Firebolt: Fast, Lightweight, Malicious Non-Interactive Vector-OLE

Michael Adjedj* Geoffroy Couteau†‡ Nikolaos Makriyannis*

October 19, 2026

Abstract

We introduce a new maliciously-secure vector oblivious linear evaluation (VOLE), building upon a recent construction of (semi-honest) Paillier-based non-interactive VOLE. Our protocol is non-interactive (the sender can adaptively reveal entries of the VOLE using a single message to the receiver) and enjoys low communication (535 bytes for each VOLE entry) and low computation (around 5ms per party, single-threaded). Our protocol can be used as a drop-in replacement for less efficient VOLEs used in a variety of threshold schemes. We exemplify it by devising a new concurrently-secure two-round two-party ECDSA protocol that significantly improves over the state of the art. Along the way, we introduce several new strategies for proving shortness of plaintexts in Paillier ciphertexts or integer commitments, and we devise a modular framework for building threshold-ECDSA in a VOLE-hybrid model with concurrent security without requiring the use of (typically less efficient) UC-secure VOLE. All of these contributions are of independent interest.

*Fireblocks. {madjedj, nikos}@fireblocks.com

†CNRS, IRIF, Université Paris Cité. couteau@irif.fr

‡Work done while the author was working as a consultant for Fireblocks.

Contents

1 Introduction **1**
1.1 Applications . . . 2

2 Technical Overview **5**
2.1 RNIVOLE from DDLOG . . . 5
2.2 An optimized encrypted discrete logarithm proof . . . 7
2.3 Optimized KZG-based proof of shortness . . . 9
2.4 Shorter proofs via batched-NIM . . . 10
2.5 Threshold ECDSA from registered VOLE . . . 12

3 Preliminaries **13**
3.1 Augmented Standalone Security Model . . . 14
3.1.1 Real-world protocol execution. . . . 14
3.1.2 Ideal-world experiment. . . . 15
3.1.3 Security definition . . . 15
3.2 Standard Lemmas . . . 16
3.3 Cryptographic Assumptions . . . 17
3.4 Paillier-ElGamal Encryption . . . 18
3.5 Distributed Discrete Logarithm . . . 18
3.6 Standard zero-knowledge proofs . . . 19
3.6.1 Definitions. . . . 19
3.6.2 Relations and promise problems. . . . 19
3.6.3 Fiat–Shamir and non-interactive zero-knowledge proofs. . . . 20

4 Short Zero-Knowledge Proof of Encrypted Discrete Logarithm **21**
4.1 Proof of Pedersen Smallness (Π_{small}) . . . 21
4.1.1 Some helpful lemmas . . . 23
4.2 Proof of Pedersen / Paillier-ElGamal Consistency (Π_{dlcom}) . . . 24
4.3 Combined Proof of Encrypted Discrete Log (Π_{encdl}) . . . 25
4.4 Cost Summary . . . 27

5 Registered Non-Interactive VOLE from Non-Interactive Multiplication **28**
5.1 Defining Registered NIVOLE . . . 29
5.1.1 Security . . . 30
5.2 Extended Paillier-ElGamal . . . 31
5.3 A RNIVOLE from Paillier-Based NIM . . . 32

6 Security Analysis **34**
6.1 A variant of the power knowledge of exponent assumption . . . 34
6.2 Security of the Registered VOLE . . . 35
6.2.1 Proof of Theorem 6.3: corrupted sender. . . . 36
6.2.2 Proof of Theorem 6.3: corrupted receiver. . . . 42

7 Batch Registered Non-Interactive VOLE **45**
7.1 Generalized Paillier-ElGamal Encryption . . . 45
7.2 Generalized Pedersen / Paillier-ElGamal Consistency ($\Pi_{k\text{-dlcom}}$) . . . 46
7.3 Combined Generalized Proof of Encrypted Discrete Log ($\Pi_{k\text{-encdl}}$) . . . 47
7.4 A Batch-RNIVOLE from Batch-NIM . . . 49
7.5 Efficiency . . . 53
7.6 A generalization of k-KZG . . . 53
7.7 Security . . . 54

References **56**

A Discussions **63**
A.1 On the Use of Knowledge of Exponent Assumptions . . . 63
A.2 On the Use of Short-Exponent Assumptions . . . 64
A.3 Why Not Fischlin? . . . 65

B 2PC ECDSA from VOLE **65**
B.1 The Protocol . . . 65
B.2 Security Analysis . . . 67

C Realizing the Setup Functionality $\mathcal{F}_{\mathsf{setup}}$ **70**
C.1 Proof of KZG Parameters ($\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$) . . . 70
C.2 Concrete Setup Protocol . . . 70
C.3 Security of the Setup Protocol . . . 71

D Description of the Standard Zero-Knowledge Protocols **73**

E On the Equivalence of Power-SEI and Power-SEDL **75**
E.1 Setting and statements . . . 76
E.2 Preliminary lemmas . . . 76
E.3 Proof of Theorem E.1 (search-to-decision) . . . 78
E.4 Proof of Theorem E.2 (decision-to-search) . . . 80

1 Introduction

In this work, we consider the problem of designing efficient protocols for vector oblivious linear evaluation (VOLE) with low communication, minimal interaction, and malicious security. A VOLE over a field $\mathbb{F}$ involves a receiver, with an input $x \in \mathbb{F}$, and a sender with inputs $(a_i, b_i) \in \mathbb{F}^2$. The parties wish to reveal $a_i \cdot x + b_i$ to the receiver, and nothing more. In recent years, VOLE has emerged as a powerful primitive with numerous applications in secure computation and threshold cryptography.

Our main contribution is Firebolt, a new VOLE protocol with unparalleled efficiency and security features. Firebolt is:

Round optimal: it requires a single round from receiver to sender to set up common parameters, and a single message to publish an encryption of x (which can be sent with the setup message, or separately). The protocol operates in the plain model: no trusted parameters are required. Afterwards, for any number of inputs (a_i, b_i), the sender can reveal $a_i \cdot x + b_i$ using a single message to the receiver. VOLEs achieving this optimal communication pattern are known as non-interactive VOLEs [BDSS25] (NIVOLEs) and are of particular interest. In particular, they form a useful special case of reusable non-interactive secure computation (NISC) and have received some attention [CDIKLOV19].

Fast: on one core of a machine with AVX512 instructions (AWS c7i.xlarge instance), after a one-time (non-interactive) setup phase, our implementation takes about 4ms per party to publish and verify the encryption of x. Afterwards, each VOLE multiplication $a_i \cdot x + b_i$ takes about 5ms of computation per party. On a processor without AVX512, the numbers grow to about 8ms per party.

Lightweight: after a one-time non-interactive setup message from the receiver, publishing an encryption of x takes only 2.5kB of communication (using interaction, this can be reduced to 1.5kB). Afterwards, each VOLE multiplication takes only 552 bytes of communication.

Secure: Firebolt is proven secure against malicious adversaries running an arbitrary number of concurrent executions of the VOLE multiplication. The security of our protocol is proven in the random oracle model under a SNARK-type knowledge of exponent assumption together with factoring-related assumptions (such as DCR) and discrete logarithm-type assumptions over RSA groups (such as the hardness of computing d given $(g^{d^i})_{i \leq k}$ for a small k when d is a random short exponent).

In addition to the above, we enhance our VOLE protocol to support *batch evaluation* to achieve even lower per-VOLE communication at the cost of a slight increase in computation: when performing VOLE multiplications $a_i \cdot x + b_i$ (for the same x) in batches of size k, the per-VOLE communication from sender to receiver drops sharply as k increases (from 322 bytes/VOLE when $k = 2$ to 98 bytes/VOLE when $k = 15$).

Comparison to the state of the art (NIVOLE). Efficient, lightweight, and round-optimal malicious VOLE protocols are useful in many settings, in particular in the context of threshold cryptography. By replacing the underlying VOLE with our new protocol, we get significant improvements in a number of applications. NIVOLEs are also motivated by applications to

efficient reusable non-interactive secure computation [CDIKLOV19] and to efficient constructions of homomorphic secret sharing [HSSCD26]. Despite their many concrete applications, there are very few maliciously-secure non-interactive VOLEs in the literature: almost all VOLE constructions we are aware of either achieve only semi-honest security, or require more than a single round to publish an encryption of the receiver input or to send a VOLE entry. We report in Table 1 the performances of Firebolt compared to state of the art malicious NIVOLEs. We note that an additional advantage of our Paillier-based approach over protocols based on class groups, beyond the fact that assumptions over class groups have received comparably less scrutiny, is that achieving side-channel resistance via constant-time implementations only adds a minor overhead to the runtime (in particular, the VroomSSL library [LHD26], which we use, implements all operations in constant time with very little overhead). By contrast, constant-time implementations of class group operations are much more experimental and around two orders of magnitude slower [DGNP25].

Table 1: Comparison of maliciously-secure non-interactive VOLEs from the literature. The runtimes have been obtained by running the fastest available implementation on one core of an AWS c7i.xlarge instance with AVX512 support; numbers marked with * have been extrapolated from microbenchmarks. The value k refers to the number of VOLE entries transmitted at once (the cost is counted per entry). Communication (Comm.) and computation ignore the one-time cost of publishing an encryption of the receiver input. Constant-time indicates whether the protocol admits an efficient constant-time implementation, and straight-line indicates whether it provides straight-line extraction from the receiver message.

Construction	Type	Constant-time	Straight-line	Comm.	Computation
ABCGJM [ABCGJM26]	Paillier+GGM	✓	✗	2kB	48ms
LLZD [LLZD25]	Class groups	✗	✗	660B	50∼100ms[1]
Firebolt	Paillier	✓	✓	552B	11ms
Firebolt ($k = 5$)	Paillier	✓	✓	167B	20ms*
Firebolt ($k = 15$)	Paillier	✓	✓	98B	40ms*

[1] LLZD does not provide dedicated benchmarks for the NIVOLE underlying their n-party threshold ECDSA protocol. The numbers have been extrapolated from their ECDSA benchmarks. We ran the implementation provided in [LLZD25] on our AWS c7i.xlarge instance, but obtained much higher runtimes.

1.1 Applications

Our core motivation for designing improved VOLE protocols stems from threshold cryptography, the branch of secure computation that enables a group of users to collectively run cryptographic algorithms, concealing the secret material from subsets of parties that are not authorized to access it. Examples include threshold signatures [Lin17; KG20; KOR23; DKLs24; ABCGJM26], cryptosystems with threshold decryption [BCDLMOT24], or distributed pseudorandom functions [NPR99; DY05; DYY06].

We showcase the benefits of our improved VOLE by focusing on two-party ECDSA as a case study. ECDSA [JMV01] is currently the most commonly deployed digital signature scheme, used e.g. over major cryptocurrencies such as Bitcoin and Ethereum. Threshold ECDSA is perhaps the most widely used form of threshold cryptography in the real world due to its use in cryptographic custody, where two-party protocols for generating ECDSA signatures are employed to sign transactions

while ensuring policy compliance and avoiding single points of failure. By using our improved VOLE within the recent two-party ECDSA protocol from [ABCGJM26], and incorporating some improvements from [BCMRS26], we obtain a new two-party ECDSA protocol which improves over the state of the art in many aspects. As a side contribution of independent interest, because the analysis of [ABCGJM26] was not modular (it was tailored to their particular VOLE construction), we provide a new modular analysis that proves full concurrent security of the threshold signature scheme given any suitable instantiation of the VOLE.

