, and an *outer protocol* that distributively generates a signature given a functionality for the inner protocol. We first cover the outer protocol before going in the details of a basic implementation of the inner protocol. Afterwards, we cover some of our (many) optimizations to the inner protocol. In particular, this includes relaxing the functionality to allow for a specific security leakage that can be shown, in the concrete context of the overall protocol we are building, to be harmless. An important note is that, as a consequence of these optimizations, we do *not* formally prove security of the components separately (as our optimized inner protocol does actually not fully realize the verifiable OLE functionality), but instead provide (in Section 6) a direct and detailed full proof of the entire protocol.

Notation. We assume some familiarity with ECDSA and basic group theory and we use multiplicative notation for group operations. Let $(\mathbb{G}, g, q)$ and $\mathsf{F} : \{0,1\}^* \to \mathbb{Z}_q$ denote the group-generator-order tuple and the message digest function associated with ECDSA. We recall that $\mathbb{G}$ is with a 'conversion function' $\mathcal{X} : \mathbb{G} \to \mathbb{Z}_q$ that deterministically maps group elements to elements modulo q with the property that $\mathcal{X}(A) = \mathcal{X}(A^{-1})$ for $A \in \mathbb{G}$. ECDSA public keys are simply elements $X = g^x \in \mathbb{G}$ where $x \xleftarrow{\$} \mathbb{Z}_q^*$ is drawn uniformly at random from the set of units modulo q. ECDSA signatures for a message digest $m = \mathsf{F}(\text{msg})$ consist of pairs $(r, s) \in \mathbb{F}_q^2$ such that $r = \mathcal{X}(g^k)$ and $s = k^{-1}(m + rx) \bmod q$ for $k \xleftarrow{\$} \mathbb{Z}_q^*$ chosen uniformly at random. Let $\mathsf{RO} : \{0,1\}^* \to \mathbb{Z}_q$ denote a random oracle that maps arbitrary strings to elements modulo q.

2.1 ECDSA at the Cost of 1 OLE

Key-generation. The parties, P_1 and P_2 execute a distributed key-generation protocol to calculate additive shares x_1 for P_1 and $x_2 \in \mathbb{Z}_q$ for P_2 of the ECDSA public key $X = g^{x_1+x_2}$. We omit a detailed description from this high-level technical overview as the process is fairly standard. (In the subsequent sections, we propose a concrete protocol for this purpose.) Write $X_i = g^{x_i}$ for $i = 1, 2$.

Signing. Assume that the parties have access to an oracle for computing the following variant of the *oblivious linear evaluation* (OLE) functionality:[2] OLE takes input $x' \in \mathbb{F}_q$ from P_1 and $a, b \in \mathbb{F}_q, X' \in \mathbb{G}$ from P_2 and returns $y = ax' + b \bmod q$ to P_1 if and only if $X' = g^{x'}$, and $\perp$ otherwise. The parties then engage in the following protocol to calculate an ECDSA signature for m. All field operations below are reduced modulo q (and we do not write this explicitly to alleviate notation).

Protocol Our 2-Party ECDSA Signing Protocol

Inputs. ECDSA public key $X \in \mathbb{G}$. Each P_i holds x_i such that $X = g^{x_1+x_2}$.

Message to be signed. msg $\in \{0,1\}^*$ and message digest $m = \mathsf{F}(\text{msg}) \in \mathbb{F}_q^*$.

Oracle. Parties have access to an OLE oracle satisfying the following functionality: OLE takes input x' from P_1 and (a, b, X') from P_2 and returns $y = ax' + b \bmod q$ to P_1 if and only if $X' = g^{x'}$, and $\perp$ otherwise.

1. P_1 samples $k_1 \xleftarrow{\$} \mathbb{Z}_q$ and sends $(R_1, C) = (g^{k_1}, X_2^{k_1})$ to P_2.
2. When P_2 receives (R_1, C), does:
   (a) Abort if (R_1, X_2, C) is not a DH tuple, i.e., $C \neq R_1^{x_2}$. Otherwise, sample $k_2 \xleftarrow{\$} \mathbb{Z}_q$.
   (b) Set $(R_2, R) \leftarrow (g^{k_2}, R_1^{k_2})$ and $r \leftarrow \mathcal{X}((R_1 \cdot g^{\mathsf{RO}(X,R_2,R,m)})^{k_2})$.
   (c) Set $(a, b) \leftarrow (k_2^{-1}r, k_2^{-1}(m + rx_2))$ and send (R_2, R) to P_1.
3. P_1, P_2 call OLE on inputs x_1 and (a, b, X_1) respectively. Let y denote P_1's output.

[2] In Appendix C, we show how to reduce this variant of the OLE functionality to standard OLE with insignificant overhead (in the ROM).

4. When receiving (R_2, R, y), P_1 does:
   (a) Abort if (R_1, R_2, R) is not a DH tuple, i.e., $R_2^{k_1} \neq R$.
   (b) Set $R \leftarrow R_2^{k_1}$, $r \leftarrow \mathcal{X}(R_2^{k_1 + \mathsf{RO}(X, R_2, R, m)})$ and $s \leftarrow (k_1 + \mathsf{RO}(X, R_2, R, m))^{-1} \cdot y$.
   (c) Output (r, s) if it is a valid signature for m.

Protocol 1: 2-Party ECDSA at the cost of 1 OLE.

Realizing the OLE Functionality and alternative taken in this work. Protocol 1 assumes that parties have access to an oracle that perfectly computes the OLE correlation. As explained earlier in this section, Protocol 1 serves a pedagogical role, providing an abstraction of the *outer* protocol in the final protocol (with the OLE call functioning as the *inner* protocol). In our final protocol, we implement the OLE over small integers using Paillier encryption and appropriate zero-knowledge proofs, following standard techniques. Furthermore, due to several optimizations, the resulting construction becomes slightly 'leaky'. As we show in the technical sections, this leakage is inconsequential to the protocol's security since only ephemeral randomness is exposed—not the actual secrets. However, it poses definitional challenges when formalizing the security model in terms of ideal functionalities. Consequently, in the formal security analysis, in order to avoid introducing contrived ideal functionalities tailored to our constructions, we have opted to prove the protocol's security in one shot rather than proving security of each component independently.

2.1.1 Sketch security Analysis

Correctness. If both parties follow the protocol correctly, then it is straightforward to see that the above process yields a random signature for m. Indeed, note that

$$\begin{aligned} y &= a \cdot x_1 + b = k_2^{-1} r x_1 + k_2^{-1}(m + r x_2) \\ &= k_2^{-1}(m + r(x_1 + x_2)). \end{aligned}$$

Thus, letting $k_1' = k_1 + \mathsf{RO}(X, R_2, R, m)$, it is easy to see that $(r, s) = (\mathcal{X}(R_2^{k_1'}), k_1'^{-1} \cdot y)$ is a valid signature for m.

Malicious P_1. We note that the protocol includes a seemingly superfluous element sent from P_1 to P_2 (namely, C). However, this element is necessary for security, as we explain shortly.
The protocol is secure against a malicious P_1 because it has almost no opportunity to cheat beyond simply aborting. For example, if P1 uses an incorrect x_1 in the OLE, the protocol will eventually abort. Slightly more formally, the protocol can be simulated for any malicious P_1 given a signature (r, s) for m, via the following simulation process.

1. When receiving (R_1, C) from P_1, do:
   (a) Extract k_1 under the knowledge of exponent (KEA) assumption if the tuple (R_1, X_2, C) is a DH tuple. If extraction fails, abort. (The simulator is able to extract k_1 when (R_1, X_2, C) is a DH tuple as KEA holds unconditionally in the generic group model—which we use to model $\mathbb{G}$.)

(b) Sample $\mu \xleftarrow{\$} \mathbb{F}_q$ and set $y = (k_1 + \mu) \cdot s$.

(c) Compute $\hat{R}$ such that $\mathcal{X}(\hat{R}) = r$ and $\hat{R}^s = g^m X^r$, and set $R_2 = \hat{R}^{(k_1+\mu)^{-1}}$.

(d) Program the random oracle such that $\mathsf{RO}(X, R_2, R_2^{k_1}, m) = \mu$.

2. When obtaining x' for the OLE call from P_1, do:

(a) Abort if $x' \neq x_1$.

(b) Return (R_2, y) on behalf of the simulated P_2 and OLE oracle.

Malicious P_2. The case of a malicious P_2 is more subtle as we require an extra security assumption which is closely related to the security of ECDSA in the 'presignatures' mode of operation (the security game of the 'presignatures' mode of operation is a variant of ECDSA security game where the attacker is allowed to choose the message to be signed after the nonce is revealed, i.e., the left hand side of the signature, r). Namely, consider the following variant of the ECDSA security game between the attacker A and a signing oracle SO:

1. *Key-Generation.* SO samples $x \xleftarrow{\$} \mathbb{Z}_q$ and returns $X = g^x$.

2. *Signing.* When prompted for signing a message digest m,

(a) SO samples $k \xleftarrow{\$} \mathbb{Z}_q$ and returns $R = g^k$.

(b) A adaptively sends tuples of the form (R, z) and SO returns $\mu \xleftarrow{\$} \mathbb{Z}_q$.

(c) A sends one of (R, z, μ) and SO returns $z^{-1}(k + \mu)^{-1}(m + rx)$ for $r = \mathcal{X}(g^{z(k+\mu)})$.

3. *Outcome.* A wins if it outputs (m^*, σ^*) s.t. σ^* is a valid signature for m^*, and m^* was never queried for signing. The pair (m^*, σ^*) is referred to as a forgery.

Assuming that A cannot win the above game except with negligible probability, it is possible to simulate the protocol in the presence of a malicious P_2, given access to SO. In this paper, we provide a formal analysis with tight security bounds for this game in the GGM—more on that in the next subsection. Below, we describe how to simulate the signing phase for a malicious P_2, given access to SO (it is assumed that the simulator extracted x_2 from P_2 in the previous step and made sure to match public key X in the protocol to the one returned by SO).

1. Call SO as in Item 2 and receive $R_1 \in \mathbb{G}$.

2. Send $(R_1, R_1^{x_2})$ to P_2 on behalf of the simulated P_1.

3. When P_2 invokes the random oracle RO on input $(X, g^{k_2'}, R_1^{k_2'}, m)$, extract k_2' similarly to the extraction k_1 in the case of corrupted P_1, call SO as in Item 2b on input (R_2', k_2') and message digest m, and obtain μ'. Program the random oracle such that $\mathsf{RO}(X, g^{k_2'}, R_1^{k_2'}, m) = \mu'$.

4. When receiving (R_2, a, b) from P_2, do:

(a) Abort if $(X, g^{k_2}, R_1^{k_2}, m)$ was never queried to RO for an extracted k_2 such that $R_2 = g^{k_2}$. Otherwise, let $\mu = \mathsf{RO}(X, g^{k_2}, R_1^{k_2}, m)$ and $r = \mathcal{X}((R_1 \cdot g^{\mu})^{k_2})$.

(b) Abort if $a \neq k_2^{-1} r$ or $b \neq k_2^{-1}(m + rx_2)$.

(c) Call SO as in Item 2c on input (R, k_2, m, μ). Obtain s from SO and output (r, s).

Faithfulness of the simulation. When P_2 follows the protocol honestly, the simulation perfectly emulates the interaction between P_1 and P_2. The challenge is to analyze the simulation's faithfulness when P_2 deviates from the protocol—specifically in Items 4a and 4b above: either P_1 never queries the RO on input $(X, g^{k_2}, R_1^{k_2}, m)$, or $a \neq k_2^{-1} r$ or $b \neq k_2^{-1}(m + rx_2)$. In these cases, as detailed in the technical sections, we show that P_2 aborts with overwhelming probability because the protocol ultimately produces an invalid signature string.
At a high level, first, in case Item 4a does not abort but (a, b) is malformed, then the protocol results in a valid signature only if P_2 guesses the secret share x_1 of P_1. Second, if Item 4a aborts, then either (R_1, R_2, R) is not a DH tuple—in which case P_1 aborts in the real protocol—or the RO was not queried on the relevant input, in which case the nonce r has full entropy from P_2's point of view, and the pair (a, b) is malformed with overwhelming probability, resulting in P_1 aborting.

2.2 Doubly-Enhanced Existential Unforgeability of ECDSA

To support the additional security assumption, we provide a security proof in the *generic group model* [Sho97]. Specifically, we show that any attacker who only makes black-box use of the elliptic curve—referred to as a *generic attacker*—and succeeds in forging a signature in the above game can be used to break the pre-image resistance of the ECDSA message digest function. Furthermore, the security reduction incurs less than a 2^5-multiplicative loss compared to the analysis of Brown [Bro02] which is essentially optimal.
To achieve the utmost generality, we prove the security bound for a weaker game (corresponding to a stronger attacker) where the attacker is allowed to choose the message digest $m \in \mathbb{F}_q^*$ *after* the group element $R \in \mathbb{G}$ is provided. The resulting security game generalizes the 're-randomized presignatures' proposal from [GS22]. Specifically, we introduce a new security game, referred to as *doubly-enhanced unforgeability game*, which is defined below (the value r below is calculated slightly differently from the value from Section 2.1.1, but it is easy to see that the two are equivalent in the random oracle model).

Security Game Unforgeability Experiment

1. *Key-Generation.* SO samples $x \xleftarrow{\$} \mathbb{Z}_q$ and returns $X = g^x$.
2. *Signing.* When prompted for signing on message digest m:
   (a) SO samples $k \xleftarrow{\$} \mathbb{Z}_q$ and returns a *presignature* $R = g^k$.
   (b) A adaptively sends tuples of the form (R, z, m) SO returns *modifiers* $\mu \xleftarrow{\$} \mathbb{Z}_q$.
   (c) A sends one of (R, z, m, μ) and SO returns $(zk + \mu)^{-1}(m + rx)$ for $r = \mathcal{X}(g^{zk+\mu})$.
3. *Outcome.* A wins if it outputs (m^*, σ^*) s.t. σ^* is a valid signature for m^*, and m^* was never queried for signing. The pair (m^*, σ^*) is referred to as a forgery.

Security Game 1: Doubly-Enhanced Unforgeability of ECDSA

Theorem 2.1 (Informal)**.** *Let* A *denote a generic attacker making at most* t *group operations and signing oracle queries and assume that* A *produces a forgery in the above security game with probability* ε*. Then, there exists an algorithmm* B *with black-box access to* A *that breaks the pre-image resistance of the ECDSA message digest function with probability* $2^{-5}\varepsilon/t$*. Furthermore, the*

running time of B *is essentially the same as the running time of* A *plus* $t \cdot u$*, where* u *is an upper bound on the number of yet-to-be-signed presignatures at any given time.*

Remark 2.2 (Comparison to [GS22]). Security Game 1 generalizes the "re-randomized presignatures" game from [GS22] in two significant ways. First, we allow the attacker to "tweak" $R = g^k$ in the exponent with a value z of their choice; for each tuple (R, z, m), a random μ is drawn, so that the candidate nonce for the signature becomes $g^{kz+\mu}$. Second, we allow the attacker to query multiple tuples (R, z', m') before choosing which one to sign. In contrast, [GS22] considers only the special case where $z = 1$ always, and the attacker is restricted to querying a single tuple $(R, 1, m)$ per presignature R. In fact, [GS22] leaves open the question of whether allowing multiple queries per presignature could harm security in some way. Here, we settle that question in the negative, showing that Security Game 1 is effectively as secure as the regular ECDSA game in the GGM.

2.2.1 The generic group model

We use the EC-generic group model of [GS22] that captures some idiosyncrasies of elliptic curves, in particular the fact that $\mathcal{X}(A) = \mathcal{X}(A^{-1})$ for every $A \in \mathbb{G}$. Formally, letting $\boldsymbol{G}$ denote the set of identifying labels of elements in $\mathbb{G}$, the EC-generic group is defined via a random bijective map $\pi : \mathbb{G} \to \boldsymbol{G}$ and a group-oracle $\mathsf{O} : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $\pi(AB) = \mathsf{O}(\pi(A), \pi(B))$ and $\mathcal{X}(\pi(A)) = \mathcal{X}(\pi(A^{-1}))$, for every $A, B \in \mathbb{G}$. That is, pairs of inverses map to the same element under $\mathcal{X}$, and the value of $\mathcal{X}(\pi(A))$ completely independent of A. In group-theoretic jargon, $(\boldsymbol{G}, *)$ is isomorphic to $(\mathbb{G}, \cdot)$ via the group-isomorphism π, letting $* : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $G * H = \mathsf{O}(G, H)$. The attacker can sample uniformly from $\boldsymbol{G}$ and efficiently decide membership in $\boldsymbol{G}$, and they have oracle access to the group-oracle O.
Additionally, it is assumed that O takes the role of SO in that (i) it samples an ECDSA secret key $x \in \mathbb{F}_q^*$ generates an ECDSA public key and returns the public key $X = \pi(g^x)$ (ii) it generates presignatures when prompted to do so by sampling $k \in \mathbb{F}_q^*$ and returning $R = \pi(g^k)$ (iii) it returns random 'modifiers' $\mu \in \mathbb{F}_q$ when queried on a suitable tuple (R, z, m) (iv) it returns signatures as per Item 2c of the security game.

2.2.2 Sketch security proof

Symbolic Lazy Evaluation. Following the approach of [GS22], our simulation employs a *lazy symbolic* evaluation of the group oracle, where the permutation π is gradually constructed as the experiment progresses. In addition, the simulator maintains X and all yet-to-be-signed presignatures R as formal random variables within its description of π, without assigning discrete-log values. Instead, when prompted (e.g., for the public key X), the simulator updates π by including the pairs $(\mathfrak{X}, X)$ and $(-\mathfrak{X}, X^{-1})$, where $\mathfrak{X}$ is a formal random variable representing the unassigned discrete logarithm of X (and a different formal variable is introduced for each presignature R).
Similarly, when a signature is requested for a tuple (R, z, m, μ), where $\mathfrak{R}$ is the formal random variable representing the discrete log of R, the simulator proceeds as follows. It samples $s \in \mathbb{F}_q^*$ at random, then replaces $\mathfrak{R}$ with $z^{-1}(m/s - \mu) + z^{-1} \cdot \hat{r}/s \cdot \mathfrak{X}$ throughout the domain of π, where $\hat{r} = \mathcal{X}(\pi(\mu + z\mathfrak{R}))$. This process consumes the presignature R and ensures that $(\hat{r}, s)$ is a valid signature for the message digest m according to π, with the nonce computed in the expected way with respect to the tuple (R, z, m, μ). The simulator then returns s to the attacker.

Injecting the pre-image challenge. The simulator's strategy for inverting the digest function on a random element $e \xleftarrow{\$} \mathbb{F}_q$ is to guess the nonce of the prospective forgery. Namely, the attacker must query the group oracle on some weighted combination of g, X and the yet-to-be-signed presignatures, and the simulation proceeds as follows. Let F denote the nonce of the prospective forgery and assuming that the simulator guessed correctly when it was first queried to the group oracle as $F = g^{\alpha} X^{\beta} \hat{R}^{\gamma}$ where $\hat{R} = R^z \cdot g^{\mu}$ according to the tuple (R, z, m, μ) (for simplicity in this high-level overview, we assume that only one presignature R is ever queried by the attacker)

1. If $\gamma = 0$ and $\alpha, \beta \neq 0$ then return F such that $\mathcal{X}(F) = \alpha^{-1} e \beta$.
2. Else, if $\gamma \neq 0$ and $\alpha \hat{r} - \beta m = 0$, for $\hat{r} = \mathcal{X}(\hat{R})$, then return F such that $\mathcal{X}(F) = m^{-1} e \hat{r}$. (If there are many tuples (R, z', m', μ') satisfying the aforementioned constraints, simply pick one of them at random.)
3. Else, return a random $Z \in \boldsymbol{G}$ and when the attacker, later, requests a signature for the tuple (R, z, m, μ) return signature $s = \gamma(e\hat{r} - \mathcal{X}(Z)m)/(\alpha \mathcal{X}(Z) - e\beta)$, for $\hat{r} = \mathcal{X}(\hat{R})$.

Security analysis. The security analysis of the above rests on three pillars. The first one is showing that the symbolic lazy evaluation experiment, without injecting the random pre-image challenge, is indistinguishable from the plain generic group experiment (except with a small error term of $O(t^2/q)$ where the hidden constant is small ≈ 5). This was shown in [GS22] and an alternative proof can be found in this document (Appendix B) for completeness.
The second pillar is showing that when injecting the random pre-image challenge, this does not affect the distribution of the adversary's view. Indeed, this is guaranteed by the fact that there exists an efficient algorithm Samp such that on a random input $w \in \mathbb{F}_q$, Samp either returns a uniformly random $A \in \boldsymbol{G} \setminus \{\mathbb{1}\}$ such that $\mathcal{X}(A) = w$ with probability at least $1/5$, or Samp returns fail (one of the sources of the security loss in our bound is the failure probability of Samp—when it does, the simulator reverts to the symbolic evaluation experiment and it does not inject the pre-image challenge). Additionally, when $\gamma \neq 0$ and $\alpha\hat{r} - \beta m \neq 0$, then the map $e \mapsto \gamma(e\hat{r} - \zeta m)/(\alpha\zeta - e\beta)$ for any fixed $\zeta \in \mathbb{F}_q^*$ is injective, and thus s is essentially uniform over $\mathbb{F}_q^*$, as in the symbolic evaluation experiment.
Finally, the last pillar of the analysis is showing the following two items in the case that $\gamma \neq 0$ and $\alpha\hat{r} - \beta m = 0$ (the first item ensures that the simulator will be able to program F in the desired way and the second item ensures there are at most three candidates for F—this is another source of the security loss because the simulator might choose the wrong candidate):

1. The tuple (R, z, m) was queried to the oracle for a modifier (and the oracle returned μ) *before* F was obtained from the oracle.
2. There are at most three tuples (R, z', m', μ') satisfying the constraints in question, i.e., for $\hat{R}' = g^{\mu'} \cdot R^{z'}$, it holds that $F = g^{\alpha'} X^{\beta'} \hat{R}'^{\gamma}$ and $\alpha'\hat{r}' - \beta' m' = 0$ where $\hat{r}' = \mathcal{X}(\hat{R}')$.

The fact that μ is random in $\mathbb{F}_q$ ensures that Item 1 holds true with all but probability $O(t^2/q)$—where the t^2-multiplicative term accounts for all the possible pairs $(F, \hat{R})$. For Item 2, assume that there exist $\{(R, z_i, m_i, \mu_i)\}_{i=1}^4$ such that for $\hat{r}_i = \mathcal{X}(g^{\mu_i} R^{z_i})$, it holds that $(\alpha - \gamma z_i^{-1})\mu_i \hat{r}_i - \beta m_i = 0$, for some $\alpha, \beta \in \mathbb{F}_q$. This means that the vector $\vec{v} = (\gamma z_i^{-1} \mu_i \hat{r}_i)_{i=1}^4$ lies on the plane in $\mathbb{F}_q^4$ generated by the vectors $(\hat{r}_i)_{i=1}^4$ and $(m_i)_{i=1}^4$. As μ_i is independent of $\hat{r}_i$ in this model, by fixing $(\hat{r}_i)_{i=1}^4$, this

means that the random vector $\vec{v}$ lies on a fixed plane—this happens with probability at most $1/q^2$. Taking into account that there are at most t^4 possible tuples $(\hat{r}_i, m_i)_{i=1}^4$, Item 2 holds true with all-but-probability t^4/q^2.

Obtaining the preimage. To conclude it remains to show that if the simulator's guesses were correct (e.g., identifying F) and no failure occurred, then the experiment above yields a preimage for the challenge e. Assume that the attacker outputs msg and (F, σ) such that, letting $f = \mathcal{X}(F)$, it holds that $F = g^{\mathsf{F}(\text{msg})/\sigma} X^{f/\sigma}$. Then each of the cases below yields that $\mathsf{F}(\text{msg}) = e$. For a suitable tuple (R, z, m, μ) with $\hat{R} = g^{\mu} R^{z}$, let α, β, γ such that $F = g^{\alpha} X^{\beta} \hat{R}^{\gamma}$ when F was first queried. Additionally, let s denote the signature returned by the signing oracle for (R, z, m, μ).

1. If $\gamma = 0$, then $F = g^{\alpha} X^{\beta} = g^{\mathsf{F}(\text{msg})/\sigma} X^{f/\sigma}$. Thus $\alpha = \mathsf{F}(\text{msg})/\sigma$ and $\beta = f/\sigma$. Using the fact that $f = \alpha^{-1} e \beta$, it follows that $\mathsf{F}(\text{msg}) = e$.
2. If $\gamma \neq 0$, then $F = g^{\alpha} X^{\beta} \hat{R}^{\gamma} = g^{\alpha + \gamma m/s} X^{\beta + \gamma \hat{r}/s} = g^{\mathsf{F}(\text{msg})/\sigma} X^{f/\sigma}$. Thus $\alpha + \gamma m/s = \mathsf{F}(\text{msg})/\sigma$ and $\beta + \gamma \hat{r}/s = f/\sigma$. If $\alpha \hat{r} - \beta m = 0$, then, using the fact that $f = m^{-1} e \hat{r}$, it follows that $\mathsf{F}(\text{msg}) = e$.
3. Else, if $\gamma \neq 0$ and $\alpha \hat{r} - \beta m \neq 0$ then $\alpha + \gamma m/s = \mathsf{F}(\text{msg})/\sigma$ and $\beta + \gamma \hat{r}/s = f/\sigma$ implies that $s = \gamma(\mathsf{F}(\text{msg})\hat{r} - fm)/(\alpha f - \mathsf{F}(\text{msg})\beta)$. Thus, using that fact that $s = \gamma(e\hat{r} - fm)/(\alpha f - e\beta)$ in this case and the map $e \mapsto \gamma(e\hat{r} - fm)/(\alpha f - e\beta)$ is injective in this regime of parameters, it follows that $\mathsf{F}(\text{msg}) = e$.

2.3 Implementation using Paillier Encryption and Integer Commitments

In this section, we describe the concrete implementation of the OLE. As mentioned in Section 2.1, we opt for a different approach than implementing our protocol by means of a random OLE correlation. Namely, we realize the functionality below by means of Paillier encryption and integer (Damgård-Fujisaki) commitments.

Functionality Committed-Receiver OLE

Common input. Group-generator-order tuple $(\mathbb{G}, g, q)$ and group element $X \in \mathbb{G}$.

Secret input. P_1 provides $x \in \mathbb{F}_q$ and P_2 provides $a, b \in \mathbb{F}_q$

Operation. If $g^x = X$, send $y = ax + b \bmod q$ to P_1. Else, abort.

Outputs. P_2 outputs nothing and P_1 outputs whatever it received from the functionality.

Realizing the above using Paillier encryption and Damgård-Fujisaki commitments is fairly standard and we outline it here for the benefit of the reader.

Parameters Generation. P_1 samples an RSA modulus N (i.e., the Paillier public key), a Damgård-Fujisaki tuple $(\hat{N}, t, s_1, s_2)$ where $\hat{N}$ is an RSA modulus consisting of strong primes and t, s_1, s_2 are random quadratic residues in $\mathbb{Z}^*_{\hat{N}}$, and sends $(N, \hat{N}, t, s_1, s_2)$, together with ZKPs for the following:

1. N is the product of exactly two primes of roughly the same size and $\gcd(N, \varphi(N)) = 1$.

2. $t \in \mathbb{Z}^*_{\hat{N}}$ and s_1, s_2 belong to the group generated by t in $\mathbb{Z}^*_{\hat{N}}$.

OLE Step 1. On input $x \in \mathbb{F}_q$ for P_1:

1. P_2 samples an ephemeral Damgård-Fujisaki tuple $(\hat{M}, v, u_1, u_2)$ and sends the tuple together with ZKP that $v \in \mathbb{Z}^*_{\hat{M}}$ and u_1, u_2 belong to the group generated by v.
2. P_1 computes $\mathcal{E} = \mathrm{enc}_N(x, \rho^N) = (1 + xN)\rho^N \bmod N^2$ for $\rho \xleftarrow{\$} \mathbb{Z}^*_N$ and, using the tuple $(\hat{M}, v, u_1, u_2)$, it generates a ZKP validating that $x' = \mathrm{dec}_{\varphi(N)}(\mathcal{E})$ is 'small' compared to N, and $g^{x'} = g^x$.

OLE Step 2. On input $a, b \in \mathbb{F}_q$ for P_2, P_2 sends $\mathcal{S} = \mathcal{E}^a \cdot \mathrm{enc}_N(b + \nu \cdot q, \mu^N) \bmod N^2$ where ν is chosen randomly and sufficiently large to mask $\lfloor (ax + b)/q \rfloor$ and $\mu \in \mathbb{Z}^*_N$. Along with $\mathcal{S}$, P_2 sends a ZKP validating that $\mathcal{S} = \mathcal{E}^{a'} \cdot \mathrm{enc}_N(b', \mu'^N) \bmod N^2$ for $\mu' \in \mathbb{Z}^*_N$ and a', b' are small compared to N; this proof requires the tuple $(\hat{N}, t, s_1, s_2)$. If no error is detected (e.g., a failed ZKP), P_1 outputs $y = \mathrm{dec}_{\varphi(N)}(\mathcal{S}) \bmod q$.

Remark 2.3. It is emphasized that in our ECDSA protocol, the parameters generation as well as the first step of the OLE are done only once, ahead of time (see Section 2.3.6). For each signature generation, the parties only consume resources for the second step of the OLE.

2.3.1 Efficiency considerations

While fairly efficient for many applications, the process described above presents a couple of 'pain points'. First, sampling strong primes for the Damgård-Fujisaki tuples is very time-consuming. For example, generating strong primes of 1024 and 1536 bits (corresponding to the bit-lengths of 2048- and 3072-bit RSA moduli) takes significantly longer than generating standard primes. Specifically, it takes between 20 and 50 times longer to generate a strong prime compared to a standard prime (depending on the size of the prime), and approximately 1 to 4 **seconds** to generate a strong RSA modulus.
Second, proving and verifying that the Damgård-Fujisaki tuple is partially well-formed, i.e., validating that $u_1, u_2 \in \langle v \rangle$, is computationally expensive. The baseline proof has a soundness of only $1/2$ and must be repeated hundreds of times to achieve the desired level of soundness.
Finally, the number of bit operations for computing $a^b \bmod m$ is $O(\log(m)^2 \cdot \log(b))$. This becomes particularly significant when encrypting Paillier ciphertexts, where the complexity is dominated by the cost of computing $\rho^N \bmod N^2$, i.e., $O(\log(N^2) \cdot \log(N))$ bit operations. This computation can become a bottleneck for the system when processing many signatures simultaneously.

2.3.2 Tough primes optimization

The standard way of generating an RSA modulus to be used with Damgård-Fujisaki commitments is to pick a product of strong primes, *i.e.*, a modulus $N = p \cdot q$ where $p = 2p' + 1$ and $q = 2q' + 1$ for primes p', q'. This was the setting assumed in the original work of Fujisaki and Okamoto [FO97] and in most follow-up works, e.g. [CPP17]. Unfortunately, and this is the core motivation underlying our choice of a different RSA modulus, sampling a product of strong primes is an extremely slow procedure: it essentially involves sampling random $n_{\mathrm{rsa}}/2$-bit integers p', testing for primality, and whenever a prime p' is found, testing for primality of $2p' + 1$ until a strong prime $p = 2p' + 1$ is found (then doing the same for q). Heuristically, the running time of this procedure is governed

by the density of primes predicted by the prime number theorem: it takes an expected number of $n_{\text{rsa}}^2/2$ tries to construct N, where each try requires testing an $n_{\text{rsa}}/2$-bit integer for primality.

Algorithm Sampling tough-prime RSA moduli

Input. Security parameter ℓ and RSA bit-length n_{rsa} s.t. 4ℓ divides n_{rsa}. Let $t \leftarrow n_{\text{rsa}}/(4\ell)$.

Operation. For $i = 1, 2$, do:

1. Sample a pool $\boldsymbol{B}_i$ of $2^{2\ell}$-sized primes.
2. Enumerate over all unordered combinations of t primes in $\boldsymbol{B}_i$.
   - Let $\{p_1, \ldots, p_t\}$ be the combination at any given iteration.
   - Set $P_i \leftarrow 2 \cdot p_1 \cdots p_t + 1$ if it is a prime number and break the loop.

Output. $N \leftarrow P_1 \cdot P_2$

Algorithm 1: Protocol for sampling tough primes

We introduce an alternative strategy for sampling RSA moduli that can be used with Damgård-Fujisaki commitments. Our procedure requires first generating $|\boldsymbol{B}_i|$ small primes, using on average $2\ell|\boldsymbol{B}_i|$ tries (where each try tests a 2ℓ-bit integer for primality) followed by an expected $n_{\text{rsa}}/2$ number of executions of Item 2 (which involves multiplying T 2ℓ-bit integers and testing an $n_{\text{rsa}}/2$-bit integer for primality). In practice, the runtime of this procedure is largely dominated by the $n_{\text{rsa}}/2$ primality tests on $n_{\text{rsa}}/2$-bit integers. Therefore, the total expected runtime to generate N is about quadratically faster (n_{rsa} versus $n_{\text{rsa}}^2/2$) than the expected time to generate a product of strong primes. In the next paragraphs, we further explain why our choice of RSA modulus does not negatively impact the security analysis of our protocols.

Security of factoring. State-of-the-art factoring algorithms, such as GNFS, have a runtime that depends solely on the size $n_{\text{rsa}} = n_{\text{rsa}}(\ell)$ of the modulus (for which we follows standard recommendation, see Section 8.1) to guarantee a runtime of the order of 2^ℓ. On the other hand, special-purpose factoring algorithms can perform better when the modulus has a specific structure. In the case where $p-1, q-1$ have only small factors, the main such algorithm is Pollard's $p-1$ algorithm [Pol74] whose runtime is quasilinear in the size of the largest prime factor of $p-1$ (or $q-1$). In our context, the runtime of Pollard's algorithm is $O(2^{2\ell} \cdot n_{\text{rsa}} \log n_{\text{rsa}})$, which is much slower than the runtime of state-of-the-art (general-purpose) factoring algorithms (which is about 2^ℓ)—see Appendix A.

Soundness of proofs. Two zero-knowledge proofs in our protocols involve proving knowledge of a (short) opening to a Damgård-Fujisaki commitment: the well-formdness of $\mathcal{E}$ is Step 1 of the OLE, and the well-formedness of $\mathcal{S}$ in step 2 of the OLE.
The security analysis of both proofs follows closely the approach laid out in [DF02] for proving knowledge of an opening to an integer commitment with the Damgård-Fujisaki commitment scheme. It provides the following guarantee: assume that the modulus N is of the form $n = p \cdot q$ with

$\gcd(p-1, q-1) = 2$, $p = q = 3 \bmod 4$, and such that $\mathbb{Z}_N$ contains many elements whose order contains only large prime factors (more formally, it should hold that a random $h \leftarrow \mathbb{Z}_N^*$ has a non-negligible probability of having order 2^ℓ-rough). Then the knowledge soundness reduces to the following two assumptions:

- Strong RSA assumption: no polynomial-time algorithm A can, given N and $v \leftarrow \mathbb{Z}_N^*$, find $u \in \mathbb{Z}_N^*$ and $t > 1$ such that $u^t = v \bmod N$ happens with non-negligible probability.
- Small order assumption: no polynomial-time algorithm A can, given N, find (b, σ) with $b^2 \neq 1$, $\sigma \leq 2^\ell$, and $b^\sigma = 1$.

It is clear from its description that the modulus N produced by Algorithm 1 satisfies all the conditions outlined above (since the only elements of $\mathbb{Z}_N^*$ whose order is not 2^ℓ-rough are the roots of unity). Furthermore, with our choice of modulus, the small order assumption holds unconditionally: by construction, all subgroups of $\mathbb{Z}_N^*$ of order larger than 2 have order at least $2^{2\ell}$. Therefore, in our setting, the soundness of the zero-knowledge proofs reduces directly to the strong RSA assumption. We note that, as of today, there is no known attack on strong RSA that does not involve factoring the modulus.

2.3.3 Fast and compact variant of Damgård-Fujisaki parameters proof

To avoid the heavy penalty in the number of repetitions, we use a variant of the protocol of [HLM25] for proving that the tuple $(\hat{M}, v, u)$ is well formed, i.e., u belongs to the group generated by v in step 1 of the OLE (for the purposes of this high-level overview we ignore the second u-term). The classic three-move zero-knowledge protocol between a prover P (the party generating the tuple) and the verifier V for this relation goes as follows (essentially the classic Schnorr-style proof for discrete-log where the verifier's challenge is a random bit):

Protocol Vanilla Damgård-Fujisaki parameters well-formedness proof

Secret Input. P holds $\lambda \in [\varphi(\hat{M})]$ such that $u = v^\lambda \bmod \hat{M}$.

1. P sends $A \leftarrow v^\alpha \bmod \hat{M}$ for $\alpha \xleftarrow{\$} [\varphi(\hat{M})]$.
2. V replies with a random challenge $e \xleftarrow{\$} \{0, 1\}$.
3. P returns $z \leftarrow \alpha + e\lambda \bmod \varphi(\hat{M})$.

Verification. V accepts if $v^z = A \cdot u^e \bmod \hat{M}$

It is straightforward to see that the above protocol only achieves a soundness of $\frac{1}{2}$ and must be repeated at significant cost. This cost arises both computationally, as it may involve *hundreds* of full-size exponentiations, and in communication, since at least ℓ elements of $\mathbb{Z}_{\hat{M}}^*$ (the z's) are transmitted, where ℓ denotes the security parameter.
In [HLM25], the authors show that the soundness of the protocol scales with the size of the challenge space (the set from which e is sampled), but this comes at the cost of similar *slackness* in the proof's guarantees. Specifically, when e is uniformly sampled from $\{1, \ldots, 2^{\ell_0}\}$, the proof guarantees that $u = \sigma \cdot v^\lambda \bmod \hat{M}$, where σ is an element of at most ℓ_0-order in $\mathbb{Z}_{\hat{M}}^*$.

In effect, increasing the challenge space enhances the protocol's soundness but also allows the prover to introduce a small 'cheating' subgroup (generated by σ in $\mathbb{Z}_{\hat{M}}$). This poses a problem for commitments of the form $c = u^m t^r \bmod \hat{M}$, where m and r are the secret and the randomizer, respectively. Specifically, it may become easy to isolate σ^m from c, thereby leaking the value $m \bmod \omega$, where ω is the size of the cheating subgroup. We explain how to address this issue in Section 2.3.5.

Optimizing computational complexity using small exponents. In this work, we further improve the protocol of [HLM25] by instructing the prover to choose *small exponents* for both λ and α. Specifically, $\lambda \leftarrow \{1, \ldots, 2^{2\ell}\}$ and $\alpha \leftarrow \{1, \ldots, 2^{2\ell+\nu}\}$, where ν is a statistical parameter chosen large enough to hide $e \cdot \lambda$, which has a bit-length of $\ell_0 + 2\ell$ (concretely, $\nu = \ell_0 + 64$ in the actual implementation). This approach results in a significant improvement in computational complexity, as the parties now perform *small* exponentiations (of size $2\ell + \nu$) rather than full-size exponentiations with a bit-length of $\log(\varphi(\hat{M})) \approx \log(\hat{M})$. Concretely, for $\ell = 128$, $\ell_0 = 32$, $\nu = 96$, and $\hat{M} \approx 2^{3072}$, this approach yields an almost 9-fold improvement in computational complexity.
In terms of security, to ensure the binding and knowledge-soundness properties of the Damgård-Fujisaki commitment, our use of small exponents relies on an additional security assumption: the so-called *small-exponent indistinguishability* (SEI) assumption. This assumption states that it is infeasible to distinguish between (v, v^{λ}) and $(v, v^{\lambda'})$ mod $\hat{M}$, where λ is drawn from $\{1, \ldots, 2^{2\ell}\}$ and λ' is drawn from $\{1, \ldots, \varphi(\hat{M})\}$. We discuss this assumption in more detail in the following section.