Comparison with the state of the art. We provide in Table 2 a comparison of Firebolt to the most efficient two-party ECDSA protocols from the literature (we omit from the table a number of classical protocols such as [GG18; DKLs18; CGGMP20; CCLST20] when their efficiency is not competitive with more recent protocols). For the sake of readability, we ignore one-time setup costs in the table, but note that all protocols have comparable setup runtimes (in the 200ms∼1s range), with the exception of [BCMRS26] which requires a much heavier setup (in the 30s∼1min range).

As shown in the table, our two-party ECDSA is highly competitive with the state of the art, enjoying low communication and fast runtimes. Among all alternatives, only [DKLs24] and [Kon26] are faster, but both lack concurrent security (an important downside which we discuss below). Furthermore, while [BCMRS26] achieves comparable runtime and better communication, it comes with severe limitations: it has a very slow and complex setup (estimated to take dozens of seconds, and for which no implementation is currently available), very large key sizes (megabytes), and requires a combination of new assumptions (including a new "filtered" variant of the sparse LPN assumption that deserves further scrutiny, and a new correlation-robust hash function proven secure in the generic group model). Among all existing alternatives, we believe that Firebolt currently provides the best tradeoff between runtime, communication, interactivity, and security. We provide more details on Firebolt's benchmark in the table below:

	Setup		**DKG**		**Sign**	
Metric	Generate	Verify	Server	Client	Server	Client
Mean Time (ms)	816.5	879.1	4.9	3.8	6.0	5.5

On concurrent security. The column "concurrent" indicates whether the protocol remains secure when an unbounded number of threshold signing sessions can be executed concurrently. The protocols of Lindell [Lin17] and Kondi [Kon26] are known to be insecure in this model as their security proof requires the simulator to abort all sessions as soon as a failure is detected (which for many deployments cannot be reliably enforced). The DKLs protocol [DKLs24] is also insecure when more than 2ℓ concurrent sessions are run in parallel, where ℓ is some tunable security parameter that sets an upper bound on the number of concurrent sessions (typically $\ell = 128$). While this is not stated explicitly in their paper [DKLs24], the insecurity of DKLs under concurrent executions has been documented in [Pol23] (see also [Ash26] for some discussion on the issue).

We note that concurrent security is an important requirement of threshold ECDSA protocols, because real-world deployments often fail to enforce the necessary condition that all signing sessions must be aborted as soon as a failure is detected. The work of [MYG24] described explicit attacks on the Lindell protocol, when used concurrently, that applied to real-world deployments of the protocol. Variants of the attack can be shown to apply to [DKLs24] and [Kon26] as well.

Table 2: Comparison of two-party ECDSA signature schemes. All estimates refer to the costs (rounds, communication, computation) of the signing phase, and ignore the (one-time) setup and key generation cost. Runtimes for DKLs, ABCGJM, Trout, LLZD, and Firebolt have been obtained by running the fastest available implementation on one core of an AWS c7i.xlarge instance with AVX512 support. Estimates for Lindell and Kondi are taken from [ABCGJM26] and [Kon26] respectively: they have been obtained on a different machine and are not directly comparable, but provide an estimate of concrete computation costs. Paillier-based operations are executed using the VroomSSL library from [LHD26], and elliptic curve operations use the Bitcoin library libsecp256k1.

Construction	Type	Concurrent	Rounds	Communication	Computation
Lindell [Lin17]	Paillier	✗	4	0.9kB	12ms
DKLs [DKLs24]	OT	✗	3	115kB	4ms
ABCGJM [ABCGJM26]	Paillier+GGM	✓	2	2kB	48ms
Trout [DGNP25]	Class groups	✓	2	3.8kB	3349ms[1]
LLZD [LLZD25]	Class groups	✓	2	1963B	267ms[1]
Kondi [Kon26]	OT	✗	2	170B	2.2ms
BCMRS [BCMRS26][2]	LPN+GGM	✓	2	128B	12ms (est.)
Firebolt	Paillier	✓	2	616B	11ms
Firebolt ($k = 15$)	Paillier	✓	2	179B	40ms

[1] Trout's slow runtime is mainly due to the use of the eVRF from [BHLS25]. Faster runtimes can likely be obtained by leveraging instead the doubly enhanced unforgeability of ECDSA as in [ABCGJM26]. For LLZD, the entry is the runtime extrapolated from their paper. We obtained a much slower runtime of 5837ms when running the code of [LLZD25] on our AWS instance and conservatively reported their lower number.

[2] BCMRS does not provide an implementation of their scheme: the runtime is extrapolated from benchmarks of individual operations and ignores all other costs. BCMRS also trades its signing performance for a very slow and highly interactive setup (optimistically estimated around 45 seconds) and very large keys (around 3MB per party). In contrast, our full (non-interactive) setup takes around 850ms per party (where one party can reuse the same setup across an arbitrary number of key generations) and the key generation only requires storing a few kilobytes.

Other applications. Beyond our "flagship" application to threshold ECDSA, our VOLE can be readily used to improve a variety of other applications. In particular, and as noted in previous works [FHKRS25; BCMRS26; HLLXBA26], a VOLE protocol suffices for distributing BBS and BBS+ signatures [BBS04; CL04; ASM06; TZ23], Boneh-Boyen signatures [BB04], GOST signatures [DD13], SM2 signatures [ZLLZH24], and Dodis-Yampolskiy verifiable random functions [DY05]. In all of these applications, replacing the underlying VOLE with our new protocol yields significant improvements compared to the state of the art.

Organization. Section 2 provides a technical overview of our contribution. Standard preliminaries, including standard cryptographic assumptions, lemmas, and protocols, are introduced in Section 3. In Section 4, we introduce our new zero-knowledge proof of knowledge of a short encrypted discrete logarithm. Section 5 introduces formally the notion of (registered) NIVOLE and our enhanced stand-alone security model, and covers our main maliciously-secure NIVOLE protocol Firebolt. Section 6 contains the detailed security analysis of Firebolt. Section 7 extends our NIVOLE to the batch setting. Appendix A discusses our use of knowledge-of-exponent and short-exponent

assumptions, and compares our approach with the Fischlin transform. In Appendix B, we introduce a threshold ECDSA game and a protocol built directly from the RNIVOLE algorithms in a hybrid model where the setup and key-generation steps are idealized, and prove its stand-alone security (which, as we observe, can be lifted to UC security via known results). Appendix C presents our concrete protocol realizing the setup functionality $\mathcal{F}_{\mathsf{setup}}$, with the standard zero-knowledge protocols it relies on described in Appendix D. Eventually, Appendix E proves the equivalence of the power-SEI and power-SEDL assumptions.

2 Technical Overview

We provide an overview of our results in this section. To avoid disrupting the flow of the presentation, we assume that the reader is familiar with a few standard assumptions and primitives over RSA groups, such as Paillier encryption and its variant, the Paillier-ElGamal encryption (we refer the unfamiliar reader to Section 3.3 and Section 3.4 of the preliminaries). We recall that a VOLE over a field $\mathbb{F}_q$ is a protocol between a receiver (holding x) and a sender (holding (a, b)) which reveals $a \cdot x + b \bmod q$ (and nothing else) to the receiver. A NIVOLE is a VOLE where the receiver can publish an encryption of x using a single message, after which the sender can adaptively send him $a_i \cdot x + b_i \bmod q$ using a single message, for any number of (a_i, b_i).

For applications to group-based threshold cryptography (such as threshold BBS/BBS+ signatures, Boneh-Boyen signatures, GOST signatures, SM2 signatures, and Dodis-Yampolskiy VRFs), it is additionally required that the encrypted input x corresponds to the discrete logarithm of a public group element X (in the basis g of the underlying DLOG-hard group $\mathbb{G}$ of order q). We call *registered* NIVOLE (or RNIVOLE) a protocol that satisfies this feature (we think of publishing X as storing a value x in a register that can later be tied to the receiver input x used in a VOLE). In the rest of this paper, we focus solely on RNIVOLE, since it is the one required in most applications we are aware of (but we note that it is straightforward to extract a plain NIVOLE, that does not connect x to X, from our registered NIVOLE).

2.1 RNIVOLE from DDLOG

Our starting point is a known construction of non-interactive VOLE from [OSY21], which builds upon the seminal non-interactive oblivious transfer protocol from [BM90] and leverages the existence of efficient distributed discrete logarithms (DDLOG) over $\mathbb{Z}_N$, where N is an RSA modulus. The DDLOG algorithm is a deterministic polytime algorithm with the following property: given $\mathcal{E}_0, \mathcal{E}_1 \in \mathbb{Z}^*_{N^2}$ such that $\mathcal{E}_1 = (1+N)^x \cdot \mathcal{E}_0 \bmod N^2$ for some $x \in \mathbb{Z}_N$, it holds that $\mathsf{DDLOG}(\mathcal{E}_1) - \mathsf{DDLOG}(\mathcal{E}_0) = x \bmod N$. Given this algorithm, fix a modulus q and a group $\mathbb{G}$ of order q with generator g. Let $X \in \mathbb{G}$. The NIVOLE from [OSY21] proceeds as follows:

1. Setup: the receiver samples an RSA modulus N, an N-th power u over $\mathbb{Z}^*_{N^2}$ and an element $t = u^d \bmod N^2$ of the subgroup generated by u. She sends (N, u, t).

2. Encryption of the receiver input: the receiver with input $x \in \mathbb{F}_q$ (with $X = g^x$) samples $\gamma \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$ and sends a Paillier-ElGamal ciphertext $(\mathcal{E}_t, \mathcal{E}_u) = ((1+N)^x \cdot t^\gamma, u^\gamma) \bmod N^2$.

3. Sender message: the sender with input (a, b) samples $\rho \xleftarrow{\$} [2^{n_\rho}]$ (where n_ρ is a suitable size bound) and computes a commitment $\mathcal{C} = t^a \cdot u^\rho \bmod N^2$. Then, the sender sets $V = \mathcal{E}_t^a \mathcal{E}_u^\rho \bmod N^2$ and $z = \mathsf{DDLOG}(V) + b \bmod q$. He sends $(\mathcal{C}, z)$.

4. Output: the receiver computes $z - \mathsf{DDLOG}(\mathcal{C}^\gamma) \bmod q$.

Above, we call $\mathcal{C}$ the *DFO commitment*, where DFO stands for Damgård-Fujisaki-Okamoto, who introduced a very similar integer commitment scheme in [FO97; DF02] (albeit over $\mathbb{Z}_N$ rather than $\mathbb{Z}_{N^2}$). For correctness, observe that $V = (1+N)^{a \cdot x}(t^\gamma)^a(u^\gamma)^\rho = (1+N)^{a \cdot x} \cdot \mathcal{C}^\gamma$, hence $z - \mathsf{DDLOG}(\mathcal{C}^\gamma) = b + (\mathsf{DDLOG}(V) - \mathsf{DDLOG}(\mathcal{C}^\gamma)) = b + a \cdot x \bmod N$. This holds modulo N, but when $a \cdot x$ is sufficiently smaller than N, the DDLOG algorithm can be tweaked to ensure that it also holds over the integers with overwhelming probability, hence it correctly produces a VOLE modulo q after the parties locally reduce their shares mod q. Security for the receiver holds under the IND-CPA security of Paillier ElGamal, and sender security is statistical. As observed in previous works [ABCGJM26; HSSCD26], the concrete efficiency of this baseline protocol can be significantly improved by letting d, γ, and ρ be *short* exponents. Concretely, under the short-exponent discrete logarithm assumption, d, γ can be taken of size 2ℓ (where ℓ is a security parameter, e.g. $\ell = 128$). Then, for $\mathcal{C}$ to statistically hide a, it suffices to take ρ of size $|a| + 2\ell + \nu$, where ν is a statistical security parameter (e.g. $\nu = 40$). When implementing this NIVOLE using the state-of-the-art VroomSSL library for fast exponentiations over RSA groups, the full protocol runs in a couple of milliseconds on one core of a laptop.

Malicious Security. The main technical challenge, to which we devote most of our effort, lies in enhancing this baseline protocol to be secure against malicious adversaries. Specifically, we seek a solution which adds a minimal communication and computation overhead to phase 3, the sender message phase (which will be executed many times in common applications), and (to a lesser extent) to phase 2, the receiver input encryption phase. In addition, because common applications might require multiple concurrent executions of the VOLE encryption and of the VOLE sender message, we aim for a straight-line simulation of phases 2 and 3 against a malicious adversary. To this end, we introduce several new highly optimized straight-line extractable NIZKs tailored to the baseline NIVOLE. We believe that some of the ideas introduced in our optimized proofs are of independent interest. Concretely, we need to enforce three things to get malicious security:

- (N, u, t) should be correct (i.e., N must be a product of two large primes, u must be an N-th power, and t must be in the subgroup generated by u)
- $(\mathcal{E}_t, \mathcal{E}_u)$ should be a valid Paillier-ElGamal encryption of a *short* plaintext x (shortness is required to ensure that $x \cdot a \ll N$, which is a necessary condition for the DDLOG to produce valid subtractive shares over the integers) equal to the discrete logarithm of X in base g.
- $\mathcal{C}$ should be a valid DFO commitment to a *short* value a.

For the first proof, of correctness of (N, u, t), we rely on standard Fiat-Shamir-compiled NIZKs adapted from [ABCGJM26]. As we neither aim for straight-line extraction nor very high performances (since this is a one-time setup message), we do not invest more effort in optimizing this step.

Both remaining proofs (of short encrypted discrete logarithm and short commitment opening) should be efficient and straight-line extractable. We briefly comment on the challenges before we move on to our solutions:

Short Paillier-ElGamal plaintext. The main issue is that range proofs are expensive. One option is to use a cut-and-choose approach (or equivalently a Σ-protocol with bit challenges), as

in [Lin17], but this yields large proofs. Another option is to use integer commitment schemes, as in [LN18; ABCGJM26; XLXADY24], but this requires the verifier to sample their own RSA modulus and prove that it is well-formed; not only is this very costly, but it also breaks the non-interactive structure of the NIVOLE. A third option is to decompose x bit-by-bit, commit to the bit decomposition over an elliptic curve, and combine a range proof based on Bulletproofs for the commitment with a simple Σ-protocol to show that the bit decomposition reconstructs to the correct plaintext. This is again expensive on the prover side.

Short DFO commitment opening. It is easy to build an efficient range proof (with slack) via a Σ-commitment, leveraging the unknown order to extract over the integers. Here, the main issue is straight-line extraction, which is crucially required in applications where the VOLE should remain secure against concurrent executions, as in applications to threshold signatures. One option is to rely on the Fischlin transform [Fis05], but this adds a large overhead. Another option is to re-commit to a separately over another group (e.g., an elliptic curve), prove plaintext equality, and leverage the generic group model as in [ABCGJM26] to obtain straight-line extraction, but the reliance on generic groups is undesirable.

In both cases, we introduce new approaches to overcome the issues mentioned. Our strategy is to introduce careful relaxations of the target statements combined with careful modifications of the baseline RNIVOLE protocol such that (1) the relaxed proofs can be achieved much more efficiently, and (2) with these modifications, the relaxed statements are sufficient to achieve malicious security. The resulting protocol we obtain achieves very strong performances; in fact, our straight-line extraction strategy for the DFO commitment yields a proof that is much shorter and computationally more efficient than the standard range proof with slack *even ignoring the issue of straight-line extraction.*

2.2 An optimized encrypted discrete logarithm proof

A baseline approach. In the baseline RNIVOLE, the goal is to prove the following statement: given $(X, \mathcal{E}_t, \mathcal{E}_u)$, there exists (x, γ) such that (1) x is short, (2) $(\mathcal{E}_t, \mathcal{E}_u) = ((1+N)^x t^\gamma, u^\gamma) \bmod N^2$, and (3) $X = g^{x \bmod q}$. As a starting point, consider the following approach:

- Break x into smaller chunks $x_1 \dots x_\tau$. Commit to $(x_1, \dots, x_\tau, \gamma)$ using Pedersen commitments over a suitable group (in our concrete instantiation, we pick the same curve as X, the ECDSA curve secp256k1, but the approach can freely use a different group).

- Use a Σ-protocol with bit challenges to prove knowledge of $(x_1, \dots, x_\tau, \gamma)$. The response of this Σ-protocol has the form $(e_i x_1 + r_{i,1}, \dots, e_i x_\tau + r_{i,\tau}, e_i \gamma + \rho_i)$, where i ranges from 1 to the number k of repetitions (typically 128 if we want to compile the proof into a NIZK), statistically masking the terms $e_i x_j$ *over the integers*. The number of chunks τ of x is chosen to ensure that $e_i x_j + r_{i,j} < q$ ($\tau = 2$ suffices in our instantiation).

- Separately, use a "one-shot" Σ-protocol to show that $x = x_1 || \dots || x_\tau$ is the value encrypted in $(\mathcal{E}_t, \mathcal{E}_u)$ and in the exponent of X.

Because of the use of bit challenges and statistical masking, the rewinding-based extraction recovers the chunks $(x_1, \dots, x_\tau)$ (and therefore x) over the integers, and guarantees a small x. However, the proof sketched above has three downsides:

1. Due to the use of bit challenges, it requires many parallel repetitions and involves a heavy communication
2. Extracting the witness requires rewinding
3. Due to the use of a one-shot Σ-protocol for proving correctness of the Paillier-ElGamal encryption, we do not actually extract a "proper" witness. Rather, the extractor obtains a witness for a "slack instance": bounded value f (a challenge difference) and a pair (x, γ') such that $(\mathcal{E}_t^f, \mathcal{E}_u^f) = ((1+N)^{xf} \cdot t^{\gamma'}, u^{\gamma'}) \bmod N^2$. However, we cannot in general deduce from these relations a valid representation of $(\mathcal{E}_t, \mathcal{E}_u)$ (indeed, such a representation might not exist).

Tolerating slack instances. Using bit challenges as well for the Paillier-ElGamal correctness would solve Item 3, but it would require exchanging several Paillier-ElGamal ciphertexts and computing several exponentiations for each repetition, which would be extremely inefficient. Rather, our strategy is to bite the bullet and adapt the NIVOLE protocol to remain secure even when a malicious receiver can use a Paillier-ElGamal ciphertext from a slack instance. In short, we devise a careful simulation strategy that succeeds given a relaxed witness, provided that the simulator receives the leakage $a \bmod f$. Then, we modify the NIVOLE to replace a with $a' = a + q \cdot r$ for a random value r of size at least $|r| = |f| + \nu$. As $q > f$, this guarantees that the leakage $a' \bmod f$ is statistically independent from a (and can therefore be simulated), yet using a' in the NIVOLE does not harm correctness as a' is short and $a' = a \bmod q$.

Straight-line extraction via factorization. Turning to Item 2, we observe that during the setup (where using rewinding is acceptable, as it is a one-time setup), the simulator can extract the factorization of N from the corrupted receiver, and use it to extract x directly from the Paillier-ElGamal ciphertext. This does not suffice, as our simulation also requires extracting γ; we therefore augment the proof with a Paillier-ElGamal encryption of γ, together with a proof that it matches the random coins of the encryption of x, which we can also extract using the factorization. We note that for several applications, straight-line extraction might not be mandatory during the encryption phase (e.g. in threshold signatures, this happens during key generation, which happens only once in a model where a single pair of parties is considered). In these settings, our proof can be made slightly more efficient by removing this additional encryption.

Reducing repetitions via pb-bit challenges. Eventually, we address Item 1 using an original and surprisingly effective strategy. We modify the NIVOLE protocol such that the encryption of x is replaced with an encryption of $[\alpha^{-1}]_q \cdot x$, where $[\cdot]_q$ denotes reduction modulo q, and $\alpha = \alpha(\mathsf{pb})$ is the smallest integer divisible by every integer between 2 and 2^{pb} (in our instantiations, we use $\mathsf{pb} = 5$ and α is a 48-bit integer). Then, both parties multiply α inside the Paillier-ElGamal ciphertext, obtaining an encryption of $x' = \alpha \cdot [\alpha^{-1}]_q \cdot x$ (which remains short). In the proof, instead of using bit challenges, we use pb-bit challenges, which reduces the number of repetitions by a factor pb. This has the effect of extracting a "slack witness" $y \cdot [z^{-1}]_q$, where y is short and z is at most pb bits long. In turn, this implies that x' (the plaintext obtained after multiplication by α) is equal to $\alpha \cdot y \cdot [z^{-1}]_N$, which is small because α cancels out the denominator z by construction. We further enhance our proof using standard batching techniques. Our final proof, compiled into a NIZK via

Fiat-Shamir, yields a straight-line extractable proof of short encrypted discrete logarithm with small communication of 1.84kB, and a little over 4ms of computation per party.

2.3 Optimized KZG-based proof of shortness

Next we discuss the proof that $\mathcal{C} = t^a \cdot u^\rho \bmod N$ can be opened to a short a. As we mentioned, while a Σ-protocol *a la* Damgård-Fujisaki [DF02] works, it does not provide straight-line extractability. Instead of using existing methods to add straight-line extractability, we employ a completely different approach that yields much better efficiency.

KZG view of the DFO commitment. The main idea is to view $\mathcal{C}$ as a KZG commitment [KZG10] to the linear polynomial $L(X) = a \cdot X + \rho$ (recall that KZG commitments commit to polynomials and enable openings of evaluations at arbitrary points). To prove that we know an opening (a, ρ) where a is short, we combine two ingredients:

- We prove shortness by receiving a random challenge e and sending a KZG opening of $\mathcal{C}$ to $L(e)$. Since e is random, checking that $L(e)$ is short suffices to guarantee that a and ρ are both simultaneously short. Recall that under the short-exponent discrete logarithm assumption, we can let the receiver use a small d and prove during setup that $t = u^d \bmod N^2$ for a small d. This in turn makes it possible for the sender to use a small ρ (of size $|a| + |d| + \nu$) to statistically mask a in $\mathcal{C}$, which allows checking smallness of $a \cdot e + \rho$ directly. Note also that verifying a KZG opening is possible given d (KZG commitments were originally designed over pairing-based groups, but pairings are not necessary in a designated-verifier setting).
- To extract (a, ρ), we leverage the fact that a KZG opening is a value π such that $\pi^d = \mathcal{C} \cdot u^{-L(e)} \cdot \pi^e \bmod N^2$. Under a variant of the power knowledge-of-exponent assumption [Gro10] which was introduced in [BFHK24], finding such a relation entails the existence of an extractor that can recover y such that $u^y = \pi$. From $y, L(e)$, using the previous expression, one can extract (a, ρ).