2.3.4 Systematic use of small-exponents

To further improve the speed of our protocol, we systematically employ small exponents throughout. Namely:

1. The Paillier ciphertext in the 1st step of the OLE is encrypted as $\mathcal{E} \leftarrow \mathrm{enc}_N(x, r^{\beta}) = (1 + xN) \cdot r_0^{\beta} \bmod N^2$ where $\beta \xleftarrow{\$} \{1, \ldots, 2^{2\ell}\}$ for a fixed, known-to-all value $r_0 \in \mathbb{Z}_N^*$ (rather than as $\mathrm{enc}_N(x, \rho^N)$ for $\rho \xleftarrow{\$} \mathbb{Z}_N^*$. This approach yields a 10-fold improvement for typical parameter choices in computing Paillier ciphertexts as the cost is primarily due to computing the randomizer.

2. The Damgård-Fujisaki commitment in step 2 of the OLE is computed as $s_1^{a'} s_2^{b'} t^r \bmod \hat{N}$, where r is chosen large enough to hide $\lambda_1 a' + \lambda_2 b'$. Here, λ_1 and λ_2 are the discrete logarithms of s_1 and s_2 with respect to t. To avoid the issue of the cheating subgroup, we use the small-exponent version of the vanilla Damgård-Fujisaki parameters proof for proving that $(\hat{N}, t, s_1, s_2)$ is well formed. This proof has the added benefit that it serves as a range proof for the λ's, ensuring that $|\lambda_1|, |\lambda_2| < 2^{2\ell+\nu}$. In our application, this allows r to be half the size of the modulus which effectively yields a twofold improvement in computation compared to the computing standard commitment.

3. Similarly, the Paillier ciphertext in step 2 of the OLE is computed as $\mathcal{E}^a(1+b'N) \cdot r_0^{\lambda_0} \bmod N^2$, where λ_0 is chosen large enough to hide $a \cdot \beta$, and β is the value chosen by P_1 in Item 1. This approach requires a range proof on β, which similarly introduces another cheating subgroup (this time for a cheating P_1). We discuss how to address this issue in Section 2.3.5. In the

end, this approach allows r_0 to be one third of the size of the modulus, effectively yielding a threefold improvement in computing $\mathcal{S}$.

4. The small exponent optimization propagates to the zero-knowledge proofs, yielding an improvement in both computation and communication. The ephemeral secrets in the proofs are chosen to be large enough to hide the actual secrets while still smaller than the full range, which benefits computational efficiency. Additionally, since the size of the ZKPs essentially corresponds to the total size of the ephemeral secrets, communication is also improved.

Security of SEI. Our protocols rely on the short-exponent indistinguishability assumption [Gen00; CKLR21]. The SEI assumption was shown in previous work [KK04] to reduce to the (much more standard and well-studied) short-exponent discrete logarithm assumption (SEDL, Definition 3.4) over prime order group. InAppendix A, we extend this reduction to the setting of unknown order groups, thereby showing that the security of our protocol can be based (in addition to other assumptions) on the hardness of the well-studied SEDL assumption (we also provide in the same Appendix a brief overview of the literature on SEDL).

2.3.5 Dealing with the cheating subgroup problem

There are two instances where the protocol faces a cheating subgroup:

1. P_2's tuple $(\hat{M}, v, u_1, u_2)$ is malformed in that $(u_1, u_2) = (\sigma_1 \cdot v^{\lambda_1}, \sigma_2 \cdot v^{\lambda_2}) \bmod \hat{M}$, where σ_1 and σ_2 are small-order elements. This creates an attack opportunity for a malicious P_2, as P_1 may inadvertently leak some bits of both x and β when computing $\mathcal{E} = \mathrm{enc}_N(x; r_0^\beta)$ in step 1 of the OLE as it is accompanied by a commitment $C = u_1^x u_2^\beta v^\rho$. (Recall that C may allow the attacker to isolate σ_1^x and $\sigma_2^\beta \bmod \hat{N}$, thus leaking the value of $x \bmod \omega_1$ and $\beta \bmod \omega_2$ where ω_i denotes the order of σ_i)

2. The ciphertext $\mathcal{E} \in \mathbb{Z}_{N^2}^*$ generated by P_1 is malformed in that $\mathcal{E} = \mathrm{enc}_N(x'; \sigma_0 \cdot r_0^\beta)$ instead of $\mathcal{E} = \mathrm{enc}_N(x'; r_0^\beta)$, where σ_0 is a small-order element, i.e., the randomizer of $\mathcal{E}$ is not a proper power of r_0. This may happen for essentially the same reason mentioned in Section 2.3.3. This creates an attack opportunity for a malicious P_1, as P_2 may inadvertently leak some bits of a when computing $\mathcal{S} = \mathcal{E}^a \cdot \mathrm{enc}_N(b'; r_0^\beta) \bmod N^2$ as $\mathcal{S}$ may allow the attacker to isolate $\sigma_0^a \bmod N^2$, thus leaking the value of $a \bmod \omega_0$ where ω_0 denotes the order of σ_0.

We first explain how to deal with Item 2. By noting that we have a good bound on the size of ω_0, and we only aim to hide the value of $a \bmod q$, our strategy is to add enough noise to a, i.e., by defining $a \leftarrow a + \mu \cdot q$, where μ is chosen randomly from a suitable range, ensuring that $\sigma_0^a \bmod N^2$ and thus $a \bmod \omega_0$ is statistically independent of $a \bmod q$ (which follows by CRT).
For Item 1, we employ a similar trick to hide x. As in the previous case, our aim is hide the value of $x \bmod q$, and it suffices to add enough noise to x, i.e., by defining $x \leftarrow x + \nu \cdot q$, where ν is chosen randomly from a suitable range, ensuring that $\sigma_1^x \bmod \hat{M}$ is statistically independent of $a \bmod q$ (which follows by CRT).

Taking Care of the Leakage of β. This is the hard case to address, as applying a similar trick as above and increasing the size of β beyond $\varphi(N)$ (the total possible range of β) defeats

the purpose of using a small exponent. Furthermore, this leakage affects the randomizer of $\mathcal{E}$, not the plaintext, and thus may in principle break the indistinguishability of Paillier encryption. Clearly, if the leakage is large enough, it becomes easy to extract the plaintext. On the other hand, our intuition suggests that from the adversary's perspective, the randomizer appears as $r_0^{\tilde{\beta}}(r_0^{\omega_2})^{\lfloor\beta/\omega_2\rfloor}$, where $\tilde{\beta} = \beta \bmod \omega_2$ may be known to the adversary but $\lfloor\beta/\omega_2\rfloor$ is hidden and sufficiently large (if β is sufficiently large) so that $(r_0^{\omega_2})^{\lfloor\beta/\omega_2\rfloor}$ is indistinguishable from a random element in $\mathbb{Z}_N^*$. Indeed, this is almost a trivial reduction to the SEI assumption *if the order ω_2 is known to the reduction/simulation.* In our case, the simulator does not know ω_2 (and, even though ω_2 is known to be small, it cannot be bruteforced either as it is computationally infeasible to decide $u_2^j = (\sigma_2 v^{\lambda_2})^j \in \langle v \rangle$ for nontrivial j's.) Consequently, our strategy in the security analysis is to simply guess the order of the group (which is guaranteed to be small), which incurs a small security loss.

Remark 2.4 (On concurrency). We note that our security proof goes through even in the concurrent setting, as we are proving unforgeability of the protocol and thus we only need to simulate for a single public key, and all other instances are simulated by simply running the code of the honest party.

2.3.6 Applications to embedded cryptocurrency wallets

We conclude this technical overview by presenting the application that motivated this work. We envision a single server S (e.g., a cryptocurrency wallet service) maintaining distinct, shared public keys with separate clients $\mathsf{C}_1, \ldots, \mathsf{C}_n$ in a star-shaped topology. To this end, we propose the following template (along the lines of the OLE protocol from the beginning of this section Section 2.3).

Setup. In a preliminary step, done ahead of time, S takes the role of P_1 and runs the parameters generation phase for computing the Paillier key N, the Damgård-Fujisaki tuple $(\hat{N}, t, s_1, s_2)$ along with the relevant zero knowledge proof. (this step is done once and for all for all clients)

Key-Generation. Every time a new client C_i joins, S and C_i run an interactive protocol to compute a fresh ECDSA public key $X^{(i)} \in \mathbb{G}$. At the same time C_i takes the role of P_2 in the OLE protocol, it retrieves and verifies the server's setup material (this is can be done non-interactively, ahead of time), and the parties execute the first step of the OLE so that C_i obtains an encryption $\mathcal{E}_i \in \mathbb{Z}_{N^2}^*$ of the server's secret-key share associated with $X^{(i)}$.

Signing. Every time the client C_i requests to sign a message digest m, S and C_i run Protocol 1 where the OLE is finalized using step 2 of the OLE protocol described at the begining of this section.

3 Preliminaries

3.1 Notation

In this document, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ denote the set of rational, integer and natural numbers, respectively, and $\mathsf{PRIMES}^{>2}$ denotes the set of odd primes. Upper case bold letters $\boldsymbol{X}, \boldsymbol{S}, \ldots$ denote sets and lower case bold letters $\boldsymbol{u}, \boldsymbol{v}, \ldots$ denote random variables. For $a, b \in \mathbb{N}$, we write $a \mid b$ for 'a divides b' and $a \nmid b$ for the negation. We write $\gcd : \mathbb{N}^2 \to \mathbb{N}$ for the greatest common divisor operation, $[a]_q$ denotes the modular reduction operation $a \bmod q$, and $\varphi(\cdot)$ denotes Euler's totient function.

Groups & Fields. For $t \in \mathbb{G}$, we write $\langle t \rangle = \{t^k \in \mathbb{G} \text{ s.t. } k \in \mathbb{Z}\}$ for the group generated by t. $(\mathbb{G}, g, q)$ will denote the group-generator-order tuple for ECDSA. We write $\mathbb{1} \in \mathbb{G}$ for the identity element. and we write $\mathbb{F}_q$ to specify that the field has q elements. For $N \in \mathbb{N}$, we write $\mathbb{Z}_N$ for the quotient group $\mathbb{Z}/N\mathbb{Z}$ and, $\mathbb{Z}_N^*$ and QR_N for the group of units and quadratic residues, respectively. For $\ell \in \mathbb{Z}$, we let $\pm\ell$ denote the interval of integers $\{-|(\ell-1)/2|, \ldots, 0, \ldots, |(\ell-1)/2|\}$ if ℓ is odd and $\{-|\ell/2|+1, \ldots, 0, \ldots, |\ell/2|\}$ otherwise.

Algorithms, Polynomials & Negligible Functions We use sans-serif letters $(\mathsf{enc}, \mathsf{dec}, \ldots)$ or $(\mathsf{S}, \mathsf{A}, \mathsf{C}, \ldots)$ to denote algorithms. We write $x \xleftarrow{\$} \boldsymbol{E}$ or $x \xleftarrow{\$} \boldsymbol{b}$ for sampling x uniformly from a set $\boldsymbol{E}$ or as a sample of $\boldsymbol{b}$ respectively, and $x \xleftarrow{\$} \mathsf{A}$ or $x \xleftarrow{\$} \mathsf{keygen}$ for sampling x according to (probabilistic) algorithms A or keygen respectively—when the algorithm is deterministic or the operation correspond to (non-randomized) assignment, we simply write '$\leftarrow$' instead of '$\xleftarrow{\$}$'. For $f : \mathbb{N} \mapsto \mathbb{R}$ we say that f is polynomially bounded and we write $f \in \mathsf{poly}$ if there exists $c \in \mathbb{N}$ such that $f(\ell) \leq \ell^c$ for all-but-finitely-many ℓ's. Furthermore, for $\varepsilon : \mathbb{N} \mapsto \mathbb{R}$, we write $\varepsilon \in 1/\mathsf{poly}$ if $1/\varepsilon$ is polynomially bounded (i.e. $1/\varepsilon \in \mathsf{poly}$). A function $\nu : \mathbb{N} \mapsto \mathbb{R}$ is negligible if for every $\varepsilon \in 1/\mathsf{poly}$ it holds that $\nu(\ell) \leq \varepsilon(\ell)$ for all-but-finitely-many ℓ's and we write neg for the set of negligible functions.

Distribution Ensembles & Indistinguishability. A distribution ensemble $\{\boldsymbol{v}_\kappa\}_{\kappa \in \mathbb{N}}$ is a sequence of random variables indexed by the natural numbers. We say two ensembles $\{\boldsymbol{v}_\kappa\}$ and $\{\boldsymbol{u}_\kappa\}$ are ε-indistinguishable and we write $\{\boldsymbol{v}_\kappa\} \overset{\varepsilon}{\equiv} \{\boldsymbol{u}_\kappa\}$ if $|\Pr[\mathsf{D}(1^\kappa, \boldsymbol{u}_\kappa) = 1] - \Pr[\mathsf{D}(1^\kappa, \boldsymbol{v}_\kappa) = 1]| \leq \varepsilon(\kappa)$ for every efficient distinguisher D, for all-but-finitely-many κ's. Finally, we write $\mathrm{SD}(\boldsymbol{u}, \boldsymbol{v})$ for the statistical distance of $\boldsymbol{u}$ and $\boldsymbol{v}$, i.e.

$$\mathrm{SD}(\boldsymbol{u}, \boldsymbol{v}) = \sup_{\boldsymbol{W}} |\Pr[\boldsymbol{v} \in \boldsymbol{W}] - \Pr[\boldsymbol{u} \in \boldsymbol{W}]|$$

3.2 Standard Lemmas

Lemma 3.1 (Generalized Forking Lemma [BN06])**.** *Fix a bound $Q \geq 1$ and a set $\boldsymbol{S}$ of size $|\boldsymbol{S}| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that, on input $(x, h_1, \cdots, h_Q)$ returns a pair, the first element of which is an integer in the range $\{0, \cdots, Q\}$ and the second element of which we refer to as a side output. Let D_{IG} be a randomized algorithm that we call the input generator. We define*

$$\varepsilon := \Pr[i^* \geq 1 \mid x \xleftarrow{\$} \mathsf{D}_{\mathsf{IG}}, (h_1, \cdots, h_Q) \xleftarrow{\$} \boldsymbol{S}^Q, (i^*, \mathsf{so}) \xleftarrow{\$} \mathcal{A}(x, h_1, \cdots, h_Q)],$$

and call ε the accepting probability *of $\mathcal{A}$. Then,*

$$\Pr[b = 1 \mid x \xleftarrow{\$} \mathsf{D}_{\mathsf{IG}}, (b, \mathsf{so}, \mathsf{so}') \xleftarrow{\$} \mathsf{F}_{\mathcal{A}}(x)] \geq \frac{\varepsilon^2}{Q} - \frac{\varepsilon}{|\boldsymbol{S}|},$$

where $\mathsf{F}_{\mathcal{A}}$ denotes the forking algorithm *associated to $\mathcal{A}$ and represented on Algorithm 2.*

Algorithm $\mathsf{F}_{\mathcal{A}}(x)$

PARAMETERS. An integer $Q \geq 1$. A set $\boldsymbol{S}$ of size $|\boldsymbol{S}| \geq 2$. A bound $\mathsf{T}_{\mathcal{A}}$ on the length of the random tape used by $\mathcal{A}$.

OPERATIONS.

1. Sample coins $\rho \xleftarrow{\$} \{0,1\}^{\mathsf{T}_{\mathcal{A}}}$ for $\mathcal{A}$.
2. Sample $(h_1, \cdots, h_Q) \xleftarrow{\$} \boldsymbol{S}^Q$.
3. $(i, \mathsf{so}) \xleftarrow{\$} \mathcal{A}(x, h_1, \cdots, h_Q; \rho)$
4. If $i = 0$, return $(0, \perp, \perp)$.
5. Sample $(h'_i, \cdots, h'_Q) \xleftarrow{\$} \boldsymbol{S}^{Q-i+1}$.
6. $(i', \mathsf{so}') \xleftarrow{\$} \mathcal{A}(x, h_1, \cdots, h_{i-1}, h'_i, \cdots, h'_Q; \rho)$.
7. If $i = i'$ and $h_i \neq h'_i$, return $(1, \mathsf{so}, \mathsf{so}')$.
8. Else, return $(0, \perp, \perp)$.

Algorithm 2: Forking algorithm associated to $\mathcal{A}$.

3.3 Security Assumptions

In the following, we denote by $\mathsf{SampleRSA}(1^\ell)$ an algorithm that returns an RSA modulus $N = p \cdot q$ where $p = q = 3 \bmod 4$, $\gcd(p-1, q-1) = 2$ and $|p| = |q| \geq n_{\mathrm{rsa}}/2 - 1$. Looking ahead, our protocols will rely on security assumptions with respect to an RSA modulus with a specific structure. We cover the concrete RSA sampling algorithm $\mathsf{SampleRSA}(1^\ell)$ used in our work in Section 4.1.

Definition 3.2 (FAC)**.** Define distribution ensemble FAC such that $N \xleftarrow{\$} \mathsf{FAC}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$. We say that *hardness of factoring holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{N \xleftarrow{\$} \mathsf{FAC}(1^\ell)} \left[(p, q) \xleftarrow{\$} \mathsf{A}(1^\ell, N) \text{ s.t. } pq = N \right] \leq \mu(\ell).$$

Definition 3.3 (SRSA)**.** Define distribution ensemble SRSA such that $(N, c) \xleftarrow{\$} \mathsf{SRSA}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$ and $c \xleftarrow{\$} \mathbb{Z}_N^*$. We say that *strong-RSA holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,c) \xleftarrow{\$} \mathsf{SRSA}(1^\ell)} \left[(e, x) \xleftarrow{\$} \mathsf{A}(1^\ell, N, c) \text{ s.t. } c = x^e \bmod N \wedge e \notin \{-1, 1\} \right] \leq \mu(\ell).$$

Definition 3.4 (SEDL over $\mathbb{Z}_N^*$)**.** Define SEDL such that $(N, t, [t^x]_N) \xleftarrow{\$} \mathsf{SEDL}(1^\ell)$ for $(N; p, q) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $t \xleftarrow{\$} \mathsf{QR}_N$ and $x \xleftarrow{\$} \pm 2^{2\ell}$. We say that *small-exponent discrete-log (SEDL) holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,t,s) \xleftarrow{\$} \mathsf{SEDL}(1^\ell)} \left[x_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, t, s) \text{ s.t. } t^{x_0} = s \bmod N \right] \leq \mu(\ell).$$

Definition 3.5 (SEI over $\mathbb{Z}_N^*$)**.** Define distribution ensemble SEI such that $(N, t, [t^{\alpha x+(1-\alpha)y}]_N, \alpha) \xleftarrow{\$} \mathsf{SEDL}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $t \xleftarrow{\$} \mathsf{QR}_N$, $x \xleftarrow{\$} \pm 2^{2\ell}$, $y \xleftarrow{\$} \pm N$ and $\alpha \xleftarrow{\$} \{0,1\}$. We say that *small-exponent indistinguishability (SEI) holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,t,s,\alpha)\xleftarrow{\$}\mathsf{SEI}(1^\ell)}\Big[\alpha_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, t, s) \text{ s.t. } \alpha_0 = \alpha\Big] \leq \frac{1}{2} + \mu(\ell).$$

3.3.1 DCR Variants

Definition 3.6 (DCR)**.** Define DCR such that $(N, [\rho^{\alpha N} \cdot c^{1-\alpha}]_{N^2}, \alpha) \xleftarrow{\$} \mathsf{DCR}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $\rho \xleftarrow{\$} \mathbb{Z}_N^*$, $c \xleftarrow{\$} \mathbb{Z}_{N^2}^*$ and $\alpha \xleftarrow{\$} \{0,1\}$. We say that *decisional composite residuosity (DCR) holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,x,\alpha)\xleftarrow{\$}\mathsf{DCR}(1^\ell)}\Big[\alpha_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, x) \text{ s.t. } \alpha_0 = \alpha\Big] \leq \frac{1}{2} + \mu(\ell).$$

Definition 3.7 (QRDCR)**.** Define QRDCR such that $(N, [\rho^{\alpha N} \cdot c^{1-\alpha}]_{N^2}, \alpha) \xleftarrow{\$} \mathsf{QRDCR}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $\rho \xleftarrow{\$} \mathsf{QR}_N$, $c \xleftarrow{\$} \mathsf{QR}_{N^2}$ and $\alpha \xleftarrow{\$} \{0,1\}$. We say that QRDCR holds true if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,x,\alpha)\xleftarrow{\$}\mathsf{QRDCR}(1^\ell)}\Big[\alpha_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, x) \text{ s.t. } \alpha_0 = \alpha\Big] \leq \frac{1}{2} + \mu(\ell).$$

Definition 3.8 (SEDCR)**.** Define SEDCR such that $(N, \rho_0, [\rho_0^{\alpha\lambda N} \cdot c^{1-\alpha}]_{N^2}, \alpha) \xleftarrow{\$} \mathsf{SEDCR}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $\rho_0 \xleftarrow{\$} \mathsf{QR}_N$, $\lambda \xleftarrow{\$} \pm 2^{2\ell}$, $c \xleftarrow{\$} \mathsf{QR}_{N^2}$ and $\alpha \xleftarrow{\$} \{0,1\}$. We say that *small-exponent DCR holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N,\rho_0,x,\alpha)\xleftarrow{\$}\mathsf{SEDCR}(1^\ell)}\Big[\alpha_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, \rho_0, x) \text{ s.t. } \alpha_0 = \alpha\Big] \leq \frac{1}{2} + \mu(\ell).$$

For any PPTM A, let $\mathsf{Adv}_{\mathsf{A}}^{\mathsf{sedcr}}(1^\ell)$ denote the advantage of A against the small-exponent DCR assumption.

Claim 3.9. *DCR implies QRDCR.*

Proof. It is enough to show that any distinguisher for QRDCR can be transformed into a distinguisher for DCR. Let $(N, x, \alpha) \xleftarrow{\$} \mathsf{DCR}(1^\ell)$ and let $x' = x^2 \bmod N^2$. We argue that (N, x', α) is identically distributed with the QRDCR distribution. First, notice that, for $x = (1+mN)\cdot\rho^N \bmod N$, it holds that $x' = (1+m'N)\cdot\rho'^N$ where $m' = 2m \bmod N$ and $\rho' = \rho^2 \bmod N$. So, since N is a Blum number (i.e., its factors are congruent to 3 mod 4) and thus squaring is an automorphism in QR_N, it follows that x' is a N-th power of a random square if $\alpha = 1$, or x' is a random element in the group QR_{N^2} (which is isomorphic to $\mathbb{Z}_N \times \mathsf{QR}_N$) as N is odd. □

Proof. For $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $\rho_0 \xleftarrow{\$} \mathsf{QR}_N$, define the following intermediate distributions.

1. (N, ρ_0, c), for $c \xleftarrow{\$} \mathsf{QR}_{N^2}$
2. $(N, \rho_0, [\rho_0^{\mu N}]_{N^2})$, for $\mu \xleftarrow{\$} \pm N$
3. $(N, \rho_0, [\rho_0^{\lambda N}]_{N^2})$, for $\lambda \xleftarrow{\$} \pm 2^\ell$.

Under the the assumpion that ρ_0 generates QR_N, notice the following. Since $x \mapsto x^N \bmod N^2$, is an injective homomorphism it follows that Items 2 and 3 are indistinguishable under SEI. Furthermore, $\rho_0^\mu \bmod N$ is $(1 - \varphi(N)/N)$-close to $\rho \bmod N$ for $\rho \xleftarrow{\$} \mathbb{Z}_{N^*}$, so Items 1 and 2 are indistinguishable under QRDCR. To conclude, since N is a tough-prime modulus, the probability that $\langle \rho_0 \rangle \neq \mathsf{QR}_N$ is at most $\left\lceil \frac{n_{\mathrm{rsa}}}{2\ell} \right\rceil \cdot \frac{1}{2^{2\ell}}$ where n_{rsa} denotes the bit-length of N. □

3.3.2 Damgård-Fujisaki to Strong-RSA reduction

Definition 3.10 (Tough RSA moduli)**.** Let $N = pq$ denote an RSA modulus. For $\ell \in \mathbb{N}$, we say that N is 2^ℓ-tough if $\gcd(p-1, q-1) = 2$ and $u \mid (p-1)/2$ (resp. $u \mid (q-1)/2$) implies $u > 2^\ell$.

Fact 3.11. *Let N denote an tough RSA modulus and let $g \in \mathsf{QR}_N$ such that $\langle g \rangle = \mathsf{QR}_N$. It holds that $\mathrm{SD}(\boldsymbol{g}, \boldsymbol{g}_0) \leq 1 - \frac{\varphi(N)}{N-1}$ for $\boldsymbol{g}$ and $\boldsymbol{g}_0$ defined below.*

1. *g^γ for $\gamma = \alpha + \beta \cdot \varphi(N)/4$ where $\alpha \xleftarrow{\$} [\varphi(N)/4]$ and $\beta \xleftarrow{\$} \{0,1\}$.*
2. *g^{γ_0} for $\gamma_0 \xleftarrow{\$} [(N-1)/2]$.*

Lemma 3.12. *Let N denote a 2^ℓ-tough RSA modulus and let $g \in \mathsf{QR}_N$ such that $\langle g \rangle = \mathsf{QR}_N$. Suppose there exists PPTM A such that for some $\varepsilon \geq 0$:*

$$\Pr_{h,f \xleftarrow{\$} \mathsf{QR}_N}\left[(x,y,z,e,C) \xleftarrow{\$} \mathsf{A}(N,g,h,f) \;\; s.t. \;\; g^x h^y f^z = C^e \wedge |e| < 2^\ell \wedge e \nmid y, z\right] \geq \varepsilon$$

Then, there exists PPTM B such that

$$\Pr\left[(c,R) \xleftarrow{\$} \mathsf{B}(N,g) \;\; s.t. \;\; g = R^c \wedge c \notin \{-1,1\}\right] \geq \varepsilon \cdot \left(2 \cdot \frac{\varphi(N)}{N-1} - \frac{3}{2}\right)$$

Proof. Define algorithm B as follows:

1. Sample $\alpha, \beta \xleftarrow{\$} [(N-1)/2]$ and set $h = g^\alpha$ and $f = g^\beta \bmod N$.
2. Execute A on input (N, g, h, f); obtain (x, y, z, e, C). Set $s = x + \alpha y + \beta z$ and let u, v denote the Bézout coefficients of e and $-s$, i.e. $ue - vs = \gcd(e, s)$.
3. Output $(R, \hat{R}, c) = (g^u C^{-v}, g^u (-C)^{-v}, e/\gcd(e,s))$ and $A = g^{s/\gcd(e,s)} \cdot C^{-e/\gcd(e,s)}$.

We begin by observing that if $e \nmid s$, then either $g = R^c$ or $g = \hat{R}^c$ and $c \notin \{-1, 1\}$, or A is a nontrivial root of 1 (which yields a suitable pair (R, c) via the factorization of N). Let $e' = e/\gcd(e,s) \notin \{-1, 1\}$ and $s' = s/\gcd(e,s)$. Fix ζ to be one of the two nontrivial root of 1. Since $p = q = 3 \bmod 4$, we know that $-1 \notin \mathsf{QR}_p, \mathsf{QR}_q$ and there exists a unique triple $(C_0, i, j) \in \mathsf{QR}_N \times \{0,1\}^2$ such that $C = C_0 \cdot (-1)^i \cdot \zeta^j$. Furthermore, since N is 2^ℓ-tough and $\gcd(e,s) \leq e < 2^\ell$, we have that $g^{s'} = C_0^{e'} \bmod N$. Consequently, either (i) $A = g^{s'} C^{-e'} \in \{\zeta, -\zeta\}$, or (ii) $g^{s'} C^{-e'} = -1$ and e' is

odd, or (iii) $g^{s'}C^{-e'} = 1$. Case (i) yields a non-trivial root of 1. For Cases (ii) and (iii), we deduce that $\hat{R}^c = g$ and $R^c = g$, respectively, since

$$\begin{cases} \hat{R}^c = \hat{R}^{e'} = g^{ue'}(-C)^{-ve'} = g^{ue'}g^{-vs'} = g^{ue'-vs'} = g \bmod N & \text{in Case (ii).} \\ R^c = R^{e'} = g^{ue'}C^{-ve'} = g^{ue'}g^{-vs'} = g^{ue'-vs'} = g \bmod N & \text{in Case (iii).} \end{cases}$$

Let $\varepsilon_0 = 1 - \varphi(N)/(N-1)$. In the remainder, we argue that the probability that $e \mid s$ is bounded above by $1/2 + 2\varepsilon_0$. Using the notation above, consider (inefficient) algorithm $\hat{\mathsf{B}}$ which similar to B except that (i) instead of sampling α, β uniformly in $[(N-1)/2]$ as in Item 1, $\hat{\mathsf{B}}$ samples $\alpha_0, \beta_0 \xleftarrow{\$} \varphi(N)/4$ and sets $(h, f) = (g^{\alpha_0}, f^{\beta_0})$, (ii) When obtaining (x, y, z, e, C) from A, $\hat{\mathsf{B}}$ samples $\mu, \nu \xleftarrow{\$} \{0,1\}$ and sets $\alpha = \alpha_0 + \mu \cdot \varphi(N)/4$ and $\beta = \beta_0 + \nu \cdot \varphi(N)/4$, (iii) all other values are set according to B. In the sequel, we consider a random execution of $\hat{\mathsf{B}}$.

Let $\boldsymbol{d}_{\mu,\nu}$ denote the event that e divides $x + (\alpha_0 + \mu \cdot \varphi(N)/4) \cdot y + (\beta_0 + \nu \cdot \varphi(N)/4) \cdot z$. Notice that if $\boldsymbol{d}_{0,\mu}$ and $\boldsymbol{d}_{1,\mu}$ both occur, then e divides $\varphi(N)/4 \cdot y$, and since N is 2^ℓ-tough and $e < 2^\ell$, then $e \mid y$. Similarly, if $\boldsymbol{d}_{\nu,0}$ and $\boldsymbol{d}_{\nu,1}$ both occur, then $e \mid z$. Thus, if $e \nmid y$ or z then $\Pr[e \mid s] \leq \frac{1}{2}$ in a random execution of $\hat{\mathsf{B}}$.

Finally, by noting that $(\alpha, \beta) \xleftarrow{\$} \hat{\mathsf{B}}$ and $(\alpha, \beta) \xleftarrow{\$} \mathsf{B}$ are statistically, $(2\varepsilon_0)$-close by Fact 3.11, and thus A's view in an execution of $\hat{\mathsf{B}}$ is statistically, $(2\varepsilon_0)$-indistinguishable from A's view in an execution of B, we deduce that $\Pr[e \mid s] \leq 1/2 + 2\varepsilon_0$ in a random execution of B. □

Corollary 3.13. *The following holds under strong RSA and SEI. For every PPTM* A*, there exists a negligible function* $\mu : \mathbb{N} \to \mathbb{R}$ *such that:*

$$\Pr_{\substack{N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell) \\ g \xleftarrow{\$} \mathsf{QR}_N, (\alpha,\beta) \xleftarrow{\$} \pm 2^{2\ell}}} \left[(x, y, z, e, C) \xleftarrow{\$} \mathsf{A}(1^\ell, N, g, g^\alpha, g^\beta) \;\; s.t. \;\; g^{x+\alpha y+\beta z} = C^e \wedge |e| < 2^\ell \wedge e \nmid y, z \right] \leq \mu(\ell).$$

3.4 Paillier Encryption, Damgård-Fujisaki & Pedersen Commitments

Definition 3.14. Paillier Encryption consists of the following algorithms.

1. *KeyGen.* On input $\ell \in \mathbb{N}$, sample $(N, n_{\text{rsa}}; p, q) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$.
   Return $(N; \varphi(N))$. (Optionally, output $\rho_0 \xleftarrow{\$} \mathbb{Z}_N^*$.)
2. *Encrypt.* On input $m \in \mathbb{Z}_N$ and N, sample $\rho \xleftarrow{\$} \mathbb{Z}_N^*$ and output
   $$\mathcal{C} = \text{enc}_N(m; \rho) = (1 + mN) \cdot \rho^N \bmod N^2.$$
3. *Decrypt.* On input $\mathcal{C} \in \mathbb{Z}_{N^2}^*$ and $\varphi(N)$, output $m = \left(\frac{[\mathcal{C}^{\varphi(N)}]_{N^2} - 1}{N} \right) \cdot \varphi(N)^{-1} \bmod N$

In this work, we will consider a small-exponent variant of Paillier Encryption, where ρ is sampled as $\rho_0^{\lambda N}$ for $\lambda \xleftarrow{\$} \pm 2^{2\ell}$. Note that via a straightfoward reduction, the IND-CPA security of small-exponent Paillier reduces to the SEDCR assumption.[3] For any PPTM A, we let $\mathsf{Adv}^{\mathsf{secpa}}_{\mathsf{A}}(1^\ell)$ denote the advantage of A against the IND-CPA security of the small-exponent Paillier Encryption.

[3]The reduction is perfectly tight for the real-or-random IND-CPA game, and looses a factor 2 in the advantage for the standard IND-CPA game.

Definition 3.15. Damgård-Fujisaki commitments in RSA groups consist of the following algorithms.

1. *KeyGen.* On input $\ell \in \mathbb{N}$, sample $(\hat{N}, n_{\text{rsa}}, t, s_1, s_2) \xleftarrow{\$} \mathsf{SampleDF}(1^\ell)$.
   (Optionally, output λ_1, λ_2 such that $t^{\lambda_1} = s_1$ and $t^{\lambda_2} = s_2 \bmod \hat{N}$.)
2. *Commit.* For $m_1, m_2 \in \mathbb{Z}$ and $(\hat{N}, t, s_1, s_2)$, sample $\rho \xleftarrow{\$} [\hat{N} \cdot 2^\ell]$ and output
$$\widetilde{C} = s_1^{m_1} s_2^{m_2} \cdot t^\rho \bmod \hat{N}.$$

Definition 3.16. Pedersen Commitments in a known-order group $(\mathbb{G}, g, h, f, q)$ where $\langle g \rangle = \langle h \rangle = \langle f \rangle = \mathbb{G}$ and $|\mathbb{G}| = q$ consist of the following algorithm.

Commit. On input $m_1, m_2 \in \mathbb{Z}$ and $(\mathbb{G}, g, h, f, q)$, sample $\rho \xleftarrow{\$} [q]$ and output
$$\widetilde{C} = g^{m_1} h^{m_2} f^\rho \in \mathbb{G}$$

3.5 Zero-Knowledge Proofs

Our protocol makes uses NIZK arising from Fiat-Shamir-compiled zero knowledge protocols for which we give a refresher. Let $\boldsymbol{R} = \{(X; w)\}$ denote an NP-relation. We define Σ-protocol Π a four-tuple of algorithms $(\mathsf{S}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ parametrized by $\boldsymbol{E}$ and $\boldsymbol{T}$ such that: (i) $\sigma \xleftarrow{\$} \mathsf{S}(1^\ell)$ denotes the setup parameter. (ii) $A \leftarrow \mathsf{P}_1(\sigma, X; w, \tau)$ denotes the provers' first message where $\tau \in \boldsymbol{T}$. (iii) $z \xleftarrow{\$} \mathsf{P}_2(\sigma, X, e; w, \tau)$ denotes the provers' second message where $e \in \boldsymbol{E}$. (iv) $\beta \leftarrow \mathsf{V}(\sigma, X, A, e, z)$ denotes the verifiers' acceptance/rejection. The tuple (σ, X, A, e, z) is called the transcript of the protocol. Additionally Π is secure if it satisfies the soundness and zero-knowledge properties with respect to $\boldsymbol{R}$, outlined in Definition 5.1.

3.5.1 NIZK and the Fiat-Shamir Transform

We use the Fiat-Shamir transform for compiling Σ-protocols into NIZKs in the random oracle model. In particular, for a ZK protocol Π with common input X and secret input x, we write $\psi \xleftarrow{\$} \Pi^{\text{FS}}(\text{aux}, X; x)$ for the transcript of the protocol resulting from the Fiat-Shamir transform. Namely, the verifier's second round message is calculated as $e = \mathtt{HS}(\text{aux}, X, A)$, where A denotes the prover's first-round message, aux denotes auxiliary contextual data and $\mathtt{HS}$ denotes the random oracle. We write $\mathsf{vrfy}_{\Pi^{\text{FS}}}(\text{aux}, x, \pi)$ for the output of the verification algorithm associated with Π^{FS} on common input X, proof ψ and auxiliary input aux. The ZK protocols for the relations below are specified in Section 5.

3.5.2 NP Relations

Our protocols rely on zero-knowledge proofs for the following NP relations:

Definition 3.17 (Well-Formed Modulus)**.** Define $\boldsymbol{R}_{\mathsf{mod}} \ni (N; p_1, p_2)$ such that $N = p_1 \cdot p_2$ where $p_1, p_2 \in \mathsf{PRIMES}^{>2}$, $(p_1, p_2) = (3, 7) \bmod 8$ and $\gcd(\varphi(N), N) = 1$.

Definition 3.18 (Small Factors)**.** Define $\boldsymbol{R}_{\mathsf{fac}} \ni (N, \ell; u, v)$ such that $N = u \cdot v$ and $u, v \in \pm\sqrt{N} \cdot 2^\ell$.

Definition 3.19 (Damgård-Fujisaki Params.)**.** Define $\boldsymbol{R}_{\mathsf{df}} \ni (\hat{N}, t, s_1, s_2; \lambda_1, \lambda_2)$ such that

$$t \in \mathbb{Z}^*_{\hat{N}}, \quad \text{and} \quad s_i = t^{\lambda_i} \bmod \hat{N}.$$

Definition 3.20 (Damgård-Fujisaki-*star* Params.)**.** Define $\boldsymbol{R}_{\mathsf{df}^*} \ni (\hat{N}, t, s_1, s_2, d; \lambda_1, \lambda_2, \omega, \sigma_1, \sigma_2)$:

$$t \in \mathbb{Z}^*_{\hat{N}}, \quad s_i = \sigma_i \cdot t^{\lambda_i} \bmod \hat{N}, \quad \sigma_1^{\omega} = \sigma_2^{\omega} = 1 \bmod \hat{N} \quad \text{and} \quad \omega \leq 2^d$$

Definition 3.21 (Discrete Log)**.** Define $\boldsymbol{R}_{\mathsf{dlog}} \ni (\mathbb{G}, g, q, X; w)$ such that $g^w = X \in \mathbb{G}$.