Plugging the leakage. However, this does not quite work, because opening the KZG commitment to $L(e)$ creates leakage. Indeed, if the receiver can compute discrete logarithms over $\mathbb{Z}^*_{N^2}$ (which it can do by forcing $\phi(N)$ to have many small factors), it obtains $a \cdot d + \rho$ from $\mathcal{C}$, and $L(e) = a \cdot e + \rho$, from which it immediately gets a. To fix this, we must add another randomizer inside $\mathcal{C}$. We therefore modify the setup to include an additional term $s = t^d = u^{d^2} \bmod N^2$. This has the side effect of requiring a slightly stronger "power-DCR" assumption for the IND-CPA security of the Paillier-ElGamal scheme extended with this additional term (a similar assumption was used in [ARS24]). Now, we set $\mathcal{C} = s^a \cdot t^{\rho_0} \cdot u^{\rho_1} \bmod N^2$, for randomizers ρ_0, ρ_1 of size to be determined later (we call $\mathcal{C}$ a DFO/KZG commitment). Given a challenge e, we compute the opening $c = a \cdot e^2 + \rho_0 \cdot e + \rho_1$, and the proof $\pi = t^a \cdot u^{a \cdot e + \rho_0} \bmod N^2$. The receiver checks that $\pi^{d-e} = \mathcal{C} \cdot u^{-c} \bmod N^2$ and that c is short. The adversary now gets to see $a \cdot d^2 + \rho_0 \cdot d + \rho_1$ and $a \cdot e^2 + \rho_0 \cdot e + \rho_1$, which can be shown to statistically hide a for a suitable choice of size for ρ_0, ρ_1. The knowledge of exponent assumption guarantees that the simulator can extract $(a, a \cdot e + \rho_0)$ from π, from which it can reconstruct (a, ρ_0, ρ_1).

Compatibility with the NIVOLE. In doing so, however, we have modified $\mathcal{C}$, and this modification propagates to the NIVOLE. The natural solution is to extend the encryption of x to $(\mathcal{E}_s, \mathcal{E}_t, \mathcal{E}_u) = ((1+N)^x s^\gamma, t^\gamma, u^\gamma) \bmod N^2$ and define $V = \mathcal{E}_s^a \mathcal{E}_t^{\rho_0} \mathcal{E}_u^{\rho_1} \bmod N^2$. However, we have so far ignored the slack of our encrypted discrete logarithm proof, covered in Section 2.2. It causes the following issue: we are not guaranteed anymore that $V = (1+N)^{a \cdot x} \cdot \mathcal{C}^\gamma \bmod N^2$. Instead, we have $V = (1+N)^{a \cdot x} \cdot (v_s^a v_t^{\rho_0} v_u^{\rho_1}) \cdot \mathcal{C}^\gamma \bmod N^2$, where the group generated by (v_s, v_t, v_u) has small order. This is a problem, because during simulation, the simulator (who could extract γ) obtains $w = a \cdot x + b \bmod q$ from the RVOLE functionality, simulates the DFO commitment $\mathcal{C}$, and must simulate $z = \mathsf{DDLOG}(V) + b \bmod q$ with $\tilde{z} = w + \mathsf{DDLOG}(\tilde{v} \cdot \mathcal{C}^\gamma)$, where $\tilde{v} = (v_s^a v_t^{\rho_0} v_u^{\rho_1})$. But this value depends on the private input a of the honest sender, which the simulator cannot know!

The solution is to leverage the fact that $\tilde{v}$ lives in a small group (of order $\tilde{\omega}$) by adding noise to a. Concretely, we replace a in $\mathcal{C}$ with $a' = a + q \cdot r$, where r is chosen large enough to ensure that $a' \bmod \tilde{\omega}$ leaks no information (statistically) about a. At the same time, $a' = a \bmod q$, hence the VOLE remains correct. Then, Sim can simulate $\tilde{v}$ without a and we are done (the full statistical indistinguishability argument is slightly more involved but follows this outline). This is the *cheating subgroup problem* of [ABCGJM26] (their Section 2.3.5): there, it is handled by relaxing the parameters relation, whereas we absorb it by randomizing a. In the NIVOLE protocol, we make the proof non-interactive by deriving e as a hash of $\mathcal{C}$ using a random oracle. This creates another slight complication: when setting $e = \mathsf{H}(\mathcal{C})$, the simulator needs to program the random oracle to emulate $\mathcal{C}$, which is possible only if $\mathcal{C}$ has never been queried before. But if u generates a group of very small order (which the receiver can enforce), $\mathcal{C}$ has small entropy and can be guessed with good probability; we fix this minor issue by adding a salt to the hash. The final sender message is $(\mathsf{salt}, \mathcal{C}, \pi, c, z)$.

Further optimizations. We introduce three additional tweaks to the above approach. First, we observe that we can save a few exponentiations by exchanging the roles of π and $\mathcal{C}$: that is, we treat π as the DFO/KZG commitment and $\mathcal{C}$ as the proof of short opening. Indeed, $\pi = t^{a'} \cdot u^{a' \cdot e + \rho_0} \bmod N^2$ *is* a valid DFO commitment to a'! With this change, we can go back to using a shorter encryption of x $(\mathcal{E}_t, \mathcal{E}_u) = ((1+Nx) \cdot t^\gamma, u^\gamma)$ and define $V = \mathcal{E}_t^{a'} \mathcal{E}_u^{a'e + \rho_0}$ using one fewer exponentiation. This changes nothing for extraction. The second tweak is to provide the N-th roots s_0, t_0, u_0 as part of the setup parameters, and set $\mathcal{C} = s_0^{a'} \cdot t_0^{\rho_0} \cdot u_0^{\rho_1} \bmod N$ and $\pi = t_0^{a'} \cdot u_0^{a' \cdot e + \rho_0} \bmod N$. This reduces the size of $\mathcal{C}, \pi$ by a factor 2 (and makes exponentiations cheaper since they are now over $\mathbb{Z}_N$) at the (mild) cost of requiring the receiver to compute $\pi^{N \cdot \gamma} \bmod N^2$ instead of π^γ, a larger exponentiation. Eventually, the last tweak is to let the receiver reverse-sample $\mathcal{C}$ from π, by setting $\mathcal{C} = u_0^c \cdot \pi^{d-e} \bmod N$ and checking that $e = \mathsf{H}(\mathsf{salt}, \mathcal{C})$. With this final change, the total communication is reduced to $(\mathsf{salt}, e, \pi, c, z)$, which amounts to 535 bytes using standard parameters.

We believe that our new zero-knowledge proof can also be integrated in other threshold schemes (using similar adaptations to accommodate for the slack of the proof) such as Lindell's protocol [Lin17] and the protocol of [ABCGJM26], and would result in significant speedups for their key generation. Both protocols are widely deployed and used commercially.

2.4 Shorter proofs via batched-NIM

The batched baseline. We also consider the batch setting, where after publishing its encrypted input x, the receiver wants to receive size-k batches of VOLE entries $(a_i \cdot x + b_i)_{i \le k}$. This is a

common setting for threshold signatures, where signing in batches reduces latency and is desirable in various scenarios. We leverage a recent "succinct" version of the NIVOLE from [OSY21], introduced concurrently in [BCMPR24] and [ARS24]. We rely on a version of the power-DCR-based succinct NIVOLE from [ARS24], which we recall below (we directly integrate optimizations such as the use of short exponents for simplicity):

- Setup: the receiver samples an RSA modulus N, a secret key $d \in [2^{2\ell}]$, an N-th power u_{-k}, and sets $u_i \leftarrow u_{-k}^{d^{k+i}} \bmod N^2$ for $i = -k$ to k. She sends $(N, u_{-k}, \ldots, u_k)$.
- Encryption of x: the receiver samples $\gamma \xleftarrow{\$} [2^{2\ell}]$ and sends an extended ciphertext $(\mathcal{E}_{-k}, \ldots, \mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k) = (u_{-k+1}^{\gamma}, \ldots u_0^{\gamma}, (1+N)^x \cdot u_1^{\gamma}, u_2^{\gamma}, \ldots u_k^{\gamma}) \bmod N^2$.
- Sender message: given k inputs $(a_i, b_i)_{i \le k}$, sample $\rho \xleftarrow{\$} [2^{n_\rho}]$ and compute a commitment $\mathcal{C} = u_0^{\rho} \cdot \prod_{i=1}^{k} u_i^{a_i} \bmod N^2$. For $i = 1$ to k, set $V_i = \mathcal{E}_{-i+1}^{\rho} \cdot \prod_{j=1}^{k} \mathcal{E}_{j-i+1}^{a_j} \bmod N^2$ and $z_i \leftarrow \mathsf{DDLOG}(V_i) + b_i \bmod q$. Return $(\mathcal{C}, z_1, \ldots, z_k)$.
- Output: the receiver computes $w_i = z_i - \mathsf{DDLOG}(\mathcal{C}^{\gamma \cdot d^{-i+1}} \bmod N^2) \bmod q$ for $i = 1$ to k.

Correctness. Correctness can be checked routinely, but in short: $V_1 = \mathcal{E}_0^{\rho} \cdot \prod_{j=1}^{k} \mathcal{E}_j^{a_j} = \mathcal{C}^{\gamma} \cdot (1+N)^{x \cdot a_1}$: the inner product in the exponent aligns x with a_1. Then, every V_i for $i > 1$ shifts by 1 how the vector $(\rho, a_1, \ldots, a_k)$ is aligned with the $\mathcal{E}_j$'s to align x with a_i, but with respect to a "shifted basis" $(u_{1-i}, \ldots, u_{k+1-i})$. Then, $\mathcal{C}^{d^{-i+1}}$ leverages the structure of the u_j's to convert $\mathcal{C}$ into a commitment to $(\rho, a_1, \ldots, a_k)$ with respect to the shifted basis. A core advantage of this variant is that the size of the sender message $(\mathcal{C}, z_1, \ldots, z_k)$ grows minimally with k: transmitting each additional VOLE entry requires sending a single element $z_i \in \mathbb{F}_q$, which is clearly optimal. This comes at the cost of a k-fold increase of the (one-time) encryption of x.

Malicious security via two KZG openings. It remains to enhance this generalization to the malicious setting. Most of the approaches we described previously extend readily to the batch setting, with one important subtlety: in the KZG-based proof of knowledge of a representation of $\mathcal{C}$, it does not suffice anymore to provide a single KZG opening π for an evaluation c at a random point e. To see this, observe that the relation $\pi^{d-e} = \mathcal{C} \cdot u_{-1}^{-c}$ guarantees, via the knowledge of exponent assumption, that a corrupted sender must know a representation of π in the basis $(u_{-k}, \ldots, u_{k-1})$, from which the simulator gets in turn a representation of $\mathcal{C}$ in the basis $(u_{-k}, \ldots, u_k)$. However, this does not suffice to simulate! The reason is that ensuring the DDLOG correctness during the simulation requires extracting a representation of $\mathcal{C}$ over the basis $(u_0, \ldots, u_k)$ to leave "room" for computing the shifted commitments $\mathcal{C}^{d^{1-i}}$ from the u_j's. Using the terminology of polynomial commitments, it does not suffice to show that $\mathcal{C}$ commits to a polynomial P that evaluates to c on e; we must additionally show that $P(X)$ is of the form $X^k \cdot Q(X)$ (it does not have "low" coefficients), where Q is a polynomial of degree at most k. To do so, we let the sender include two KZG commitments instead of one: we let c be defined as $Q(e)$ instead of $P(e)$ and write $Q(X) - Q(e) = (X - e) \cdot Q'(X)$. Then, we commit to both $Q'(X)$ with π and $X^k \cdot Q'(X)$ with π', which the receiver can verify by checking $\pi^{d^k} = \pi'$. This effectively ensures simultaneously that $\mathcal{C}$ commits to a polynomial with short coefficients, but also that it is of the form $X^k \cdot Q(X)$. Formalizing extraction requires introducing a new but natural generalization of the power knowledge of exponent assumption, which belongs to the same Uber-Knowledge family of assumptions defined in [BFHK24] (and therefore holds unconditionally in the generic group model).