Definition 3.22 (Encrypted Discrete Log)**.** Define $\boldsymbol{R}_{\mathsf{dlenc}} \ni (\mathbb{G}, g, q, N, \mathcal{C}, \rho, n_x, n_\lambda, X, \ell; x, \lambda, \ell', \sigma)$

$$\text{s.t. } x \in \pm 2^{n_x}, \lambda \in \pm 2^{n_\lambda} \text{ and } \begin{cases} g^x = X \in \mathbb{G} \\ \mathcal{C} = \mathrm{enc}_N(x; \rho^\lambda \cdot \sigma^N) \in \mathbb{Z}^*_{N^2} \\ (\sigma^N)^{\ell'} = 1 \bmod N^2 \text{ and } \ell' \leq \ell \end{cases}.$$

Definition 3.23 (Well-Formed Signature)**.** Define $\boldsymbol{R}_{\mathsf{sig}} \ni (N, \rho, \mathcal{E}, \mathcal{S}, n_a, n_b; a, b, \eta, \mu)$ for

$$a \in \pm 2^{n_a}, b \in \pm 2^{n_b} \text{ and } \begin{cases} \mathcal{S}, \mathcal{E} \in \mathbb{Z}^*_{N^2} \\ \mathcal{S} = \mathrm{enc}_N(a; \rho^\eta) \cdot \mathcal{E}^b \bmod N^2 \in \mathbb{Z}^*_{N^2} \end{cases}$$

3.6 The EC-Generic Group Model

Let $\boldsymbol{G}$ denote the labels of the elements in the prime-order elliptic curve $(\mathbb{G}, g, q)$. As $\mathbb{G}$ denotes an elliptic curve, every $(A, A^{-1}) \in \mathbb{G}^2$ has the form $((\alpha, 0), (\alpha, 1))$ or $((\alpha, 1), (\alpha, 0)) \in \boldsymbol{G}^2$. Recall the projection function $\mathcal{X} : \boldsymbol{G} \to \mathbb{F}_q$ such that $\mathcal{X}(H) = \eta \in \mathbb{F}_q$ for $H \in \boldsymbol{G}$ of the form $(\eta, 0)$ or $(\eta, 1)$. Notice that this map is efficiently invertible $\mathcal{X}^{-1} : \mathbb{F}_q \to \{\{G, H\} \text{ s.t. } G, H \in \boldsymbol{G}\} \cup \{\perp\}$ such that

$$\mathcal{X}^{-1} : x \mapsto \begin{cases} \{H, H^{-1}\} & \text{if } \exists H \text{ s.t. } \mathcal{X}(H) = x \\ \perp & \text{otherwise} \end{cases}.$$

We Groth and Shoup's model of the *elliptic-curve generic group model* for prime-order curves (EC-GGM) [GS22]: the generic group is defined via a random bijective map $\pi : \mathbb{G} \to \boldsymbol{G}$ that preserves $\mathcal{X}()$ and a group-oracle $\mathsf{O} : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $\pi(gh) = \mathsf{O}(\pi(g), \pi(h))$ and π preserves the invariant $\mathcal{X}(\pi(g)) = \mathcal{X}(\pi(g^{-1}))$, for every g, $h \in \mathbb{G}$. In group-theoretic jargon, $(\boldsymbol{G}, *)$ is isomorphic to $(\mathbb{G}, \cdot)$ via the group-isomorphism π, letting $* : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $G * H = \mathsf{O}(G, H)$. The participants can sample uniformly from $\boldsymbol{G}$ and efficiently decide membership in $\boldsymbol{G}$, and they have oracle access to the group-oracle O.

4 2-Party Threshold ECDSA

In this section, we introduce our two-party threshold ECDSA protocol. The main protocol comprises three parts:

1. A *non-interactive setup*, executed once and for all by the server, and reused across all its key generation and signing protocols with different clients. In this phase, the server generates a Paillier modulus and Damgård-Fujisaki parameters, and proves their well-formedness. Additionally, each client should verify the setup before interacting with the server.

2. An *interactive key-generation*, where the server and a client interactively generate an ECDSA public key $X \in \mathbb{G}$ together with additive shares x_1, x_2 of the corresponding secret key $x \in \mathbb{Z}_q$, and a Paillier encryption of x_2.

3. A *signing* protocol, where the server and the client interactively generate a signature for a message m with secret key x, using the shares and the Paillier encryption generated during key-generation to homomorphically compute the signature.

All parts rely on non-interactive zero-knowledge proofs obtained by compiling Σ-protocols with the Fiat-Shamir transform. Concretely,

- The non-interactive setup (Section 4.2) uses proofs $\pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}$, and π_{df} proving respectively the well-formedness of the Paillier modulus (Definition 3.17), the smallness of its factor (Definition 3.18), and the correctness of the Damgård-Fujisaki parameters (Definition 3.19).
- The interactive key-generation (Section 4.3) uses proofs $\psi_{\mathsf{df}^*}, \psi_{\mathsf{dlog}}$, and ψ_{dlenc} proving respectively (Definition 3.19), knowledge of discrete logarithm (Definition 3.21), and the encrypted discrete-logarithm relation (Definition 3.22) respectively.
- the signing protocol (Section 4.4) uses the proof ψ_{sig} proving well-formedness of the signature (Definition 3.23).

All the above zero-knowledge proofs except ψ_{dlog} and ψ_{sig} are proofs of membership in the corresponding language. In contrast, ψ_{dlog} and ψ_{sig} are *proofs of knowledge*. We note that our protocols can be executed concurrently, in which case the Fiat-Shamir transform does not suffice to compile Σ-protocols into proofs of knowledge: instead, the Fischlin transform [Fis05] or the Kondi-shelat transform [Ks22] should be used. However, these alternatives would incur a significant overhead. In line with our philosophy of seeking the fastest possible protocol, and given that our protocols rely anyway on the fact that ECDSA satisfies doubly-enhanced unforgeability in the generic group model, we adopt a different strategy: we show that the plain Fiat-Shamir transform applied to ψ_{dlog} and ψ_{sig} yields NIZKs which are *straight-line extractable in the generic group model* (unconditionally and under the strong-RSA assumption, respectively).

4.1 Sampling the RSA modulus

In this section, we cover our algorithm for sampling the RSA modulus used in our protocols, together with parameters (v, u_1, u_2) for the Damgård-Fujisaki (multi-)integer commitment scheme, see Algorithm 3 below. We also denote by $\mathsf{SampleRSA}(1^\ell, n_{\mathrm{rsa}})$ the algorithm that executes only the steps 1 to 5 of Algorithm 3 and outputs $(N; p, q)$.

Description of the algorithm. The algorithm $\mathsf{SampleDF}$ samples parameters for the Damgård-Fujisaki integer commitment scheme over an RSA group. In more details, $\mathsf{SampleDF}(1^\ell, n_{\mathrm{rsa}})$ outputs:

Public parameters. An RSA modulus $N = p \cdot q$ where $p = q = 3 \bmod 4$, $\gcd(p-1, q-1) = 2$, and each of $(p-1)/2, (q-1)/2$ is a product of $T = \lceil n_{\mathrm{rsa}}/4\ell \rceil$ 2ℓ-bit random primes. A basis $v \in \mathbb{Z}_N^*$, and two elements $u_1, u_2 \in \langle v \rangle$.

Secret parameters. The factors (p, q) of N, and 2ℓ-bit exponents (λ_1, λ_2) such that $u_i = v^{\lambda_i}$ for $i = 1, 2$.

Subroutines. The algorithm uses the following subroutines:

- We write $p \leftarrow \mathsf{Prime}_1(2\ell)$ to denote the sampling of a uniformly random 2ℓ-bit prime $p = 1 \bmod 4$.
- $\mathsf{isPrime}(1^\ell, p)$ is a probabilistic procedure that, on input p, returns a bit b such that:
  1. If $p \in \mathsf{PRIMES}_{>2}$, $\mathsf{isPrime}(1^\ell, p) = 1$.
  2. If $p \notin \mathsf{PRIMES}_{>2}$, $\Pr[\mathsf{isPrime}(1^\ell, p) = 0] \geq 1 - 1/2^\ell$ (where the probability is over the randomness of $\mathsf{isPrime}$).

Algorithm $\mathsf{SampleDF}(1^\ell)$

PARAMETERS.

- $n_{\mathrm{rsa}} \in \mathbb{N}$ is a function of ℓ and C denotes a constant.
- $T \leftarrow \lceil n_{\mathrm{rsa}}/4\ell \rceil$ and let B be such that $\binom{B}{T} > C \cdot n_{\mathrm{rsa}}$.

OPERATIONS. Set $N \leftarrow 1$.

1. Sample $(p_1, \cdots, p_B) \xleftarrow{\$} \mathsf{Prime}(2\ell)$.
2. **For each** size-T subset S of $\{1, \cdots, B\}$,
   - $p \leftarrow 2 \cdot \prod_{i \in S} p_i + 1$
   - If $\mathsf{isPrime}(1^\ell, p)$, set $\mathsf{foundprime} \leftarrow 1$, then **break**.
3. If $\mathsf{foundprime} = 0$, **go to** Item 1.
4. $N \leftarrow N \cdot p$.
5. If $\log N < n_{\mathrm{rsa}}$, set $q \leftarrow p$ and **go to** Item 1.
6. Sample $v \xleftarrow{\$} \mathsf{QR}_N$ and $(\lambda_1, \lambda_2) \xleftarrow{\$} [2^{2\ell}]$.
7. Set $(u_1, u_2) \leftarrow (v^{\lambda_1}, v^{\lambda_2})$.
8. **Output** $(N, v, u_1, u_2; p, q, \lambda_1, \lambda_2)$.

Algorithm 3: Algorithm for sampling a tough RSA modulus N and Damgård-Fujisaki parameters

4.2 Non-Interactive Setup

This section covers the non-interactive setup phase of our 2-party threshold ECDSA protocol. The main purpose of this protocol is to generate a Paillier modulus and Damgård-Fujisaki parameters together with proofs of correct generation. The non-interactive setup is the heavy bulk of our protocol, but a main advantage of our approach is that it can be executed *once and for all* by the server, and reused across distributed key-generations with arbitrarily many clients.

Algorithm Generate Auxiliary Signing Material

Group-generator-order. $(\mathbb{G}, g, q)$.

Input. Security parameter ℓ, $\text{aux} = (sid, \mathsf{S})$.

Server Operations (S).

1. Sample $(N, n_{\text{rsa}}; p_1, p_2) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$ s.t. $(p_1, p_2) = (3, 7) \bmod 8$.
   - Generate Paillier-Blum well-formedness proof $\pi_{\mathsf{mod}} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{mod}}(\text{aux}, N; p_1, p_2)$.
   - Generate small factors proof $\pi_{\mathsf{fac}} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{fac}}(\text{aux}, N; p_1, p_2)$
   - Sample $\rho_0 \xleftarrow{\$} \mathbb{Z}^*_N$ and set $\rho = \rho_0^{2N} \bmod N^2$.
2. Sample a Damgård-Fujisaki tuple $(\hat{N}, n_{\text{rsa}}, t, s_1, s_2; \hat{p}, \hat{q}, \lambda_1, \lambda_2) \xleftarrow{\$} \mathsf{SampleDF}(1^\ell, n)$.
   - Generate Damgård-Fujisaki params proof $\pi_{\mathsf{df}} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{df}}(\text{aux}, \hat{N}, t, s_1, s_2; \lambda_1, \lambda_2)$.
3. Generate $h, f \in \mathbb{G}$ as nothing-up-my-sleeve numbers using $(\mathbb{G}, g, q)$ and aux as a seed.

   Public Output. $\mathsf{SetupData} = (N, \hat{N}, \rho_0, \rho, t, s_1, s_2, h, f, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{df}})$.

   Secret Output. $\varphi(N)$

Client Operations (C). When obtaining $\mathsf{SetupData} \in \{0,1\}^*$, do:

1. Parse $\mathsf{SetupData} = (N, \hat{N}, \rho_0, \rho, t, s_1, s_2, h, f, \pi^{\mathsf{mod}}, \pi^{\mathsf{fac}}, \pi^{\mathsf{ped}})$.

   Let $n_{\text{rsa}} \leftarrow \mathsf{SampleRSA}(1^\ell)$.
2. Verify that $N, \hat{N} \geq 2^{n_{\text{rsa}}}$, $\rho_0 \in \mathbb{Z}^*_N$, $\rho = \rho_0^{2N} \bmod N^2$ and $t, s_1, s_2 \in \mathbb{Z}^*_{\hat{N}}$.
3. Verify that $h, f \in \mathbb{G}$ can be reconstructed according to Item 3 of Algorithm 4.
4. Verify the ZK proofs:
   (a) $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{mod}}}(\text{aux}, N, \pi_{\mathsf{mod}}) = 1$
   (b) $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{fac}}}(\text{aux}, N, \pi_{\mathsf{fac}}) = 1$
   (c) $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{df}}}(\text{aux}, \hat{N}, t, s_1, s_2, \pi_{\mathsf{df}}) = 1$

Algorithm 4: Non-interactive setup protocol executed ahead-of-time by the server

4.3 Interactive Key-Generation

This section covers the distributed 2-party generation of an ECDSA public key, together with shares of the associated secret key and a Paillier encryption of the server share. In the following, we assume that the server has executed the non-interactive setup of Section 4.2, and that the client has successfully verified the setup. In Protocol 2 below, we let n denote the length of an RSA modulus that will be generated by the client. Note that we do not set $n = n_{\text{rsa}}$ because the client modulus plays a different role: unlike the modulus generated by the server, it is an *ephemeral* RSA modulus, generated and used solely for the purpose of enabling an efficient zero-knowledge proof of equality between the Paillier-encrypted secret-key share x_2 of the server, and the discrete logarithm

of the server's (multiplicative) share X_2 of the public key. In particular, revealing the factorization of the ephemeral modulus after completion of Protocol 2 would not cause any security issue.
In practice, this means that we can reasonably set n to be smaller than n_{rsa} and set a timeout for the client. Concretely, we set $n_{\text{rsa}} = 3072$ and $n = 2048$ in our implementation, and set the client timeout to 1 minute. This corresponds to making the (conservative) assumption that factoring a 2048-bit RSA modulus within 1 minute is not feasible (note that according to the NIST documentation [Bar20], a 2048-bit RSA modulus corresponds to a 112-bit security level).

Protocol Generate Address – i.e. PK

Common Input. $(\mathbb{G}, g, q)$, $\mathsf{SetupData}$ and identifier $ssid = (sid, \texttt{keygen}, \mathsf{C}, \ldots)$ and $d \in \mathbb{N}$.

Operations. The following operations are executed in *sequential* order.

1. S: Sample $x_2' \xleftarrow{\$} \pm 2^{n_x}$, and do:
   - Set $X_2 \leftarrow g^{x_2'}$ and $B \leftarrow \mathsf{RO}(ssid, \mathsf{S}, X_2)$.
   - Send B to C.
2. C: Sample $x_1 \xleftarrow{\$} \mathbb{F}_q$, $(\hat{M}, n_{\text{rsa}}, v, u_1, u_2; \hat{p}, \hat{q}, \lambda_1, \lambda_2) \xleftarrow{\$} \mathsf{SampleDF}(1^\ell)$, and do:
   - Generate $\psi_{\mathsf{df}^*} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{df}^*}(ssid, \mathsf{C}, \hat{M}, v, u_1, u_2; \lambda_1, \lambda_2)$.
   - Set $X_1 \leftarrow g^{x_1}$ and $\psi_{\mathsf{dlog}} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{dlog}}(ssid, \mathsf{C}, X_1; x_1)$.
   - Send $(X_1, \hat{M}, v, u_1, u_2, \psi_{\mathsf{df}^*}, \psi_{\mathsf{dlog}})$ to S.
3. S: Sample $\beta \xleftarrow{\$} \pm 2^{n_\lambda}$, and do:
   - Check that $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{df}^*}}(ssid, \mathsf{C}, \hat{M}, v, u_1, u_2, \psi_{\mathsf{df}^*}) = 1$.
   - Check that $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{dlog}}}(ssid, \mathsf{C}, X_1, \psi_{\mathsf{dlog}}) = 1$.
   - Set $\mathcal{E} \leftarrow \text{enc}_N(x_2'; \rho^\beta)$ and generate proof
   $$\psi_{\mathsf{dlenc}} \xleftarrow{\$} \Pi^{\text{FS}}_{\mathsf{dlenc}}(ssid, \mathsf{S}, \mathbb{G}, g, q, X_2, \mathcal{E}, N, \rho, n_x, n_\lambda, \hat{M}, v, u_1, u_2; x_2', \beta)$$
   - Send $(X_2, \mathcal{E}, \psi^{\mathsf{dlenc}})$ to C.
4. C:
   - Check that $\mathcal{E} \in \mathbb{Z}^*_{N^2}$ and $B = \mathsf{RO}(ssid, \mathsf{S}, X_2)$.
   - Verify proof $\mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{dlenc}}}(ssid, \mathsf{S}, \mathbb{G}, g, q, X_2, \mathcal{E}, N, \rho, n_x, n_\lambda, \hat{M}, v, u_1, u_2, \psi_{\mathsf{dlenc}}) = 1$.

Public Output. $X = X_1 \cdot X_2 \in \mathbb{G}$ and $\mathcal{E}$.

Secret Output. S outputs $x_2 = x_2' \bmod q$ and C outputs x_1

Protocol 2: Interactive key-generation protocol.

4.4 Distributed Signing

This section covers our main protocol, the signing protocol. Because Protocol 3 can be invoked many times by any (client, server) pair, its efficiency is the most crucial aspect of our approach.

Protocol Signing

Common Input. $(\mathbb{G}, g, h, f, q)$, $\mathsf{SetupData}$ and $ssid = (sid, \mathtt{sign}, \mathrm{msg}, \mathsf{C}, X, \mathcal{E}, \ldots)$.

Message digest. $m = \mathsf{F}(\mathrm{msg})$

Private Input. $\mathsf{S}: \varphi(N) \in \mathbb{N}$. $\mathsf{C}: x_1 \in \mathbb{F}_q$.

Operations. The following operations are executed in *sequential* order.

1. S: Sample $k_2 \xleftarrow{\$} \mathbb{F}_q$, and do:
   - Set $R_2 \leftarrow g^{k_2}$ and $Y \leftarrow X_1^{k_2}$.
   - Send (R_2, Y).
2. C: Check that $R_2 \neq \mathbb{1}$ and $R_2^{x_1} = Y$, and do:
   - Sample $k_1 \xleftarrow{\$} \mathbb{F}_q$ and $(\lambda_0, \mu, \mu') \xleftarrow{\$} (\pm 2^{n\lambda_0}) \times (\pm\lfloor 2^{n_a}/q \rfloor) \times (\pm\lfloor 2^{n_b}/q \rfloor)$.
   - Set $R_1 \leftarrow g^{k_1}$ and $R \leftarrow R_2^{k_1}$
   - $r = \mathcal{X}((R_2 \cdot g^{\mathsf{RO}(X,R_1,R,m)})^{k_1})$ and $\mathcal{S} = \mathrm{enc}_N(u; \rho^{\lambda_0}) \cdot \mathcal{E}^v \bmod N^2$ where
   $$u = [k_1^{-1}(m + rx_1)]_q + \mu q, \qquad v = [k_1^{-1} r]_q + \mu' q$$
   - Generate $\psi_{\mathsf{sig}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{sig}}(ssid, \mathbb{G}, g, h, f, N, \rho, n_a, n_b, \hat{N}, t, s_1, s_2, \mathcal{S}, \mathcal{E}; u, v, \lambda_0)$
   - Send $(R_1, R, \mathcal{S}, \psi_{\mathsf{sig}})$.
3. S: Set $r \leftarrow \mathcal{X}(R_1^{k_2 + \mathsf{RO}(X,R_1,R,m)})$ and do:
   - Check that $R_1 \neq \mathbb{1}$, $\mathcal{S} \in \mathbb{Z}^*_{N^2}$ and $R = R_1^{k_2}$.
   - Check that $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{sig}}}(ssid, \mathbb{G}, g, h, f, N, \rho, n_a, n_b, \hat{N}, t, s_1, s_2, \mathcal{S}, \mathcal{E}, \psi_{\mathsf{sig}}) = 1$.

S's Output.

(r, σ) for $\sigma = \mathrm{dec}_{\varphi(N)}(\mathcal{S}) \cdot (k_2 + \mathsf{RO}(X, R_1, R, m))^{-1} \bmod q$ iff (r, σ) is valid.

Protocol 3: Threshold signing protocol.

5 Underlying ZK Protocols

This section covers the zero-knowledge proofs used in our non-interactive setup, distributed key-generation, and signing protocols respectively. All protocols are described as interactive (three-round) protocols, and are compiled into non-interactive zero-knowledge proofs via the Fiat-Shamir transform. We note that the well-formed signature proof (Protocol 6) is used as proof *of knowledge*

in our threshold ECDSA protocol. While the Fiat-Shamir transform does not preserve knowledge extractability in a concurrent setting in general (because it requires rewinding for extraction), our security analysis shows that these protocols (when compiled with Fiat-Shamir) are straight-line extractable in the generic group model and random oracle model. Our protocol also relies on the straight-line extractability of the standard (Fiat-Shamir-compiled) Schnorr Σ-protocol for knowledge of a discrete logarithm, which we also prove to hold in the generic group and random oracle models.

Definition 5.1. Let $\boldsymbol{R} = \{(X; w)\}$ denote an NP-relation and let ℓ denote the security parameter. A Σ-protocol Π is a four-tuple of algorithms $(\mathsf{S}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ parametrized by $\boldsymbol{E}$ and $\boldsymbol{T}$ such that:

0. $\sigma \xleftarrow{\$} \mathsf{S}(1^\ell)$ denotes the setup parameter.
1. $A \leftarrow \mathsf{P}_1(\sigma, X; w, \tau)$ denotes the provers' first message where $\tau \in \boldsymbol{T}$.
2. $z \leftarrow \mathsf{P}_2(\sigma, X, e; w, \tau)$ denotes the provers' second message where $e \in \boldsymbol{E}$.
3. $\beta \leftarrow \mathsf{V}(\sigma, X, A, e, z)$ denotes the verifiers' acceptance/rejection.

The tuple (σ, X, A, e, z) is called the transcript of the protocol. Security properties:

HVZK. For any $\sigma \xleftarrow{\$} \mathsf{S}$, there exists H such that the following are μ-statistically close:

(a) (σ, X, A, e, z) for $\tau \xleftarrow{\$} \boldsymbol{T}$, $e \xleftarrow{\$} \boldsymbol{E}$, $A \leftarrow \mathsf{P}_1(\sigma, X; w, \tau)$, $z \leftarrow \mathsf{P}_2(\sigma, X, e; w, \tau)$

(b) (σ, X, A', e', z') for $e' \xleftarrow{\$} \boldsymbol{E}$, $(A', z') \xleftarrow{\$} \mathsf{H}(\sigma, X, e')$ such that $\mathsf{V}(\sigma, X, A', e', z') = 1$.

Soundness. If $(X; w) \notin \boldsymbol{R}$, then the following holds for every $(\mathsf{P}_1^*, \mathsf{P}_2^*)$:

$$\Pr_{\substack{e \xleftarrow{\$} \boldsymbol{E}, \sigma \xleftarrow{\$} \mathsf{S}(1^\ell) \\ A \xleftarrow{\$} \mathsf{P}_1^*(\sigma, X)}} \left[z \xleftarrow{\$} \mathsf{P}_2^*(\sigma, X, e) \text{ s.t. } \mathsf{V}(\sigma, X, A, e, z) = 1 \right] \leq \lambda$$

Let $\Pi_{\boldsymbol{R}}$ be a Σ-protocol for a relation $\boldsymbol{R} = \{(X, ; w)\}$, and let $\Pi_{\boldsymbol{R}}^{\mathrm{FS}}$ denote the NIZK obtained by applying the Fiat-Shamir transform to $\Pi_{\boldsymbol{R}}$ (see Section 3.5.1). If $\Pi_{\boldsymbol{R}}$ is μ-HVZK, there exists an efficient simulator $\mathsf{Sim}_{\boldsymbol{R}}$ for $\Pi_{\boldsymbol{R}}^{\mathrm{FS}}$ that is given programming access to the random oracle $\mathtt{HS} : \{0,1\}^* \to \boldsymbol{E}$: on input (σ, X), $\mathsf{Sim}_{\boldsymbol{R}}$ samples $e' \xleftarrow{\$} \boldsymbol{E}$ and $(A', z') \xleftarrow{\$} \mathsf{H}(\sigma, X, e')$ (where H is the sampling algorithm given by Definition 5.1). If (σ, X, A') was never queried before, Sim programs $\mathtt{HS}(\sigma, X, A') = e'$ and outputs (A', e', z'); otherwise, it outputs $\perp$. The following is standard:

- For every fixed $\sigma \in \mathrm{supp}(\mathsf{S}^*)$ and $(X; w) \in \boldsymbol{R}$, the distribution $\mathsf{Sim}_{\boldsymbol{R}}(\sigma, X)$ conditioned on not outputting $\perp$ is μ-statistically close to the distribution sampled by $\Pi_{\boldsymbol{R}}^{\mathrm{FS}}(\sigma, X; w)$.
- For every fixed $\sigma \in \mathrm{supp}(\mathsf{S}^*)$ and $(X; w) \in \boldsymbol{R}$,

$$\Pr[\mathsf{Sim}_{\boldsymbol{R}}(\sigma, X) = \perp] \leq \frac{Q_{\mathtt{HS}}^2}{2^m},$$

where m denotes a lower bound on the min-entropy of the distribution $\{A' : (A', z') \xleftarrow{\$} \mathsf{H}(\sigma, X, e')\}$.

Remark 5.2. For all zero-knowledge proofs considered in this work, it holds that the min-entropy of the distribution $\{A' : (A', z') \xleftarrow{\$} \mathsf{H}(\sigma, X, e')\}$ is at least 2ℓ. Therefore, without loss of generality, if $\Pi_{\boldsymbol{R}}$ is μ-HVZK, we can bound the advantage of any adversary, making at most $\mathcal{Q}_{\mathtt{HS}}$ queries to the random oracle, in distinguishing $\Pi^{\mathrm{FS}}_{\boldsymbol{R}}(\sigma, X; w)$ from $\mathsf{Sim}_{\boldsymbol{R}}(\sigma, X)$ by $\mu + Q^2_{\mathtt{HS}}/2^{2\ell}$.

Notation 5.3. We denote $\mathsf{Sim}_{\boldsymbol{R}}$ the simulator described above for $\Pi^{\mathrm{FS}}_{\boldsymbol{R}}$.

Notation 5.4. We say that Π is μ-HVZK and λ-sound with respect to $\boldsymbol{R}$ if it satisfies the security properties outlined in Definition 5.1. Additionally, in the sequel, we demonstrate that some of our protocols satisfy the 'special' soundness property, which allows for the extraction of a witness for the NP-relation from two suitable transcripts of the protocol, although we do not formally define this property.

5.1 Small Factors Proof (Π_{fac})

Protocol Small Factors

Common Input. $N = pq \in \mathbb{Z}$ s.t. $2^n > N \geq 2^{n-1}$, soundness parameter ℓ, and aux $\in \{0,1\}^*$. Strong prime d such that $d > 2^{n+2\ell+2\nu+3}$ and $g, h \in \mathrm{QR}_d$.

Preliminary step. Set $\widetilde{P} = g^p h^r \bmod d$ and $\widetilde{Q} = g^q h^s \bmod d$ for random $r, s \in \mathbb{Z}_{(d-1)/2}$.

Operations.

1. P sends $(d, \widetilde{P}, \widetilde{Q})$ and $(\widetilde{A}, \widetilde{B}, \widetilde{C}) = (g^\alpha \cdot h^\rho, g^\beta \cdot h^\sigma, \widetilde{Q}^\alpha \cdot h^\mu)$ for
$$(\alpha, \beta, \rho, \sigma, \mu) \xleftarrow{\$} (\pm 2^{n/2+\ell+\nu})^2 \times \mathbb{Z}^3_{(d-1)/2}.$$
2. V sends $e \in \pm 2^\ell$
3. P returns $(z_1, z_2, \lambda_1, \lambda_2, w)$ such that
$$\begin{cases} z_1 = \alpha + ep \\ z_2 = \beta + eq \\ \lambda_1 = \rho + er \bmod (d-1)/2 \\ \lambda_2 = \sigma + es \bmod (d-1)/2 \\ w = \mu + e(-sp) \bmod (d-1)/2 \end{cases}$$

Verifier's Output. Reconstruct g, h from (d, aux) and accept if $z_1, z_2 \in \pm 2^{n/2+\ell+\nu}$ and
$$\begin{cases} g^{z_1} h^{\lambda_1} = \widetilde{A} \cdot \widetilde{P}^e \bmod d \\ g^{z_2} h^{\lambda_2} = \widetilde{B} \cdot \widetilde{Q}^e \bmod d \\ \widetilde{Q}^{z_1} h^w = \widetilde{C} \cdot g^{Ne} \bmod d \end{cases}$$

Protocol 4: Small factor proof Π_{fac}.

Proposition 5.5. *It holds that* Π_{fac} *is* $(2^{-\nu+1})$*-HVZK with respect to* $\boldsymbol{R}_{\mathsf{fac}}$.

Proof. Define $\mathsf{trans}' = (\widetilde{P}', \widetilde{Q}', \widetilde{A}', \widetilde{B}', \widetilde{C}', e', z_1', z_2', \lambda_1', \lambda_2', w') \xleftarrow{\$} \mathsf{Sim}_{\boldsymbol{R}_{\mathsf{fac}}}(N, d, g, h)$ as follows. Let $\widetilde{P}', \widetilde{Q}' \xleftarrow{\$} \mathsf{QR}_d$, $\lambda_1', \lambda_2', w' \xleftarrow{\$} \mathbb{Z}_{(d-1)/2}$ and $z_1', z_2' \xleftarrow{\$} \pm 2^{n/2+\ell+\nu}$. Sample $e' \xleftarrow{\$} \pm 2^{\ell}$ and set $(\tilde{A}', \tilde{B}', \tilde{C}') = (g^{z_1'} h^{\lambda_1'} \cdot \widetilde{P}'^{-e'}, g^{z_2'} h^{\lambda_2'} \cdot \widetilde{Q}'^{-e'}, Q^{z_1'} h^{w'} \cdot g^{-Ne'}) \bmod d$. We note that the statistical distance between trans' and the transcript of Π_{fac} is bounded by the statistical distance of (z_1, z_2) and (z_1', z_2'), as all other values are sampled identically or they are set accordingly. Since $\mathrm{SD}(z_1, z_1') = \mathrm{SD}(z_2, z_2') \leq 2^{-\nu}$, the claim follows. □

5.1.1 Special Soundness of Π_{fac}

Lemma 5.6 (Special Soundness)**.** *Using the notation above, let* $\mathsf{trans} = (\widetilde{A}, \widetilde{B}, \widetilde{C}, e, z_1, z_2, \lambda_1, \lambda_2, w)$ *and* $\mathsf{trans}' = (\widetilde{A}, \widetilde{B}, \widetilde{C}, e', z_1', z_2', \lambda_1', \lambda_2', w')$ *denote two accepting transcripts of Protocol 4. Then, there exists an extractor* E *such that at least one of the following holds true.*

1. $\alpha \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *and* $g^{\alpha} = h \bmod d$.
2. $(u, v) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $N = uv$ *and* $|u|, |v| \leq 2^{n/2+\ell+\nu}$

Proof. Write $\Delta_{(\cdot)}$ for the relevant differences, e.g., $\Delta_e = e - e'$. Deduce that $N = pq \bmod d$ for $p = \Delta_{z_1}/\Delta_e \bmod d$ and $q = \Delta_{z_2}/\Delta_e \bmod d$. Otherwise, the equation below yields a discrete logarithm relation between g and h.

$$\widetilde{Q}^{\Delta_{z_1}} h^{\Delta_w} = \left(g^{\Delta_{z_2}/\Delta_e} h^{\Delta_{\lambda_2}/\Delta_e}\right)^{\Delta_{z_1}} h^{\Delta_w} = g^{N\Delta_e} \bmod d$$

If not, i.e. $\Delta_e^2 N = \Delta_{z_1}\Delta_{z_2} \bmod (d-1)/2$, by noting that $N \cdot 2^{2\ell} < d/2$ and $|\Delta_{z_1} \cdot \Delta_{z_2}| \leq 2^{n+2\ell+\nu+2} < d/2$, it follows that $\Delta_e^2 \cdot N = \Delta_{z_1}\Delta_{z_2}$ Finally, for $f_1 = \gcd(\Delta_e^2, u)$ and $f_2 = \Delta_e^2/f_1$ it holds that $f_1 \mid \Delta_{z_1}$ and $f_2 \mid \Delta_{z_2}$. In conclusion, $N = (\Delta_{z_1}/f_1) \cdot (\Delta_{z_2}/f_2)$. □

5.2 Well-Formed Modulus (Π_{mod})

Protocol Well-Formed Modulus

Common Input. $N \in \mathbb{Z}$ such that $N = 1 \bmod 4$.

Secret Input. Odd primes p_1, p_2 such that $N = p_1 p_2$ and $\gcd(N, \varphi(N)) = 1$.

Operations.

1. V sends $y \xleftarrow{\$} \mathbb{Z}_N^*$
2. P returns $(a, b, x, z) \in \{0,1\}^2 \times \mathbb{Z}_N^{*2}$ such that
   - (a, b) are uniquely determined so that $y_0 = (-1)^a \cdot 2^{Nb} \cdot y \in \mathsf{QR}_N$
   - x is chosen at random as one of the 4-th roots of y_0.
   - z is the unique N-th root of y modulo N.

Verifier's Output. Set $w = 2^N \bmod N$ and accept if

$$\begin{cases} N = 1 \bmod 4 \text{ and } N \text{ is a composite number} \\ 2 \text{ has Jacobi symbol } -1 \\ x^4 = (-1)^a \cdot w^b \cdot y \bmod N \in \mathbb{Z}_N^* \\ y = z^N \bmod N \in \mathbb{Z}_N^* \end{cases}$$

Protocol 5: Well-Formed Modulus Π_{mod}.

Proposition 5.7. *If $(p_1, p_2) = (3, 7) \bmod 8$, then Π_{mod} is 0-HVZK with respect to $\boldsymbol{R}_{\mathsf{mod}}$. Additionally, Π_{mod} is λ-sound with respect to $\boldsymbol{R}_{\mathsf{mod}}$, for*

$$\lambda = \max\left\{\frac{1}{4}, \frac{1}{\gcd(N, \varphi(N))}\right\}.$$

Proof. The soundness property is a corollary of Lemma 5.8. For the HVZK property, define ZK simulator which on input N returns (y, a, b, x, z) for $(a, b, \hat{x}) \xleftarrow{\$} \{0,1\}^N \times \mathbb{Z}_N^*$ and $z = \hat{x}^4 \cdot (-1)^a \cdot 2^{-b} \bmod N$, $x = \hat{x}^N \bmod N$ and $y = z^N \bmod N$. We conclude that the tuple (y, a, b, x, z) is identically distributed with the transcript of Π_{mod}, because, under the assumption that $(p_1, p_2) = (3, 7) \bmod 8$, $-1 \notin \mathsf{QR}_N$ of Jacobi symbol 1, $2 \in \mathbb{Z}_N^*$ has Jacobi symbol -1, and $w = 2^N \in \mathbb{Z}_N^*$ has Jacobi symbol -1, since N is odd. □

Lemma 5.8. *Using the notation from Protocol 5, the following hold true.*

1. *If $\gcd(N, \varphi(N)) \neq 1$, then the verifier accepts with probability at most $1/\gcd(N, \varphi(N))$.*
2. *If N is square free and has more than three prime factors, then the verifier accepts with probability at most $1/4$.*
3. *If $N = p_1 p_2$ for distinct primes p_1, p_2 and $(p_i, p_{3-i}) \neq (3, 7) \bmod 8$ for some $i \in \{1, 2\}$, then the verifier accepts with probability at most $1/4$.*
4. *If N is prime, then the verifier accepts with probability at most 0.*

Proof. **Item 1**: The first item is immediate since only $|\mathbb{Z}_N^*|/\gcd(N, \varphi(N))$ elements in $\mathbb{Z}_N^*$ admit an N-th root.
Item 2: Observe that if N has more than four prime factors then the verifer accepts with probability at most $1/4$ because at most $|\mathbb{Z}_N^*|/2^4$ elements admit fourth roots and thus the set $\{(-1)^a \cdot w^b \cdot x^4 \bmod N \mid (a, b, x) \in \{0,1\}^2 \times \mathbb{Z}_N^*\}$ contains at most $|\mathbb{Z}_N^*|/4$ elements. If N has exactly three factors, under the assumption that N is square free, say $N = pqr$, we note that (WLOG) $r = 1 \bmod 4$ because $N = 1 \bmod 4$. Furthermore, for such an r, only $1/4$ of the elements in $\mathbb{Z}_r^*$ admit fourth roots. Therefore, it follows that the set $\{(-1)^a \cdot w^b \cdot x^4 \bmod N \mid (a, b, x) \in \{0,1\}^2 \times \mathbb{Z}_N^*\}$ admits at most $4 \cdot (|\mathbb{Z}_p^*|/2) \cdot (|\mathbb{Z}_q^*|/2) \cdot (|\mathbb{Z}_p^*|/4) = |\mathbb{Z}_N^*|/4$ elements.
Item 3: if $p_1, p_2 = 1 \bmod 4$, then, as above, only $1/16$-th of the elements in $\mathbb{Z}_N^*$ admit 4-th roots which means that the verifier accepts with probability $1/4$—the remaining cases: N is even, (p_1, p_2) or $(p_2, p_1) = (1, -1) \bmod 4$, and $p_1 = p_2 \bmod 8$ are ruled out by the checks $N = 1 \bmod 4$ and 2 has Jacobi symbol -1.
Item 4: Immediate as N is tested for primality.

□

5.3 Well-Formed Signature (Π_{sig})

Protocol Well-Formed Signature

Common Input. Group-generators-order tuple $(\mathbb{G}, g, h, f, q)$, modulus N and $\rho \in \mathbb{Z}^*_{N^2}$, and Damgård-Fujisaki params $(\hat{N}, t, s_1, s_2)$ and $(\mathcal{S}, \mathcal{E}) \in (\mathbb{Z}^*_{N^2})^2$ s.t. $\mathcal{S} = \mathrm{enc}_N(a; \rho^{\lambda_0}) \cdot \mathcal{E}^b \bmod N^2$ and

$$(a, b, \lambda_0) \in (\pm 2^{n_a}) \times (\pm 2^{n_b}) \times (\pm 2^{n_{\lambda_0}}).$$

Auxiliary Input. $\widetilde{P} = s_1^a s_2^b t^\mu \bmod \hat{N}$ and $\widetilde{U} = g^a h^b f^\gamma$ for $\mu \xleftarrow{\$} \pm 2^{n_{\mathsf{t}\rho}}$ and $\gamma \xleftarrow{\$} \mathbb{F}_q$.

Operations. (Recall $\varepsilon = \ell + \nu$).

1. P samples $(\alpha, \beta, \gamma', \mu', \lambda') \xleftarrow{\$} (\pm 2^{n_a+\varepsilon}) \times (\pm 2^{n_b+\varepsilon}) \times \mathbb{F}_q \times (\pm 2^{n_{\mathsf{t}\rho}+\varepsilon}) \times \pm(2^{n_{\lambda_0}+\varepsilon})$ and sends $(\widetilde{P}, \widetilde{U})$ and $(\widetilde{V}, \widetilde{B}, \mathcal{D})$ such that
$$\begin{cases} \widetilde{V} = g^\alpha h^\beta f^{\gamma'} \in \mathbb{G} \\ \widetilde{B} = s_1^\alpha s_2^\beta t^{\mu'} \bmod \hat{N} \\ \mathcal{D} = \mathrm{enc}_N(\alpha; \rho^{\lambda'}) \cdot \mathcal{E}^\beta \bmod N^2. \end{cases}$$
2. V sends $e \xleftarrow{\$} \pm 2^\ell$
3. P returns $(z_1, z_2, w_0, w_1, w_2)$ such that
$$\begin{cases} z_1 = \alpha + ea \\ z_2 = \beta + eb \\ w_0 = \gamma' + e \cdot \gamma \bmod q \\ w_1 = \mu' + e \cdot \mu \\ w_2 = \lambda' + e \cdot \lambda_0 \end{cases}$$

Verifier's Output. Accept if $z_1, z_2 \in (\pm 2^{n_a+\varepsilon}) \times (\pm 2^{n_b+\varepsilon})$ and

$$\begin{cases} g^{z_1} h^{z_2} f^{w_0} = \widetilde{V} \cdot \widetilde{U}^e \in \mathbb{G} \\ s_1^{z_1} s_2^{z_2} t^{w_1} = \widetilde{B} \cdot \widetilde{P}^e \bmod \hat{N} \in \mathbb{Z}^*_{\hat{N}} \\ \mathrm{enc}_N(z_1; \rho^{w_2}) \cdot \mathcal{E}^{z_2} = \mathcal{D} \cdot \mathcal{S}^e \bmod N_2 \in \mathbb{Z}^*_{N^2} \end{cases}$$

Protocol 6: Well-Formed Signature Proof Π_{sig}.

Proposition 5.9. *Π_{sig} is $(5 \cdot 2^{-\nu})$-HVZK with respect to $\boldsymbol{R}_{\mathsf{sig}}$.*

Proof. The proof is similar to the proof of Proposition 5.5 and it is omitted from the text. □

5.3.1 Special Soundness of Π_{sig}

Lemma 5.10 (Special Soundness)**.** *Using the notation above, let* $\mathsf{trans} = (\tilde{V}, \tilde{B}, \mathcal{D}, e, z_1, z_2, w_1, w_2)$ *and* $\mathsf{trans}' = (\tilde{V}, \tilde{B}, \mathcal{D}, e', z_1', z_2', w_1', w_2')$ *denote two accepting transcripts of Protocol 6. Then, there*

exists an extractor E *given auxiliary input* $\rho_0 \in \mathbb{Z}^*_{N^2}$ *with* $\rho_0^N = \rho \bmod N^2$ *such that at least one of the following holds true.*

1. $(\alpha, \beta, \gamma, \delta) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $\delta \nmid \gcd(\alpha, \beta)$ *and* $s_1^\alpha s_2^\beta t^\gamma = \tilde{P}^\delta \bmod \hat{N}$.
2. $k \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $k \notin \{1, N\}$, $|k| < \ell$ *and* $\gcd(k, N) \neq 1$.
3. $(u, v, \eta, \zeta) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $\mathcal{S} = \mathrm{enc}_N(u, \eta^N) \cdot \mathcal{E}^v \bmod N^2$, $\widetilde{U} = g^u f^v h^\zeta$,
$$u, v \in (\pm 2^{n_a + \varepsilon}) \times (\pm 2^{n_b + \varepsilon}).$$

Proof. Let $(\Delta_e, \Delta_1, \Delta_2, \Delta_{w_0}, \Delta_{w_1}, \Delta_{w_2}) \leftarrow (e - e', z_1 - z_1', z_2 - z_2', w_0 - w_0', w_1 - w_1', w_2 - w_2')$. If $\Delta_e \nmid \gcd(\Delta_1, \Delta_2)$, output $(\Delta_1, \Delta_2, \Delta_{w_1}, \Delta_e)$ and we are in case 1. Else if $\Delta_e \neq 1, N$ and $\gcd(\Delta_e, N) \neq 1$, output Δ_e and we are in case 2. Else, output (u, v, η, ζ) as follows: Set $u = (z_1 - z_1')/\Delta_e$ and $v = (z_2 - z_2')/\Delta_e$. Notice that

$$\mathrm{enc}_N(u\Delta_e; \rho^{\Delta_{w_2}}) \cdot \mathcal{E}^{v\Delta_e} = \mathcal{S}^{\Delta_e} \bmod N^2.$$

Thus $(\mathrm{enc}_N(u; 1) \cdot \mathcal{E}^v \cdot \mathcal{S}^{-1})^{\Delta_e} = \rho^{-\Delta_{w_2}} = (\rho_0^{-\Delta_{w_2}})^N \bmod N^2$ which yields η from Fact 5.11. Finally, set $\zeta = \Delta_{w_0} \cdot \Delta_e^{-1} \bmod q$, and deduce that $\widetilde{U} = g^u h^v f^\zeta$, as required. □

Fact 5.11. *Let* $M, k, \ell \in \mathbb{N}$ *such that* $\gcd(k, \ell) = 1$ *and* $k, \ell \leq M$. *For* $C, x \in \mathbb{Z}^*_M$ *such that* $C^k = x^\ell \bmod M$, *there exists efficiently computable (in* $|M|$*)* $y \in \mathbb{Z}^*_M$ *such that* $C = y^\ell \bmod M$.

Proof. Let (μ, ν) denote the Bézout coefficients of (k, ℓ). Notice that $\mu k + \nu \ell = 1$ by assumption. So, setting $y = x^\mu \cdot C^\nu \bmod N$, it holds that $C = y^\ell \bmod M$ since

$$C = C^{\mu k + \nu \ell} = (C^k)^\mu (C^\nu)^\ell = (x^\ell)^\mu (C^\nu)^\ell = (x^\mu \cdot C^\nu)^\ell = y^\ell \bmod M$$

□

5.3.2 Partial Straight-Line Extractability of Π_{sig} in the GGM

Claim 5.12 (Partial Straight-Line Extractability in the GGM)**.** *Let* $\mathtt{HS} : \{0,1\}^* \to \mathbb{Z}_q$ *be a random oracle. Let* $\mathbb{G}$ *be modeled as a generic group* $\boldsymbol{G}$ *of order* q *with group oracle* $\mathcal{O}$ *(as defined in Section 3.6). Fix the group-generators* $(g, h, f) \in \boldsymbol{G}^3$, *modulus* N, $\rho \in \mathbb{Z}^*_{N^2}$, *and Damgård-Fujisaki params* $(\hat{N}, t, s_1, s_2)$. *Set* $\mathsf{pp} \leftarrow (\mathbb{G}, g, h, f, q, N, \rho, \hat{N}, t, s_1, s_2)$. *Let* $(\mathcal{S}, \mathcal{E}) \in (\mathbb{Z}^*_{N^2})^2$ *and* $\mathrm{aux} = (\mathrm{aux}^*, \tilde{P}, \tilde{U}) \in \mathbb{Z}_{\hat{N}} \times \boldsymbol{G}$ *be auxiliary data, where* aux^* *denotes some additional auxiliary contextual data. For every (possibly malicious) prover* P^* *outputting an accepting proof* ψ_{sig} *with nonzero probability, let* $\mathcal{Q}^*(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$ *denote the list of all queries to* $\mathtt{HS}$ *and* $\mathcal{O}$ *(together with their answers) made by* $\mathcal{P}^*$ *on input* $(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$. *Then, there exists an extractor* $\mathrm{GGMExt}_{\mathsf{sig}}$ *such that*

$$\Pr[\tilde{U} = g^a h^b f^\gamma \mid \mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{sig}}}(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E}, \psi_{\mathsf{sig}}) = 1 \ \wedge \ \neg \mathsf{Coincidence}] \geq 1 - \frac{1}{2^{\ell+1}},$$

where the probability is over $\psi_{\mathsf{sig}} \leftarrow \mathsf{P}^*(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$ *and* $(a, b, \gamma) \xleftarrow{\$} \mathrm{GGMExt}_{\mathsf{sig}}(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E}, \psi_{\mathsf{sig}}, \mathcal{Q}^*(\mathrm{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E}))$, *and* $\mathsf{Coincidence}$ *denotes the event (defined in Section 6.1) that* $\mathrm{GGMExt}_{\mathsf{sig}}$ *fails to maintain unique representations of all group oracle queries.*

Proof. Given a malicious prover P^* running on input $(\text{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$ making queries $\mathcal{Q}^* = \mathcal{Q}^*(\text{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$, we let $\text{GGMExt}_{\mathsf{sig}}$ construct from $\mathcal{Q}^*$ representations of all queries in lists $\boldsymbol{B}, \boldsymbol{L}$ as specified in Section 6.1. At any point of the execution, if a flag Coincidence is raised, $\text{GGMExt}_{\mathsf{sig}}$ outputs $\perp$ and terminates.

- $\text{GGMExt}_{\mathsf{sig}}$ runs $\psi_{\mathsf{sig}} \xleftarrow{\$} \mathsf{P}^*(\text{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E})$, and parses ψ_{sig} as $(\widetilde{V}, \widetilde{B}, \mathcal{D}, z_1, z_2, w_0, w_1, w_2)$.
- If the proof ψ^{sig} verifies, $\text{GGMExt}_{\mathsf{sig}}$ retrieves the representation $\tilde{U} = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$ from $\boldsymbol{L}$ (it exists necessarily as $\tilde{U}$ gets added to $\boldsymbol{L}$ after $(\text{aux}, \mathsf{pp}, \mathcal{S}, \mathcal{E}, \tilde{V}, \tilde{B}, \mathcal{D})$ is queried to HS if it was not already there).
- If $\gamma_Z = 0$ for all $Z \in \boldsymbol{B} \setminus \{g, h, f\}$, $\text{GGMExt}_{\mathsf{sig}}$ sets $(a, b, \gamma) \leftarrow (\gamma_g, \gamma_h, \gamma_f)$.
- Else, $\text{GGMExt}_{\mathsf{sig}}$ raises a flag FailExt, output $\perp$, and terminates.

We now show that

$$\Pr[\mathsf{FailExt} \mid \neg\mathsf{Coincidence}] \leq \frac{1}{2^{\ell+1}}.$$

Proof. Assume FailExt is raised and no flag coincidence is raised. Since ψ^{sig} verifies, we have

$$g^{z_1} h^{z_2} f^{w_0} = \tilde{V} \cdot \tilde{U}^e.$$

Let $Z \in \boldsymbol{L} \setminus \{g, h, f\}$ such that $\gamma_Z \neq 0$ in the representation $\tilde{U} = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$. By uniqueness of the representation, we have $\gamma'_Z + e \cdot \gamma_Z = 0$. As e is sampled by Sim uniformly at random over $\pm 2^\ell$ after $(\mathcal{S}, \tilde{P}, \tilde{U}, \tilde{V}, \tilde{B}, \mathcal{D})$ is queried to HS (independently of the representations of $\tilde{U}$ and $\tilde{V}$), this happens with probability at most $1/2^{\ell+1}$. This concludes the proof. □

□

5.4 Discrete Logarithm (Π_{dlog})

Protocol 5.13 (Discrete Logarithm)**.**

Common Input. Group-tuple $(\mathbb{G}, g, q)$ and $X = g^x$.

Operations.

1. P samples $\alpha \xleftarrow{\$} \mathbb{Z}_q$ and sends $A = g^\alpha \in \mathbb{G}$.
2. V sends $e \xleftarrow{\$} \pm 2^\ell$.
3. P returns $z = \alpha + e \cdot x$.

Verifier's Output. Accept if $X^e A = g^z$.

The following is standard:

Proposition 5.14. *It holds that Π_{dlog} is 0-HVZK with respect to $\boldsymbol{R}_{\mathsf{dlog}}$.*

Claim 5.15 (Extractability in the GGM)**.** *Let* $\mathtt{HS} : \{0,1\}^* \to \mathbb{Z}_q$ *be a random oracle. Let* $(\mathbb{G}, g, q)$ *be modeled as a generic group* $(\boldsymbol{G}, g, q)$ *with group oracle* $\mathcal{O}$ *(as defined in Section 3.6), let* $X \in \boldsymbol{G}$ *be a group element and* aux *be some auxiliary contextual data. For every (possibly malicious) prover* P^* *outputting an accepting proof* ψ_{dlog} *with nonzero probability, let* $\mathcal{Q}^*(\text{aux}, X)$ *denote the list of all queries to* $\mathtt{HS}$ *and* $\mathcal{O}$ *(together with their answers) made by* $\mathcal{P}^*$ *on input* (aux, X)*. Then, there exists an extractor* $\text{GGMExt}_{\mathsf{dlog}}$ *such that*

$$\Pr[X = g^x \mid \mathsf{vrfy}_{\Pi^{\text{FS}}_{\mathsf{dlog}}}(\text{aux}, X, \psi_{\mathsf{dlog}}) = 1 \ \wedge \ \neg\mathsf{Coincidence}] \geq 1 - \frac{1}{q},$$

where the probability is over $\psi_{\mathsf{dlog}} \leftarrow \mathsf{P}^*(\text{aux}, X)$ *and* $x \xleftarrow{\$} \text{GGMExt}_{\mathsf{dlog}}(\text{aux}, X, \psi_{\mathsf{dlog}}, \mathcal{Q}^*(\text{aux}, X))$*, and* Coincidence *denotes the event (defined in Section 6.1) that* $\text{GGMExt}_{\mathsf{dlog}}$ *fails to maintain unique representations of all group oracle queries.*

Proof. Given a malicious prover P^*, we let $\text{GGMExt}_{\mathsf{dlog}}$ maintain representations f all queries in lists $\boldsymbol{B}, \boldsymbol{L}$ as specified in Section 6.1. At any point of the execution, if a flag Coincidence is raised, $\text{GGMExt}_{\mathsf{dlog}}$ outputs $\perp$ and terminates.

- $\text{GGMExt}_{\mathsf{dlog}}$ runs $\psi_{\mathsf{dlog}} \xleftarrow{\$} \mathsf{P}^*(\text{aux}, X)$, and parses ψ_{dlog} as (A, z).
- If the proof ψ^{dlog} verifies, $\text{GGMExt}_{\mathsf{dlog}}$ retrieves the representation $X = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$ from $\boldsymbol{L}$ (it exists necessarily as X gets added to $\boldsymbol{L}$ after (aux, X, A) is queried to $\mathtt{HS}$ if it was not already there).
- If $\gamma_Z = 0$ for all $Z \in \boldsymbol{B} \setminus \{g\}$, $\text{GGMExt}_{\mathsf{dlog}}$ sets $x \leftarrow \gamma_g$.
- Else, $\text{GGMExt}_{\mathsf{dlog}}$ raises a flag FailExt, output $\perp$, and terminates.

We now show that $\Pr[\mathsf{FailExt} \mid \neg\mathsf{Coincidence}] \leq \frac{1}{q}$. Assume FailExt is raised and no flag coincidence is raised. Since ψ^{dlog} verifies, we have $X^e \cdot A = g^z$. Write $X = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$ and $A = \prod_{Z \in \boldsymbol{B}} Z^{\delta_Z}$. Let Z be any element of $\boldsymbol{B} \setminus \{g\}$ such that $\gamma_Z \neq 0$. Because $X^e \cdot A$ has a unique representation in $\boldsymbol{L}$ as g^z, we necessarily have $\gamma_Z + e \cdot \delta_Z = 0$. As e is sampled by $\mathtt{HS}$ on input (aux, X, A) uniformly at random over $\mathbb{Z}_q$ (independently of the representations of X and A), this happens with probability at most $1/q$. This concludes the proof. □

5.5 Encrypted Discrete Log (Π_{dlenc})

Protocol Encrypted Discrete Log

Common Input. Group-tuple $(\mathbb{G}, g, q)$ and $X = g^x$, modulus N and $\rho \in \mathbb{Z}^*_{N^2}$, parameters $(\hat{M}, u_1, u_2, v)$ and $\mathcal{E} \in \mathbb{Z}^*_{N^2}$ such that $\mathcal{E} = \text{enc}_N(x; \rho^\lambda) \bmod N^2$ and $x \in \pm 2^{n_x}$, $\lambda \in \pm 2^{n_\lambda}$.

Auxiliary Input. $\widetilde{P} = u_1^x u_2^\lambda v^\mu \bmod \hat{M}$ for $\mu \xleftarrow{\$} \pm\hat{M} \cdot 2^\nu$.

Operations. (Recall $\varepsilon = \ell + \nu$).

1. P samples $(\alpha, \lambda', \mu') \xleftarrow{\$} (\pm 2^{n_x+\varepsilon}) \times (\pm 2^{n_\lambda+\varepsilon}) \times (\pm \hat{M} \cdot 2^{\nu+\varepsilon})$ and sends $\widetilde{P}$ and

$$\begin{cases} A = g^\alpha \in \mathbb{G} \\ \widetilde{B} = u_1^\alpha u_2^{\lambda'} v^{\mu'} \bmod \hat{M} \\ \mathcal{D} = \mathrm{enc}_N(\alpha; \rho^{\lambda'}) \bmod N^2. \end{cases}$$

2. V sends $e \xleftarrow{\$} \pm 2^\ell$

3. P returns (z_1, z_2, z_3) such that $\begin{cases} z_1 = \alpha + e \cdot x \\ z_2 = \lambda' + e \cdot \lambda \\ z_3 = \mu' + e \cdot \mu \end{cases}$

Verifier's Output. Accept if $(z_1, z_2) \in \pm 2^{n_x+\varepsilon} \times \pm 2^{n_\lambda+\varepsilon}$ and

$$\begin{cases} g^{z_1} = A \cdot X^e \in \mathbb{G} \\ u_1^{z_1} u_2^{z_2} v^{z_3} = \widetilde{B} \cdot \widetilde{P}^e \bmod \hat{M} \in \mathbb{Z}^*_{\hat{M}} \\ \mathrm{enc}_N(z_1; \rho^{z_2}) = \mathcal{D} \cdot \mathcal{E}^e \bmod N^2 \in \mathbb{Z}^*_{N^2} \end{cases}$$

Protocol 7: Encrypted Discrete Log Proof Π_{dlenc}.

Proposition 5.16. *It holds that Π_{dlenc} is $(2^{-\nu+2})$-HVZK with respect to $\boldsymbol{R}_{\mathsf{dlenc}}$.*

Proof. The proof is similar to the proof of Proposition 5.5 and it is omitted from the text. □

Claim 5.17 (Special Soundness)**.** *Using the notation above, let* $\mathsf{trans} = (A, \tilde{B}, \mathcal{D}, e, z_1, z_2, z_3)$ *and* $\mathsf{trans}' = (A, \tilde{B}, \mathcal{D}, e', z_1', z_2', z_3')$ *denote two accepting transcripts of Protocol 7. Then, there exists an extractor* E *given auxiliary input* $\rho_0 \in \mathbb{Z}^*_{N^2}$ *with* $\rho_0^N = \rho \bmod N^2$ *such that at least one of the following holds true.*

1. $(\alpha, \beta, \gamma, \delta) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $\delta \nmid \gcd(\alpha, \beta)$ *and* $u_1^\alpha u_2^\beta v^\gamma = \tilde{P}^\delta \bmod \hat{M}$.
2. $k \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $k \notin \{1, N\}$, $k < 2^\ell$ *and* $\gcd(k, N) \neq 1$.
3. $(x_1, x_2, \sigma_0, \Delta_e) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that*
   (a) $(x_1, x_2, \sigma_0) \in \pm 2^{n_x+\nu+1} \times \pm 2^{n_\lambda+\nu+1} \times \mathbb{Z}^*_{N^2}$,
   (b) $X = g^{x_1} \in \mathbb{G}$ *and* $\mathcal{E} = (1+N)^{x_1} \rho^{x_2} \sigma_0^N \bmod N^2$,
   (c) $(\sigma_0^N)^{\Delta_e} = 1 \bmod N^2$ *and* $|\Delta_e| \leq 2^\ell$.

In particular, this implies that assuming the hardness of strong RSA and N does not admit small factors, it necessarily holds that the extractor produces the triple (x_1, x_2, σ_0) satisfying the conditions of Item 3.

Proof. Let $(\Delta_e, \Delta_1, \Delta_2, \Delta_3) \leftarrow (e' - e, z_1' - z_1, z_2' - z_2, z_3' - z_3)$. By the verification equations,

- $u_1^{\Delta_1} u_2^{\Delta_2} v^{\Delta_3} = \tilde{P}^{\Delta_e} \bmod \hat{M}$,

- $g^{\Delta_1} = X^{\Delta_e} \in \mathbb{G}$ and $\mathrm{enc}_N(\Delta_1; \rho^{\Delta_2}) = \mathcal{E}^{\Delta_e} \bmod N^2$.

If $\Delta_e \nmid \Delta_1, \Delta_2$, output $(\Delta_1, \Delta_2, \Delta_3, \Delta_e)$ thus satisfying Item 1. Else, if $\Delta_e \notin \{1, N\}$ and $\gcd(\Delta_e, N) \neq 1$, output Δ_e as per Item 2. Else, output (x_1, x_2, σ_0) as follows. Set $(x_1, x_2) \leftarrow (\Delta_1/\Delta_e, \Delta_2/\Delta_e)$. This immediately implies that $(x_1, x_2) \in \pm 2^{n_x+\varepsilon+1} \times \pm 2^{n_\lambda+\varepsilon+1}$ and $X = g^{x_1}$ since Δ_e is coprime to q. The last equation becomes $(\rho_0^{\Delta_2})^N = (\mathcal{E} \cdot \mathrm{enc}_N(-x_1; 1))^{\Delta_e} \bmod N^2$. Since $\gcd(\Delta_e, N) = 1$ compute $\eta \in \mathbb{Z}_{N^2}^*$ using Fact 5.11 such that $\mathrm{enc}_N(x_1; \eta^N) = \mathcal{E} \bmod N^2$ and set $\sigma_0 = \eta \cdot \rho_0^{-x_2} \bmod N^2$. Deduce that

$$\begin{cases} \mathcal{E} = (1 + x_1 N) \cdot \eta^N = (1 + x_1 N) \cdot \rho^{x_2} \sigma_0^N \bmod N^2 \\ \sigma_0^{N\Delta_e} = \mathcal{E}^{\Delta_e} \cdot \mathrm{enc}_N(-\Delta_1; \rho^{-\Delta_2}) = 1 \bmod N^2 \end{cases}$$

In conclusion, since $|\Delta_e| \leq 2^\ell$ the conditions of Item 3 are satisfied. □

5.6 Damgård-Fujisaki Parameters (Π_{df})

Protocol Damgård-Fujisaki Params Proof

Common Input. $\hat{N} \in \mathbb{Z}$ and (s_1, s_2, t) s.t. $\{t^{\lambda_i} = s_i \bmod \hat{N}\}_{i=1,2}$ for $\max_i(|\lambda_i|) < 2^{2\ell}$.

Operations.

1. P sends $A = t^\alpha \bmod \hat{N}$ for $\alpha \xleftarrow{\$} \pm 2^{2\ell+\nu}$
2. V replies with $e_1, e_2 \xleftarrow{\$} \{0, 1\}$
3. P sends $z = \alpha + e_1\lambda_1 + e_2\lambda_2$.

Verifier's Output. Accept if $t \in \mathbb{Z}_{\hat{N}}^*$, $z \in \pm 2^{2\ell+\nu}$ and $t^z = A \cdot s_1^{e_1} s_2^{e_2} \bmod \hat{N}$.

Protocol 8: Σ-protocol for proving correctness of the Damgård-Fujisaki parameters (s_1, s_2, t).

Proposition 5.18. *It holds that Π_{df} is $(2^{-\nu+1})$-HVZK and (2^{-1})-sound with respect to $\boldsymbol{R}_{\mathsf{df}}$.*

Proof. The HVZK property is similar to the proof of Proposition 5.5 and it is omitted from the text. The soundness property follows from the claim below.

Claim 5.19. *Using the notation from Protocol 8, if $s_i \notin \langle t \rangle$ or $[\lambda_i]_{\mathsf{ord}(t)} \notin \pm 2^{2\ell+\nu+1}$, for some $i \in [2]$, then the Verifier accepts with probability at most $1/2$.*

Proof. We prove the claim for arbitrary tuple $(s_1, \ldots, s_n, t)$, with $n = 2$ being the special case above. First, suppose that $s_i \notin \langle t \rangle$ for *all* $i \in [n]$. Observe that for accepting transcripts $(A, e_1, \ldots, e_n, z)$ and $(A, e'_1, \ldots, e'_n, z)$, if $(e_1, \ldots, e_n)$ and $(e'_1, \ldots, e'_n)$ have hamming distance 1, then at least one of the s_i's belongs to $\langle t \rangle$. Consequently, since every code in $\mathbb{Z}_2^n$ of size $2^{n-1} + 1$ has minimal distance at most 1, it follows that the verifier accepts with probability at most $1/2$ in this case. Next, suppose that $s_1, \ldots, s_m \notin \langle t \rangle$ and $s_{m+1}, \ldots, s_n \in \langle t \rangle$ then the problem is reduced to tuples of the form $(s_1, \ldots, s_m, t)$, and the verifier accepts with probability at most $1/2$ by the previous argument. Finally, write $\omega_t = \mathsf{ord}(t)$, and assume that $s_1, \ldots, s_n \in \langle t \rangle$ and $[\lambda_1]_{\omega_t} \notin \pm 2^{2\ell+\nu+1}$. Fix $A \in \mathbb{Z}_{\hat{N}}^*$ and $e_1, \ldots, e_n \in \{0, 1\}$ and $\beta \in \pm\,\omega_t$ such that $t^\beta = A \prod_{j=2}^n s_j^{e_j}$ and assume that $(A, e_1, \ldots, e_n, z)$ is a

valid transcript. Note that $z = \beta \bmod \omega_t$ if $e_1 = 0$ and $z = \lambda_1 + \beta \bmod \omega_t$, otherwise. Furthermore, for some $\mu \in \{-1, 0, 1\}$, observe that

$$|[\lambda_1 + \beta]_{\omega_t}| = |[\lambda_1]_{\omega_t} + \beta + \mu \cdot \omega_t]| \geq |[\lambda_1]_{\omega_t} + \mu \cdot \omega_t]| - |\beta|$$

Thus, if $\beta \in \pm 2^{2\ell+\nu}$ and $[\lambda_1 + \beta]_{\omega_t} \in \pm 2^{2\ell+\nu}$ then

$$|[\lambda_1]_{\omega_t} + \mu \cdot \omega_t]| \leq |[\lambda_1 + \beta]_{\omega_t}| + |\beta| \leq 2^{2\ell+\nu}. \tag{1}$$

By Equation (1), using the fact that $[\lambda_1]_{\omega_t} \in \pm\,\omega_t$, it follows that

$$\begin{cases} |[\lambda_1]_{\omega_t}| \leq 2^{2\ell+\nu} & \text{if } \mu = 0 \\ \omega_t \leq 2^{2\ell+\nu+1} & \text{if } \mu \neq 0 \end{cases}$$

In both cases, $|[\lambda_1]_{\omega_t}| \leq 2^{2\ell+\nu}$. In conclusion, either $\beta \notin \pm 2^{2\ell+\nu}$ or $[\lambda_1 + \beta]_{\omega_t} \notin \pm 2^{2\ell+\nu}$, and thus the Verifier accepts with probability at most $1/2$. □

□

5.7 Damgård-Fujisaki-*star* Parameters ($\Pi_{\mathsf{df}*}$)

The following protocol is adapted from [HLM25].

Protocol Damgård-Fujisaki-*star* Params Proof

Soundness & ZK Parameter: ℓ_0 and ν

Common Input. $\hat{N} \in \mathbb{Z}$ and (s_1, s_2, t) such that $\{t^{\lambda_i} = s_i \bmod \hat{N}\}_{i=1,2}$.

Operations.

1. P sends $A = t^{\alpha} \bmod \hat{N}$ for $\alpha \xleftarrow{\$} \pm 2^{\ell_0+\nu} \cdot \max_i(2^{\lceil \log(\lambda_i) \rceil})$
2. V replies with $e_1, e_2 \xleftarrow{\$} \{0, \ldots, 2^{\ell_0} - 1\}$
3. P sends $z = \alpha + e_1\lambda_1 + e_2\lambda_2$.

Verifier's Output. Accept if $t \in \mathbb{Z}^*_{\hat{N}}$ and $t^z = A \cdot s_1^{e_1} s_2^{e_2} \bmod \hat{N}$.

Protocol 9: Σ-protocol for proving relaxed correctness of the Damgård-Fujisaki parameters (s_1, s_2, t).

Proposition 5.20. *Let $\omega = \max\{\omega_1, \omega_2\}$ where $\omega_i = \min\{i \in \mathbb{N} \;\; s.t. \;\; s_i^i \in \langle t \rangle\}$. It holds that $\Pi_{\mathsf{df}*}$ is $(2^{-\nu+1})$-HVZK and $(1/\omega + 2^{-\ell_0})$-sound with respect to $\boldsymbol{R}_{\mathsf{df}*}$.*

Proof. See [HLM25]. □

6 Security Analysis

In this section, we prove the following:

Theorem 6.1. *Assume the doubly-enhanced unforgeability of ECDSA, the* SEDCR *assumption, and the strong-RSA assumption. Then, when modeling the hash function* RO *as a random oracle and the group* $\mathbb{G}$ *as a generic group* $\boldsymbol{G}$*, the probability that a corrupted party outputs a valid signature* σ^* *on message* m^**, after interacting concurrently in any (polynomial) number of key generation sessions and signing sessions on messages* $m \neq m^*$ *of its choice is negligible.*

The precise bounds of the theorem vary depending on whether we consider a corrupted client or a corrupted server; the exact bounds are given in Theorem 6.4 and Theorem 6.8 respectively. In the following, given an adversary Adv that corrupts either the client C or the server S, We describe a simulator Sim that emulates the behavior of the honest party and interacts with an ECDSA challenger, a generic group oracle $\mathsf{O}_{\mathbb{G}}$, and a (programmable) random oracle RO.

6.1 Lazy symbolic interfacing with the GGM and the signing oracle

Security analyses in the GGM typically consider a simulator Sim that plays the role of an *interface* between the adversary and the generic group $\boldsymbol{G}$, and records the adversary's queries to the group-oracle O. We recall in this section a standard technical lemma that states, informally, that every group element G produced by the adversary (either in messages exchanged in the protocol, through queries to the group oracle, or to the random oracle) is either *basic* (it has never been produced before as the output of a group oracle queries), or the simulator can find a *unique* representation of G in terms of basic elements. In this work, we also need a generalization of this lemma for the setting where Sim additionally interacts with a signing oracle. Similar generalizations were previously shown to hold in [Bro02; GS22]; we require a slight variant tailored to our doubly enhanced unforgeability game.

Representations of group elements. Formally, we consider a simulator Sim maintaining two lists: a list $\boldsymbol{B}$ of elements of $\boldsymbol{G}$, and a list $\boldsymbol{L}$, initially empty. The list $\boldsymbol{B}$ contains all the *basic* elements (that have not been produced as a result of group operations) and $\boldsymbol{L}$ contains the representations all elements seen by Sim (either basic elements or produced as a result of group operations, of which Sim will know a representation in terms of basic elements). The entries of $\boldsymbol{L}$ are formal equality of the form $P = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$ (where $\gamma_Z = 0$ for all but one Z if $P \in \boldsymbol{B}$), where the indices $Z \in \boldsymbol{B}$ are treated as formal variables. Throughout the emulation, Sim will maintain a unique representation for each element of $\boldsymbol{L}$, and abort if an element has two distinct representations.

Handling queries to the group oracle. Whenever Sim receives (or observes in a query to an oracle) a group element $P \in \boldsymbol{G}$, it checks whether $\boldsymbol{L}$ contains a representation $P = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$. If it does not, Sim adds P to $\boldsymbol{B}$ and the trivial representations $P = P, P^{-1} = P^{-1}$ to $\boldsymbol{L}$. Whenever Sim observes a query $\mathsf{O}(P_1, P_2)$ to the group oracle with answer P,

- For $j \in \{1, 2\}$, if $\boldsymbol{L}$ does not contain a representation $P_j = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z}$, Sim adds P_j to $\boldsymbol{B}$ and the trivial representations $P_j = P_j, P_j^{-1} = P_j^{-1}$ to $\boldsymbol{L}$.
- Sim retrieves the unique representations $P_1 = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z^1}$ and $P_2 = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z^2}$ of (P_1, P_2) and adds to $\boldsymbol{L}$ the representations $P = \prod_{Z \in \boldsymbol{B}} Z^{\gamma_Z^1 + \gamma_Z^2}$ and $P^{-1} = \prod_{Z \in \boldsymbol{B}} Z^{-\gamma_Z^1 - \gamma_Z^2}$.
- If P or P^{-1} has two distinct representations in $\boldsymbol{L}$ (where representations are distinct if their are not equal as formal monomials), Sim raises a flag Coincidence.

Interacting with the signing oracle. Throughout the course of the simulation, Sim might interact with the signing oracle SO to turn an attacker against the protocol into an attacker against the (doubly enhanced) unforgeability of ECDSA.

- Initialization: Sim receives (g, X) from the signing oracle. It adds g, X to $\boldsymbol{B}$ and the trivial representations $g = g, g^{-1} = g^{-1}, X = X, X^{-1} = X^{-1}$ to $\boldsymbol{L}$.
- Presignatures: Whenever Sim sends a request (m, δ) for a presignature to the signing oracle, upon receiving the answer R, it adds R, R^{-1} to $\boldsymbol{B}$ and the trivial representations $R = R, R^{-1} = R^{-1}$ to $\boldsymbol{L}$. If R or R^{-1} has two distinct representations in $\boldsymbol{L}$, Sim raises a flag Coincidence.
- Modifier: after sending a pair (ℓ, ψ) (for a presignature query (m, δ) with answer R) and receiving a signature s, Sim
  - computes $\mu = \mathsf{RO}(X \cdot g^{\delta}, R^{\ell}, \psi, m)$
  - sets $\hat{r} = \mathcal{X}((R \cdot g^{\mu})^{\ell})$
  - substitutes all occurrences of R in the right hand side of representations in $\boldsymbol{L}$ with $g^{(m+\hat{r}\delta)\cdot(s\ell)^{-1}} \cdot X^{\hat{r}\cdot(s\ell)^{-1}}$.

6.1.1 Unique representations

It is a standard lemma (that follows from the analysis of Theorem 1 in [Sho97]) that when Sim interfaces only with the group oracle, the probability of raising a flag Coincidence is at most Q_{O}^2/q, where Q_{O} denotes the total number of queries to the group oracle. A variant of the lemma accounting for interactions with a signing oracle was stated in [Bro02, Lemma 1]. A further generalization of this lemma, accounting for presignatures, was established in [GS22]; all variants are proven via a relatively straightforward use of the Schwarz-Zippel lemma. In this work, we use a slight variants of the latter that also accounts for the use of modifiers in our doubly-enhanced ECDSA experiment:

Lemma 6.2 (Unique representations)**.** *Let Q denote a bound on the total number of queries to either the group oracle or the signing oracle. Then,*

$$\Pr[\mathsf{Coincidence}] \leq \frac{O(Q^2)}{q}.$$

Proof. The proof of Lemma 6.2 proceeds via a straightforward application of Theorem B.1:

Game_0**.** This is the original experiment, where Sim behaves as described in Section 6.1 and interacts with a true generic group oracle $\boldsymbol{G}$ and signing oracle SO.

Game_1**.** In this game, the reduction emulates $\boldsymbol{G}$ and SO via the lazy symbolic evaluator from Section 7.1.1. By Theorem B.1, the advantage of any adversary in distinguishing Game_0 from Game_1 is at most $O(Q^2/|\boldsymbol{G}|)$ (where for large enough Q, the constant in the $O(\cdot)$ is smaller than 6 and approaches 5). Therefore,

$$|\Pr_{\mathsf{Game}_0}[\mathsf{Coincidence}] - \Pr_{\mathsf{Game}_1}[\mathsf{Coincidence}]| \leq \frac{O(Q^2)}{q}.$$

Furthermore, observe that by design, the lazy symbolic evaluator associates a unique group element to any exponent *viewed as its formal representation*: if two elements differ in their representation, the lazy symbolic evaluator associates them to distinct group elements (or aborts). Therefore,

$$\Pr_{\mathsf{Game}_1}[\mathsf{Coincidence}] = 0,$$

which concludes the proof. □

6.1.2 Knowledge of exponents

The knowledge of exponent assumption, initially introduced in [Dam92] and generally called KEA1, states that for any PPT adversary $\mathcal{A}$, there exists an extractor $E_{\mathcal{A}}$ such that the following holds:

$$\Pr[\exists x, (X,Y) = (g^x, h^x) \wedge x' \neq x \ : \ (g,h) \leftarrow \mathbb{G}^2, (X,Y) \leftarrow \mathcal{A}(g,h), x' \leftarrow E_{\mathcal{A}}(g,h)] = \mathrm{neg}(\lambda),$$

where the probability is taken over the random sampling of g, h and the coins of $\mathcal{A}$. We restate below the well-known result that KEA1 holds in the generic group model (with a universal extractor E that works for every adversary $\mathcal{A}$):

Lemma 6.3. *Let $\boldsymbol{G}$ be a generic group, and let $\mu : \mathbb{G} \to \boldsymbol{G}$ be the associated bijective mapping. Fix a generic Q-query adversary $\mathcal{A}$ and let E run the lazy symbolic simulator* Sim *described in Section 6.1. Sample $(g,h) \leftarrow \boldsymbol{G}^2$ and $(X,Y) \leftarrow \mathcal{A}(h,g)$. E outputs x' if $\boldsymbol{L}$ contains representations $X = g^{x'}$ and $Y = h^{x'}$; else, E outputs $\perp$. Then:*

$$\Pr[E \text{ returns } \perp \wedge Y = h^{\mu^{-1}(X)}] \leq Q/q.$$

Lemma 6.3 is a classical result and has been proven in several works [Den06; AF07] in the setting where no signing oracle is provided, using the standard lazy symbolic emulation of the generic group model from[Sho97]. It is relatively immediate to observe that these analyses can be ported directly to our setting by replacing the lazy symbolic evaluator with the one described in Section 6.1 and relying on Theorem B.1 to argue that the failure probability of the extractor does not change significantly when the generic group is replaced with the lazy symbolic evaluator. We omit the straightforward details.

6.2 The client is corrupted, single key-generation

We consider an adversary $\mathcal{A}$ corrupting the client C and interacting with the honest server S as follows:

1. S executes the non-interactive setup procedure from Algorithm 4.
2. S and C engage in a single execution of the key-generation procedure from Protocol 2.
3. S and C engage in polynomially-many signing sessions (Protocol 3) on messages chosen by $\mathcal{A}$. Let $\mathsf{N_{ss}}$ denote the number of signing sessions and $(m_i)_{i \leq \mathsf{N_{ss}}}$ denote the signed messages.

At the end of the interaction, $\mathcal{A}$ outputs a pair (m^*, σ^*). We prove the following:

Theorem 6.4. *Let $\mathcal{A}$ denote an adversary corrupting C, making at most Q_{RO} queries to the random oracle RO and Q_{O} queries to the generic group $\boldsymbol{G}$. Let $\mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}}(1^\ell)$ denote the probability that $\mathcal{A}$ outputs a valid forgery (m^*, σ^*) where σ^* verifies and $m^* \notin (m_i)_{i \le \mathsf{N_{ss}}}$. Then there exists a PPT adversary $\mathcal{B}$ (with running time comparable to that of $\mathcal{A}$) such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}} \le & \left(2\mathsf{Adv}^{\mathsf{ecdsa}}_{\mathcal{B}} + \frac{Q_{\mathsf{O}}}{q} + 2^{-\nu+\ell_0} + \mathsf{Adv}^{\mathsf{sedcr}}_{\mathcal{B}}\right) \cdot \frac{2^{\ell_0}}{1 - Q_{\mathsf{RO}}/2^\ell} \\ & + Q_{\mathsf{RO}} \cdot \left(\frac{1}{2^\ell} + \frac{2}{q} + \frac{4Q_{\mathsf{RO}}}{2^{2\ell}}\right) + \frac{\sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{srsa}}_{\mathcal{B}}}}{2} + 1.25 \cdot 2^{-\nu} + \frac{6Q_{\mathsf{O}}^2 + 1}{q},\end{aligned}$$

where ν denotes a statistical security parameter, $\ell_0 = \lceil \ell / r_{\mathsf{df}*} \rceil$, and $r_{\mathsf{df}*}$ is the number of repetitions of the proof $\Pi_{\mathsf{df}*}$.

The rest of this section is dedicated to the proof of Theorem 6.4. Assume that

$$\Pr[\sigma^* \text{ verifies } \wedge \ m^* \notin (m_i)_{i \le \mathsf{N_{ss}}}] \ge \varepsilon,$$

for some non-negligible ε, where the randomness is taken over the coins of $\mathcal{A}$ and S. We show via a sequence of game hops how to construct from $\mathcal{A}$ an attack against the doubly-enhanced existential unforgeability of ECDSA.
Let Q_{RO} and Q_{O} denote the total number of queries made by $\mathcal{A}$ to the random oracle RO and the total number of queries made by both Adv and Sim to the generic group oracle O respectively. In any game Game_i, we write $\Pr[\mathsf{Game}_i]$ to denote the probability that $\mathcal{A}$ outputs an accepting signature σ^* on a message m^* (that was not previously signed) at the end of Game_i. We proceed through a sequence of game hops.