Concrete sender message. Combining our enhanced KZG-based methodology with the optimizations introduced in the non-batch setting, the final sender message is $(\mathsf{salt}, e, \pi, \pi', c, z_1, \ldots, z_k)$: it contains a single fixed prefix $(\mathsf{salt}, e, \pi, \pi')$ (of size 0.77kB for standard parameters), an integer $c = \rho + \sum_{i=1}^{k} a_i \cdot e^i$ which grows as $(k + O(1)) \cdot \ell$, and an $\mathbb{F}_q$ value for each proof. A last technicality is that to enforce the shortness condition required for DDLOG correctness, c must be shorter than the RSA modulus, which puts an upper bound on the maximum supported batch size. For a standard choice of parameters, this amounts to a bound $k \leq 15$. For $k = 15$ and using standard parameters ($\log N = 3072$, $\ell = 128$, $\log q = 256$), the per-entry communication cost is only 106 bytes.

2.5 Threshold ECDSA from registered VOLE

Modular treatment without UC overkill. We demonstrate the efficiency benefits of our new (registered) NIVOLE by building a two-round two-party ECDSA protocol with concurrent security. Our construction simply plugs Firebolt within the protocol from [ABCGJM26], integrating a small optimization from [BCMRS26] along the way (which shaves two elements of $\mathbb{G}$). However, the security analysis of [ABCGJM26] is monolithic: it proves security of the full protocol, and does not allow a modular treatment of the underlying RNIVOLE. Most previous works [Lin17; DGNP25; LLZD25] suffer from the same limitation. Some works adopt a more modular approach [DKLs24; BCMRS26] but place themselves in the UC setting, which restricts the instantiation to use UC-secure NIVOLE. We observe that this is overkill: UC-secure NIVOLE can be significantly less efficient (in particular, our construction is not UC secure, due to its use of knowledge of exponent assumptions). Furthermore, we introduce a slight enhancement to the stand-alone security model which provides a better tradeoff: we demonstrate that a RNIVOLE secure in our enhanced stand-alone model suffices to yield threshold signatures with full concurrent security. Furthermore, we show that Firebolt is secure in this variant, and the proof of security is essentially the same as the proof of security in the (regular) stand-alone model.

The enhanced stand-alone model. The aim of the enhanced model is to capture reductions to *games*—reactive functionalities that interact with the attacker—for protocols in which certain components are idealized via helper functionalities that are atomic and stateless. Concretely, we augment the plain stand-alone model in three ways: (1) the target functionality is a game, so that game-based guarantees such as unforgeability are captured directly; (2) the attacker is allowed to write on the parties' public-input tape at the beginning of each round, which captures concurrency and adversarial selection of inputs; and (3) we incorporate the random oracle model, domain-separated so that composition works cleanly. This enhancement captures precisely the required security features for common threshold schemes, including in particular threshold signatures (where the adversary adaptively supplies public messages to be signed in the security game) or threshold PRF/VRF (where the adversary adaptively supplies public inputs to the PRF/VRF).

Composition. In the end, we prove security of a hybrid protocol with respect to this game, and we invoke the modular composition theorem to say that we can replace all the helper functionalities with real interactive protocols. Concretely, we prove that our two-party ECDSA protocol, built directly from the RNIVOLE algorithms, realizes an ECDSA-with-presignatures game (which is known to be unforgeable [ABCGJM26; GS22]) in a hybrid model where the setup and key-generation steps are idealized helper functionalities; the composition theorem then lets us instantiate the latter with their concrete (Firebolt-based) realizations while retaining the game-based guarantee.

Compatibility with knowledge assumptions. A key benefit of remaining in the stand-alone setting is that the model coexists with knowledge-of-exponent assumptions, whose extractors require the code and random coins of the machine producing the relevant group elements—the well-known obstacle to using such assumptions in the UC framework, where the environment's coins are external and cannot be handed to the extractor. In our model, the extractor is only ever invoked inside the security reduction, where the simulator implements everything surrounding the attacker—including the helper functionalities, whose coins are then part of the simulator's own random tape (see Remark 3.5). This is precisely what allows instantiating the VOLE with Firebolt, whose security relies on knowledge of exponent assumptions. We believe that this framework is of independent interest and could facilitate the use of more efficient cryptographic protocols (which often lack a proof in the UC model) within threshold schemes in a more modular way.

Concurrency within a session. Our model (with its public input tape) captures concurrency naturally, and we use it to show that the VOLE register and extend operations admit arbitrary concurrency, both in the VOLE game of Section 5 and in the ECDSA game of Section B. We recall that the RNIVOLE breaks down into three steps: init (a receiver operation), register (a receiver operation), and extend (a sender operation). The only step that does not support full concurrency is init, a one-time operation performed ahead of time to realize the remaining VOLE operations: our concrete instantiation of the setup requires rewinding and, as such, does not support concurrency directly. The register and extend steps, in contrast, enjoy straightline extractability, and we show that they can be scheduled in whatever order the attacker pleases. This feature is inherited by the ECDSA protocol: signing consists (essentially) of a single extend operation, and such operations can be interleaved and scheduled at will. The bottom line is that, within a single session (parameterized by a given ECDSA public key), there is no limitation on the number of concurrent signing executions.

Concurrency across sessions. What about concurrency across distinct sessions, i.e. two different public keys? This is captured at a different level. Specifically, our security analysis reduces a single session to a game (in this case, an ECDSA oracle). This game comes with a winning condition for the attacker — creating a forgery — that is efficiently testable, since the forgery verifies under the public key. As such, concurrency at this level is proven via a guessing argument: the reduction guesses which session the forgery targets, injects the target key there and applies the simulation strategy (including rewinding where necessary), and simulates all other sessions by simply running the code of the honest parties on fresh randomness. If the distribution of the public keys is the same in the simulated and in the actual execution, this simulation technique yields a forgery in the underlying unforgeability game, at the cost of a loss polynomial in the number of instances.

To sum up, our model places no restriction on concurrency where it matters: any number of public keys, and any number of signings per public key, can be run concurrently and scheduled at the attacker's choosing.

3 Preliminaries

Notations. Given an integer k, we write $[k]$ for the set $\{0, \ldots, k-1\}$ and $[\cdot]_k$ for the function that reduces its input modulo k.

Parameters. The following parameters are used throughout the paper. ℓ is the computational security parameter and ν the statistical security parameter: ν bounds the statistical distance incurred by each masking ("smudging") argument and by the simulation of each zero-knowledge proof. n_{rsa} denotes the bitlength of the RSA modulus: $\mathsf{SampleRSA}(1^\ell)$ returns $N \in [2^{n_{\mathrm{rsa}}-1}, 2^{n_{\mathrm{rsa}}})$. The intended instantiation is $(\ell, \nu, n_{\mathrm{rsa}}) = (128, 40, 3072)$; we require $n_{\mathrm{rsa}}/2 > \ell + \nu + 1$ and $\log(q+1) \le 2\ell$ for the elliptic-curve order q. The zero-knowledge proofs with small per-repetition challenge spaces are repeated until they reach a target soundness level: s denotes the target soundness parameter (soundness error $\approx 2^{-\mathsf{s}}$), while r_{mod} and $r_{\mathsf{kzg_{PE}}}$ denote the number of parallel repetitions of Π_{mod} and $\Pi_{\mathsf{kzg_{PE}}}$ (cf. the costs paragraph of Section 3.6). Within the shortness proofs of Section 4, the committed integer is split into τ chunks of κ bits each; a single repetition of Π_{small} uses challenges in $\pm 2^{\mathsf{pb}}$ (and is repeated $\lfloor \mathsf{s}/\mathsf{pb} \rfloor$ times, with masking slack $\varepsilon' := \lceil \log_2 \tau \rceil + \mathsf{pb} + \nu$), and a single repetition of Π_{dlcom} uses challenges in $\pm 2^{\mathsf{s}_0}$ (and is repeated $\lceil \mathsf{s}/\mathsf{s}_0 \rceil$ times, with masking slack $\varepsilon := \mathsf{s}_0 + \nu$). The public scalar α of Π_{encdl} is a product of small prime powers, chosen so that every denominator in $\pm 2^{\mathsf{pb}+1}$ that the extractor of Π_{small} can produce divides α (cf. Protocol 3). Finally, n_x denotes the bound on the encrypted plaintext in the slack instances of the encrypted-discrete-logarithm promise problem (Definition 4.7).

3.1 Augmented Standalone Security Model

We use a simulation-based security definition for multiparty protocols. We deviate from textbook definitions so we take a moment to describe the model and the security definition we use.

3.1.1 Real-world protocol execution.

We work in the standalone synchronous ROM setting of [Lin16; Gol04]. Parties are oracle-aided PPT interactive Turing machines, pairwise connected by authenticated synchronous channels and equipped with access to a random oracle RO via dedicated query/answer tapes. Each party also has a dedicated *public-input tape* that the adversary may write to. The honest parties' initial inputs and the static corruption pattern are fixed at the start of the execution; the adversary $\mathcal{A}$ controls the corrupted parties.

A protocol Π in the $\mathcal{F}_1, \ldots, \mathcal{F}_k$-hybrid model is a tuple of parties with access to functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$. As per the usual model, functionality calls are atomic: input and output are exchanged within the same round, and the functionality performs no other task during that round. In each synchronous round, $\mathcal{A}$ may write a value to the public-input tape (delivered identically to all parties); $\mathcal{A}$ does not communicate with the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$ beyond choosing the inputs of the corrupted parties. Each party then reads its incoming tapes, locally computes (including RO queries), and writes outgoing messages and functionality inputs. At termination, each honest party writes its output.

Remark 3.1 (The purpose of the public input tape). The public input tape allows the adversary to write arbitrary values under the restriction that these values are common to all parties. The purpose of this convention is twofold: (1) it specifically captures certain capabilities of the attacker that are not captured by the standard model such as the ability to choose messages for signing in the game-based unforgeability experiment for the signature schemes that we are interested in; (2) it allows to capture the notion of concurrency inside the protocol, as the attacker inside the protocol can instruct the parties which executions to run or which subfunctionalities to call.

Adversarial Model. The attacker corrupts up to all but one of the parties; corrupted parties are controlled maliciously (the attacker may send arbitrary messages on their behalf) and rushingly (in each round, the attacker chooses the corrupted parties' messages after seeing the honest parties' messages). In addition, in our model the attacker may write arbitrary values to the public-input tape. Finally, as is standard in the dishonest-majority, security-with-abort setting, whenever a hybrid call $\mathcal{F}_i$ delivers output to the corrupted parties, the attacker may—after seeing those outputs—abort the functionality, preventing the honest parties from receiving their own outputs.

We write $\mathrm{REAL}^{\Pi,\mathcal{A}}_{1^\ell,\vec{x},z}$ for the joint distribution of the honest parties' outputs and the output of $\mathcal{A}$ (wlog its view) in the real experiment, where ℓ is the security parameter, $\vec{x}$ is the tuple of secret inputs of the honest parties at the beginning of the protocol, and z is the attacker's auxiliary input.

3.1.2 Ideal-world experiment.

We prove security by showing that the real protocol emulates a trusted party computing a functionality $\mathcal{G}$. However, the functionality $\mathcal{G}$ in the ideal experiment operates differently from the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$ in the real experiment. $\mathcal{G}$ is a *game*: a reactive functionality that interacts with the adversary, whereas each $\mathcal{F}_i$ is a stateless secure-function-evaluation functionality. We focus on the following class of functionalities.[1]

Definition 3.2 (Game)**.** A functionality $\mathcal{G}$ is a *game* if the honest parties provide inputs only at the start of the execution: once their initial inputs are delivered, every subsequent input to $\mathcal{G}$ is supplied by the adversary. Equivalently, the honest parties' outputs are a deterministic function of their initial inputs, $\mathcal{G}$'s random tape, and the adversary's adaptively-chosen inputs.