Game_0**.** This is the normal game, where the simulator Sim emulates honestly the role of S. We have $\Pr[\mathsf{Game}_0] \ge \varepsilon$.

Game_1**.** In this game, Sim interfaces with the generic group oracle as prescribed in Section 6.1, and emulates the random oracle lazily (that is, for each previously unseen query query from $\mathcal{A}$, it samples a uniformly random output $\mathsf{RO}(\mathsf{query}) \leftarrow \{0,1\}^{2\ell}$). If Sim does not abort, this game is perfectly indistinguishable from the previous game, hence by Lemma 6.2:

$$|\Pr[\mathsf{Game}_0] - \Pr[\mathsf{Game}_1]| \le \Pr[\mathsf{Coincidence}] \le \frac{6 \cdot Q_{\mathsf{O}}^2}{q}.$$

Game_2**.** In this game, we delay the choice of the content of the commitment B sent by the server in Protocol 2:

- In step 1 of Protocol 2, instead of setting $B = \mathsf{RO}(ssid, \mathsf{S}, X_2)$, Sim samples $B \leftarrow \{0,1\}^{2\ell}$. Sim plays as in Game_1 otherwise, and in particular, samples $x_2' \leftarrow \pm 2^{n_x}$ uniformly and sets $X_2 = g^{x_2'}$.
- In step 3 of Protocol 2, before sending X_2, Sim checks whether the tuple $(ssid, \mathsf{S}, X_2)$ had been queried by $\mathcal{A}$. If it was queried, it raises a flag $\mathsf{Fail}_{\mathsf{RO}}$ and aborts the simulation. Else, it programs RO to output B on input $(ssid, \mathsf{C}, X_2)$, and sends $(X_2, \mathcal{E}, \psi_{\mathsf{dlenc}})$.

Because X_2 is a uniformly random group element sampled independently of $\mathcal{A}$, the probability that $\mathcal{A}$ placed a query $(ssid, \mathsf{C}, X_2)$ before Sim sends X_2 is at most Q_{RO}/q. Conditioned on not raising a flag $\mathsf{Fail}_{\mathsf{RO}}$, the view of $\mathcal{A}$ in Game_2 is perfectly identical to its view in Game_1 and we have:

$$|\Pr[\mathsf{Game}_1] - \Pr[\mathsf{Game}_2]| \leq \Pr[\mathsf{Fail}_{\mathsf{RO}}] \leq \frac{Q_{\mathsf{RO}}}{q}.$$

Game_3. We let $\mathsf{Sim}_{\mathsf{df}}, \mathsf{Sim}_{\mathsf{mod}}, \mathsf{Sim}_{\mathsf{fac}}$, and $\mathsf{Sim}_{\mathsf{dlenc}}$ denote zero-knowledge simulators for the proofs ψ_{df}, ψ_{mod}, ψ_{fac}, and ψ_{dlenc} respectively (all simulators are implicitly given programmable access to the random oracle). In this game, Sim emulates all proofs using their respective zero-knowledge simulator, programming the random oracle whenever needed. We denote ψ^*_{df}, ψ^*_{mod}, ψ^*_{fac}, and ψ^*_{dlenc} the simulated proofs. We have

$$|\Pr[\mathsf{Game}_3] - \Pr[\mathsf{Game}_2]| \leq 2^{-\nu} + 2^{-\nu+2} + 4 \cdot Q^2_{\mathsf{RO}}/2^{2\ell},$$

using the fact that Π_{df}, Π_{mod}, $\Pi_{\mathsf{fac}}, \Pi_{\mathsf{dlenc}}$ are $2^{-\nu+1}$-HVZK, 0-HVZK, $2^{-\nu+1}$-HVZK, and $2^{-\nu+2}$-HVZK respectively, as well as Remark 5.2.

Game_4. In this game, when receiving the proof ψ^{dlog} in Protocol 2, Sim stores all queries of query-answer pairs of $\mathcal{A}$ to RO and $\mathcal{O}$ in $\mathcal{Q}^*$ and invokes the generic group model extractor $\mathrm{GGMExt}_{\mathsf{dlog}}$ from Claim 5.15 on input $(ssid, \mathsf{C}, X_1, \psi_{\mathsf{dlog}})$ and query-answer list $\mathcal{Q}^*$. If $\mathrm{GGMExt}_{\mathsf{dlog}}$ outputs $\perp$, Sim aborts. Otherwise, Sim obtains the discrete logarithm x_1 of X_1. We have

$$|\Pr[\mathsf{Game}_4] - \Pr[\mathsf{Game}_3]| \leq \frac{1}{q}.$$

Proof. The only difference between Game_4 and Game_4 happens if Sim does not abort in Game_3 but Sim aborts in Game_4. Conditioned on Sim not aborting in Game_3, no flag $\mathsf{Coincidence}$ is raised and the proof ψ_{dlog} verifies, in which case it follows from Claim 5.15 that $\mathrm{GGMExt}_{\mathsf{dlog}}$ outputs x_1 with probability at least $1 - 1/q$. □

Game_5. In this game, we let Sim sample an ECDSA secret key x ahead of time. Then, instead of sampling x_2 in the interactive key generation protocol (protocol 2), Sim does the following:

- In step 1 of Protocol 2, Sim samples and sends $B \leftarrow \{0,1\}^{2\ell}$.
- In step 3 of Protocol 2, after extracting x_1, Sim sets $x_2 \leftarrow x - x_1$, $X_2 \leftarrow g^{x_2}$, and randomly samples $x'_2 \leftarrow \pm 2^{n_x}$ such that $x'_2 = x_2 \bmod q$. If no query $(ssid, X_2, \mathsf{S})$ was made to RO, Sim programs $B = \mathsf{RO}(ssid, X_2, \mathsf{S})$ and behaves as in Game_4 for the rest of the game. Else, if a query $(ssid, X_2, \mathsf{S})$ was already made to RO, Sim aborts.

If no query $(ssid, X_2, \mathsf{S})$ was already made to RO, Game_4 is perfectly identical to Game_5. Furthermore, as x is uniformly random over $\mathbb{Z}_q$ and X_2 fully determines $x = \mathsf{dlog}_g(X_2 \cdot X_1)$, this happens with probability at most Q_{RO}/q. Hence:

$$|\Pr[\mathsf{Game}_5] - \Pr[\mathsf{Game}_4]| \leq \frac{Q_{\mathsf{RO}}}{q}.$$

Game_6. In this game, when receiving the proof ψ^{sig} in Protocol 3, Sim stores all queries of query-answer pairs of $\mathcal{A}$ to RO and $\mathcal{O}$ in $\mathcal{Q}^*$ and invokes the generic group model extractor $\mathrm{GGMExt}_{\mathsf{sig}}$ from Claim 5.12 on input $(\mathrm{aux}^* = (ssid, \mathsf{C}), \tilde{P}, \tilde{U}, (\mathbb{G}, g, h, f, q, N, \rho, \hat{N}, t, s_1, s_2), \mathcal{S}, \mathcal{E}, \psi_{\mathsf{sig}})$ and query-answer list $\mathcal{Q}^*$. If $\mathrm{GGMExt}_{\mathsf{sig}}$ outputs $\perp$, Sim aborts. Otherwise, Sim obtains the representation (a, b, μ) of $\tilde{U} = g^a h^b f^\gamma$. We have

$$|\Pr[\mathsf{Game}_5] - \Pr[\mathsf{Game}_6]| \leq \frac{1}{2^{\ell+1}}.$$

Proof. The only difference between Game_6 and Game_5 happens if Sim does not abort in Game_5 but Sim aborts in Game_6. Conditioned on Sim not aborting in Game_5, no flag $\mathsf{Coincidence}$ is raised and the proof ψ_{sig} verifies, in which case it follows from Claim 5.12 that $\mathrm{GGMExt}_{\mathsf{sig}}$ outputs (a, b, μ) with probability at least $1 - 1/2^{\ell+1}$. □

Game_7. In this game, Sim is not given the Paillier decryption key anymore, and does not decrypt the ciphertext $\mathcal{S}$ to compute σ at the end of Protocol 3. Rather, after receiving the proof $\psi_{\mathsf{sig}} = (\widetilde{P}, \widetilde{U}, \widetilde{B}, \mathcal{D}, e, z_1, z_2, w_1, w_2)$ in each signing session and extracting a triple $(a, b, \gamma) \in \mathbb{Z}_q^3$ such that $\tilde{U} = g^a h^b f^\gamma$, Sim uses the quantity $a + x_2 b$ instead of $\mathrm{dec}_{\varphi(N)}(\mathcal{S})$; that is, it computes σ as $\sigma = (a + x_2 b) \cdot (k_2 \cdot \mathsf{RO}(X, R_1^{k_2}, m))^{-1} \bmod q$. We show:

Lemma 6.5. *Assume that there exists a non-negligible δ such that*

$$|\Pr[\mathsf{Game}_7] - \Pr[\mathsf{Game}_6]| \geq \delta.$$

Then there exists a polynomial-time algorithm $\mathcal{B}$ which breaks the strong-RSA assumption (Definition 3.3) with advantage

$$\mathsf{Adv}^{\mathsf{srsa}}_{\mathcal{B}}(1^\ell) \geq \left(\frac{\delta^2}{Q_{\mathsf{RO}}} + \frac{\delta}{2^{2\ell}}\right) \cdot \left(\frac{5}{2} - 2\frac{\phi(N)}{N}\right) > \frac{2 \cdot \delta^2}{Q_{\mathsf{RO}}}.$$

Proof. Let $\mathsf{BadCiphertext}$ denote the event that there exists a signing session such that $\mathrm{dec}_{\varphi(N)}(\mathcal{S}) \neq a + x_2 b \bmod q$. Clearly, if $\mathsf{BadCiphertext}$ does not happen, Sim's behavior is identical in Game_6 and Game_7. That is, $|\Pr[\mathsf{Game}_7] - \Pr[\mathsf{Game}_6]| \leq \Pr[\mathsf{BadCiphertext}]$. We now bound $\Pr[\mathsf{BadCiphertext}]$; let $\delta \geq |\Pr[\mathsf{Game}_7] - \Pr[\mathsf{Game}_6]| \geq \Pr[\mathsf{BadCiphertext}]$. Let $\mathcal{A}$ denote the following algorithm: on input $\mathsf{SetupData} = (N, \hat{N}, \rho_0, \rho, t, s_1, s_2, h, f, \psi_{\mathsf{mod}}, \psi_{\mathsf{fac}}, \psi_{\mathsf{df}})$ and a Q_{RO}-tuple of uniformly random elements $(h_1, \cdots, h_{Q_{\mathsf{RO}}}) \in (\{0,1\}^{2\ell})^{Q_{\mathsf{RO}}}$, it emulates the interaction between Sim and Adv in Game_7, answering the j-th random oracle query of Adv with h_j. Eventually, if Sim raises a flag $\mathsf{BadCiphertext}$ after receiving a proof $\psi_{\mathsf{sig}} = (\widetilde{P}, \widetilde{U}, \widetilde{B}, \mathcal{D}, e, z_1, z_2, w_1, w_2)$ with $e = h_{i^*}$, it outputs $(i^*, \psi_{\mathsf{sig}})$. Otherwise, it outputs $(0, \perp)$.
By the Generalized Forking Lemma (Lemma 3.1), denoting D_{IG} the distribution of (simulated) $\mathsf{SetupData}$ sampled by Sim, there exists an algorithm $\mathsf{F}_{\mathcal{A}}$ that, on input $\mathsf{SetupData}$, returns a triple $(b, \mathsf{so}, \mathsf{so}')$ with $b \in \{0, 1\}$ such that

$$\mathsf{p}_{\mathsf{fork}} = \Pr[b = 1 \mid \mathsf{SetupData} \leftarrow \mathsf{D}_{\mathsf{IG}}, (b, \mathsf{so}, \mathsf{so}') \leftarrow \mathsf{F}_{\mathcal{A}}(\mathsf{SetupData})] \geq \frac{\delta^2}{Q_{\mathsf{RO}}} - \frac{\delta}{2^{2\ell}},$$

and when $b = 1$, $(\mathsf{so}, \mathsf{so}') = (\psi_{\mathsf{sig}}, \psi'_{\mathsf{sig}})$ where $\psi_{\mathsf{sig}} = (\widetilde{P}, \widetilde{U}, \widetilde{B}, \mathcal{D}, h_i, z_1, z_2, w_1, w_2)$, $\psi'_{\mathsf{sig}} = (\widetilde{P}, \widetilde{U}, \widetilde{B}, \mathcal{D}, h'_i, z'_1, z'_2, w'_1, w'_2)$ are two accepting proofs, and $L_{\tilde{U}} = a + b \cdot Z_h + \gamma \cdot Z_f$ with

$a + x_2 \cdot b \neq \mathrm{dec}_{\varphi(N)}(\mathcal{S}) \bmod q$. In particular, by Lemma 5.10, we get that there exists an extractor E given auxiliary input $\rho_0 \in \mathbb{Z}^*_{N^2}$ with $\rho_0^N = \rho \bmod N^2$ such that at least one of the following holds true:

1. $\delta \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that $\delta \nmid \gcd(\alpha, \beta)$ and $s_1^\alpha s_2^\beta t^\gamma = \tilde{P}^\delta \bmod \hat{N}$, or
2. $k \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that $k \notin \{1, N\}$, $|k| < 2^\ell$ and $\gcd(k, N) \neq 1$.

The second case above is ruled out by the small factor proof ψ_{fac} from $\mathsf{SetupData}$, which guarantees that no factor of N can be smaller than $\sqrt{N}/2^\ell \gg 2^\ell$ (since $N > 2^{4\ell}$). The first case above yields to a contradiction to the strong RSA assumption with probability $\mathsf{p}_{\mathsf{fork}} \cdot \left(\frac{5}{2} - 2\frac{\phi(N)}{N}\right)$ by Lemma 3.12. This concludes the proof of Lemma 6.5 □

Before moving on to the next games, we recall a few parameters (we refer the reader to Section 8.1 for a detailed breakdown of our parameters) and introduce a few notations. The challenge space for the proof $\psi_{\mathsf{df}*}$ computed via $\Pi^{\mathrm{FS}}_{\mathsf{df}*}$ is $\pm 2^{\ell_0}$, where $\ell_0 = \lceil \ell / r_{\mathsf{df}*} \rceil$. $r_{\mathsf{df}*}$ denotes the repetition parameter for the Σ-protocol $\Pi_{\mathsf{df}*}$, chosen such that the soundness error is at most $2^{-\ell}$. Eventually, x_2 is sampled from $\pm 2^{n_x}$ with $2^{n_x} \geq q \cdot 2^{\nu_{\mathrm{leak}}}$, and $\nu_{\mathrm{leak}} = \ell_0 + 40$ (40 being some fixed choice of statistical security parameter). Given the Damgård-Fujisaki parameters (u_1, u_2, v), let ω denote the maximal order of an element σ_i such that u_i/σ_i is in the subgroup generated by v. Rewrite x_2', β as follows:

- $x_2' = x_{\mathrm{leak}} + \omega \cdot x_2''$ with $x_{\mathrm{leak}} < \omega$,
- $\beta = \beta_{\mathrm{leak}} + \omega \cdot \beta'$ with $\beta_{\mathrm{leak}} < \omega$.

With this ω-ary decomposition, we can rewrite $\mathcal{E} = \mathrm{enc}_N(x_2'; \rho^\beta)$ as $\mathcal{E} = \mathcal{E}_{\mathrm{leak}} \cdot (\mathcal{E}')^\omega \bmod N^2$, with $\mathcal{E}_{\mathrm{leak}} = \mathrm{enc}_N(x_{\mathrm{leak}}; \rho^{\beta_{\mathrm{leak}}})$ and $\mathcal{E}' = \mathrm{enc}_N(x_2''; \rho^{\beta'})$.

Game_8. In this game, we let Sim sample a *guess* $\tilde{\omega} \leftarrow [2^{\ell_0}]$ for the value of ω. Then, Sim samples $x_{\mathrm{leak}}, \beta_{\mathrm{leak}} \leftarrow \mathbb{Z}_{\tilde{\omega}}$, $x_2'' \leftarrow \{0, \cdots, \lfloor 2^{n_x}/\tilde{\omega} \rfloor\}$, $\beta' \leftarrow \{0, \cdots, \lfloor 2^{n_\lambda}/\tilde{\omega} \rfloor\}$, and $\mu \leftarrow \hat{M} \cdot 2^\nu$. Sim sets

- $x_2' = x_{\mathrm{leak}} + \tilde{\omega} \cdot x_2''$, $X_2 \leftarrow g^{x_2'}$, and $\beta = \beta_{\mathrm{leak}} + \tilde{\omega} \cdot \beta'$,
- $\mathcal{E} = \mathcal{E}_{\mathrm{leak}} \cdot (\mathcal{E}')^{\tilde{\omega}} \bmod N^2$, with $\mathcal{E}_{\mathrm{leak}} = \mathrm{enc}_N(x_{\mathrm{leak}}; \rho^{\beta_{\mathrm{leak}}})$ and $\mathcal{E}' = \mathrm{enc}_N(x_2''; \rho^{\beta'})$,
- $\tilde{P} = u_1^{x_{\mathrm{leak}}} u_2^{\beta_{\mathrm{leak}}} v^\mu \bmod \hat{M}$.

Let us denote E the event that there exists $\omega < 2^{\ell_0}$ such that ω is the maximal order of any element σ_i for which u_i/σ_i is in the subgroup generated by v (for $i = 1, 2$). By the soundness of $\Pi^{\mathrm{FS}}_{\mathsf{df}*}$, for any adversary making at most Q_{RO} queries to the random oracle and outputting $(u_1, u_2, v, \hat{M})$ together with an accepting proof $\psi_{\mathsf{df}*}$, the probability of E is at least $1 - Q_{\mathsf{RO}}/2^\ell$. Furthermore, conditioned on E and on Sim's guess $\tilde{\omega}$ being correct, i.e., $\tilde{\omega} = \omega$, Game_8 is distributed identically as Game_7. That is,

$$\Pr[\mathsf{Game}_7 \mid E] = \Pr[\mathsf{Game}_8 \mid E, \omega = \tilde{\omega}] \geq \Pr[\mathsf{Game}_7] - Q_{\mathsf{RO}}/2^\ell.$$

Furthermore,

$$\begin{aligned}\Pr[\mathsf{Game}_8] &= \Pr[\mathsf{Game}_8 \mid E, \omega = \tilde{\omega}] \cdot \Pr[E, \omega = \tilde{\omega}] + \Pr[\mathsf{Game}_8 \mid \neg E \vee \omega \neq \tilde{\omega}] \cdot (1 - \Pr[E, \omega = \tilde{\omega}]) \\ &= \Pr[\mathsf{Game}_8 \mid E, \omega = \tilde{\omega}] \cdot \frac{1 - Q_{\mathsf{RO}}/2^{\ell}}{2^{\ell_0}} + \Pr[\mathsf{Game}_8 \mid \neg E \vee \omega \neq \tilde{\omega}] \cdot (1 - \Pr[E, \omega = \tilde{\omega}]) \\ &\geq (\Pr[\mathsf{Game}_7] - Q_{\mathsf{RO}}/2^{\ell}) \cdot \frac{1 - Q_{\mathsf{RO}}/2^{\ell}}{2^{\ell_0}}.\end{aligned}$$

Game_9. In this game, Sim plays as in Game_8 except that he samples $\mathcal{E}'$ as a uniformly random ciphertext over QR_{N^2}. Distinguishing Game_9 from Game_8 reduces immediately to the SEDCR assumption, and we have

$$|\Pr[\mathsf{Game}_9] - \Pr[\mathsf{Game}_8]| \leq \mathsf{Adv}^{\mathsf{sedcr}}_{\mathcal{B}}(1^{\ell}),$$

where $\mathcal{B}$ denotes the adversary that receives an SEDCR challenge $\mathcal{E}'$ and runs the interaction between Sim and $\mathcal{A}$ as in Game_9 (using the challenge $\mathcal{E}'$ instead of sampling it at random).

Game_{10}. In this game, Sim plays as in Game_9, except that it samples $x_2 \leftarrow \mathbb{Z}_q$ (independently of x_{leak}) and sets $X_2 \leftarrow g^{x_2}$. We show:

$$|\Pr[\mathsf{Game}_9] - \Pr[\mathsf{Game}_{10}]| \leq 2^{-40}.$$

Proof. In Game_9, x_2 is sampled as $[x_{\text{leak}} + \tilde{\omega} \cdot x_2'' \bmod q]$ with $x_2'' \leftarrow [\lfloor 2^{n_x}/\tilde{\omega} \rfloor]$. as $q \gg \tilde{\omega}$ is prime, $\tilde{\omega}$ and q are coprime. Using the parameter selection outlined in Section 8.1, $\lfloor 2^{n_x}/\tilde{\omega} \rfloor \geq 2^{n_x - \ell_0} \geq q \cdot 2^{40}$. Therefore, the distribution of $[\tilde{\omega} \cdot x_2'' \bmod q]$ is within statistical distance 2^{-40} of the uniform distribution over $\mathbb{Z}_q$, which implies that $x_2 = [x_{\text{leak}} + \tilde{\omega} \cdot x_2'' \bmod q]$ is within distance 2^{-40} of the uniform distribution over $\mathbb{Z}_q$ in Game_9. This concludes the proof. □

Game_{11}. In this game, when receiving the pair (R_1, R) in Protocol 3 and instead of checking that $R = R_1^{k_2}$, Sim retrieves the representations of (R_1, R) from $\boldsymbol{L}$. If $R_1 \neq g^{k_1}$ or $R \neq R_2^{k_1}$, Sim aborts. Otherwise, Sim obtains the discrete logarithm k_1 of R_1. By Lemma 6.3, we have

$$|\Pr[\mathsf{Game}_{10}] - \Pr[\mathsf{Game}_{11}]| \leq \frac{Q_{\mathsf{O}}}{q}.$$

Proof. The only difference between Game_{11} and Game_{10} happens if Sim does not abort in Game_{10} (hence $R = R_1^{k_2}$) but Sim aborts in Game_{11}. If this happens with probability more than Q_{O}/q, we get a contradiction to Lemma 6.3 by receiving a random KEA1 challenge (g, R_2), constructing $Y = R_2^{x_1}$ (recall that x_1 is extracted from ψ^{dlog}) and outputting the pair (R, R_1) obtained from C. □

Game_{12}. In this game, Sim does not sample x anymore at the beginning of Protocol 2. Instead, Sim interacts with the signing oracle SO (Experiment 7.1):

- At the beginning of 2, Sim receives $X \in \boldsymbol{G}$ from SO.
- In step 1 of Protocol 3, Sim sends $(m, 0)$ to SO and receives R_2 from SO. After extracting $x_1 = \mathsf{dlog}_g(X_1)$, Sim sends $(R_2, Y = R_2^{x_1})$ to C.

- In step 3 of Protocol 3, after receiving R_1 and extracting $k_1 = \mathsf{dlog}_g(R_1)$, and $(a, b, \gamma) \in \mathbb{Z}_q^3$ from C, Sim checks whether $a = k_1^{-1} \cdot (m + rx_1) \bmod q$ and $b = k_1^{-1} \cdot r \bmod q$.
- If this checks fails, Sim aborts.
- Else, Sim sends k_1 to SO, receives σ, and outputs (σ, r).

During the above interactions with SO, Sim updates the lists $\boldsymbol{B}, \boldsymbol{L}$ as prescribed in Section 6.1. We prove the following:

$$|\Pr[\mathsf{Game}_{12}] - \Pr[\mathsf{Game}_{11}]| \leq \mathsf{Adv}^{\mathsf{ecdsa}}_{\mathcal{B}}(1^\ell),$$

Where $\mathcal{B}$ denotes the adversary defined in the proof below.

Proof. If it holds that $a = k_1^{-1} \cdot (m + rx_1) \bmod q$ and $b = k_1^{-1} r \bmod q$, then denoting $\mu = \mathsf{RO}(X, R_1, R, m)$, we have $\sigma = (a + x_2 b) \cdot (k_2 + \mu)^{-1} = (m + rx) \cdot (k_1 \cdot (k_2 + \mu))^{-1} \bmod q$ in Game_{12}, hence the signature computed in Game_{12} is identical to the signature obtained by Sim in Game_{11}. Therefore, Game_{12} and Game_{11} differ only when $a \neq k_1^{-1} \cdot (m + rx_1) \bmod q$ or $b \neq k_1^{-1} \cdot r \bmod q$, yet the signature $(r, \sigma = (a + x_2 b) \cdot (k_2 + \mu)^{-1})$ verifies. It is straightforward to check that if $a \neq k_1^{-1} \cdot (m + rx_1) \bmod q$ and $b = k_1^{-1} \cdot r \bmod q$, the equation $\sigma = (a + x_2 b) \cdot (k_2 + \mu)^{-1} = (m + rx) \cdot (k_1 \cdot (k_2 + \mu))^{-1} \bmod q$ cannot be satisfied, as it simplifies to

$$\begin{aligned} & a + x_2 b = (m + rx) \cdot k_1^{-1} \bmod q \\ \implies & a + xb - x_1 b = m k_1^{-1} + xb \bmod q \\ \implies & a - x_1 r k_1^{-1} = m k_1^{-1} \bmod q \\ \implies & a = k_1^{-1} \cdot (m + rx_1) \bmod q. \end{aligned}$$

Therefore, we necessarily have $b \neq k_1^{-1} \cdot r \bmod q$. But then,

$$\begin{aligned} & a + x_2 b = (m + rx) \cdot k_1^{-1} \bmod q \\ \implies & k_1(a - x_1 b) - m = x(r - k_1 b) \bmod q \\ \implies & x = k_1(a - x_1 b) \cdot (r - k_1 b)^{-1} \bmod q, \end{aligned}$$

Hence, we let $\mathcal{B}$ denote the adversary that runs the interaction between Sim and $\mathcal{A}$, playing the role of Sim, (who knows x_1, k_1), and using it as above to extract the ECDSA secret key x. Using x, outputting a forgery is straightforward. □

Remark 6.6. The reduction given above extracts the ECDSA secret key given a distinguisher between Game_{11} and Game_{12}. Assuming the doubly-enhanced unforgeability of ECDSA is therefore somewhat of an overkill here: it suffices to assume that it is hard for an adversary playing the doubly-enhanced unforgeability game to output the ECDSA secret key, which is a much weaker assumption.

In Game_{12}, Sim only interacts with the signing oracle SO. Therefore, any forgery produced by $\mathcal{A}$ at the end of its interaction with Sim in Game_{12} yields a contradiction to the doubly-enhanced unforgeability of ECDSA. This concludes the proof of Theorem 6.4.

6.3 The client is corrupted, multiple key-generation

We now address the more general setting where the corrupted client and the server can engage in an arbitrary number of key generations. We consider an adversary $\mathcal{A}$ corrupting the client C and interacting with the honest server S as follows:

1. S executes the non-interactive setup procedure from Algorithm 4 with session id *sid*.
2. At any time, S and C engage in an execution of the key-generation procedure from Protocol 2 with subsession id *ssid*.
3. For each subsession id *ssid*, S and C engage in polynomially-many signing sessions (Protocol 3) on messages chosen by $\mathcal{A}$. Let $\mathsf{N}_{\mathsf{ss}}(ssid)$ denote the number of signing sessions with *ssid* and $(m^s sid_i)_{i \leq \mathsf{N}_{\mathsf{ss}}}$ denote the signed messages.

Corollary 6.7. *Let $\mathcal{A}$ denote an adversary corrupting C, making at most Q_{RO} queries to the random oracle RO and $\mathcal{Q}_{\mathsf{O}}$ queries to the generic group $\boldsymbol{G}$. Let $\mathsf{Adv}^{\mathsf{mtecdsa}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{mtecdsa}}_{\mathcal{A}}(1^\ell, t)$ denote the probability that $\mathcal{A}$ outputs a valid forgery (m^*, σ^*) with respect to at least one of t subsessions ssid where σ^* verifies and $m^* \notin (m_i)_{i \leq \mathsf{N}_{\mathsf{ss}}(ssid)}$. Then there exists a PPT adversary $\mathcal{B}$ (with running time comparable to that of $\mathcal{A}$) such that*

$$\mathsf{Adv}^{\mathsf{mtecdsa}}_{\mathcal{A}} \leq \frac{\sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{srsa}}_{\mathcal{B}}}}{2} + \frac{Q^2_{\mathsf{RO}}}{2^{2\ell}} t \cdot \mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{B}}.$$

We sketch below of the analysis of the single key generation setting extends to the multi-instance setting. We note that the reason why this is not immediately straightforward is that we cannot simply apply the simulator of Section 6.2 for each of the *ssid*, because the simulation involves *guessing* the order ω of a "cheating subgroup" that the adversary can introduce (this is the change introduced in Game_8). Concretely, the probability of guessing correctly is at least $1/2^{\ell_0} = 1/2^8$ (using our concrete choice of parameters with $r_{\mathsf{df}*} = 16$, $\ell = 128$, and $\ell_0 = \ell / r_{\mathsf{df}*}$). However, when simulating t key generation instances in parallel, the probability of guessing correctly would drop to 2^{-8t}.

At a high level, we circumvent this issue by simulating *only one branch*, hoping that the adversary will output an ECDSA forgery within this branch (this induces a $1/t$ loss, but this is the expected loss when moving from unforgeability to 1-out-of-t unforgeability, where the adversary wins if they manage to produce a forgery for one out of t public keys). However, using the simulator in one branch requires simulating without the knowledge of the factorization of N in all other branches (otherwise, the game hop in Game_9, invoking the SEDCR assumption, cannot be made). Therefore, for all other branches, we use a partial simulation that knows all secrets of S (x_2, k_2) except for the factorization of N, and plays as the honest server, except that it does not decrypt the ciphertext $\mathcal{S}$ and uses instead the extraction-based strategy of Game_7. Details follow.

Game_0**.** This is the normal game, where the simulator Sim emulates honestly the role of S. We have $\Pr[\mathsf{Game}_0] \geq \varepsilon$.

Game_1**.** In this game, Sim simulates the proof π_{mod} in Algorithm 4 using the zero-knowledge simulator $\mathsf{Sim}_{\mathsf{mod}}$ of $\Pi^{\mathrm{FS}}_{\mathsf{mod}}$. We have

$$|\Pr[\mathsf{Game}_1] - \Pr[\mathsf{Game}_0]| \leq Q^2_{\mathsf{RO}} / 2^{2\ell},$$

using the fact that Π_{mod} is 0-HVZK as well as Remark 5.2.

Game_2. In this game, Sim is not given the Paillier decryption key anymore, and does not decrypt the ciphertext $\mathcal{S}$ to compute σ at the end of Protocol 3. Instead, in all signing sessions and for all *ssid*'s, Sim plays exactly as in Game_7 of Section 6.2. By the same analysis, if there exists a non-negligible δ such that

$$|\Pr[\mathsf{Game}_2] - \Pr[\mathsf{Game}_1]| \geq \delta,$$

then there exists a polynomial-time algorithm $\mathcal{B}$ which breaks the strong-RSA assumption (Definition 3.3) with advantage

$$\mathsf{Adv}^{\mathsf{srsa}}_{\mathcal{B}}(1^\ell) \geq \left(\frac{\delta^2}{Q_{\mathsf{RO}}} + \frac{\delta}{2^{2\ell}}\right) \cdot \left(\frac{5}{2} - 2\frac{\phi(N)}{N}\right) > \frac{2\delta^2}{Q_{\mathsf{RO}}}.$$

Game_3. In this game, Sim guesses a subsession id *ssid*. For the guessed subsession, Sim plays exactly as the simulator in Game_{12} of Section 6.2. Note that in this subsession, Sim does not know the ECDSA secrey key; rather, it interacts with the signing oracle SO and produces a forgery if the adversary produces a forgery in this subsession. For all other subsessions, Sim plays as in Game_2.

6.4 The server is corrupted

We consider an adversary $\mathcal{A}$ corrupting the server S and interacting with the honest client C as follows:

1. C receives the non-interactive setup produced by S and runs the *client operations* of Algorithm 4.
2. S and C engage in a single execution of the key-generation procedure from Protocol 2.
3. S and C engage in polynomially-many signing sessions (Protocol 3) on messages chosen by $\mathcal{A}$. Let N_{ss} denote the number of signing sessions and $(m_i)_{i \leq \mathsf{N}_{\mathsf{ss}}}$ denote the signed messages.

At the end of the interaction, $\mathcal{A}$ outputs a pair (m^*, σ^*). We prove the following:

Theorem 6.8. *Let $\mathcal{A}$ denote an adversary corrupting S, making at most Q_{RO} queries to the random oracle RO and $\mathcal{Q}_{\mathsf{O}}$ queries to the generic group $\boldsymbol{G}$. Let $\mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}}(1^\ell)$ denote the probability that $\mathcal{A}$ outputs a valid forgery (m^*, σ^*) where σ^* verifies and $m^* \notin (m_i)_{i \leq \mathsf{N}_{\mathsf{ss}}}$. Then there exists PPT adversaries $\mathcal{B}, \mathsf{F}_{\mathcal{B}}, \mathsf{F}_{\mathcal{C}}$ (with running time comparable to that of $\mathcal{A}$) such that*

$$\mathsf{Adv}^{\mathsf{tecdsa}}_{\mathcal{A}} \leq \frac{Q_{\mathsf{O}}^2 + Q_{\mathsf{O}}}{q} + \frac{4Q_{\mathsf{RO}}^2 + Q_{\mathsf{RO}}}{2^{2\ell}} + 7 \cdot 2^{-\nu} + \theta + \mathsf{Adv}^{\mathsf{ecdsa}}_{\mathcal{B}},$$

where

$$\theta = \left(\mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^\ell) + \left(Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}} + \frac{1}{2^{2\ell}} + \frac{Q_{\mathsf{RO}}}{2^\ell}\right)^{1/2} + \frac{1}{2^{\nu-1}} + \frac{1}{2^{2\ell}}\right)^{1/2},$$

where $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}$ is a shorthand for $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3})$.

Let $\mathcal{A}$ be an adversary corrupting the server S and outputting a bit b after the interaction. Let Q_{RO} and $Q_{\mathcal{O}}$ denote the total number of queries made by $\mathcal{A}$ to the random oracle RO and the total number of queries made by both $\mathcal{A}$ and Sim to the generic group oracle $\mathcal{O}$ respectively. In any game Game_i, we write $\Pr[\mathsf{Game}_i]$ to denote the probability that $\mathcal{A}$ outputs 1 in Game_i. We proceed through a sequence of game hops.

Game_0. This is the normal game, where the simulator Sim emulates honestly the role of C.

Game_1. In this game, Sim interfaces with the generic group oracle as prescribed in Section 6.1, and emulates the random oracle lazily (that is, for each previously unseen query query from $\mathcal{A}$, it samples a uniformly random output $\mathsf{RO}(\mathsf{query}) \leftarrow \{0,1\}^{2\ell}$). If Sim does not abort, this game is perfectly indistinguishable from the previous game, hence by Lemma 6.2:

$$|\Pr[\mathsf{Game}_0] - \Pr[\mathsf{Game}_1]| \leq \Pr[\mathsf{Coincidence}] \leq \frac{Q_{\mathsf{O}}^2}{q}.$$

Game_2. In this game, Sim aborts whenever a hash collision happens, that is, whenever $\mathsf{RO}(\mathsf{query}) = \mathsf{RO}(\mathsf{query}')$ for two distinct queries $\mathsf{query}, \mathsf{query}'$ from $\mathcal{A}$. Since all outputs of RO are sampled uniformly at random from $\{0,1\}^{2\ell}$, we have

$$|\Pr[\mathsf{Game}_1] - \Pr[\mathsf{Game}_2]| \leq \frac{Q_{\mathsf{RO}}^2}{2^{2\ell}}.$$

Game_3. In this game, Sim aborts whenever the server S opens the hash commitment B to $(ssid, \mathsf{S}, X_2)$ in step 4 of Protocol 2 but the server had not made any oracle query to RO oin input $(ssid, \mathsf{S}, X_2)$ before sending B. Since this happens only if the server can find a hash preimage to B, we have

$$|\Pr[\mathsf{Game}_2] - \Pr[\mathsf{Game}_3]| \leq \frac{Q_{\mathsf{RO}}}{2^{2\ell}}.$$

From this game onward, we can assume that whenever S sends B in step 4 of Protocol 2, Sim can retrieve a uniquely-defined preimage $(ssid, \mathsf{S}, X_2)$ of B from its list of queries (conditioned on not aborting).

Game_4. We let We let $\mathsf{Sim}_{\mathsf{dlog}}, \mathsf{Sim}_{\mathsf{df}^*}$, and $\mathsf{Sim}_{\mathsf{sig}}$ denote zero-knowledge simulators for the proofs ψ_{dlog}, ψ_{df^*}, and ψ_{sig} respectively (all simulators are implicitly given programmable access to the random oracle). In this game, Sim emulates all proofs using their respective zero-knowledge simulator, programming the random oracle whenever needed. We denote ψ^*_{dlog}, $\psi^*_{\mathsf{df}^*}$, and ψ^*_{sig} the simulated proofs. We have:

$$|\Pr[\mathsf{Game}_3] - \Pr[\mathsf{Game}_4]| \leq 7 \cdot 2^{-\nu} + 3 \cdot \frac{Q_{\mathsf{RO}}^2}{2^{2\ell}},$$

using the fact that Π_{dlog}, Π_{df}, $\Pi_{\mathsf{sig}}, \Pi_{\mathsf{dlenc}}$ are 0-HVZK, $2^{-\nu+1}$-HVZK, and $5 \cdot 2^{-\nu}$-HVZK respectively, as well as Remark 5.2.

Game_5. In this game, when receiving the pair (R_2, Y) in Protocol 3 and instead of checking that $Y = R_2^{x_1}$, Sim retrieves the representations of (R_2, Y) from $\boldsymbol{L}$. If $R_2 \neq g^{k_2}$ or $Y \neq X_1^{k_2}$, Sim aborts. Otherwise, Sim obtains the discrete logarithm k_2 of R_2. By Lemma 6.3, we have

$$|\Pr[\mathsf{Game}_4] - \Pr[\mathsf{Game}_5]| \leq \frac{Q_{\mathsf{O}}}{q}.$$

Proof. The only difference between Game_5 and Game_4 happens if Sim does not abort in Game_4 (hence $Y = R_2^{x_1}$) but Sim aborts in Game_5. If this happens with probability more than Q_{O}/q, we get a contradiction to Lemma 6.3 by receiving a random KEA1 challenge (g, X_1) and outputting the pair (R_2, Y) obtained from S. □

Game_6. In this game, Sim does not sample x_1 anymore in step 2. Instead, Sim retrieves X_2 from the unique preimage $(ssid, \mathsf{S}, X_2)$ of B, samples $x \leftarrow \mathbb{F}_q$, sets $X \leftarrow g^x$, and sends $X_1 \leftarrow X/X_2$. Note that X_1 is distributed exactly as in Game_5, and that Sim computes ψ_{dlog} using $\mathsf{Sim}_{\mathsf{dlog}}$ (hence it does not need to know x_1).
In step 3 of Protocol 3, Sim does not compute u, v and send $\mathcal{S}$ as before. Instead, Sim computes the partial signature $\sigma' \leftarrow k_1^{-1}(m + rx) \bmod q$. Then, it samples $\mu \leftarrow \pm\lfloor 2^{n_a}/q \rfloor$, $\lambda_0 \leftarrow 2^{n_{\lambda_0}}$, $z \leftarrow \pm 2^{\varepsilon}$ and $\mathsf{S}^* \leftarrow \mathrm{enc}_N(\sigma' + \mu q; \rho^{\lambda_0}) \cdot \mathcal{E}^{q \cdot z}$. Sim sends $(R_1, R_1^{k_2}, \mathcal{S}^*, \psi^*_{\mathsf{sig}})$. We prove the following lemma:

Lemma 6.9. *There exists algorithms $\mathsf{F}_{\mathcal{B}}$, $\mathsf{F}_{\mathcal{C}}$ with running time comparable to that of $\mathcal{A}$ such that*

$$|\Pr[\mathsf{Game}_5] - \Pr[\mathsf{Game}_6]| \leq \left(\mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^\ell) + \left(Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}} + \frac{1}{2^{2\ell}} + \frac{Q_{\mathsf{RO}}}{2^\ell}\right)^{1/2} + \frac{1}{2^{\nu-1}} + \frac{1}{2^{2\ell}}\right)^{1/2},$$

where $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}$ is a shorthand for $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3})$.

Proof. Let $\delta = |\Pr[\mathsf{Game}_5] - \Pr[\mathsf{Game}_6]|$. Consider the following algorithm $\mathcal{B}$: it receives 1^ℓ and a Q_{RO}-tuple of uniformly random elements $(h_1, \cdots, h_{Q_{\mathsf{RO}}}) \in (\{0,1\}^{2\ell})^{Q_{\mathsf{RO}}}$. Then, it emulates the interaction between Sim and $\mathcal{A}$ in Game_6, answering the j-th random oracle query of Adv with h_j. Eventually, if case 1 occurs after receiving a proof $\psi_{\mathsf{dlenc}} = (A, \widetilde{B}, D, e, z_1, z_2, z_3)$ with $e = h_{i^*}$, it outputs $(i^*, \psi_{\mathsf{dlenc}})$. Otherwise, it outputs $(0, \perp)$. By the Generalized Forking Lemma (Lemma 3.1), denoting $\mathsf{D}_{\mathsf{IG}} : r \mapsto 1^\ell$, there exists an algorithm $\mathsf{F}_{\mathcal{B}}$ that, on input 1^ℓ, returns a triple $(b, \mathsf{so}, \mathsf{so}')$ with $b \in \{0,1\}$ such that

$$\mathsf{p}_{\mathsf{fork}} = \Pr[b = 1 \mid (b, \mathsf{so}, \mathsf{so}') \leftarrow \mathsf{F}_{\mathcal{B}}(1^\ell)] \geq \frac{\delta^2}{Q_{\mathsf{RO}}} - \frac{\delta}{2^{2\ell}},$$

and when $b = 1$, $(\mathsf{so}, \mathsf{so}') = (\psi_{\mathsf{dlenc}}, \psi'_{\mathsf{dlenc}})$ where $\psi_{\mathsf{dlenc}} = (A, \widetilde{B}, D, e, z_1, z_2, z_3)$ and $\psi'_{\mathsf{dlenc}} = (A, \widetilde{B}, D, e', z'_1, z'_2, z'_3)$. Then, by Claim 5.17, when $b = 1$, there exists an extractor E such that one of the following hold:

1. $(\alpha, \beta, \gamma, \delta) \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that $\delta \nmid \gcd(\alpha, \beta)$ and $u_1^\alpha u_2^\beta v^\gamma = \tilde{P}^\delta \bmod \hat{N}$.
2. $k \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that $k \notin \{1, N\}$, $k < 2^\ell$ and $\gcd(k, N) \neq 1$.
3. $(x'_2, r, \sigma_0, \Delta_e) \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that
   (a) $(x'_2, \lambda^*, \sigma_0) \in \pm 2^{n_x+\nu+1} \times \pm 2^{n_\lambda+\nu+1} \times \mathbb{Z}^*_{N^2}$,
   (b) $X = g^{x'_2} \in \mathbb{G}$ and $\mathcal{E} = (1+N)^{x'_2} \rho^{\lambda^*} \sigma_0^N \bmod N^2$,
   (c) $(\sigma_0^N)^{\Delta_e} = 1 \bmod N^2$ and $|\Delta_e| \leq 2^\ell$.

From Item 1, we directly get a contradiction to the strong RSA assumption. Therefore, with probability at least $\mathsf{p}_{\mathsf{fork}} - \mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^\ell)$, we must land in Item 2 or in Item 3.

Now, assume that we land in Item 2 with probability at least δ', and consider the following algorithm $\mathcal{C}$: it receives 1^ℓ and a Q_{RO}-tuple of uniformly random elements $(h_1, \cdots, h_{Q_{\mathsf{RO}}}) \in (\{0,1\}^{2\ell})^{Q_{\mathsf{RO}}}$. Then it runs $\mathsf{F}_{\mathcal{B}}(1^\ell)$ with random tape $(h_1, \cdots, h_{Q_{\mathsf{RO}}})$ until Item 2 occurs (for simplicity and readability, we locally "reset" the notations and let $\mathsf{so}, \mathsf{so}', i^*, e$ be temporarily overwritten by the second application of the forking lemma, before returning to the analysis of the first forking lemma and "reloading" the original notations). It retrieves the proof $\psi_{\mathsf{fac}} = (\widetilde{P}, \widetilde{Q}, \widetilde{A}, \widetilde{B}, \widetilde{C}, e, z_1, e_2, \lambda_1, \lambda_2, w)$ and returns $(i^*, \psi_{\mathsf{fac}})$, where i^* is such that $e = h_{i^*}$. By the Generalized Forking Lemma (Lemma 3.1), there exists an algorithm $\mathsf{F}_{\mathcal{C}}$ that, on input 1^ℓ, returns a triple $(b, \mathsf{so}, \mathsf{so}')$ with $b \in \{0,1\}$ such that

$$\mathsf{p}'_{\mathsf{fork}} = \Pr[b = 1 \mid (b, \mathsf{so}, \mathsf{so}') \leftarrow \mathsf{F}_{\mathcal{B}}(1^\ell)] \geq \frac{(\delta')^2}{Q_{\mathsf{RO}}} - \frac{\delta'}{2^{2\ell}},$$

and when $b = 1$, $(\mathsf{so}, \mathsf{so}') = (\psi_{\mathsf{fac}}, \psi'_{\mathsf{fac}})$ where $\psi_{\mathsf{fac}} = (\widetilde{P}, \widetilde{Q}, \widetilde{A}, \widetilde{B}, \widetilde{C}, e, z_1, e_2, \lambda_1, \lambda_2, w)$ and $\psi'_{\mathsf{fac}} = (\widetilde{P}, \widetilde{Q}, \widetilde{A}, \widetilde{B}, \widetilde{C}, e', z'_1, e'_2, \lambda'_1, \lambda'_2, w')$. Then, by Lemma 5.6, there exists an extractor E such that

1. $\alpha \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ and $g^\alpha = h \bmod d$.
2. $(u, v) \leftarrow \mathsf{E}(\mathsf{so}, \mathsf{so}')$ such that $N = uv$ and $|u|, |v| \leq 2^{n/2+\ell+\nu}$

The first item contradicts the discrete logarithm assumption over QR_d. Therefore, with probability at least $\mathsf{p}'_{\mathsf{fork}} - \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3})$, it must hold that $N = uv$ and $|u|, |v| \leq 2^{n/2+\ell+\nu}$. But then, by the soundness of Π_{mod} and via Fiat-Shamir, except with probability $Q_{\mathsf{RO}}/2^\ell$, it holds that[4] $N = p_1 \cdot p_2$ where $p_1, p_2 \in \mathsf{PRIMES}^{>2}$. it follows that N is a product of two primes of size at least $2^{n/2-\ell-\nu} \gg 2^\ell$, which contradicts Item 2. Hence, we must have

$$\mathsf{p}'_{\mathsf{fork}} - \frac{Q_{\mathsf{RO}}}{2^\ell} = \frac{(\delta')^2}{Q_{\mathsf{RO}}} - \frac{\delta'}{2^{2\ell}} - \frac{Q_{\mathsf{RO}}}{2^\ell} \leq \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3}),$$

which implies in particular that $\delta' \leq \sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3}) + 1/2^{2\ell} + Q_{\mathsf{RO}}/2^\ell}$.
It remains to analyze the case where we land in Item 3. Assume for now that we have $(x'_2, \lambda^*, \sigma_0)$ such that:

1. $(x'_2, \lambda^*, \sigma_0) \in \pm 2^{n_x+\nu+1} \times \pm 2^{n_\lambda+\nu+1} \times \mathbb{Z}^*_{N^2}$,
2. $X = g^{x'_2} \in \mathbb{G}$ and $\mathcal{E} = (1+N)^{x'_2} \rho^{\lambda^*} \sigma_0^N \bmod N^2$,
3. $(\sigma_0^N)^{\Delta_e} = 1 \bmod N^2$ and $|\Delta_e| \leq 2^\ell$.

We compare the distributions of $\mathcal{S}$ in Game_5 and $\mathcal{S}^*$ in Game_6 and show that they are statistically close. More precisely, we show:

Claim 6.10. *Conditioned on Item 3 happening, the distributions of $\mathcal{S}$ in Game_5 and $\mathcal{S}^*$ in Game_6 are $2^{-\nu+1}$-close to uniform.*

[4] We use $\ell/2$ repetitions of Π_{mod}, which has soundness error at most $\max\{1/4, 1/\gcd(N, \phi(N))\}$. However, Π_{mod} also guarantees that $N = 1 \bmod 4$ (with zero soundness error, as the verifier checks it directly) which combined with the fact that $N = uv$ and $|u|, |v| \leq 2^{n/2+\ell+\nu}$, implies that $\gcd(N, \phi(N)) > 4$, hence the soundness error for each repetition is at most $1/4$.

Proof. In Game_5, we have

$$\mathcal{S} = (1+N)^{u+x_2'v} \cdot \rho^{\lambda_0+\lambda^* v} \cdot (\sigma_0^N)^v,$$

where $u = [k_1^{-1}(m+rx_1)]_q + \mu q$, $v = [k_1^{-1}r]_q + \mu' q$ for λ_0, μ, μ' sampled uniformly from $(\pm 2^{n_{\lambda_0}}) \times (\pm\lfloor 2^{n_a}/q \rfloor) \times (\pm\lfloor 2^{n_b}/q \rfloor)$. In Game_6, we have

$$\mathcal{S}^* = (1+N)^{\sigma' + (\mu + zx_2')q} \cdot \rho^{\lambda_0 + \lambda^* zq} \cdot (\sigma_0^N)^{qz},$$

where $\sigma' = k_1^{-1}(m+rx) \bmod q$, $z \leftarrow \pm 2^{\varepsilon}$, and λ_0, μ are sampled as in Game_6. We have:

$$\begin{aligned}
u + x_2'v &= [k_1^{-1}(m+rx_1)]_q + \mu q + x_2'([k_1^{-1}r]_q + \mu' q) \bmod N \\
&= [k_1^{-1}(m+rx_1)]_q + x_2'[k_1^{-1}r]_q + q \cdot (\mu + x_2'\mu') \bmod N \\
&\approx_{2^{-\varepsilon}} [k_1^{-1}(m+rx_1) + x_2'k_1^{-1}r]_q + \mu \cdot q \bmod N \\
&= [k_1^{-1}(m+rx)]_q + \mu \cdot q = \sigma' + \mu q \bmod N \\
&\approx_{2^{-\varepsilon}} \sigma' + (\mu + zx_2')q \bmod N
\end{aligned}$$

where $\approx_{2^{-\varepsilon}}$ denotes $1/2^{\varepsilon}$-statistical closeness and follows from the fact that $x_2'\mu'$ is upper bounded by $2^{n_x+\nu+1+n_b-\kappa}$ and $x_2'x$ is upper bounded by $2^{\varepsilon+n_x+\nu+1}$, while μ was sampled from $\pm\lfloor 2^{n_a}/q \rfloor \supset 2^{n_x+\nu+n_b-\kappa+\varepsilon} \supset \pm 2^{\varepsilon+n_x+\nu+1}$ using our choice of parameters (Section 8.1). Then,

$$\lambda_0 + \lambda^* v \approx_{2^{-\varepsilon}} \lambda_0 + \lambda^* zq,$$

because λ_0 is sampled from $\pm 2^{n_{\lambda_0}}$ and n_{λ_0} is at least ε larger than $n_b + n_\lambda + \nu + 1$ (which bounds the bitlength of $\lambda^* v$) and than $\varepsilon + \kappa$ (which bounds the bitlength of zq) using again the parameters from Section 8.1. Eventually,

$$\begin{aligned}
v &= ([k_1^{-1}r]_q + \mu' q) \bmod \Delta_e \\
&= (\mu' + [k_1^{-1}r]_q \cdot q^{-}1) \cdot q \bmod \Delta_e
\end{aligned}$$

because $q \gg 2^{\ell} \geq |\Delta_e|$ is prime, hence coprime to Δ_e. Since μ' is sampled from $\pm\lfloor 2^{n_b}/q \rfloor \supset \pm 2^{\varepsilon}$, it holds that $(\sigma_0^N)^v \bmod N^2$ is $2^{\ell-\varepsilon} = 2^{-}\nu$-close to being uniform over the subgroup generated by σ_0^N (importantly, note that we have already shown that μ' was $2^{-\varepsilon}$-statistically masked by μ in the term $u + x_2'$). Similarly, since z us sampled from $\pm 2^{\varepsilon}$, $(\sigma_0)^{qz} \bmod N^2$ is $2^{-\nu}$-close to uniform over the subgroup generated by σ_0^N. We conclude the proof of the claim using the fact that $2 \cdot 2^{-\varepsilon} + 2^{-\nu} \leq 2^{-\nu+1}$. □

Combining everything, the probability that $\mathsf{F}_{\mathcal{B}}$ returns a triple $(b, \mathsf{so}, \mathsf{so}')$ with $b = 1$ can be bounded by

$$\mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^{\ell}) + \sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}} + \frac{1}{2^{2\ell}} + \frac{Q_{\mathsf{RO}}}{2^{\ell}}} + \frac{1}{2^{\nu-1}},$$

where $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}$ is a shorthand for $\mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}}(1^{n+2\ell+2\nu+3})$. Hence,

$$\begin{aligned}
\frac{\delta^2}{Q_{\mathsf{RO}}} - \frac{\delta}{2^{2\ell}} &\leq \mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^{\ell}) + \sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}} + \frac{1}{2^{2\ell}} + \frac{Q_{\mathsf{RO}}}{2^{\ell}}} + \frac{1}{2^{\nu-1}} \\
\implies \delta^2 &\leq \mathsf{Adv}^{\mathsf{srsa}}_{\mathsf{F}_{\mathcal{B}}}(1^{\ell}) + \sqrt{Q_{\mathsf{RO}} \cdot \mathsf{Adv}^{\mathsf{dlog}}_{\mathsf{F}_{\mathcal{C}}} + \frac{1}{2^{2\ell}} + \frac{Q_{\mathsf{RO}}}{2^{\ell}}} + \frac{1}{2^{\nu-1}} + \frac{1}{2^{2\ell}},
\end{aligned}$$

which concludes the proof. □

Game_7. In this game, Sim does not sample x anymore at the beginning of Protocol 2. Instead, Sim interacts with the signing oracle SO (Experiment 7.1):

- At the beginning of 2, Sim receives $X \in \boldsymbol{G}$ from SO.
- In step 1 of Protocol 3, Sim stores R_2 and extracts $k_2 = \mathsf{dlog}_g(R_2)$.
- In step 2 of Protocol 3, Sim sends $(m, 0)$ to SO and receives R_1 from SO. Sim sends R_1 to C.
- In step 3 of Protocol 3, Sim sends k_2 to SO, and receives σ. Sim sets $\sigma' \leftarrow (k_2 + \mathsf{RO}(X, R_1, R_1^{k_2}, m)) \cdot \sigma \bmod q$ and proceeds as in Game_6 to construct $\mathcal{S}^*$ from σ'.

During the above interactions with SO, Sim updates the lists $\boldsymbol{B}, \boldsymbol{L}$ as prescribed in Section 6.1. Since SO returns $\sigma = (k_1 \cdot (k_2 + \mathsf{RO}(X, R_1, R_1^{k_2}, m)) \cdot (m + rx) \bmod q$, it holds that $(k_2 + \mathsf{RO}(X, R_1, R_1^{k_2}, m)) \cdot \sigma = k_1^{-1} \cdot (m + rx) \bmod q$, hence $\mathcal{S}^*$ is distributed exactly as in Game_6 and we have

$$\Pr[\mathsf{Game}_7] = \Pr[\mathsf{Game}_6].$$

In Game_7, Sim only interacts with the signing oracle SO. Therefore, any forgery produced by $\mathcal{A}$ at the end of its interaction with Sim in Game_8 yields a contradiction to the doubly-enhanced unforgeability of ECDSA. This concludes the proof.

7 Unforgeability in the Generic Group Model

In this section, we present our security analysis for the doubly-enhanced security experiment of ECDSA. For generality, and as is common in certain applications, our analysis holds even when considering additive derivation by a public tweak. Specifically, for $X \in \mathbb{G}$, there is a deterministic algorithm D that outputs a set $\boldsymbol{\Delta}_X \leftarrow \mathsf{D}(X)$, and the attacker may query signatures for a related key $X' = X \cdot g^\delta$ for any $\delta \in \boldsymbol{\Delta}_X$. As in [GS22], this additional capability impacts our security bound by a factor of $1/|\boldsymbol{\Delta}_X|$ which is inconsequential for the plain case of $\boldsymbol{\Delta}_X = \{0\}$. It is assumed that random elements in $\boldsymbol{\Delta}_X$ can be efficiently sampled, and membership can be verified efficiently using a suitable witness.

Experiment 7.1 (Two-Round Security Experiment)**.** *Consider the following game between adversary* A *parameterized by* $j \in \mathrm{poly}$ *and a special ECDSA signing oracle* SO*:*

Common Input. $\kappa \in \mathbb{N}$*, EC-algorithm* $\mathsf{group}(\cdot)$*, random oracle* RO *and function family* $\boldsymbol{F}_\kappa$*.*

0. SO *return* $(\mathbb{G}, g, q) \xleftarrow{\$} \mathsf{group}(1^\kappa)$ *and hash function* $\mathsf{F} \xleftarrow{\$} \boldsymbol{F}_\kappa$*.*

1. SO *samples* $x \xleftarrow{\$} \mathbb{F}_q$ *and returns* $X = g^x \in \mathbb{G}$ *and deterministic* D *s.t.* $\boldsymbol{\Delta}_X \leftarrow \mathsf{D}(X) \subseteq \mathbb{F}_q$*.*

2. In $j(\kappa)$ *concurrent iterations, letting* i *denote the index of the iteration:*

(a) A *sends* $(m_i, \delta_i) \in \mathbb{F}_q \times \boldsymbol{\Delta}_X$ *to* SO*.*

(b) SO *samples* $k_i \xleftarrow{\$} \mathbb{F}_q$ *and sends* $R_i = g^{k_i}$ *to* A*.*

(c) A *sends* $\ell_i \in \mathbb{F}_q \setminus \{0\}$ *and* $\psi_i \in \{0,1\}^*$ *to* SO*.*

(d) SO *sets* $\mu_i = \mathsf{RO}(X \cdot g^{\delta_i}, R_i^{\ell_i}, \psi_i, m_i)$ *and* $\hat{r}_i = \mathcal{X}((R_i \cdot g^{\mu_i})^{\ell_i})$ *and returns*

$$s_i = \ell_i^{-1} \cdot (k_i + \mu_i)^{-1}(m_i + \hat{r}_i \cdot (x + \delta_i)) \bmod q.$$

3. A *returns* $(\mathrm{msg}, \delta, r, \sigma) \in \{0,1\}^* \times \mathbb{F}_q^2$.

Output. $b_{\mathsf{A}} = 1$ *if* $\begin{cases} \mathcal{X}((g^{\mathsf{F}(\mathrm{msg})}(X \cdot g^{\delta})^r)^{1/\sigma}) = r \\ (\mathsf{F}(\mathrm{msg}), \delta) \notin \{(m_i, \delta_i)\}_{i \in [j]} \end{cases}$, *and* 0 *otherwise.*

Experiment 7.2 (Doubly Enhanced Unforgeability Experiment)**.** *Consider the following game between adversary* A *parameterized by* $j \in \mathrm{poly}$ *and a special ECDSA signing oracle* SO*:*

Common Input. $\kappa \in \mathbb{N}$*, EC-algorithm* $\mathsf{group}(\cdot)$ *and function family* $\boldsymbol{F}_\kappa$.

0. SO *return* $(\mathbb{G}, g, q) \leftarrow \mathsf{group}(1^\kappa)$ *and hash function* $\mathsf{F} \xleftarrow{\$} \boldsymbol{F}_\kappa$.

1. SO *samples* $x \xleftarrow{\$} \mathbb{F}_q$ *and returns* $X = g^x \in \mathbb{G}$ *and deterministic* D *s.t.* $\boldsymbol{\Delta}_X \leftarrow \mathsf{D}(X) \subseteq \mathbb{F}_q$.

2. In $j(\kappa)$ *concurrent iterations, letting* i *denote the index of the iteration:*

(a) SO *samples* $k_i \xleftarrow{\$} \mathbb{F}_q$ *and sends* $R_i = g^{k_i}$ *to* A.

(b) A *adaptively sends distinct* $\{(m_{i,\ell}, z_{i,\ell}, \delta_{i,\ell}) \in \mathbb{F}_q \times \boldsymbol{\Delta}_X\}_{\ell=1}^{j(\kappa)}$ *to* SO.

SO *returns* $\mu_{i,\ell} \xleftarrow{\$} \mathbb{F}_q$ *for each tuple* $(m_{i,\ell}, z_{i,\ell}, \delta_{i,\ell})$.

(c) A *sends* $\ell_0 \in \mathbb{N}$.

(d) SO *sets* $(m_i, z_i, \mu_i, \delta_i) = (m_{i,\ell_0}, z_{i,\ell_0}, \mu_{i,\ell_0}, \delta_{i,\ell_0})$ *and* $\hat{r}_i = \mathcal{X}(R_i^{z_i} g^{\mu_i})$ *and returns*

$$s_i = (z_i k_i + \mu_i)^{-1}(m_i + \hat{r}_i \cdot (x + \delta_i)) \bmod q.$$

3. A *returns* $(\mathrm{msg}, \delta, r, \sigma) \in \{0,1\}^* \times \boldsymbol{\Delta}_X \times \mathbb{F}_q^2$.

Output. $b_{\mathsf{A}} = 1$ *if* $\begin{cases} \mathcal{X}((g^{\mathsf{F}(\mathrm{msg})}(X \cdot g^{\delta})^r)^{1/\sigma}) = r \\ (\mathsf{F}(\mathrm{msg}), \delta) \notin \{(m_i, \delta_i)\}_{i \in [j]} \end{cases}$, *and* 0 *otherwise.*

Definition 7.3. Using the notation above, we say that A breaks the unforgeability of Experiment 7.1 (resp. Experiment 7.2) if there exists a polynomial λ such that $\Pr[b_{\mathsf{A}} = 1] \geq 1/\lambda$ for infinitely many $\kappa \in \mathbb{N}$. Otherwise, if no such A exists, we say that Experiment 7.1 (resp. Experiment 7.2) is unforgeable.

Theorem 7.4. *If Experiment 7.2 is unforgeable then Experiment 7.1 is unforgeable in the ROM.*

Proof. Immediate. □

7.1 The Generic Group Model

Let $\boldsymbol{G}$ denote the labels of the elements in the elliptic curve $(\mathbb{G}, g, q)$. As $\mathbb{G}$ denotes an elliptic curve, every $(A, A^{-1}) \in \mathbb{G}^2$ has the form $((\alpha, 0), (\alpha, 1))$ or $((\alpha, 1), (\alpha, 0)) \in \boldsymbol{G}^2$. For every label $(\alpha, \beta) \in \mathbb{F}_q \times \{0,1\}$, it is assumed that there is an efficient algorithm for checking that $(\alpha, \beta) \in \boldsymbol{G}$, using the description of the elliptic curve group (i.e., the defining equation). Recall the projection function $\mathcal{X} : \boldsymbol{G} \to \mathbb{F}_q$ such that $\mathcal{X}(H) = \eta \in \mathbb{F}_q$ for $H \in \boldsymbol{G}$ of the form $(\eta, 0)$ or $(\eta, 1)$. Notice that this map is efficiently invertible $\mathcal{X}^{-1} : \mathbb{F}_q \to \{\{G, H\} \text{ s.t. } G, H \in \boldsymbol{G}\} \cup \{\perp\}$ such that

$$\mathcal{X}^{-1} : x \mapsto \begin{cases} \{H, H^{-1}\} & \text{if } \exists H \text{ s.t. } \mathcal{X}(H) = x \\ \perp & \text{otherwise} \end{cases}.$$

Brief overview of the EC Generic Group Model. The generic group is defined via a random bijective map $\pi : \mathbb{G} \to \boldsymbol{G}$ that preserves $\mathcal{X}$ and a group-oracle $\mathsf{O}_{\mathbb{G}} : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $\pi(gh) = \mathsf{O}_{\mathbb{G}}(\pi(g), \pi(h))$ and $\mathcal{X}(\pi(g)) = \mathcal{X}(\pi(g^{-1}))$, for every $g, h \in \mathbb{G}$. In group-theoretic jargon, $(\boldsymbol{G}, *)$ is isomorphic to $(\mathbb{G}, \cdot)$ via the group-isomorphism π, letting $* : \boldsymbol{G} \times \boldsymbol{G} \to \boldsymbol{G}$ such that $G * H = \mathsf{O}_{\mathbb{G}}(G, H)$.

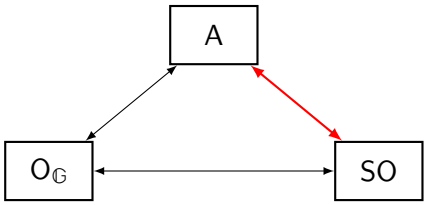

Figure 1: Pictorial representation of the different algorithms/oracles in the GGM. The red arrow represents Experiment 7.2. Note that both A and SO query $\mathsf{O}_{\mathbb{G}}$ for the group operations.

7.1.1 Symbolic lazy evaluation

1. Initialization.
   (a) $\pi \leftarrow \{(0, \mathbb{1})\}$
   (b) Sample $g \xleftarrow{\$} \boldsymbol{G}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.
   (c) Sample $X \xleftarrow{\$} \boldsymbol{G}$ and $\pi \leftarrow \pi \cup \{(\mathfrak{R}_1, X), (-\mathfrak{R}_1, X^{-1})\}$. (If $X \in \text{range}(\pi)$, abort)
   (d) Set $\boldsymbol{V} \leftarrow \{\mathfrak{R}_1\}$ and $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set) and $c = 1$.
   (e) For all $P \in \boldsymbol{G}$ such that $\mathcal{X}(P) = 0$, execute the group addition query on input $(P, \mathbb{1})$.
   (f) Return (g, X)
2. Group query. On input $i = i_0 + \sum_{\mathfrak{R} \in \boldsymbol{V}} i_{\mathfrak{R}} \cdot \mathfrak{R}$, do:
   (a) If $i \notin \text{domain}(\pi)$
      i. Sample $P \xleftarrow{\$} \boldsymbol{G}$. (If $P \in \text{range}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P), (-i, P^{-1})\}$.
   (b) Return $\pi(i)$
3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:
   (a) For $j \in \{1, 2\}$, if $P_j \notin \text{range}(\pi)$:
      i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$. (If $i \in \text{domain}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.
   (b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.
4. Presignatures. Set $c \leftarrow c + 1$, and do:
   (a) Sample $R \xleftarrow{\$} \boldsymbol{G}$. (If $R \in \text{range}(\pi)$, abort).
   (b) Set $\pi \leftarrow \pi \cup \{(\mathfrak{R}_c, R), (-\mathfrak{R}_c, -R)\}$, $\boldsymbol{V} \leftarrow \boldsymbol{V} \cup \{\mathfrak{R}_c\}$ and $R \leftarrow \boldsymbol{R} \cup \{R\}$.

(c) Return R.

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^2$, do:
   (a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$
      i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q^*$. (If $\mu + z\pi^{-1}(R) \in \text{domain}(\pi)$), abort)
      ii. Invoke the group query on input $\mu + z\pi^{-1}(R)$.
      iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$.
   (b) Return $\lambda(R, \delta, z, m)$
6. Signature. Letting $\mathfrak{R} \in \boldsymbol{V}$ such that $\pi(\mathfrak{R}) = R$, on input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:
   (a) Let $\mu = \lambda(R, \delta, z, m)$. Obtain $\hat{R}$ by invoking the group query on input $\mu + z \cdot \mathfrak{R}$.
   (b) Set $\hat{r} = \mathcal{X}(\hat{R})$ and sample $s \xleftarrow{\$} \mathbb{F}_q^*$.
   (c) Substitute $s^{-1}(m + \hat{r} \cdot (\mathfrak{R}_1 + \delta))$ for $\pi^{-1}(\hat{R})$ throughout $\text{domain}(\pi)$.
      Abort if any two polynomials collapse, i.e., there's a collision.
   (d) Remove all records of R and $\mathfrak{R}$ from $\boldsymbol{R}$, $\boldsymbol{V}$ and λ.
   (e) Return $(\hat{r}, s)$

Theorem 7.5. *Assume that* A *breaks the unforgeability of Experiment 7.2 in the GGM with probability* ε*, then,* A *breaks the unforgeability in the symbolic lazy evaluation experiment with probability* $\varepsilon - \text{err}$ *where* $\text{err} \approx 5t^2/q$ *and the exact error term is specified in Theorem B.1.*

Proof. Corollary of Theorem B.1. □

7.2 Doubly-Enhanced Unforgeability in the GGM

Parameter choice (size of q). Hereafter, we assume that q is sufficiently large such that $1 - \left(\frac{3}{4} + \frac{q^{-1/2}}{\sqrt{2}}\right) \geq 1/5$ and $(1 - 1/q) \geq 1/5$. A straightforward calculation shows that this condition is satisfied for any $q \geq 2^8$.

Theorem 7.6. *Let* t *denote an upper bound on the number of group and signing-oracle queries of* A *and assume that* A *breaks the unforgeability of Experiment 7.2 in the GGM with probability* ε*. Then, there exists algorithm* B *with black-box access to* A *such that*

$$\Pr_{\substack{e \xleftarrow{\$} \mathbb{F}_q \\ x \xleftarrow{\$} \mathsf{B}(e)}} [e = \mathsf{F}(x)] \geq (\varepsilon - \text{err}) \cdot \frac{1 - 2t/q}{25|\boldsymbol{\Delta}_X|t} - 13t^2/(q-1) + t^2/q + t^4/q^2.$$

Proof. The theorem above is a corollary of Theorem 7.5 and Propositions 7.10 and 7.11 by considering an algorithm B that runs one of B_1, B_2, B_3 from Proposition 7.11 with probability 1/5, 3/5 and 1/5 respectively. □

7.2.1 Preliminary Facts

Definition 7.7. For $\mathbb{D} = \{(\alpha, \beta, \gamma, \zeta, m, r, e) \in \mathbb{F}_q^7 \text{ s.t. } \alpha\zeta \neq e\beta\}$, define $\mathsf{ssig} : \mathbb{D} \to \mathbb{F}_q$ such that

$$\mathsf{ssig} : (\alpha, \beta, \gamma, \zeta, m, r, e) \mapsto \gamma \cdot \frac{er - \zeta m}{\alpha\zeta - e\beta}$$

Claim 7.8. *Fix $\alpha, \beta, \gamma, \zeta, m, r \in \mathbb{F}_q$ such that $\gamma, \zeta \neq 0$. Define function $\mathsf{ssig}' : e \mapsto \mathsf{ssig}(\alpha, \beta, \gamma, \zeta, m, r, e) \in \mathbb{F}_q$ where e takes values $\mathbb{D}' = \{e \in \mathbb{F}_q \text{ s.t. } (\alpha, \beta, \gamma, \zeta, m, r, e) \in \mathbb{D}\}$. If $r\alpha - \beta m \neq 0$, then ssig' is* injective.

Proof.

$$\begin{aligned}
\mathsf{ssig}'(e_1) = \mathsf{ssig}'(e_2) &\quad \Leftrightarrow \\
\frac{e_1 r - \zeta m}{\alpha\zeta - e_1\beta} = \frac{e_2 r - \zeta m}{\alpha\zeta - e_2\beta} &\quad \Leftrightarrow \\
e_1 r\alpha\zeta - e_1 r e_2\beta - \zeta^2 m\alpha + \zeta m e_2\beta = e_2 r\alpha\zeta - e_2 r e_1\beta - \zeta^2 m\alpha + \zeta m e_1\beta &\quad \Leftrightarrow \\
e_1 r\alpha\zeta + \zeta m e_2\beta = e_2 r\alpha\zeta + \zeta m e_1\beta &\quad \Leftrightarrow \\
\zeta(e_1 - e_2)(r\alpha - \beta m) = 0 &\quad \Leftrightarrow \\
e_1 = e_2 &
\end{aligned}$$

□

Some facts about the ECDSA conversion function. We will be using the following lemma from [GS22].

Lemma 7.9. *For the identifying labels $\boldsymbol{G}$ of an eliptic curve encoded by the tuple $(\mathbb{G}, g, q)$, there exists an efficient randomized algorithm Samp such that for $A \xleftarrow{\$} \mathsf{Samp}(x)$ where x is drawn uniformly from $\mathbb{F}_q^*$:*

1. $\Pr[A \in \{\mathsf{fail}\} \cup \boldsymbol{G}] = 1$ *and* $\Pr[A = \mathsf{fail}] \leq \frac{3}{4} + \frac{q^{-1/2}}{\sqrt{2}}$
2. *If $A \in \boldsymbol{G}$, then $\mathcal{X}(A) = x$.*
3. *For any $P \in \boldsymbol{G} \setminus \{\mathbb{1}\}$, $\Pr[A = P \mid A \neq \mathsf{fail}] = 1/(q-1)$.*

7.2.2 Simulation

We define three algorithms, $\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3$, with black-box access to A, which simulate both SO and $\mathsf{O}_{\mathbb{G}}$ as described below. On input $e \xleftarrow{\$} \mathbb{F}_q$, algorithm B_i samples $j_0 \xleftarrow{\$} [t]$ and proceeds as follows: B_i initializes $\mathsf{ctr} = 0$, incrementing ctr after each operation. (When the description states 'B_i **only**:', it indicates that only B_i performs this step, while $\{\mathsf{B}_j\}_{j \neq i}$ ignore the operation in question.)

1. Initialization.
   (a) $\pi \leftarrow \{(0, \mathbb{1})\}$
   (b) Sample $g \xleftarrow{\$} \boldsymbol{G}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.

(c) Sample $X \xleftarrow{\$} \boldsymbol{G}$ and $\pi \leftarrow \pi \cup \{(\mathfrak{R}_1, X), (-\mathfrak{R}_1, X^{-1})\}$. (If $X \in \text{range}(\pi)$, abort)

(d) Pick $\delta_0 \leftarrow \boldsymbol{\Delta}_X$, and invoke the group query on input $\delta_0 + \mathfrak{R}_1$. Let $X_0 = \pi(\delta + \mathfrak{R}_1)$.

(e) For all $P \in \boldsymbol{G}$ such that $\mathcal{X}(P) = 0$, execute the group addition query on input $(P, \mathbb{1})$.

(f) Set $\boldsymbol{V} \leftarrow \{\mathfrak{R}_1\}$, $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set) and $c = 1$.

(g) Return (g, X)

2. Group query. On input $i = \alpha + \beta(\delta_0 + \mathfrak{R}_1) + \sum_{\mathfrak{R} \in \boldsymbol{V} \setminus \{\mathfrak{R}_1\}} i_{\mathfrak{R}} \cdot \mathfrak{R}$, do:

(a) If $i \notin \text{domain}(\pi)$

i. If $\mathsf{ctr} = j_0$:

A. B_1 **only**: Choose $Z \in \boldsymbol{G}$ such that $\mathcal{X}(Z) = \alpha^{-1} e \beta$ using algorithm Samp from Lemma 7.9 on input $\alpha^{-1} e \beta$. If Z is not well defined or $Z \in \text{range}(\pi)$, go to Item 2(a)ii.

B. B_2 **only**: If $i = \alpha' + \beta'(\delta_0 + \mathfrak{R}_1) + \gamma'(\mu + z\mathfrak{R})$ where for $((R, \delta, z, m), \mu) \in \lambda$ with $R = \pi(\mathfrak{R})$, $\hat{r} = \mathcal{X}(\pi(\mu + z\mathfrak{R}))$ and $\hat{m} = m + \hat{r}(\delta - \delta_0)$, it holds that $\alpha' \hat{r} + \beta' \hat{m} = 0$, then set Z subject to $\mathcal{X}(Z) = \hat{m}^{-1} e \hat{r}$, using algorithm Samp from Lemma 7.9 on input $\hat{m}^{-1} e \hat{r}$ (if there are many pairs $(\hat{m}, \hat{r})$ satisfying the aforementioned conditions, simply pick one at random). If Z is not well defined or $Z \in \text{range}(\pi)$, go to Item 2(a)ii.

C. B_3 **only**: Let $\boldsymbol{S} \leftarrow \{\mathfrak{R} \text{ s.t. } i_{\mathfrak{R}} \neq 0\} \setminus \{\mathfrak{R}_1\}$, sample $Z \xleftarrow{\$} \boldsymbol{G}$ and set $Z_0 \leftarrow Z$ (If $Z \in \text{range}(\pi)$, abort).

ii. Else, choose $Z \xleftarrow{\$} \boldsymbol{G}$ (If $Z \in \text{range}(\pi)$, abort) .

Set $\pi \leftarrow \pi \cup \{(i, Z), (-i, Z^{-1})\}$

(b) Return $\pi(i)$

3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:

(a) For $j \in \{1, 2\}$, if $P_j \notin \text{range}(\pi)$:

i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$. (If $i \in \text{domain}(\pi)$, abort)

ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.

(b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.

4. Presignatures. Set $c \leftarrow c + 1$, and do:

(a) Sample $R \xleftarrow{\$} \boldsymbol{G}$. (If $R \in \text{range}(\pi)$, abort).

(b) Set $\pi \leftarrow \pi \cup \{(\mathfrak{R}_c, R), (-\mathfrak{R}_c, R^{-1})\}$ and $R \leftarrow \boldsymbol{R} \cup \{R\}$.

(c) Return R.

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^2$, do:

(a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$

i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q^*$. (If $\mu + z\pi^{-1}(R) \in \text{domain}(\pi)$, abort)

ii. Invoke the group query on input $\mu + z\pi^{-1}(R)$

iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$.

(b) Return $\lambda(R, \delta, z, m)$

6. Signature. Letting $\mathfrak{R}$ such that $\pi(\mathfrak{R}) = R$, on input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:

(a) Let $\mu = \lambda(R, \delta, z, m)$. Obtain $\hat{R}$ by invoking the group query on input $\mu + z \cdot \mathfrak{R}$.

(b) Set $\hat{r} = \mathcal{X}(\hat{R})$ and $\hat{m} = m + \hat{r}(\delta - \delta_0)$.

(c) i. B_3 **only**: If $\boldsymbol{S} = \{\mathfrak{R}\}$, $Z_0 = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$, $\hat{r}\alpha - \beta\hat{m} \neq 0$, set $s = \mathsf{ssig}(\alpha, \beta, \gamma, \mathcal{X}(Z), \hat{m}, \hat{r}, e)$ If Z_0 does not fulfill these conditions or $s = 0$, set $s \xleftarrow{\$} \mathbb{F}_q^*$.

ii. Else, sample $s \xleftarrow{\$} \mathbb{F}_q^*$.

(d) Substitute $s^{-1}(m + \hat{r} \cdot (\mathfrak{R}_1 + \delta))$ for $\mu + z\mathfrak{R}$ throughout $\text{domain}(\pi)$.

Abort if any two polynomials collapse, i.e., there's a collision.

(e) Remove all records of R and $\mathfrak{R}$ from $\boldsymbol{R}$, $\boldsymbol{V}$, $\boldsymbol{S}$ and λ.