In the ideal experiment, honest parties deliver their initial inputs to $\mathcal{G}$; subsequently, $\mathcal{G}$ interacts directly with the ideal adversary $\mathcal{S}$ for all further inputs; at termination, $\mathcal{G}$ delivers outputs to the parties and $\mathcal{S}$ produces an output. There is no public-input tape in the ideal world: since $\mathcal{G}$ is a game (Definition 3.2), $\mathcal{G}$ computes the honest parties' outputs from their initial inputs and from $\mathcal{S}$'s adaptively-chosen inputs (where the adaptivity depends on whatever leakage $\mathcal{G}$ exposes to $\mathcal{S}$).

We write $\mathrm{IDEAL}^{\mathcal{G},\mathcal{S}}_{1^\ell,\vec{x},z}$ for the joint distribution of the honest parties' outputs and $\mathcal{S}$'s output in the ideal experiment.

3.1.3 Security definition

Definition 3.3 (Computational indistinguishability)**.** A *distribution ensemble* indexed by a set I is a sequence of random variables $X = \{X(i)\}_{i\in I}$, where each $X(i)$ takes values in $\{0,1\}^*$. Two distribution ensembles $X = \{X(\ell,i)\}_{(\ell,i)\in\mathbb{N}\times I_\ell}$ and $Y = \{Y(\ell,i)\}_{(\ell,i)\in\mathbb{N}\times I_\ell}$ are *computationally indistinguishable*, written $X \approx Y$, if for every non-uniform PPT distinguisher D there exists a negligible function $\mathrm{neg}(\cdot)$ such that for all $\ell \in \mathbb{N}$ and $i \in I_\ell$,

$$\left|\Pr[D(1^\ell, X(\ell,i)) = 1] - \Pr[D(1^\ell, Y(\ell,i)) = 1]\right| \le \mathrm{neg}(\ell).$$

Definition 3.4 (Secure realization)**.** A protocol Π *securely realizes* a functionality $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that

$$\left\{\mathrm{REAL}^{\Pi,\mathcal{A}}_{1^\ell,\vec{x},z}\right\}_{(\ell,\vec{x},z)} \approx \left\{\mathrm{IDEAL}^{\mathcal{G},\mathcal{S}}_{1^\ell,\vec{x},z}\right\}_{(\ell,\vec{x},z)}.$$

[1] In the conference submission (and its rebuttal), games were called *adversary-reactive functionalities*; only the name has changed.

Remark 3.5 (Compatibility with knowledge assumptions). Our proofs rely on knowledge-of-exponent assumptions, whose extractors require the code and random coins of the machine producing the relevant group elements. This is the well-known reason such assumptions do not fit the UC framework: the environment's coins are external and cannot be handed to the extractor. In our stand-alone model the situation is different. The extractor is only ever invoked inside the security reduction, where the entire experiment surrounding $\mathcal{A}$ — the emulated honest parties, the (lazily sampled, domain-separated) random oracle, and the helper functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$ — is implemented by the simulator. In particular, while each $\mathcal{F}_i$ hides its coins from the parties in the real execution, the $\mathcal{F}_i$ do not exist in the ideal world: the simulator is in charge of emulating them, so their coins are part of the simulator's own random tape. The extractor is thus simply fed the coins used to implement the attacker's view.

Remark 3.6 (Modular composition). We note that this formalization allows us to invoke Canetti's modular composition theorem [Can00]: if a hybrid protocol $\Pi^{\mathcal{F}}$ securely realizes a game $\mathcal{G}$ in the $\mathcal{F}$-hybrid model, and ρ securely realizes the (non-game) functionality $\mathcal{F}$, then the composition Π^{ρ} securely realizes $\mathcal{G}$.

3.2 Standard Lemmas

Lemma 3.7 (Splitting Lemma [PS96; PS00]). *Let $A \subset X \times Y$ such that $\Pr[(x,y) \in A] \geq \varepsilon$. For any $\varepsilon' < \varepsilon$, if one defines $B = \{(x,y) \in X \times Y \mid \Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon - \varepsilon'\}$, then it holds that:*

$$(i)\ \Pr[B] \geq \varepsilon', \quad (ii)\ \forall (x,y) \in B, \Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon - \varepsilon', \quad (iii)\ \Pr[B \mid A] \geq \varepsilon'/\varepsilon.$$

Lemma 3.8 (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 1.*

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

3.3 Cryptographic Assumptions

Some of the assumptions below are reproduced verbatim from [ABCGJM26]. We denote by $\mathsf{SampleRSA}(1^\ell)$ an algorithm that returns a strong RSA modulus $N = p \cdot q$ where $p = q = 3 \bmod 4$, $\gcd(p-1, q-1) = 2$, $(p-1)/2, (q-1)/2$ are prime, and $|p| = |q| \geq n_{\mathrm{rsa}}/2 - 1$.

Assumption 3.9 (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

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

Assumption 3.10 (Power-SEDL over $\mathbb{Z}_N^*$)**.** Define k-Power-SEDL (PSEDL_k) such that $(N, t, [t^x]_N, \ldots, [t^{x^k}]_N) \xleftarrow{\$} \mathsf{PSEDL}_k(1^\ell)$ 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 k-power discrete-log (Power-SEDL) assumption holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N, t, t_1, \ldots, t_k) \xleftarrow{\$} \mathsf{PSEDL}_k(1^\ell)} \left[x_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, t, t_1, \ldots, t_k) \text{ s.t. } t^{x_0} = t_1 \bmod N \right] \leq \mu(\ell).$$

Assumption 3.11 (Power-SEI over $\mathbb{Z}_N^*$)**.** Define distribution ensemble PSEI_k such that $(N, t, ([t^{\alpha x^i + (1-\alpha) y^i}]_N)_{i \leq k}, \alpha) \xleftarrow{\$} \mathsf{PSEI}_k(1^\ell)$ 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 the *$k$-power small-exponent indistinguishability (Power-SEI) assumption holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr_{(N, t, (t_1, \ldots, t_k), \alpha) \xleftarrow{\$} \mathsf{PSEI}_k(1^\ell)} \left[\alpha_0 \xleftarrow{\$} \mathsf{A}(1^\ell, N, t, (t_1, \ldots, t_k)) \text{ s.t. } \alpha_0 = \alpha \right] \leq \frac{1}{2} + \mu(\ell).$$

The 1-power SEDL and 1-power SEI assumptions are called SEDL and SEI respectively; their equivalence was proved in [ABCGJM26] (Appendix A). In Section E, we generalize this result to the k-power setting, giving a formal treatment of the reduction (which is only sketched in [ABCGJM26]) with explicit error bounds:

Corollary 3.12 (Corollary E.3)**.** *k-Power-SEDL implies k-Power-SEI. Under the factoring assumption (Assumption 3.9, itself implied by* DCR*), the two assumptions are equivalent.*

Assumption 3.13 (QR)**.** Define QR such that $(N, [\rho^{2\alpha} \cdot c^{1-\alpha}]_N, \alpha) \xleftarrow{\$} \mathsf{QR}(1^\ell)$ for $N \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $\rho \xleftarrow{\$} \mathbb{Z}_N^*$, $c \xleftarrow{\$} \mathbb{Z}_N^*$ and $\alpha \xleftarrow{\$} \{0,1\}$. We say that *quadratic residuosity (QR) holds true* if for every PPTM A, there exists a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

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

Assumption 3.14 (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

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

3.4 Paillier-ElGamal Encryption

Definition 3.15 (Paillier-ElGamal encryption)**.** The Paillier-ElGamal encryption $\mathsf{PE} = (\mathsf{PE.KeyGen}, \mathsf{PE.Enc}, \mathsf{PE.Dec})$ consists of the following algorithms.

1. $\mathsf{PE.KeyGen}(1^\ell)$. On input $\ell \in \mathbb{N}$ in unary,
   - Sample $(N, n_{\mathrm{rsa}}; p, q) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$.
   - Sample $u_0 \xleftarrow{\$} \mathbb{J}_N$, $d \xleftarrow{\$} \mathbb{Z}_{\varphi(N)}$, and set $t_0 = u_0^d \bmod N$, $u = u_0^{2N} \bmod N^2$, $t = t_0^{2N} \bmod N^2$.

   Return $(\mathsf{pk_{PE}} = (N, u_0, u, t_0, t);\ \mathsf{sk_{PE}} = (p, q, d))$.
2. $\mathsf{PE.Enc}_{\mathsf{pk_{PE}}}(m; \gamma)$. On input $m \in \mathbb{Z}_N$, $\mathsf{pk_{PE}} = (N, u_0, u, t_0, t)$, and random coins $\gamma \xleftarrow{\$} \mathbb{Z}_N^*$, output
$$(\mathcal{E}_t, \mathcal{E}_u) = ((1 + mN) \cdot t^\gamma, u^\gamma) \bmod N^2.$$
3. $\mathsf{PE.Dec}_{\mathsf{sk_{PE}}}(\mathcal{E}_t, \mathcal{E}_u)$. On input $\mathcal{E}_t, \mathcal{E}_u \in \mathbb{Z}_{N^2}^*$ and $\mathsf{sk_{PE}} = (p, q, d)$, output $m = \mathsf{DLOG}_N(\mathcal{E}_t / \mathcal{E}_u^d \bmod N^2)$, where $\mathsf{DLOG}_N(X)$ returns x if $X = (1 + xN) \bmod N^2$, and $\perp$ otherwise.

3.5 Distributed Discrete Logarithm

Definition 3.16. The procedure $\mathsf{DDLOG}(\mathcal{E})$ is defined as follows: on input $\mathcal{E} \in \mathbb{Z}_{N^2}^*$,

- Parse $\mathcal{E}$ as $\mathcal{E} = \mathcal{E}_L + N \cdot \mathcal{E}_R$ with $(\mathcal{E}_L, \mathcal{E}_R) \in \mathbb{Z}_N^2$. If $\mathcal{E}_L \notin \mathbb{Z}_N^*$, return $\perp$.
- Else, return $\mathcal{E}_R \cdot \mathcal{E}_L^{-1} \bmod N$.

We have the following lemma from [OSY21] showing that the DDLOG algorithm turns divisive shares of $(1+N)^x$ over $\mathbb{Z}_{N^2}^*$ into subtractive shares of x over $\mathbb{Z}_N$:

Lemma 3.17. *Fix any $x \in \mathbb{Z}_N$. Let $\mathcal{E}_0 \in \mathbb{Z}_{N^2}^*$ and let $\mathcal{E}_1 := (1+N)^x \cdot \mathcal{E}_0 \bmod N^2$. Then if $\mathsf{DDLOG}(\mathcal{E}_0) \neq \perp$, it holds that $\mathsf{DDLOG}(\mathcal{E}_1) \neq \perp$ and*

$$\mathsf{DDLOG}(\mathcal{E}_1) - \mathsf{DDLOG}(\mathcal{E}_0) = x \bmod N.$$

3.6 Standard zero-knowledge proofs

This section collects the standard zero-knowledge notions and proofs used throughout the paper. We first recall the terminology of relations, languages, and promise problems; we otherwise assume familiarity with the basic notions of interactive proofs (completeness, public-coin protocols, statistical distance). We do not define soundness or knowledge extraction generically: all our proofs target promise problems, and the guarantee obtained from two (or more) accepting transcripts is stated per protocol, as extraction of a witness in the protocol-specific slack set. We then give the relations and promise problems underlying the three setup proofs: Π_{mod} and Π_{fac}, both taken from [ABCGJM26], and $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$, a generalization of the DFO parameters proof of [ABCGJM26]. We conclude with the protocols themselves and their soundness and zero-knowledge guarantees.