(f) Return $(\hat{r}, s)$

7.2.3 Analysis

In this section we analyze the probability that the simulation from Section 7.2.2 results in a 'good' outcome, i.e. for $\mathcal{A}$'s output $x \in \{0,1\}^*$, $e = \mathsf{F}(x)$. At the end of the experiment, it is assumed that A outputs a forgery $(f, \sigma) \in \mathbb{F}_q^2$ for message $x \in \{0,1\}^*$ and public key $X_0 = g^{\delta_0} \cdot X$. Let $F \in \boldsymbol{G}$ such that $\mathcal{X}(F) = f$ and observe that, by assumption, $F = g^{\mathsf{F}(x)/\sigma} \cdot X_0^{f/\sigma}$. We begin by observing that the following outcomes are 'good' for the reduction.

1. There exist $\alpha, \beta \in \mathbb{F}_q$ such that $F = g^{\alpha} \cdot X_0^{\beta}$ and $\mathcal{X}(F) = \alpha^{-1} e \beta$.

2. There exists $\alpha, \beta \in \mathbb{F}_q$, $\gamma, s \in \mathbb{F}_q^*$, and $((R, \delta, z, m), \mu) \in \lambda$ such that, letting $\hat{R} = g^{\mu} R^z$, $\hat{r} = \mathcal{X}(\hat{R})$ and $\hat{m} = m + \hat{r}(\delta - \delta_0)$,

(a) $\hat{R} = g^{\hat{m}/s} X_0^{\hat{r}/s}$, $F = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$, $\alpha\hat{r} - \hat{m}\beta = 0$ and $\hat{m} f = e\hat{r}$.

(b) $\hat{R} = g^{\hat{m}/s} X_0^{\hat{r}/s}$, $F = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$, $\alpha\hat{r} - \hat{m}\beta \neq 0$ and $s = \mathsf{ssig}(\alpha, \beta, \gamma, f, \hat{m}, \hat{r}, e)$.

For Item 1, deduce that

$$F = g^{\alpha} \cdot X_0^{\beta} = g^{\mathsf{F}(x)/\sigma} \cdot X_0^{f/\sigma} \Rightarrow$$
$$\begin{cases} \alpha = \mathsf{F}(x)/\sigma \\ \beta = f/\sigma \end{cases} \Rightarrow \begin{cases} \alpha = \mathsf{F}(x)/\sigma \\ \sigma = \alpha^{-1} e \end{cases} \Rightarrow \quad \mathsf{F}(x) = e \tag{2}$$

For Item 2a,

$$\begin{cases} F = g^{\alpha} \cdot X_0^{\beta} \cdot \hat{R}^{\gamma} = g^{\mathsf{F}(x)/\sigma} \cdot X_0^{f/\sigma} \\ \hat{R} = g^{\hat{m}/s} X_0^{\hat{r}/s} \end{cases} \Rightarrow \begin{cases} \hat{m}/s = (\mathsf{F}(x)/\sigma - \alpha)/\gamma \\ \hat{r}/s = (f/\sigma - \beta)/\gamma \end{cases} \Rightarrow$$
$$\hat{m} \cdot (f/\sigma - \beta)/\gamma = \hat{r} \cdot (\mathsf{F}(x)/\sigma - \alpha)/\gamma \Rightarrow \tag{3}$$
$$\hat{m} \cdot f/\sigma = \hat{r} \cdot \mathsf{F}(x)/\sigma \quad \Rightarrow \quad \mathsf{F}(x) = \hat{r}^{-1} f \hat{m} \quad \Rightarrow \quad \mathsf{F}(x) = e \tag{4}$$

For Item 2b,

$$\begin{cases} F = g^{\alpha} \cdot X_0^{\beta} \cdot \hat{R}^{\gamma} = g^{\mathsf{F}(x)/\sigma} \cdot X_0^{f/\sigma} \\ \hat{R} = g^{\hat{m}/s} X_0^{\hat{r}/s} \end{cases} \Rightarrow \begin{cases} \alpha + \hat{m}\gamma/s = \mathsf{F}(x)/\sigma \\ \beta + \hat{r}\gamma/s = f/\sigma \end{cases} \Rightarrow$$

$$f \cdot (\alpha + \hat{m}\gamma/s) = \mathsf{F}(x) \cdot (\beta + \hat{r}\gamma/s) \Rightarrow \tag{5}$$

$$s = \gamma \cdot \frac{\mathsf{F}(x)\hat{r} - \hat{m}f}{\alpha f - \mathsf{F}(x)\beta} \quad \Rightarrow \quad s = \mathsf{ssig}(\alpha, \beta, \gamma, f, \hat{m}, \hat{r}, \mathsf{F}(x)) \tag{6}$$

Note that Equation (5) and $\alpha f - \mathsf{F}(x)\beta = 0$ implies $\mathsf{F}(x)\hat{r} - \hat{m}f = 0$ implies $\alpha\hat{r} - \hat{m}\beta = 0$, which we have ruled out by assumption. Thus, s is well defined in Equation (6). To conclude, since $s = \mathsf{ssig}(\alpha, \beta, f, \gamma, \hat{m}, \hat{r}, \mathsf{F}(x))$ and $\mathsf{ssig}' : e \mapsto \mathsf{ssig}(\alpha, \beta, \gamma, f, \hat{m}, \hat{r}, e)$ is injective in this regime of parameters, it follows that $\mathsf{F}(x) = e$.

Evaluating the probability of each event. In the sequel, for $((R, \delta_i, z_i, m_i), \mu_i) \in \lambda$, we write $\hat{R}_i = g^{\mu_i} R^{z_i}$, $\hat{r}_i = \mathcal{X}(\hat{R})$ and $\hat{m}_i = m_i + \hat{r}_i(\delta_i - \delta_0)$.

Proposition 7.10. *Let E denote the following event. For all $A \in \mathrm{range}(\pi)$, letting R denote the last presignature in the expression of A queried to the signing oracle,*

1. *If $A = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$ and $\alpha\hat{r} - \beta\hat{m} = 0$ for $((R, \delta, z, m), \mu) \in \lambda$ then $\hat{R}$ was obtained from the group oracle before A was, and A was first queried as a weighted combination of g, X and R, i.e., no other presignatures appear in the expression of A.*
2. $\left|\{((R, \delta_i, z_i, m_i), \mu_i) \in \lambda \;\; s.t. \;\; A = g^{\alpha_i} X_0^{\beta_i} \hat{R}_i^{\gamma_i} \wedge \alpha_i\hat{r}_i - \beta_i\hat{m}_i = 0\}\right| \leq 3$

Then, $\Pr[\neg E] \leq 13t^2/(q-1) + t^2/q + t^4/q^2$.

Proof. The above is shown in Section 7.2.4. □

Let $\hat{R}$ denote the last nonce that was queried to the signing oracle such that F was recorded as $F = g^{\alpha} X^{\beta} \hat{R}^{\gamma}$. Let $d = |\mathbf{\Delta}_X|$.

Proposition 7.11. *Using the notation above, under the assumption that event E from Proposition 7.10 occurred, it holds that:*

1. *If $\gamma = 0$, then Item 1 occurs for B_1 with prob. $(1 - 2t/q)/5dt$.*
2. *Else if $\alpha\hat{r} - \hat{m}\beta = 0$, then Item 2a occurs for B_2 with prob. $(1 - 2t/q)/15dt$.*
3. *Else, Item 2b occurs for B_3 with prob. $(1 - 2t/q)/5dt$.*

Proof. In each of the above cases, we argue that the tuple $(\hat{R}, F, \alpha, \beta, \gamma, s)$ satisfies the desired expression. Recall that for algorithm Samp, Lemma 7.9 tells that $\Pr[A = \mathsf{fail} \text{ s.t. } A \xleftarrow{\$} \mathsf{Samp}(x) \text{ and } x \xleftarrow{\$} \mathbb{F}_q^*] \leq 4/5$ (**Case 1.**) Suppose that F was queried at the j-th group or signing query. If $\gamma = 0$, then F was queried as a linear combination of g and X_0 and it suffices that the following conditions are met:

- A returns a forgery for X_0 and $j_0 = j$—probability $1/dt$.
- $\mathcal{X}(F) = \alpha^{-1}e\beta$ is well defined and $F \notin \mathrm{range}(\pi)$—probability $(1 - 2t/q)/5$.

- B_1 chose Item 2(a)iA when assigning F—probability 1.

(**Case 2.**) Suppose that F was queried at the j-th group or signing query. If F was queried after $\hat{R}$ then it suffices that the conditions below are met:

- A returns a forgery for X_0 and $j_0 = j$—probability $1/dt$.
- $\mathcal{X}(F) = \hat{m}^{-1}e\hat{r}$ is well defined and $F \notin \text{range}(\pi)$—probability $(1 - 2t/q)/5$.
- B_2 chose $(\hat{m}, \hat{r})$ in Item 2(a)iB when assigning F—probability $1/3$.

(**Case 3.**) Suppose that F was queried at the j-the group or signing query. It suffices that the conditions below are met:

- A returns a forgery for X_0 and $j_0 = j$—probability $1/dt$.
- $s = \mathsf{ssig}(\alpha, \beta, \gamma, \mathcal{X}(Z), \hat{m}, \hat{r}, e) \neq 0$ is well defined—probability $(1 - 2/q) \geq (1 - 2t/q)/5$.

□

7.2.4 Proof of Proposition 7.10

Proposition 7.12 (Restatement of Proposition 7.10)**.** *Let E denote the following event. For all $A \in \text{range}(\pi)$, letting R denote the last presignature in the expression of A queried to the signing oracle,*

1. *If $A = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$ and $\alpha\hat{r} - \beta\hat{m} = 0$ for $((R, \delta, z, m), \mu) \in \lambda$ then $\hat{R}$ was obtained from the group oracle before A was, and A was first queried as a weighted combination of g, X and R, i.e., no other presignatures appear in the expression of A.*
2. $\left|\{((R, \delta_i, z_i, m_i), \mu_i) \in \lambda \;\; s.t. \;\; A = g^{\alpha_i} X_0^{\beta_i} \hat{R}_i^{\gamma_i} \wedge \alpha_i\hat{r}_i - \beta_i\hat{m}_i = 0\}\right| \leq 3$

Then, $\Pr[\neg E] \leq 13t^2/(q-1) + t^2/q + t^4/q^2$.

We define the following events. For $A \in \text{range}(\pi)$, let $\hat{R} \in \boldsymbol{G}$ denote the last modified presignature in the expression of A was queried to the signing oracle, and let $s_0 \in \mathbb{F}_q^*$ denote the signature returned by the signing oracle associated with the penultimate presignature in the expression of A. Let U_1 denote the event that there exists $A \in \text{range}(\pi)$ such that:

1. A was first queried as an expression of at least two distinct presignatures.
2. $A = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$ and $\alpha\hat{r} - \beta\hat{m} = 0$ when s_0 was obtained.
3. $\hat{R}$ was obtained from the oracle before s_0 was obtained.

Let U_2 denote the event that there exists $A \in \text{range}(\pi)$ such that:

1. $A = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$ and $\alpha\hat{r} - \beta\hat{m} = 0$ when s_0 was obtained.
2. $\hat{R}$ was obtained from the oracle after A and s_0.

Let U_3 denote the event that there exists $A \in \text{range}(\pi)$ such that:

$$\left|\{((R, \delta_i, z_i, m_i), \mu_i) \in \lambda \;\text{ s.t. }\; A = g^{\alpha_i} X_0^{\beta_i} \hat{R}_i^{\gamma_i} \wedge \alpha_i\hat{r}_i - \beta_i\hat{m}_i = 0\}\right| \geq 4$$

Outline of the proof. The proof of Proposition 7.10 is broken down as follows. First, we show that $\Pr[U_1] \leq 9t^2/(q-1)$. Second, we show that $\Pr[U_2] \leq 4t^2/(q-1) + t^2/q$. Finally, we show that $\Pr[U_3 \mid \neg U_1 \wedge \neg U_2] \leq t^4/q^2$. The desired bound on $\Pr[\neg E]$ follows via the claim below.

Claim 7.13. $\Pr[\neg E] \leq \Pr[U_1] + \Pr[U_2] + \Pr[U_3 \mid \neg U_1 \wedge \neg U_2]$

Proof. Assume that for some $A \in \text{range}(\pi)$ we have $A = g^{\alpha} X_0^{\beta} \hat{R}^{\gamma}$ and $\alpha\hat{r} - \beta\hat{m} = 0$. Depending on the order that $A, \hat{R}$ and s_0 were obtained, the table below identifies if U_1 or U_2 occurred. Notice that for the orderings identified with '†', A was first queried as a weighted combination of g, X and R, i.e., no other presignatures appear in the expression of A, and $\hat{R}$ was obtained from the oracle before A was, as per event E.

Order	$(A, s_0, \hat{R})$	$(A, \hat{R}, s_0)$	$(\hat{R}, s_0, A)$	$(\hat{R}, A, s_0)$	$(s_0, \hat{R}, A)$	$(s_0, A, \hat{R})$
Event	U_2	U_1	†	U_1	†	U_2

To conclude, if U_1 and U_2 do not occur, then E holds true as long as U_3 does not occur. □

In the sequel, we will also be using the fact below (corollary of Hasse's theorem from [GS22]).

Fact 7.14. *Let $(\mathbb{G}, g, q)$ be the group-generator-order tuple associated with a prime-order elliptic curve, and let $a \in \mathbb{F}_q$. Then,* $\Pr_{k \xleftarrow{\$} \mathbb{F}_q^*}\left[\mathcal{X}(g^k) = a\right] \leq \frac{4}{q-1}$.

Proving that $\Pr[U_1] \leq 9t^2/(q-1)$. By Claim 7.15 the probability that there exist $\hat{m}_1, \hat{r}_1, \hat{m}_1, \hat{r}_2$ such that $\hat{m}_1\hat{r}_2 = \hat{r}_1\hat{m}_2$ is at most $8t^2/(q-1)$. Therefore, by Claim 7.16, the probability that U_1 occurs is at most $9t^2/(q-1)$, since there at most t^2 possible pairs $(A, \hat{R})$.

Claim 7.15. *For any $\{((R_i, \delta_i, z_i, m_i), \mu_i) \in \lambda\}_{i=1}^2$, it holds that* $\Pr_{\hat{r}_1, \hat{r}_2}[\hat{m}_1\hat{r}_2 = \hat{r}_1\hat{m}_2] \leq \frac{8}{q-1}$.

Proof. If $\delta_1 = \delta_2$, then $\hat{m}_1\hat{r}_2 = \hat{m}_2\hat{r}_1$ is equivalent to $m_1\hat{r}_2 = m_2\hat{r}_1$ and thus equality happens with probability at most $1/q$ because $m_1, m_2 \neq 0$. Otherwise, fix an arbitrary $\hat{r}_1$ such that $\hat{m}_1 - \hat{r}_1(\delta_2 - \delta_0) \neq 0$ (there is at least one since $\delta_1 \neq \delta_2$). Notice that

$$\begin{aligned} \hat{m}_1\hat{r}_2 = \hat{r}_1\hat{m}_2 \Leftrightarrow \\ \hat{m}_1\hat{r}_2 = \hat{r}_1(m_2 + \hat{r}_2(\delta_2 - \delta_0)) \Leftrightarrow \\ \hat{r}_2(\hat{m}_1 - \hat{r}_1(\delta_2 - \delta_0)) = \hat{r}_1 m_2 \end{aligned}$$

In this case, $\Pr_{\hat{r}_2}[\hat{m}_1\hat{r}_2 = \hat{r}_1\hat{m}_2] \leq \frac{4}{q-1}$ by Fact 7.14. Furthermore, using a similar argument, $\Pr_{\hat{r}_1}[\hat{m}_1 - \hat{r}_1(\delta_2 - \delta_0) = 0] \leq \frac{4}{q-1}$. Thus, $\Pr_{\hat{r}_1, \hat{r}_2}[\hat{m}_1\hat{r}_2 = \hat{r}_1\hat{m}_2] \leq 8/(q-1)$. □

Claim 7.16. *Let $(\hat{m}_1, \hat{m}_2, \hat{r}_1, \hat{r}_2) \in \mathbb{F}_q^4$ such that $\hat{m}_1\hat{r}_2 \neq \hat{r}_1\hat{m}_2$, and $\alpha, \beta \in \mathbb{F}_q$. Then,*

$$\Pr_{\gamma' \xleftarrow{\$} \mathbb{F}_q^*}\left[\hat{r}_1(\alpha + \gamma'\hat{m}_2) - \hat{m}_1(\beta + \gamma'\hat{r}_2) = 0\right] \leq 1/(q-1)$$

Proof. The claim is immediate since $\hat{m}_1\hat{r}_2 \neq \hat{r}_1\hat{m}_2$. □

Proving that $\Pr[U_2] \leq 4t^2/(q-1)+t^2/q$. This bound follows from Claim 7.17 under the following observation. For $R \in \boldsymbol{R}$, assume that $A = g^{\alpha} X_0^{\beta} R^{\gamma}$, for $\gamma \neq 0$. Then, if (R, δ, z, m) is queried for the modifier after A was queried, the probability that $\hat{r}\alpha' - \hat{m}\beta' = 0$ for $A = g^{\alpha'} X_0^{\beta'} \hat{R}^{\gamma}$, corresponds to the probability that $\hat{r}(\alpha - \gamma z^{-1}\mu) - \beta(m + \hat{r}(\delta - \delta_0)) = 0$ for random, independent $\hat{r}$ and μ, and μ is the answer of the oracle for the modifier query (R, δ, z, m). Thus, by setting $\gamma' \leftarrow \gamma z^{-1}$ and $\delta' \leftarrow \delta - \delta_0$, the bound $\Pr[U_2] \leq 4t^2/(q-1) + t^2/q$ follows from Claim 7.17 since there are at most t^2 possible pairs $(A, \hat{R})$.

Claim 7.17. *For any $\alpha, \beta, \delta' \in \mathbb{F}_q$, $\gamma' \in \mathbb{F}_q^*$ it holds that* $\Pr_{\mu, \hat{r}}[(\alpha - \gamma'\mu)\hat{r} - \beta(m + \hat{r}\delta') = 0] \leq \frac{4}{q-1} + \frac{1}{q}$

Proof. Fix μ such that $\alpha - \mu\gamma' - \beta\delta' \neq 0$. In this case, by Fact 7.14

$$\Pr_{\hat{r}}\left[(\alpha - \gamma'\mu)\hat{r} - \beta(m + \hat{r}\delta') = 0\right] = \Pr_{\hat{r}}\left[\hat{r} = \frac{\beta m}{\alpha - \gamma\mu' - \beta\delta'}\right] = \frac{4}{q-1}$$

Furthermore, since $\Pr_{\mu}[\alpha - \gamma'\mu - \beta\delta' = 0] = 1/q$, the claim follows. □

Proving that $\Pr[U_3 \mid \neg U_1 \wedge \neg U_2] \leq t^4/q^2$. This bound is a corollary Claim 7.18 given the following observation. Assume there exists $\{((R, \delta_i, z_i, m_i), \mu_i) \in \lambda\}_{i=1}^4$ and $\alpha, \beta \in \mathbb{F}_q$ such that $A = g^{\alpha - \gamma z_i^{-1}\mu_i} X_0^{\beta} \hat{R}_i^{z_i^{-1}\gamma_i} \wedge (\alpha - \gamma z_i^{-1}\mu_i)\hat{r}_i - \beta\hat{m}_i = 0$. By reassigning $\gamma_i' \leftarrow \gamma z_i^{-1}\mu_i$, it holds that

$$\begin{pmatrix} \gamma_1'\hat{r}_1 \\ \vdots \\ \gamma_4'\hat{r}_4 \end{pmatrix} \in \left\langle \begin{pmatrix} \hat{r}_1 \\ \vdots \\ \hat{r}_4 \end{pmatrix}, \begin{pmatrix} \hat{m}_1 \\ \vdots \\ \hat{m}_4 \end{pmatrix} \right\rangle$$

Since γ_i' is independent of $\hat{r}_i$ and $\hat{m}_i$ and there are at most t^4 possible tuples $(\hat{m}_i, \hat{r}_i)_{i=1}^4$, the bound follows by the claim below.

Claim 7.18. *Let $\{\hat{m}_i\}_{i=1}^4 \subseteq \mathbb{F}_q$,$\{\hat{r}_i\}_{i=1}^4 \subseteq \mathbb{F}_q^*$ and $\{\gamma_i' \xleftarrow{\$} \mathbb{F}_q\}_{i=1}^4$. Then*

$$\Pr_{\gamma_i'}\left[\begin{pmatrix} \gamma_1'\hat{r}_1 \\ \vdots \\ \gamma_4'\hat{r}_4 \end{pmatrix} \in \left\langle \begin{pmatrix} \hat{r}_1 \\ \vdots \\ \hat{r}_4 \end{pmatrix}, \begin{pmatrix} \hat{m}_1 \\ \vdots \\ \hat{m}_4 \end{pmatrix} \right\rangle\right] \leq \frac{1}{q^2}$$

Proof. By noting that $\vec{v} = (\gamma_1'\hat{r}_1, \ldots, \gamma_4'\hat{r}_4)$ is simply a random vector in $\mathbb{F}_q^4$, the probability that $\vec{v}$ lies on a fixed plane is at most $1/q^2$. □

8 Benchmarks

8.1 Concrete Parameters

In the protocol there are 15 parameters of interest:

1. The size of the Elliptic curve $\kappa = \lfloor \log(|\mathbb{G}|) \rfloor$
2. The soundness, noise-flooding and slackness parameters ℓ, ν and $\varepsilon = \ell + \nu$

3. The size of the RSA moduli $n_{\text{rsa}}(\ell) = [112 \to 2048, 128 \to 3072]$
4. The repetitions numbers $r_{\mathsf{mod}}, r_{\mathsf{df}}, r_{\mathsf{df}^*}$ for Π_{mod}, Π_{df} and Π_{df^*}.
5. The leakage-toleration parameter $\nu_{\text{leak}} = \nu$.
6. The range parameters $n_x = \kappa + \nu_{\text{leak}}, n_\lambda = 2 \cdot \ell + \nu_{\text{leak}}$ for Π_{dlenc},
7. The range parameters $n_b = \kappa + \ell + \nu$, $n_a = n_x + \varepsilon + n_b + \nu$, $n_{\lambda_0} = n_\lambda + \varepsilon + n_b + \nu$ for Π_{sig}.
8. The Damgård-Fujisaki randomness domain $n_{\mathrm{t}\rho} = \max(n_a, n_b) + 2\ell + 2\nu$

How to set r_{df^*}. For $\ell_0 \geq 32$ (the challenge-space size) in Π_{df^*}, by Proposition 5.20, the attacker can inject a cheating subgroup of size ω in Π_{df^*} with probability at most $\max\{((4 + 10^{-3})/\omega)^{r_{\mathsf{df}^*}}, 2^{-15.99 \cdot r_{\mathsf{df}^*}}\}$. Thus, for $r_{\mathsf{df}^*} \geq 16$, cheating subgroups of size $\omega \geq 2^{10} = 1024$ will be accepted with probability at most $((4 + 10^{-3})/1024)^{16} \leq 2^{-127.99}$ which corresponds to a target security level of 128 bits. Under this choice of parameters, accounting for any potential leakage by a cheating subroup of size at most 1024, parameter ν_{leak} (the added noise to the server's share and randomizer) is chosen such that at least $\nu_{\text{leak}} - 10$ bits of noise are added to the plaintext x_2' and randomizer β, ensuring that x_2' mod q remains hidden and β is large enough to compensate for the 10 bits of leakage.

Suggested Parameters. We express all parameters as a function of κ, ℓ, ν, meaning that each parameter is obtained as a weighted sum of κ, ℓ and ν, For instance, $n_a = \mathbf{2} \cdot \kappa + \mathbf{2} \cdot \ell + \mathbf{4} \cdot \nu$. NB: The parameters below are set assuming $(r_{\mathsf{df}^*}, \ell_0) \geq (16, 32)$ (see paragraph above).

input \ prm	ε	ν_{leak}	r_{mod}	r_{df}	n_x	n_λ	n_a	n_b	n_{λ_0}	$n_{\mathrm{t}\rho}$
κ	0	0	0	0	1	0	2	1	1	2
ℓ	1	0	1/2	1	0	2	2	1	4	4
ν	1	1	0	0	1	1	4	1	4	6

The above yields the following concrete values for $\kappa = 256$, $\nu = 64$, $r_{\mathsf{df}^*} = 16$, $\ell_0 = 32$.

bit-sec \ prm	ℓ	ε	ν_{leak}	r_{mod}	r_{df}	n_x	n_λ	n_a	n_b	n_{λ_0}	$n_{\mathrm{t}\rho}$
112	112	176	64	56	112	320	288	992	432	960	1344
128	128	192	64	64	128	320	320	1024	448	1024	1408

Choosing the strong prime d in Π_{fac}. Recall that Π_{fac} is parametrized by a strong prime d known to all. This strong prime has size at least $n_{\text{rsa}} + 2\ell + 2\nu$ which for $(n_{\text{rsa}}, \ell, \nu) = (3072, 128, 64)$ means that $d > 2^{3456}$. Finding a strong prime of this size during execution time can severely impact performance. Instead, since this prime is a public value known to all, we suggest using a hardcoded value (and instructing the parties to verify that it is indeed a strong prime). We recommend using $d = 2^{3459} + 1169115$ which is the smallest strong prime that can be encoded with no less than 3460 bits.

8.2 Experimental Benchmarks

We implement the protocol in C/C++ using OpenSSL v1.1.0 for underlying modular/elliptic curve operations.
Benchmarks were performed on a Intel(R) Core(TM) i7-1365U CPU, and run on one thread. Timings are given in microseconds. Bandwidth requirements are given in kilobytes.

Protocol	Operation	Average	Min.	Max.	10th perc.	99th perc.	band.
Setup	Generation	532363	345455	1063288	409301	861254	86.54
	Verification	514371	489814	638177	507499	543917	N/A
KeyGen	Protocol 2 Step 2	89820	46694	241917	62565	196767	5.81
	Protocol 2 Step 3	13961	13142	41791	13489	19458	2.89
	Protocol 2 Step 4	5517	5217	17388	5388	6005	N/A
Sign	Protocol 3 Step 1	401	377	1074	390	435	0.06
	Protocol 3 Step 2	29447	28228	55532	28963	31793	1.92
	Protocol 3 Step 3	18562	17807	37169	18262	20195	N/A

Table 2: Benchmarks on 1000 executions for 128-bits security

Protocol	Operation	Average	Min.	Max.	10th perc.	99th perc.	band.
Setup	Generation	229505	165778	405945	190950	346518	88.04
	Verification	450156	429693	537248	443908	480626	N/A
KeyGen	Protocol 2 Step 2	84756	2858	475591	22832	269740	4.61
	Protocol 2 Step 3	4444	4285	12349	4308	5595	2.07
	Protocol 2 Step 4	2568	2389	7218	2428	3206	N/A
Sign	Protocol 3 Step 1	396	377	1079	384	431	0.06
	Protocol 3 Step 2	14401	13866	19014	14172	15063	1.55
	Protocol 3 Step 3	8133	7704	15489	7945	8880	N/A

Table 3: Benchmarks on 1000 executions for 112-bits security

Prime type / Bit size	Regular		Tough		Strong	
	average	std dev.	average	std dev.	average	std dev.
1024	23.2	12.6	37.1	18.4	586.7	555.8
1536	70.5	35.9	98.9	53.9	3.6×10^3	4.2×10^3
2048	204.7	137.1	235.5	143.1	14.8×10^3	15.3×10^3

Table 4: Prime Generation Benchmarks (in milliseconds)

Code Repo	Language	Protocol	Rounds	Comp.	Comm.
[Tau21]	GoLang	DKLs18 (Ver. 2018)	2	28084 µs	135.42 KB
[Neu18]	Rust	DKLs18	3	2952 µs	135.42 KB
[Sil23]	Rust	DKLs23	3	29445 µs	115.32 KB
[Unb19]	C++	Lindell17	4	12400 µs	0.90 KB
This paper	C/C++		2	47486 µs	1.98 KB

Table 5: Comparative Benchmarks of signing phases of different protocols for 128 bits security

References

[AF07] Masayuki Abe and Serge Fehr. "Perfect NIZK with Adaptive Soundness". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Berlin, Heidelberg, Feb. 2007, pp. 118–136. DOI: 10.1007/978-3-540-70936-7_7 (cit. on p. 47).

[Bar20] Elaine Barker. *Recommendation for Key Management: Part 1 – General*. NIST Special Publication 800-57 Part 1 Rev. 5. National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.SP.800-57pt1r5. URL: https://doi.org/10.6028/NIST.SP.800-57pt1r5 (cit. on pp. 7, 32).

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020 (cit. on p. 7).

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Berlin, Heidelberg, Aug. 2004, pp. 41–55. DOI: 10.1007/978-3-540-28628-8_3 (cit. on p. 1).

[BHLS24] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. *Exponent-VRFs and Their Applications*. Cryptology ePrint Archive, Paper 2024/397. 2024. URL: https://eprint.iacr.org/2024/397 (cit. on pp. 5–7).

[BHLS25a] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. *Presentation of "Exponent-VRFs and Their Applications"*. Talk at Eurocrypt 2025. May 2025. URL: https://iacr.org/submit/files/slides/2025/eurocrypt/eurocrypt2025/365/365_slides.pptx (cit. on p. 8).

[BHLS25b] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. *Private communication regarding "Exponent-VRFs and Their Applications"*. Personal communication. Email correspondence with the authors on May 15, 2025. May 2025 (cit. on p. 7).

[BMP22] Constantin Blokh, Nikolaos Makriyannis, and Udi Peled. *Efficient Asymmetric Threshold ECDSA for MPC-based Cold Storage*. Cryptology ePrint Archive, Report 2022/1296. 2022. URL: https://eprint.iacr.org/2022/1296 (cit. on p. 1).

[BN06] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, 2006, pp. 390–399. DOI: 10.1145/1180405.1180453 (cit. on p. 22).

[BP23] Luís T. A. N. Brandão and Rene Peralta. *NIST First Call for Multi-Party Threshold Schemes*. Tech. rep. NIST IR 8214C (Initial Public Draft). Public comment period closed on April 10, 2023. National Institute of Standards and Technology, 2023. DOI: 10.6028/NIST.IR.8214C.ipd. URL: https://doi.org/10.6028/NIST.IR.8214C.ipd (cit. on p. 1).

[Bro02] Daniel R. L. Brown. *Generic Groups, Collision Resistance, and ECDSA*. Contributions to IEEE P1363a. Updated version for "The Exact Security of ECDSA." Available from http://grouper.ieee.org/groups/1363/. Feb. 2002 (cit. on pp. 12, 45, 46).

[Bro05] Daniel R. L. Brown. "Generic Groups, Collision Resistance, and ECDSA". In: *DCC* 35.1 (2005), pp. 119–152. DOI: 10.1007/s10623-003-6154-z (cit. on p. 2).

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *Journal of Cryptology* 13.1 (Jan. 2000), pp. 143–202. DOI: 10.1007/s001459910006 (cit. on p. 90).

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd FOCS*. IEEE Computer Society Press, Oct. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888 (cit. on p. 90).

[CC18] Pyrros Chaidos and Geoffroy Couteau. "Efficient Designated-Verifier Non-interactive Zero-Knowledge Proofs of Knowledge". In: *EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. LNCS. Springer, Cham, 2018, pp. 193–221. DOI: 10.1007/978-3-319-78372-7_7 (cit. on pp. 3, 80, 81).

[CDGLN18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. "The Wonderful World of Global Random Oracles". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Cham, 2018, pp. 280–312. DOI: 10.1007/978-3-319-78381-9_11 (cit. on p. 90).

[CEP83] E.R Canfield, Paul Erdös, and Carl Pomerance. "On a problem of Oppenheim concerning "factorisatio numerorum"". In: *Journal of Number Theory* 17.1 (1983), pp. 1–28. ISSN: 0022-314X. DOI: https://doi.org/10.1016/0022-314X(83)90002-1. URL: https://www.sciencedirect.com/science/article/pii/0022314X83900021 (cit. on p. 83).

[CGGMP20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1769–1787. DOI: 10.1145/3372297.3423367 (cit. on pp. 2–4).

[CGKR22] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. "Sharp: Short Relaxed Range Proofs". In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 609–622. DOI: 10.1145/3548606.3560628 (cit. on p. 3).

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. "Practical UC security with a Global Random Oracle". In: *ACM CCS 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM Press, Nov. 2014, pp. 597–608. DOI: 10.1145/2660267.2660374 (cit. on p. 90).

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. "Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments". In: *EUROCRYPT 2021, Part III*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. LNCS. Springer, Cham, Oct. 2021, pp. 247–277. DOI: 10.1007/978-3-030-77883-5_9 (cit. on pp. 3, 20, 80).

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Cham, 2017, pp. 321–350. DOI: 10.1007/978-3-319-56614-6_11 (cit. on pp. 3, 16).

[Dam92] Ivan Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 445–456. DOI: 10.1007/3-540-46766-1_36 (cit. on p. 47).

[Den06] Alexander W. Dent. *The Hardness of the DHK Problem in the Generic Group Model*. Cryptology ePrint Archive, Report 2006/156. 2006. URL: https://eprint.iacr.org/2006/156 (cit. on p. 47).

[DF02] Ivan Damgård and Eiichiro Fujisaki. "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order". In: *ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. LNCS. Springer, Berlin, Heidelberg, Dec. 2002, pp. 125–142. DOI: 10.1007/3-540-36178-2_8 (cit. on pp. 4, 17).

[DKLs18a] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. *Secure Two-party Threshold ECDSA from ECDSA Assumptions*. Cryptology ePrint Archive, Report 2018/499. 2018. URL: https://eprint.iacr.org/2018/499 (cit. on pp. 6, 8).

[DKLs18b] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Secure Two-party Threshold ECDSA from ECDSA Assumptions". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 980–997. DOI: 10.1109/SP.2018.00036 (cit. on pp. 6, 8).

[DKLs19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Threshold ECDSA from ECDSA Assumptions: The Multiparty Case". In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 1051–1066. DOI: 10.1109/SP.2019.00024 (cit. on p. 5).

[DKLs24] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Threshold ECDSA in Three Rounds". In: *2024 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2024, pp. 3053–3071. DOI: 10.1109/SP54263.2024.00178 (cit. on pp. 2, 6, 8).

[Fir] *Fireblocks*. URL: https://www.fireblocks.com/ (cit. on p. 1).

[Fis05] Marc Fischlin. "Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors". In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 152–168. DOI: 10.1007/11535218_10 (cit. on p. 29).

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. “Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations”. In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Berlin, Heidelberg, Aug. 1997, pp. 16–30. DOI: 10.1007/BFb0052225 (cit. on p. 16).

[For] *Fordefi*. URL: https://fordefi.com/ (cit. on p. 1).

[Gen00] Rosario Gennaro. “An Improved Pseudo-random Generator Based on Discrete Log”. In: *CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Springer, Berlin, Heidelberg, Aug. 2000, pp. 469–481. DOI: 10.1007/3-540-44598-6_29 (cit. on pp. 3, 20, 80).

[GG18] Rosario Gennaro and Steven Goldfeder. “Fast Multiparty Threshold ECDSA with Fast Trustless Setup”. In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859 (cit. on pp. 2–4).

[Gol09] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. 1st. USA: Cambridge University Press, 2009. ISBN: 052111991X (cit. on p. 90).

[GS22] Jens Groth and Victor Shoup. “On the Security of ECDSA with Additive Key Derivation and Presignatures”. In: *EUROCRYPT 2022, Part I*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13275. LNCS. Springer, Cham, 2022, pp. 365–396. DOI: 10.1007/978-3-031-06944-4_13 (cit. on pp. 2, 3, 8, 12–14, 28, 45, 46, 60, 64, 69).

[HLM25] Iftach Haitner, Yehuda Lindell, and Nikolaos Makriyannis. *Integer Commitments, Old and New Tools*. Cryptology ePrint Archive, Paper 2025/081. 2025. URL: https://eprint.iacr.org/2025/081 (cit. on pp. 4, 18, 19, 44).

[HN04] Johan Håstad and Mats Naslund. “The security of all RSA and discrete log bits”. In: *Journal of the ACM (JACM)* 51.2 (2004), pp. 187–230 (cit. on p. 80).

[HW18] Susan Hohenberger and Brent Waters. “Synchronized Aggregate Signatures from the RSA Assumption”. In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Cham, 2018, pp. 197–229. DOI: 10.1007/978-3-319-78375-8_7 (cit. on p. 3).

[Inc16] Apple Inc. *Local and Remote Notification Programming Guide*. https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/. Apple Push Notification payloads are limited to 4096 bytes (4 KB). 2016 (cit. on p. 6).

[Iso] *Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification*. ISO/IEC 18004:2024. Geneva, Switzerland: International Organization for Standardization, 2024 (cit. on p. 5).

[KK04] Takeshi Koshiba and Kaoru Kurosawa. “Short Exponent Diffie-Hellman Problems”. In: *PKC 2004*. Ed. by Feng Bao, Robert Deng, and Jianying Zhou. Vol. 2947. LNCS. Springer, Berlin, Heidelberg, Mar. 2004, pp. 173–186. DOI: 10.1007/978-3-540-24632-9_13 (cit. on pp. 3, 5, 20, 80–82).

[KNR24] Julia Kastner, Ky Nguyen, and Michael Reichle. "Pairing-Free Blind Signatures from Standard Assumptions in the ROM". In: *CRYPTO 2024, Part I*. LNCS. Springer, Cham, Aug. 2024, pp. 210–245. DOI: 10.1007/978-3-031-68376-3_7 (cit. on p. 3).

[Ks22] Yashvanth Kondi and abhi shelat. "Improved Straight-Line Extraction in the Random Oracle Model with Applications to Signature Aggregation". In: *ASIACRYPT 2022, Part II*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13792. LNCS. Springer, Cham, Dec. 2022, pp. 279–309. DOI: 10.1007/978-3-031-22966-4_10 (cit. on p. 29).

[Lin17] Yehuda Lindell. "Fast Secure Two-Party ECDSA Signing". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 613–644. DOI: 10.1007/978-3-319-63715-0_21 (cit. on pp. 2, 4–7).

[LL94] Chae Hoon Lim and Pil Joong Lee. "More Flexible Exponentiation with Precomputation". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Berlin, Heidelberg, Aug. 1994, pp. 95–107. DOI: 10.1007/3-540-48658-5_11 (cit. on p. 80).

[LLZD25] Yingjie Lyu, Zengpeng Li, Hong-Sheng Zhou, and Xudong Deng. *Threshold ECDSA in Two Rounds*. Cryptology ePrint Archive, Paper 2025/1696. 2025. DOI: 10.1145/3719027.3765176. URL: https://eprint.iacr.org/2025/1696 (cit. on p. 3).

[LN18] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1837–1854. DOI: 10.1145/3243734.3243788 (cit. on p. 2).

[LW88] Douglas L Long and Avi Wigderson. "The Discrete Logarithm Hides O(logn) Bits". In: *SIAM Journal on Computing* 17.2 (1988), pp. 363–372 (cit. on p. 80).

[MYG23] Nikolaos Makriyannis, Oren Yomtov, and Arik Galansky. *Practical Key-Extraction Attacks in Leading MPC Wallets*. Cryptology ePrint Archive, Report 2023/1234. 2023. URL: https://eprint.iacr.org/2023/1234 (cit. on pp. 4, 5).

[Neu18] NeuCrypt. *mpecdsa*. https://gitlab.com/neucrypt/mpecdsa. Available at https://gitlab.com/neucrypt/mpecdsa. 2018 (cit. on pp. 6, 73).

[OW96] Paul C. van Oorschot and Michael J. Wiener. "On Diffie-Hellman Key Agreement with Short Exponents". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 332–343. DOI: 10.1007/3-540-68339-9_29 (cit. on p. 80).

[Per86] René Peralta. "Simultaneous Security of Bits in the Discrete Log". In: *EUROCRYPT'85*. Ed. by Franz Pichler. Vol. 219. LNCS. Springer, Berlin, Heidelberg, Apr. 1986, pp. 62–72. DOI: 10.1007/3-540-39805-8_8 (cit. on p. 80).

[Pol74] John M Pollard. "Theorems on factorization and primality testing". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 76. 3. Cambridge University Press. 1974, pp. 521–528 (cit. on p. 17).

[Pol78] John M Pollard. "Monte Carlo methods for index computation (mod p)". In: *Mathematics of computation* 32.143 (1978), pp. 918–924 (cit. on p. 80).

[PS98] Sarvar Patel and Ganapathy S. Sundaram. "An Efficient Discrete Log Pseudo Random Generator". In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Berlin, Heidelberg, Aug. 1998, pp. 304–317. DOI: 10.1007/BFb0055737 (cit. on p. 80).

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725 (cit. on p. 1).

[Sch98] Claus P. Schnorr. *Almost All Discrete Log Bits Are Simultaneously Secure*. Cryptology ePrint Archive, Report 1998/020. 1998. URL: https://eprint.iacr.org/1998/020 (cit. on p. 80).

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18 (cit. on pp. 12, 46, 47).

[Sil23] Silence Laboratories. *silent-shard-dkls-23-ll*. https://github.com/silence-laboratories/silent-shard-dkls23-ll. Available at https://github.com/silence-laboratories/silent-shard-dkls23-ll. 2023 (cit. on pp. 6, 73).

[Sod] *Sodot*. URL: https://www.sodot.dev/ (cit. on p. 1).

[Tau21] TaurusHQ. *multi-party-sig: DKLS18 Protocols*. https://github.com/taurusgroup/multi-party-sig/tree/main/protocols/doerner. Available at https://github.com/taurusgroup/multi-party-sig/tree/main/protocols/doerner. 2021 (cit. on p. 73).

[Unb19] Unbound Security. *blockchain-crypto-mpc*. https://github.com/unboundsecurity/blockchain-crypto-mpc. Available at https://github.com/unboundsecurity/blockchain-crypto-mpc. 2019 (cit. on pp. 6, 73).

[Uti] *Utila*. URL: https://utila.io/ (cit. on p. 1).

[XALCCXYZ23] Haiyang Xue, Man Ho Au, Mengling Liu, Kwan Yin Chan, Handong Cui, Xiang Xie, Tsz Hon Yuen, and Chengru Zhang. "Efficient Multiplicative-to-Additive Function from Joye-Libert Cryptosystem and Its Application to Threshold ECDSA". In: *ACM CCS 2023*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda. ACM Press, Nov. 2023, pp. 2974–2988. DOI: 10.1145/3576915.3616595 (cit. on p. 6).

[XAXYC21] Haiyang Xue, Man Ho Au, Xiang Xie, Tsz Hon Yuen, and Handong Cui. "Efficient Online-friendly Two-Party ECDSA Signature". In: *ACM CCS 2021*. Ed. by Giovanni Vigna and Elaine Shi. ACM Press, Nov. 2021, pp. 558–573. DOI: 10.1145/3460120.3484803 (cit. on pp. 2, 4, 6).

[Zen] *Zengo*. URL: https://zengo.com/ (cit. on p. 1).

A Analysis of the Assumptions

A.1 The short-exponent indistinguishability assumption

In this section, we analyze the hardness of the short-exponent indistinguishability assumption. We prove its equivalence with the short-exponent discrete logarithm assumption, an old and well-established assumptions, over groups of unknown order. This generalizes and improves over previous results from [Gen00; KK04; CC18].

A.1.1 Background on the short-exponent discrete logarithm assumption

Fix a cyclic group $\mathbb{G}$ of order $|\mathbb{G}|$. Given an integer $T < |\mathbb{G}|$, the T-bounded short-exponent discrete logarithm (T-SEDL) assumption states that no PPT algorithm can, on input (g, g^x) where $g \xleftarrow{\$} \mathbb{G}$ and $x \xleftarrow{\$} \{0, \cdots, T\}$, find x with non-negligible probability.

SEDL is essentially as old as the discrete logarithm itself in cryptography: it originally appears in the seminal paper of Pollard [Pol78] (who introduced the catching-kangaroo method to solve T-SEDL in $O(\sqrt{T})$ group operations) and was commonly assumed in cryptographic schemes during the pre-elliptic-curves era (for example, the use of a short secret exponent is explicitly standardized in DSS for the DSA algorithm). The work of [OW96] provided the first concrete cryptanalysis of SEDL over finite fields and showed that $\mathbb{G}$ must have a large prime-order subgroup. In addition, [OW96] also discusses the impact of small factors in the order of the subgroup on secure values of T. In a nutshell, since the discrete logarithm modulo each small factor is easily computed, they remark that a bitsize equal to the logarithm of the product of small factors is always leaked. For this reason, when dealing with short-exponent discrete logarithm assumptions, it is much more convenient to choose a group whose order does not contain any small factors.

The work of [Gen00], building upon an earlier work of [PS98], was the first to establish the the equivalence between SEDL and the short-exponent indistinguishability assumption (the name is from [CKLR21]) over the group $\mathbb{Z}_p^*$ (for a prime p). This result was later extended in [KK04] over arbitrary prime-order groups. Eventually, this was later extended in [CC18] to the setting of composite-order groups; albeit specifically in the setting where T is sufficiently close to $|\mathbb{G}|$.

However, the roots of the equivalence between SEI and SEDL are much older: distinguishing $\{g^x \mid x \xleftarrow{\$} \mathbb{Z}_p\}$ from $\{g^x \mid x \xleftarrow{\$} [q]\}$ (with $q \ll p$) is, in essence, the problem of predicting the most significant bits of x (when defining "short" as "smaller than $p/2$", this translates to extracting the MSB of x), a goal which has been the subject of a long line of work starting with [Per86] and followed by Long and Widgerson [LW88], Håstad and Näslund [HN04], Schnorr [Sch98], and many more.

The SEDL assumption has also been studied in the context of achieving faster exponentiation via precomputation within a fixed memory budget [LL94], in the context of security in the preprocessing setting (where the attacker has a long preprocessing time and can store an advice). Eventually, the study of fast SEDL algorithm is tightly related to finding efficient algorithms for ElGamal decryption, BGN decryption, or distributed discrete logarithm procedures.

A.1.2 Reduction to SEDL

As discussed above, the SEI assumption was shown to be equivalent to the SEDL assumption in several settings: over $\mathbb{Z}_p^*$ in [Gen00], over all prime-order groups in [KK04], and over the subgroup

QR_N of quadratic residues over $\mathbb{Z}_N^*$ (where N is a product of strong primes) in [CC18] (this last result is restricted to the setting where the bound on the short exponent is significantly larger than $\sqrt{N}$).

In this section, we introduce a new reduction that proves the equivalence between SEI and SEDL over an arbitrary group $\mathbb{G}$ of unknown order (when a sufficiently good bound is known on $|\mathbb{G}|$. This improves significantly over the reductions in [KK04] and [CC18]. A downside of our reduction is that it is restricted to short exponents sampled from a domain of the form $S_i = \mathbb{Z}_{\lfloor |\mathbb{G}|/2^i \rfloor}$. While this bound cannot be computed exactly for general unknown-order groups, we note that the reduction trivially extends to short exponents sampled from any distribution statistically close to the uniform distribution over S_i. When a sufficiently good bound on $|\mathbb{G}|$ is known, this is easy to do. Typically, for the case of interest in this work, $\mathbb{G} = \mathsf{QR}_N$ and $|\mathbb{G}| = \varphi(N)/4 = N/4 - (p+q-1)/4 = N/4 - O(\sqrt{N})$, hence the uniform distribution over $\mathbb{Z}_{\lfloor N/2^{i+2} \rfloor}$ is statistically close to the uniform distribution over $\mathbb{Z}_{\lfloor |\mathbb{G}|/2^i \rfloor}$. In our work, we choose i such that $\lfloor |\mathbb{G}|/2^i \rfloor \approx 2^{2\ell}$ (where ℓ is set to 128 in practice).

Revisiting the reduction of [KK04]. To design our improved reduction, we revisit the SEI-to-SEDL reduction provided in [KK04]. We start by remarking that their proof works with a flavor of SEI that differs from the flavor considered here. We refer to this flavor as *short-exponent in the high bits*. For a given group $\mathbb{G}$ of known, odd order, their reduction considers indistinguishability between the following distributions:

$$A_i = \{(g, g^x) \mid g \xleftarrow{\$} \mathbb{G} \text{ and } x \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|} \cap (2^i)\mathbb{Z}\}.$$

The extreme case A_0 corresponds to full size exponents. Furthermore, when moving from A_i to A_{i+1} the size of the space of exponents reduces by one bit. Let A_c be the target SEDL distribution. By a standard hybrid argument, it holds that if there exists a PPT adversary $\mathcal{A}$ distinguishing A_0 and A_c with advantage ε, then there exists an intermediate value $0 \leq i < c$ and a PPT algorithm $\mathcal{A}'$ distinguishing A_i and A_{i+1} with advantage at least ε/c.

It remains to argue that a distinguisher between A_i and A_{i+1} yields an attack against the SEDL assumption. To do so, consider the following distributions for $b \in \{0, 1\}$:

$$B_i^b = \{(g, g^x) \mid g \xleftarrow{\$} \mathbb{G}, x \xleftarrow{\$} \mathbb{Z}_{\lfloor |\mathbb{G}|/2^i \rfloor} \text{ s.t. } \mathsf{lsb}(x) = b\}.$$

Given a distinguisher $\mathcal{B}$ against B_i^0, B_i^1, [KK04] constructs a distinguisher $\mathcal{A}$ against A_i, A_{i+1} as follows: on input (g, g^x) from either A_i or A_{i+1}, run $b \leftarrow \mathcal{B}(g, g^{x/2^i})$ and output b. Observe that if (g, g^x) was sampled from A_i, then $(g, g^{x/2^i})$ is distributed as a random sample from B_i^1, while if (g, g^x) was sampled from A_{i+1}, $(g, g^{x/2^i})$ is distributed as a random sample from B_i^0. Then, an algorithm $\mathcal{B}$ distinguishing B_i^0 from B_1^i translates to an algorithm predicting, given a sample (g, g^x), the least significant bit of $x \in \mathbb{Z}_{\lfloor |\mathbb{G}|/2^i \rfloor}$ (with non-negligible advantage). Note that raising the second menber to the power 2^{-i} is achieved by computing the inverse of 2^i modulo the order of the group (which is odd and known). Now, given algorithm to compute the LSB of x for B_i with advantage close to 1 easily gives a bit-by-bit approach to recover the discrete logarithm x: if x is even, we can go to the next bit by considering $(g, g^{x/2})$; for an odd x we compute g^{x-1} and proceed as before. Furthermore, an imprecise predictor for the LSB in B_i can be easily re-randomized to provide a very precise predictor[5] for $B_{i+\delta}$.

[5] In [KK04], this is expressed slightly differently, by guessing the δ most significant bits before re-randomizing. However, this amounts to the same idea.

Handling groups of unknown order. The analysis of [KK04] is restricted to groups of known order due to the need to compute the inverse of 2^i. However, we make the following simple observation: over a group of unknown order, we can achieve the same result by *scaling the basis*. In this approach, however, we still need the order of the group to be odd. Concretely, given a distinguisher $\mathcal{B}$ against B_i^0, B_i^1, we construct a slightly different distinguisher $\mathcal{A}'$: on input (g, g^x) sampled from A_i or A_{i+1}, $\mathcal{A}'$ outputs $b \leftarrow \mathcal{B}(g^{2^i}, g^x)$. Observe that the distribution of $h = g^{2^i}$ is uniform over $|\mathbb{G}|$, and $g^x = h^{x/2^i}$, hence $\mathcal{A}'$ has the same advantage as $\mathcal{A}$ against A_i, A_{i+1}. But the new reduction does not need to compute the modular inverse of 2^i anymore. Similarly, after extracting the LSB of x, the bit-by-bit extraction of x proceeds by repeating the process with (g^2, g^x) if x is even (instead of $(g, g^{x/2})$) or with (g^2, g^{x-1}) is x is odd.
This generalizes the reduction of [KK04] to groups of unknown order, albeit only for a flavor of SEDL with "short exponents in the high bits". Below, we extend this analysis to the case of short exponents as considered in our work, for short-exponent bounds of a special form.

Reducing the standard SEI with special bounds to SEDL. Let us define:

$$A_i' = \{(g, g^x) \mid g \xleftarrow{\$} \mathbb{G}, x \xleftarrow{\$} \mathbb{Z}_{\lfloor |\mathbb{G}|/2^i \rfloor}\}.$$

We now show that the indistinguishability of A_i' and $A_0' = A_0$ is equivalent to the indistinguishability of A_i and A_0. Indeed, if (g, h) is an element of A_i' then (g, h^{2^i}) is an element of A_i. Moreover, if (g, h) is an element of A_0' then (g, h^{2^i}) is an element of A_0. In the other direction, an element (g, h) of A_i can be sent to the element (g^{2^i}, h) of A_i'. Thanks to these two efficiently computable maps, we directly conclude that A_i' is indistinguishable from A_0' if and only if A_i is indistinguishable from A_0.

A.2 Factoring tough-RSA numbers compared standard RSA numbers

Since we are using RSA numbers with a very specific structure, one might worry that factorization of these numbers might be much easier than the factorization of ordinary RSA numbers. We would like to argue about the security of our choice beyond the rather obvious remark that none of the known factoring algorithms can exploit our chosen structure of tough-RSA numbers.
First remark that the structure implies two dictinct properties. The first property avoids any small prime factors in the group order. The second property fixes the size of factors to remain smaller than $2^{2\ell}$.
As remarked in Section A.1.2, without the first property, we would have to increase the bitsize of small exponents to compensate for the presence of the small factors. Thus, it make perfect sense to compare the security of tough-RSA to the security of RSA number without any small factor in the group order. Also, from a heuristic point of view, it is very unclear how the absence of short factors could help while factoring.
To get confidence in our assumption that tough-RSA numbers are hard to factor, we propose to compare the hardness of factoring standard RSA numbers and RSA numbers where $p-1$ and $q-1$ have all their factors below $2^{2\ell}$. It then makes sense to argue that if the hardness of factoring are comparable for these numbers, then the hardness of factoring when small factors are forbidden are also comparable.
To achieve that, we want to estimate the probability that a random standard RSA number has no factors higher than $2^{2\ell}$ in $p-1$ and $q-1$. It also makes sense to consider the case where only one of $p-1$ and $q-1$ satisfies the $2^{2\ell}$-smoothness assumption. Indeed, known factoring algorithms

that require prime factors with a special property work even when a single prime has the property. Thus, it is reasonable to assume that an algorithm exploiting $2^{2\ell}$-smoothness of $p-1$ would also work with a single special prime.
To estimate the probability that a number X is B-smooth, we use the Canfield-Erdös-Pomerance [CEP83] approximation $\rho(\log(X)/\log(B)$ where ρ is the Dickman-Rho function. This yields the following results:

- For $(112, 2048)$, the probability for any given prime to have $p-1$ smooth enough is approximately $\rho(512/112) \approx 1/882$.
- For $(128, 2048)$, the probability for a single prime is approximately $\rho(4) \approx 1/204$.
- For $(128, 3072)$, the probability for a single prime is approximately $\rho(6) \approx 1/51000$.

As a consequence, for long-term keys of 3072, a fraction larger than 2^{-15} of standard RSA numbers would be affected by a specific algorithm improving on the principle of the $P-1$ algorithm. If both factors are required to be smooth the probability would be higher than 2^{-32} and still far from negligible.

B Symbolic Evaluation to Lazy Evaluation Reduction

In this section, we show that the symbolic evaluation form Section 7.1.1 is indistinguishable from the GGM up to an error term of $O(t^2/q)$ (with a small hidden constant ≈ 5), where t denotes the number of queries of the attacker. Namely, we prove the following theorem.

Theorem B.1. *For an algorithm* D *interacting with the group oracle in the GGM or the lazy symbolic evaluation from Section 7.1.1 making at most* t *queries to the group oracle, let* $\mathrm{out}_{\mathsf{D}}$ *and* $\mathrm{out}'_{\mathsf{D}}$ *denote the output of* D *when interacting with the respective oracle. Then, for every* D*,*

$$\left|\Pr_{\mathsf{D}}[\mathrm{out}_{\mathsf{D}} = 1] - \Pr_{\mathsf{D}}[\mathrm{out}'_{\mathsf{D}} = 1]\right| \leq \frac{(t+5)(t+4)}{q} + 2 \cdot \frac{(t+4)(2t+9)+2t}{q-1}$$

Proof. The above is a corollary of Propositions B.2, B.3 and B.5 stated further below. □

B.1 Lazy Evaluation 1

Let $\mathsf{LazyEval}_1$ denote the following group oracle defined for the following operations. This oracle corresponds to the standard lazy evaluation oracle of the GGM, suitably modified for our use case.

1. Initialization.
   (a) $\pi \leftarrow \{(0, \mathbb{1})\}$
   (b) Sample $g \xleftarrow{\$} \boldsymbol{G} \setminus \{\mathbb{1}\}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.
   (c) Sample $x \xleftarrow{\$} \mathbb{F}_q^*$ and $X \xleftarrow{\$} \boldsymbol{G}$. Set $\pi \leftarrow \pi \cup \{(x, X), (-x, X^{-1})\}$.
   (d) Set $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set)
   (e) Return (g, X).
2. Group query. On input $i \in \mathbb{F}_q$, do:

(a) If $i \notin \text{domain}(\pi)$

i. Sample $P \xleftarrow{\$} \boldsymbol{G}$ (if $P \in \text{range}(\pi)$, repeat this step)

ii. Set $\pi \leftarrow \pi \cup \{(i, P), (-i, P^{-1})\}$.

(b) Return $\pi(i)$

3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:

(a) For $j \in \{1, 2\}$, if $P_j \notin \text{range}(\pi)$:

i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$ (if $i \in \text{domain}(\pi)$, repeat this step)

ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.

(b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.

4. Presignatures. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$, and do:

(a) If $i \notin \text{domain}(\pi)$, invoke the group query on input i to obtain R.

(b) Return $R = \pi(i)$ and set $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup \{R\}$. .

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^{*2}$, do:

(a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$

i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q$

ii. Invoke the group query on input $\mu + z \cdot \pi^{-1}(R)$

iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$.

(b) Return $\lambda(R, \delta, z, m)$

6. Signature. On input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:

(a) Let $\mu = \lambda(R, \delta, z, m)$ and $\hat{R} = \pi(\mu + z\pi^{-1}(R))$

(b) Set $\hat{r} = \mathcal{X}(\hat{R})$ and remove all records of R from $\boldsymbol{R}$ and λ

(c) Return $(\hat{r}, s)$ for $s \leftarrow (\pi^{-1}(\hat{R}))^{-1}(m + \hat{r}(x + \delta)) \bmod q$.
(If $m + \hat{r}(x + \delta) = 0$ or $\hat{r} = 0$, return fail)

B.2 Lazy Evaluation 2

Let $\mathsf{LazyEval}_2$ denote the following group oracle defined for the following operations. Changes with respect to $\mathsf{LazyEval}_1$ are highlighted.

1. Initialization.

(a) $\pi \leftarrow \{(0, \mathbb{1})\}$

(b) Sample $g \xleftarrow{\$} \boldsymbol{G} \setminus \{\mathbb{1}\}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.

(c) Sample $x \xleftarrow{\$} \mathbb{F}_q^*$ and $X \xleftarrow{\$} \boldsymbol{G}$. (If $X \in \text{range}(\pi)$, abort)

(d) Set $\pi \leftarrow \pi \cup \{(x, X), (-x, X^{-1})\}$.

(e) Set $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set)

(f) For all $P \in \boldsymbol{G}$ such that $\mathcal{X}(P) = 0$, execute the group addition query on input $(P, \mathbb{1})$.

(g) Return (g, X)

2. Group query. On input $i \in \mathbb{F}_q$, do:

   (a) If $i \notin \text{domain}(\pi)$

      i. Sample $P \xleftarrow{\$} \boldsymbol{G}$. (If $P \in \text{range}(\pi)$, abort)

      ii. Set $\pi \leftarrow \pi \cup \{(i, P), (-i, P^{-1})\}$.

   (b) Return $\pi(i)$

3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:

   (a) For $j \in \{1, 2\}$, if $P_j \notin \text{range}(\pi)$:

      i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$ (if $i \in \text{domain}(\pi)$, abort)

      ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.

   (b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.

4. Presignatures. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$ (If $i \in \text{domain}(\pi)$, abort), and do:

   (a) Invoke the group query on input i.

   (b) Return $R = \pi(i)$ and set $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup \{R\}$.

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^{*2}$, do:

   (a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$

      i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q$ (If $\mu + z \cdot \pi^{-1}(R) \in \text{domain}(\pi)$, abort)

      ii. Invoke the group query on input $\mu + z \cdot \pi^{-1}(R)$

      iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$.

   (b) Return $\lambda(R, \delta, z, m)$

6. Signature. On input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:

   (a) Let $\mu = \lambda(R, \delta, z, m)$ and $\hat{R} = \pi(\mu + z\pi^{-1}(R))$

   (b) Set $\hat{r} = \mathcal{X}(\hat{R})$ and remove all records of R from $\boldsymbol{R}$ and λ

   (c) Return $(\hat{r}, s)$ for $s \leftarrow (\pi^{-1}(\hat{R}))^{-1}(m + \hat{r}(x + \delta)) \bmod q$.
   (If $m + \hat{r}(x + \delta) = 0$, return fail)

Proposition B.2. *For an algorithm* D *interacting with* $\mathsf{LazyEval}_1$ *or* $\mathsf{LazyEval}_2$ *making at most* t *queries to the group oracle, let* $\text{out}_{\mathsf{D}}^{\mathsf{LazyEval}_i}$ *denote the output of* D *when interacting with* $\mathsf{LazyEval}_i$. *Then, for every* D,

$$\left| \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{LazyEval}_1} = 1\right] - \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{LazyEval}_2} = 1\right] \right| \leq \frac{(t+5)(t+4)}{q}$$

Proof. Note that the two experiments are distinguishable only if $\mathsf{LazyEval}_2$ aborts. At any given query, $\mathsf{LazyEval}_2$ aborts with probability $\frac{|\pi|}{q}$. After initialization, we have $|\pi| \leq 9$. Since each query adds at most two elements to π, the size of π at the i-th query is at most $9 + 2(i-1)$. Therefore, the total probability that $\mathsf{LazyEval}_2$ aborts over t queries is at most

$$\sum_{i=-2}^{t} \frac{9 + 2(t-1)}{q} = \frac{t+3}{2} \cdot \frac{9 + 2t + 1}{q}$$
$$= \frac{(t+5)(t+3)}{q}$$

□

B.3 Symbolic Evaluation 1

Let $\mathsf{SymbLazyEval}_1$ denote the following group oracle defined for the following operations. Changes with respect to $\mathsf{LazyEval}_2$ are highlighted.

1. Initialization.
   (a) $\pi \leftarrow \{(0, \mathbb{1})\}$
   (b) Sample $g \xleftarrow{\$} \boldsymbol{G} \setminus \{\mathbb{1}\}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.
   (c) Sample $x \xleftarrow{\$} \mathbb{F}_q^*$ and $X \xleftarrow{\$} \boldsymbol{G}$. (If $X \in \text{range}(\pi)$, abort)
   (d) Set $\rho = \{(x, X), (-x, X^{-1})\}$and $\pi \leftarrow \pi \cup \{(\mathfrak{R}_1, X), (-\mathfrak{R}_1, X^{-1})\}$.
   (e) Set $\boldsymbol{V} \leftarrow \{\mathfrak{R}_1\}$ and $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set) and $c = 1$.
   (f) For all $P \in \boldsymbol{G}$ such that $\mathcal{X}(P) = 0$, execute the group addition query on input $(P, \mathbb{1})$.
   (g) Return (g, X)
2. Group query. On input $i \in \mathbb{F}_q$, do:
   (a) If $i \notin \text{domain}(\pi)$
      i. Sample $P \xleftarrow{\$} \boldsymbol{G}$. (If $P \in \text{range}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P), (-i, P^{-1})\}$.
   (b) Return $\pi(i)$
3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:
   (a) For $j \in \{1, 2\}$, if $P_j \notin \text{range}(\pi)$:
      i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$. (If $i \in \text{domain}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.
   (b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.
4. Presignatures. Set $c \leftarrow c + 1$, and do:
   (a) Sample $R \xleftarrow{\$} \boldsymbol{G}$. (If $R \in \text{range}(\pi)$, abort).

(b) Set $\rho \leftarrow \rho \cup \{(i, R), (-i, R^{-1})\}$.

(c) Set $\pi \leftarrow \pi \cup \{(\mathfrak{R}_c, R), (-\mathfrak{R}_c, R^{-1})\}$, $\boldsymbol{V} \leftarrow \boldsymbol{V} \cup \{\mathfrak{R}_c\}$ and $R \leftarrow \boldsymbol{R} \cup \{R\}$.

(d) Return R.

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^{*2}$, do:

(a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$

i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q$. (If $\mu + z \cdot \pi^{-1}(R) \in \text{domain}(\pi)$, abort)

ii. Invoke the group query on input $\mu + z\pi^{-1}(R)$.

iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$ and $\rho \leftarrow \rho \cup (\mu + z \cdot \rho^{-1}(R), \hat{R})$.

(b) Return $\lambda(R, \delta, z, m)$

6. Signature. On input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:

(a) Let $\mu = \lambda(R, \delta, z, m)$, $\hat{R} = \mu + z \cdot \pi^{-1}(\mathfrak{R})$ and $\hat{r} = \mathcal{X}(\hat{R})$.

(b) Set $s = (\rho^{-1}(\hat{R}))^{-1}(m + \hat{r} \cdot (x + \delta))$ if $m + \hat{r} \cdot (x + \delta) \neq 0$ and $s \xleftarrow{\$} \mathbb{F}_q^*$, otherwise.

(c) Substitute $s^{-1}(m + \hat{r} \cdot (\mathfrak{R}_1 + \delta))$ for $\pi^{-1}(\hat{R})$ throughout $\text{domain}(\pi)$.

Abort if any two polynomials collapse, i.e., there's a collision.

(d) Remove all records of R and $\mathfrak{R}$ from $\boldsymbol{R}$, $\boldsymbol{V}$ and λ.

(e) Return $(\hat{r}, s)$

Proposition B.3. *For an algorithm* D *interacting with* $\mathsf{LazyEval}_2$ *or* $\mathsf{SymbLazyEval}_1$ *making at most* t *queries to the group oracle, let* $\text{out}_{\mathsf{D}}^{\mathsf{LazyEval}_2}$ *and* $\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_1}$ *denote the output of* D *when interacting with the corresponding oracle. Then, for every* D,

$$\left| \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{LazyEval}_2} = 1\right] - \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_1} = 1\right] \right| \leq \frac{2(t+4)(2t+9) + 4t}{q-1}$$

Proof. Notice that the two experiments are distinguishable only in the following cases:

1. It occurred that $m + \hat{r} \cdot (x + \delta) = 0$ in Item 6b
2. Two polynomials in $\text{domain}(\pi)$ collapsed in Item 6c
3. One of the highlighted items would have aborted according to $\mathsf{LazyEval}_2$ (because $i \notin \text{domain}(\pi)$ according to $\mathsf{SymbLazyEval}_1$ and yet $i = i_0 + \sum_{\mathfrak{R} \in \boldsymbol{V}} i_{\mathfrak{R}} \cdot \rho^{-1}(\pi(\mathfrak{R}))$ for $i_0 + \sum_{\mathfrak{R} \in \boldsymbol{V}} i_{\mathfrak{R}} \cdot \mathfrak{R} \in \text{domain}(\pi)$).

We show that the first two items occur with probability at most t/q, $(t+4)(2t+9)/q$, respectively, and, assuming Item 2 does not occur, Item 3 occurs with probability at most $(t+4)(2t+9)/q$ which immediately implies the claimed bound. The first item follows from Fact B.4 (stated further below) as each $\hat{r}$ is random. For the second item, we count how many points $P, P' \in \text{range}(\pi)$ are susceptible to collapse. Assume that when $\hat{R} = \pi(\mu + z\mathfrak{R}_2)$ is queried for a signature, for some $\mathfrak{R}_2 \in \boldsymbol{V}$, and P and P' admit the following expressions in $\text{domain}(\pi)$:

$$\begin{cases} P = i_0 + i_1\mathfrak{R}_1 + i_2(\mu + z\mathfrak{R}_2) + \sum_{\mathfrak{R} \in \boldsymbol{V} \setminus \{\mathfrak{R}_1, \mathfrak{R}_2\}} i_{\mathfrak{R}} \cdot \mathfrak{R} \\ P' = i_0 + i'_1\mathfrak{R}_1 + i'_2(\mu + z\mathfrak{R}_2) + \sum_{\mathfrak{R} \in \boldsymbol{V} \setminus \{\mathfrak{R}_1, \mathfrak{R}_2\}} i'_{\mathfrak{R}} \cdot \mathfrak{R} \end{cases}$$

Note that the polynomials associated with P and P' can collide only if $i_j \neq i'_j$ for $j \in \{0,1,2\}$ and $i_{\mathfrak{R}} = i'_{\mathfrak{R}}$ for all $\mathfrak{R} \in \boldsymbol{V} \setminus \{\mathfrak{R}_1, \mathfrak{R}_2\}$. Consequently, if P and P' are potential colliding pairs but do not collide when the signature associated with $\hat{R}$ is released, they cannot collide in any subsequent signature query. Therefore, each pair $\{P, P'\}$ is counted at most once as a potentially colliding pair. Since there are $\binom{|\pi|}{2}$ such pairs, the Schwartz-Zippel lemma implies that the probability that two polynomials collide is at most

$$\binom{|\pi|}{2} \cdot \frac{1}{q-1} = \frac{(9+2t)(9+2t-1)}{2(q-1)} = \frac{(2t+9)(t+4)}{q-1}$$

To conclude, under the condition that no two polynomials collide, we apply Schwartz-Zippel once again to deduce that the probability that Item 3 occurs is at most $(2t+9)(t+4)/(q-1)$. □

Fact B.4. *Let $(\mathbb{G}, g, q)$ be the group-generator-order tuple associated with a prime order elliptic curve, and let $a \in \mathbb{F}_q$. Then, $\Pr_{k \leftarrow \mathbb{F}_q^*}\left[\mathcal{X}(g^k) = a\right] \leq \frac{4}{q-1}$.*

B.4 Symbolic Evaluation 2

Let $\mathsf{SymbLazyEval}_2$ denote the following group oracle defined for the following operations. This oracle corresponds to the lazy symbolic oracle from Section 7.1.1.

1. Initialization.
   (a) $\pi \leftarrow \{(0, \mathbb{1})\}$
   (b) Sample $g \xleftarrow{\$} \boldsymbol{G}$ and set $\pi \leftarrow \pi \cup \{(1, g), (-1, g^{-1})\}$.
   (c) Sample $X \xleftarrow{\$} \boldsymbol{G}$ and $\pi \leftarrow \pi \cup \{(\mathfrak{R}_1, X), (-\mathfrak{R}_1, X^{-1})\}$. (If $X \in \text{range}(\pi)$, abort)
   (d) Set $\boldsymbol{V} \leftarrow \{\mathfrak{R}_1\}$ and $\boldsymbol{R} \leftarrow \varnothing$ (the presignatures set) and $c = 1$.
   (e) For all $P \in \boldsymbol{G}$ such that $\mathcal{X}(P) = 0$, execute the group addition query on input $(P, \mathbb{1})$.
   (f) Return (g, X)
2. Group query. On input $i = i_0 + \sum_{\mathfrak{R} \in \boldsymbol{V}} i_{\mathfrak{R}} \cdot \mathfrak{R}$, do:
   (a) If $i \notin \text{domain}(\pi)$
      i. Sample $P \xleftarrow{\$} \boldsymbol{G}$. (If $P \in \text{range}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P), (-i, P^{-1})\}$.
   (b) Return $\pi(i)$
3. Group addition. On input $(P_1, P_2) \in \boldsymbol{G}^2$, do:
   (a) For $j \in \{1,2\}$, if $P_j \notin \text{range}(\pi)$:
      i. Sample $i \xleftarrow{\$} \mathbb{F}_q^*$. (If $i \in \text{domain}(\pi)$, abort)
      ii. Set $\pi \leftarrow \pi \cup \{(i, P_j), (-i, P_j^{-1})\}$.
   (b) Return the group query output on input $\pi^{-1}(P_1) + \pi^{-1}(P_2)$.
4. Presignatures. Set $c \leftarrow c + 1$, and do:

(a) Sample $R \xleftarrow{\$} \boldsymbol{G}$. (If $R \in \text{range}(\pi)$, abort).

(b) Set $\pi \leftarrow \pi \cup \{(\mathfrak{R}_c, R), (-\mathfrak{R}_c, R^{-1})\}$, $\boldsymbol{V} \leftarrow \boldsymbol{V} \cup \{\mathfrak{R}_c\}$ and $R \leftarrow \boldsymbol{R} \cup \{R\}$.

(c) Return R.

5. Modifier. On input $(R, \delta, z, m) \in \boldsymbol{R} \times \boldsymbol{\Delta} \times \mathbb{F}_q^{*2}$, do:

(a) If $(R, \delta, z, m) \notin \text{domain}(\lambda)$

i. Sample $\mu \xleftarrow{\$} \mathbb{F}_q$. (If $\mu + z\pi^{-1}(\mathfrak{R}) \in \text{domain}(\pi)$), abort)

ii. Invoke the group query on input $\mu + z\pi^{-1}(\mathfrak{R})$.

iii. Set $\lambda \leftarrow \lambda \cup ((R, \delta, z, m), \mu)$.

(b) Return $\lambda(R, \delta, z, m)$

6. Signature. On input $(R, \delta, z, m) \in \text{domain}(\lambda)$, do:

(a) Let $\mu = \lambda(R, \delta, z, m)$ and $\hat{R} = \pi(\mu + z \cdot \pi^{-1}(R))$.

(b) Set $\hat{r} = \mathcal{X}(\hat{R})$ and sample $s \xleftarrow{\$} \mathbb{F}_q^*$.

(c) Substitute $s^{-1}(m + \hat{r} \cdot (\mathfrak{R}_1 + \delta))$ for $\pi^{-1}(\hat{R})$ throughout $\text{domain}(\pi)$.

Abort if any two polynomials collapse, i.e., there's a collision.

(d) Remove all records of R and $\mathfrak{R}$ from $\boldsymbol{R}$, $\boldsymbol{V}$ and λ.

(e) Return $(\hat{r}, s)$

Proposition B.5. *For an algorithm* D *interacting with* $\mathsf{SymbLazyEval}_2$ *or* $\mathsf{SymbLazyEval}_1$ *making at most* t *queries to the group oracle, let* $\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_1}$ *and* $\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_2}$ *denote the output of* D *when interacting with the corresponding oracle. Then, for every* D*,*

$$\left| \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_2} = 1\right] - \Pr_{\mathsf{D}}\left[\text{out}_{\mathsf{D}}^{\mathsf{SymbLazyEval}_1} = 1\right] \right| = 0$$

C 'Verifiable OLE' from OLE in the ROM

Protocol 'Verifiable OLE' from Single-Call Standard OLE in the ROM

Inputs. P_1 has input $x \in \mathbb{F}_q$, P_2 has input $a, b \in \mathbb{F}_q, X \in \mathbb{G}$ such that $a \neq 0$.

Random Oracle. Let $\mathtt{HS}_0 : \mathbb{G} \to \mathbb{F}_q^2$ denote a random oracle that outputs two field elements when taking a group element as input.

1. P_2 samples $\mu, \nu \xleftarrow{\$} \mathbb{F}_q$ and sets $a' \leftarrow a\mu$, $b' \leftarrow b\mu + \nu$, $Y' \leftarrow X^{a'} \cdot g^{b'}$.
2. In parallel:
   (a) P_1, P_2 call OLE on inputs x and (a', b'). Let y' denote P_1's output.
   (b) P_2 sends $(\alpha, \beta) \leftarrow \mathtt{HS}_0(Y') + (\mu, \nu)$ to P_1
3. P_1 outputs $y \leftarrow (y' - \nu') \cdot \mu'^{-1}$ for $(\nu', \mu') \leftarrow (\alpha, \beta) - \mathtt{HS}_0(Y')$

Protocol 10: 'Verifiable OLE' from Single-Call Standard OLE in the ROM

We show that Protocol 10 realizes the following functionality in the sense of the standard simulation-based paradigm [Can00; Can01; Gol09]. The random oracle is modeled as a *strict* global functionality in the sense of [CJS14; CDGLN18].

Functionality 'Verifiable OLE' Functionality

Inputs. P_1 has input $x \in \mathbb{F}_q$, P_2 has input $a, b \in \mathbb{F}_q, X \in \mathbb{G}$ such that $a \neq 0$.

Outputs.

(a) P_1 obtains $y \leftarrow ax + b$ if $g^x = X$ and $\perp$ otherwise.

(b) P_2 obtains nothing.

Functionality 1: 'Verifiable OLE' Functionality

Theorem C.1. *Protocol 10 realizes Functionality 1 in the random oracle model.*

Proof. **Corrupted** P_2. Write (a', b') for the pair used P_2 in the verifiable OLE call and (α, β) for the values sent to the simulated P_1 in Item 2b of Protocol 10. Set $(\nu', \mu') \leftarrow (\alpha, \beta) - \mathtt{HS}_0(X^{a'} \cdot g^{b'})$ and send $(\hat{a}, \hat{b}) \leftarrow (a', (b' - \nu')) \cdot \mu'^{-1}$ to the OLE oracle. The aforementioned process is equivalent to an honest execution of P_2 with inputs $(\hat{a}, \hat{b}, X)$. **Corrupted** P_1. Let $\hat{x}$ denote P_1's input to the verifiable OLE call. If $\hat{x} = x$, then send x to the OLE oracle, obtain y, sample $y', \mu, \nu \xleftarrow{\$} \mathbb{F}_q$ subject to $(y' - \nu) \cdot \mu^{-1} = y$, set $(\alpha, \beta) \leftarrow \mathtt{HS}_0(g^{y'}) + (\mu, \nu)$ and return (y', α, β) to P_1. Else, send $\perp$ to the functionality and return $(\hat{y}, \alpha, \beta) \xleftarrow{\$} \mathbb{F}_q^3$ to P_1. This concludes the simulation. To see why the simulation is indistinguishable from the real execution, notice that if P_1 uses the wrong input, by Proposition C.2, it holds that $a'x + b'$ (and thus $g^{a'x+b'}$) has full entropy given $a'\hat{x} + b'$ and x. Therefore, unless P_1 guesses $g^{a'x+b'}$, thanks to the random oracle, P_1's view bears no correlation to P_2's inputs. As such, by Proposition C.2, the distinguishing advantage is at most $1/q$. □

Proposition C.2. *Let $a', b' \xleftarrow{\$} \mathbb{F}_q$ and $y' \leftarrow a'x + b'$. For any $x, v \in \mathbb{F}_q$ and $\delta \in \mathbb{F}_q^*$, it holds that*

$$H_\infty(y' \mid a'(x + \delta) + b' = v) = \log(q).$$

Proof. Fix an arbitrary $u \in \mathbb{F}_q$ and deduce that

$$\begin{aligned}
\Pr\big[y' = u \mid a'(x+\delta) + b' = v\big] &= \Pr\big[a'(x+\delta) + b' - a'\delta - b' = u \mid a'(x+\delta) + b' = v\big] \\
&= \Pr\big[a'(x+\delta) + b' - a'\delta = u \mid a'(x+\delta) + b' = v\big] \\
&= \Pr\big[a'\delta = v - u \mid a'(x+\delta) + b' = v\big] \\
&= \Pr\big[a'\delta = v - u\big] = \frac{1}{q}
\end{aligned}$$

The last equality follows from the fact that a' and $a'(x + \delta) + b'$ are independent given that a' and b' are idependent. □