3.6.1 Definitions.

Definition 3.18 (Relation and language)**.** Given a set $\mathcal{I}$ of *problem instances* and a set $\mathcal{W}$ of *witnesses*, a *relation* is a subset $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{W}$ which defines a *language* $\mathcal{L} \subseteq \mathcal{I}$ by the equivalence

$$\mathfrak{x} \in \mathcal{L} \iff \exists\, \mathfrak{w} \in \mathcal{W} : (\mathfrak{x}, \mathfrak{w}) \in \mathcal{R}.$$

Definition 3.19 (Promise problem)**.** For a language $\mathcal{L}$ defined for some relation $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{W}$, a *promise problem* is a pair of disjoint sets $(\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}}) \subseteq \mathcal{I}^2$ such that $\mathcal{I} \setminus \mathcal{L}_{\mathsf{no}} = \mathcal{L}$ for some language $\mathcal{L} \subseteq \mathcal{I}$. Elements in $\mathcal{L}_{\mathsf{yes}}$ and $\mathcal{L}_{\mathsf{no}}$ are referred to as "yes" and "no" instances, respectively. Every promise problem implicitly defines a "slack" set $\mathcal{L}_{\mathsf{slack}} = \mathcal{I} \setminus \mathcal{L}_{\mathsf{no}}$ of instances for which the underlying relation holds. To simplify notation, promise problems will be defined by specifying the slack and yes instances.

We only fix the following convention: a proof for a promise problem $(\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}})$ is δ-*HVZK* if the transcript of an honest execution on any yes instance can be simulated in PPT within statistical distance δ (*perfect* HVZK when $\delta = 0$).

3.6.2 Relations and promise problems.

The proofs Π_{mod} and Π_{fac} from [ABCGJM26] prove well-formedness of an RSA modulus through the following two promise problems.

Definition 3.20 (Well-formed modulus language)**.** Define the *Well-formed modulus language* mod such that:

$$\mathsf{mod} = \left\{ (\ell, N) \in \mathbb{N}^2 \;\middle|\; \begin{array}{l} N = pq \in [2^{n_{\mathrm{rsa}}-1}, 2^{n_{\mathrm{rsa}}}),\ \gcd(N, \varphi(N)) = 1, \\ p, q \in \mathsf{PRIMES} \text{ and } (p, q) = (3, 7) \bmod 8 \end{array} \right\}.$$

The underlying instance-witness pair is specified as $(\mathfrak{x}, \mathfrak{w}) = ((\ell, N), (p, q))$.

Definition 3.21 (Small factors)**.** The *Small Factors* promise problem fac has statement $(\ell, N) \in \mathbb{N}^2$.

- *Yes instances.* Tuples for which $N = uv \in [2^{n_{\mathrm{rsa}}-1}, 2^{n_{\mathrm{rsa}}})$ with $u, v \in \pm 2^{n_{\mathrm{rsa}}/2}$.
- *Slack instances.* Tuples for which $N = uv \in [2^{n_{\mathrm{rsa}}-1}, 2^{n_{\mathrm{rsa}}})$ with $u, v \in \pm 2^{n_{\mathrm{rsa}}/2+\ell+\nu}$.

Finally, the proof $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$ of correctness of the KZG parameters (Protocol 10, Section D) targets the following promise problem.

Definition 3.22 (KZG parameters)**.** The *KZG parameters* promise problem $\mathsf{kzg}_{\mathsf{PE}}$ is defined as follows. For tuples $(N, (u_{-k+1+i})_{i=1}^{2k-1}) \in \mathbb{N} \times (\mathbb{Z}_N^*)^{2k-1}$, define:

- *Yes instances.* $\exists d \in \pm 2^{2\ell}$ s.t. for $i \in [2k-1]$: $u_{-k+1+i} = u_{-k+1}^{d^i} \bmod N$.
- *Slack instances.* $\exists d \in \pm 2^{2\ell+\nu+1}$ s.t. for $i \in [2k-1]$: $u_{-k+1+i} = u_{-k+1}^{d^i} \bmod N$.

3.6.3 Fiat–Shamir and non-interactive zero-knowledge proofs.

This paper relies on three standard zero-knowledge proofs:

- Π_{mod}, which proves membership in the well-formed modulus language mod (Definition 3.20);
- Π_{fac}, which proves the small factors promise problem fac (Definition 3.21);
- $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$, which proves the KZG parameters promise problem $\mathsf{kzg}_{\mathsf{PE}}$ (Definition 3.22).

Π_{mod} and Π_{fac} are taken from [ABCGJM26], while $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$ generalizes the DFO parameters proof of [ABCGJM26] from independent bases to the chained bases $u_{-k+1+i} = u_{-k+1}^{d^i}$, proving all links of the chain with a single shared response. All three are public-coin; their full descriptions are deferred to Section D (Protocols 8, 9 and 10, respectively).

All proofs are made non-interactive following the Fiat–Shamir paradigm: Given a public-coin protocol Π, we denote by Π^{FS} the resulting non-interactive proof, and write $\pi \xleftarrow{\$} \Pi^{\mathrm{FS}}(\mathrm{aux}, \mathfrak{x}; \mathfrak{w})$ for proof generation and $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}}(\mathrm{aux}, \mathfrak{x}, \pi) \in \{0,1\}$ for verification. Here, $\mathrm{aux} \in \{0,1\}^*$ denotes a context string — the session identifier, the parties' identities, and all public values fixed so far — which is prepended to every random-oracle query of the Fiat–Shamir transform, for domain separation.

We conclude with the security claims for the three proofs, using the notation of the corresponding protocols. The claims for Π_{mod} and Π_{fac} are proven in [ABCGJM26]; the claim for $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$ directly follows the proof of the DFO parameters proof of [ABCGJM26].

Lemma 3.23 (HVZK)**.** Π_{mod} *is perfect HVZK with respect to the language* mod *(Definition 3.20),* Π_{fac} *is* $(2^{-\nu+1})$*-HVZK with respect to the promise problem* fac *(Definition 3.21), and* $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$ *is* $(2^{-\nu})$*-HVZK with respect to the promise problem* $\mathsf{kzg}_{\mathsf{PE}}$ *(Definition 3.22).*

Lemma 3.24 (Soundness of Π_{mod} [ABCGJM26])**.** *If* N *is not a product of two primes* p_1, p_2 *with* $(p_1, p_2) = (3, 7) \bmod 8$ *and* $\gcd(N, \varphi(N)) = 1$*, then for every (unbounded) prover strategy the verifier of Protocol 8 accepts with probability at most* $1/3$*, and at most* $1/4$ *whenever* $\gcd(N, \varphi(N)) \neq 3$ *(this last condition is efficiently checkable).*

Lemma 3.25 (Special soundness of Π_{fac} [ABCGJM26])**.** *Let* $\mathsf{trans} = ((p_0, \widetilde{P}, \widetilde{Q}, \widetilde{A}, \widetilde{B}, \widetilde{C}), e, (z_1, z_2, \lambda_1, \lambda_2, w))$ *and* $\mathsf{trans}' = ((p_0, \widetilde{P}, \widetilde{Q}, \widetilde{A}, \widetilde{B}, \widetilde{C}), e', (z_1', z_2', \lambda_1', \lambda_2', w'))$ *denote two accepting transcripts of Protocol 9 with identical first message and* $e \neq e'$*. Then, there exists a PPT extractor* E *such that at least one of the following holds true.*

1. $\alpha \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* $g^\alpha = h \bmod p_0$.
2. $(u, v) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *such that* (u, v) *is a witness for a slack instance of* fac *(Definition 3.21).*

Lemma 3.26 (Special soundness of $\Pi_{\mathsf{kzg_{PE}}}$)**.** *Given two accepting transcripts* $((A_{-k+1+j})_{j=1}^{2k-1}, e, z)$ *and* $((A_{-k+1+j})_{j=1}^{2k-1}, e', z')$ *of Protocol 10 with identical first message and* $e \neq e'$*, the exponent* $d := (z - z') \cdot (e - e')$ *is a witness for a slack instance of* $\mathsf{kzg_{PE}}$ *(Definition 3.22). In particular, the verifier accepts a no instance with probability at most* $1/2$*..*

The slack bound $\pm 2^{2\ell+\nu+1}$ of Definition 3.22 accounts precisely for this extraction: $d = (z-z')\cdot(e-e')$ from two accepting transcripts with the same first message and $e \neq e'$ (no division is needed since $|e - e'| = 1$), where $z, z' \in \pm 2^{2\ell+\nu}$, hence $d \in \pm 2^{2\ell+\nu+1}$.

Costs. With Fiat–Shamir and r_{mod} (resp. $r_{\mathsf{kzg_{PE}}}$) parallel repetitions, achieving soundness error $4^{-r_{\mathsf{mod}}}$ (resp. $2^{-r_{\mathsf{kzg_{PE}}}}$), the proofs communicate $r_{\mathsf{mod}} \cdot (2n_{\mathrm{rsa}} + 2)$ bits for $\Pi^{\mathrm{FS}}_{\mathsf{mod}}$, roughly $5 \log p_0 + n_{\mathrm{rsa}} + 2\ell + 2\nu$ bits for $\Pi^{\mathrm{FS}}_{\mathsf{fac}}$, and $r_{\mathsf{kzg_{PE}}} \cdot (2\ell + \nu + 1)$ bits for $\Pi^{\mathrm{FS}}_{\mathsf{kzg_{PE}}}$ (recomputing the first messages from the verification equations). For $(n_{\mathrm{rsa}}, \ell, \nu) = (3072, 128, 40)$ and $(r_{\mathsf{mod}}, r_{\mathsf{kzg_{PE}}}) = (64, 128)$, this amounts to 49.2kB, 2.6kB, and 4.8kB, respectively. Verification is dominated by the r_{mod} full-length exponentiations $z^N \bmod N$ of Π_{mod} and the $3\, r_{\mathsf{kzg_{PE}}}$ short exponentiations of $\Pi_{\mathsf{kzg_{PE}}}$.

4 Short Zero-Knowledge Proof of Encrypted Discrete Logarithm

This section formalizes the proof of short encrypted discrete logarithm outlined in Section 2.2. We build the proof in three steps. We first introduce Π_{small}, a proof that a tuple of Pedersen commitments opens to small values. We then introduce Π_{dlcom}, a proof that a tuple of Pedersen commitments is consistent with a Paillier-ElGamal ciphertext. Finally, we combine the two into Π_{encdl}, our proof of encrypted discrete logarithm. We conclude with a summary of the concrete costs in Section 4.4.

4.1 Proof of Pedersen Smallness (Π_{small})

Protocol Pedersen Smallness

Common Input. $g, h \in \mathbb{G}$ and $Y_1, \ldots, Y_\tau \in \mathbb{G}$ such that $Y_j = g^{y_{j-1}} \cdot h^{r_{j-1}}$ for $\{y_j\}_{j=0}^{\tau-1} \in (\pm 2^\kappa)^\tau$ and $\{r_j\}_{j=0}^{\tau-1} \in \mathbb{F}_q^\tau$.

Operations. (Let $\varepsilon' = \lceil \log_2 \tau \rceil + \mathsf{pb} + \nu$. Repeated $\lfloor \mathsf{s}/\mathsf{pb} \rfloor$ times for soundness $\approx 2^{-\mathsf{s}}$.)

1. P samples $a \xleftarrow{\$} \pm 2^{\kappa+\varepsilon'}$ and $\rho \xleftarrow{\$} \mathbb{F}_q$, and sends
$$A \;=\; g^a \cdot h^\rho \;\in\; \mathbb{G}.$$
2. V sends $e_1, \ldots, e_\tau \xleftarrow{\$} \pm 2^{\mathsf{pb}}$ independently.
3. P returns z and w such that
$$z \;=\; a + \sum_{j=1}^{\tau} e_j\, y_{j-1}, \qquad w \;=\; \rho + \sum_{j=1}^{\tau} e_j\, r_{j-1} \bmod q.$$

Verifier's Output. Accept if $z \in \pm 2^{\kappa+\varepsilon'}$ and

$$g^z \cdot h^w \;=\; A \cdot \prod_{j=1}^{\tau} Y_j^{e_j} \;\in\; \mathbb{G}.$$

Protocol 1: Proof of Pedersen Smallness Π_{small}.

Proposition 4.1 (Extraction for Π_{small})**.** *Assume there exists some prover* P^* *that succeeds in convincing the verifier* V *in* Π_{small} *with probability at least* λ*. Then, there exists an extractor* E *that interacts with* P^* *and, with probability* $(\lambda/2)^{\tau+2} - 2 \cdot 2^{-\mathsf{s}}$*, outputs one of the following:*

1. $(c_i, d_i, r_i)_{i\in[\tau]}$ *such that* $c_i \in \pm 2^{\kappa+\varepsilon'+1}$*,* $d_i \in \pm 2^{\mathsf{pb}+1}$*,* $r_i \in \mathbb{Z}_q$*, and* $Y_i = g^{c_i \cdot d_i^{-1}} \cdot h^{r_i}$*.*
2. $x \in \mathbb{Z}_q$ *such that* $g^x = h$*.*

Furthermore, the complexity of E *boils down to* $\tau + 2$ *executions of* P^* *run in parallel, and thus the running time of* E *equals that of* P^**.*

Proof. Via the splitting lemma (Lemma 3.7), there exists an extractor that interacts with P^* and, with probability at least $(\lambda/2)^{\tau+2}$, obtains valid transcripts for the same first message A_i with different challenges $e_1^{(i)}, \ldots, e_\tau^{(i)}$, where i ranges over the repetitions of the proof. For each repetition i, let M_i be the $(\tau+1) \times (\tau+1)$ matrix whose k-th row ($k = 1, \ldots, \tau+1$) is $(1, e_1^{(k)}, \ldots, e_\tau^{(k)})$, the challenges from the k-th transcript in repetition i. Assuming that at least one of the matrices is invertible (the failure event is handled at the end of the proof), we can solve for the vector $\vec{y} = (y_0, \ldots, y_{\tau-1})$ of the discrete logarithms of the Y_i's and for the vector $\vec{r} = (r_0, \ldots, r_{\tau-1})$ of the randomizers, as well as the decommitment to $A_i = g^{a_i} h^{\rho_i}$ for $a_i = z_i - \sum_{j=1}^{\tau} e_j^{(i)} y_{j-1}$ and $\rho_i = w_i - \sum_{j=1}^{\tau} e_j^{(i)} r_{j-1} \bmod q$. If the solution is in range (i.e., satisfies case 1 of the proposition) we are done. Else, the last transcript is an additional accepting transcript for the same first messages $A_1, \ldots, A_{\lfloor \mathsf{s}/\mathsf{pb} \rfloor}$ with fresh challenges $e_1'^{(i)}, \ldots, e_\tau'^{(i)}$ and responses $(z'^{(i)}, w'^{(i)})$ per repetition i. For each repetition i, since the last transcript accepts, $g^{z'^{(i)}} h^{w'^{(i)}} = A_i \cdot \prod_j Y_j^{e_j'^{(i)}}$, and since the extracted witnesses satisfy the same equation, $g^{z'^{(i)}} h^{w'^{(i)}} = g^{a_i + \sum_j e_j'^{(i)} y_{j-1}} h^{\rho_i + \sum_j e_j'^{(i)} r_{j-1}}$. If $(z'^{(i)}, w'^{(i)}) \neq (a_i + \sum_j e_j'^{(i)} y_{j-1},\ \rho_i + \sum_j e_j'^{(i)} r_{j-1} \bmod q)$ for some i, then $g^{z'^{(i)} - (a_i + \sum_j e_j'^{(i)} y_{j-1})} = h^{(\rho_i + \sum_j e_j'^{(i)} r_{j-1}) - w'^{(i)}}$ with nontrivial exponent mod q, yielding $\log_g h$. It remains to deal with the case where the obtained matrices are non-invertible or the last transcript is consistent with the extracted witness. Consider the following experiment, which is identical to the extraction strategy:

1. Make $\tau + 1$ runs of the protocol. If the obtained transcripts are not valid or the induced matrices are all not invertible, output 0 and halt.
2. Run the protocol one more time, and if the last transcript is valid and consistent with the extracted witness (i.e., $z'^{(i)} = a_i + \sum_j e_j'^{(i)} y_{j-1}$ and $w'^{(i)} = \rho_i + \sum_j e_j'^{(i)} r_{j-1} \bmod q$ for all i), output 1, else output 0.

By the splitting lemma and Lemma 4.3, the probability that extraction occurs in step 1 is at least $(\lambda/2)^{\tau+1} - 2^{-\mathsf{s}}$. Second, conditioned on the experiment still running and the first message of the proof being "good", the probability that the experiment outputs 1 is at least $\lambda/2 - 2^{-\mathsf{s}}$, where the $2^{-\mathsf{s}}$ loss comes from Lemma 4.4. Putting everything together, the probability that the experiment terminates in 1 is at least $(\lambda/2)^{\tau+2} - 2 \cdot 2^{-\mathsf{s}}$. □

Batch Verification. As noted above, the above proof is repeated $\lfloor \mathsf{s}/\mathsf{pb} \rfloor$ times to achieve soundness $2^{-\mathsf{s}}$. From the verifier's perspective, this consists of checking $\lfloor \mathsf{s}/\mathsf{pb} \rfloor$ equations of the form $g^{z_{j-1}} h^{w_{j-1}} = A_j \cdot \prod_{i=1}^{\tau} Y_i^{e_{i,j}}$ for $j = 1, \ldots, \lfloor \mathsf{s}/\mathsf{pb} \rfloor$. The following claim shows that these checks can be batched into a single check of the same form, with only a small loss in soundness.

Claim 4.2 (Batch verification soundness)**.** *Let $\mathsf{s}_0 \in \mathbb{N}$ and $(z_j, w_j, e_{1,j}, \ldots, e_{\tau,j}, A_j)_{j=1}^{\lfloor \mathsf{s}/\mathsf{pb} \rfloor}$ be transcripts of the above protocol for tuple $(Y_i)_{i=1}^{\tau}$. Assume that for some j, $g^{z_{j-1}} h^{w_{j-1}} \neq A_j \cdot \prod_{i=1}^{\tau} Y_i^{e_{i,j}}$. Then, for $\rho_1, \ldots, \rho_{\lfloor \mathsf{s}/\mathsf{pb} \rfloor} \xleftarrow{\$} \pm 2^{\mathsf{s}_0}$ independently, it holds that*

$$\Pr\left[g^{\sum_j \rho_j z_{j-1}} h^{\sum_j \rho_j w_{j-1}} = \left(\prod_{i=1}^{\tau} Y_i^{\sum_j \rho_j e_{i,j}} \right) \cdot \left(\prod_j A_j^{\rho_j} \right) \right] \leq 2^{-\mathsf{s}_0}. \tag{1}$$

Proof. Suppose that $g^{z_0} h^{w_0} \neq A_1 \cdot \prod_{i=1}^{\tau} Y_i^{e_{i,1}}$. Fix all the ρ_j's but ρ_1 and notice that Equation (1) boils down to $\left(g^{-z_0} h^{-w_0} \cdot A_1 \cdot \prod_{i=1}^{\tau} Y_i^{e_{i,1}} \right)^{\rho_1} = B$ for some $B \in \mathbb{G}$, which holds with probability at most $2^{-\mathsf{s}_0}$ over the choice of ρ_1. □

4.1.1 Some helpful lemmas

Lemma 4.3. *Let $M \in (\pm 2^{\mathsf{pb}})^{(\tau+1)\times(\tau+1)}$ denote a random matrix such that $M_{i,*} = (1, e_{i,1}, \ldots, e_{i,\tau})$ where $e_{i,j} \leftarrow \pm 2^{\mathsf{pb}}$. For $\mathsf{pb} \geq 5$, it holds that $\Pr[\det(M) = 0 \bmod q] \leq 2^{1/20 - \mathsf{pb}}$.*

Proof. For linearly independent $(1, \vec{e}_1), \ldots, (1, \vec{e}_k)$, for a random $(1, \vec{e}_{k+1})$, the probability that it is linearly dependent on the previous ones is at most $2^{-\mathsf{pb} \cdot (\tau - k + 1)}$: by Gaussian elimination over $\mathbb{F}_q$ the $\tau - k + 1$ independent constraints on $(1, \vec{e}_{k+1})$ can be triangulated into pivot/free form, and since $q > 2^{\mathsf{pb}+1}$ each pivot has at most one preimage in $\pm 2^{\mathsf{pb}}$. By union bound, the probability that M is not invertible is at most $\sum_{k=0}^{\tau} 2^{-\mathsf{pb} \cdot (\tau - k + 1)} = \sum_{j=1}^{\tau+1} 2^{-\mathsf{pb} \cdot j} \leq \frac{2^{-\mathsf{pb}}}{1 - 2^{-\mathsf{pb}}} \leq \frac{32}{31} \cdot 2^{-\mathsf{pb}} \leq 2^{1/20 - \mathsf{pb}}$. □

Lemma 4.4. *Fix $i \in [\tau]$ and let $y_i \in \mathbb{Z}_q$ such that $d \cdot y_i = c \bmod q$ implies $c \notin \pm 2^{\kappa + \varepsilon' + 1}$ or $d \notin \pm 2^{\mathsf{pb}+1}$. For any a and $(y_j)_{j \in [\tau] \setminus i}$ in $\mathbb{Z}_q$, for $(e_j \xleftarrow{\$} \pm 2^{\mathsf{pb}})_{j \in [\tau]}$, it holds that*

$$\Pr\left[a + \sum_{j=1}^{\tau} e_j y_j \in \pm 2^{\kappa + \varepsilon'} \bmod q \right] \leq 2^{-\mathsf{pb}}.$$

Proof. Fix e_j for $j \in [\tau] \setminus i$. Assume that for $e \neq e' \in \pm 2^{\mathsf{pb}}$ it holds that $a + \sum_{j \in [\tau] \setminus i} e_j y_j + e y_i \in \pm 2^{\kappa + \varepsilon'} \bmod q$ and $a + \sum_{j \in [\tau] \setminus i} e_j y_j + e' y_i \in \pm 2^{\kappa + \varepsilon'} \bmod q$. Then, by subtracting the two equations, we get that $(e - e') y_i = c \bmod q$ for some $c \in \pm 2^{\kappa + \varepsilon' + 1}$, contradicting the assumption on y_i. Thus, there is at most one choice of e_i such that $a + \sum_{j=1}^{\tau} e_j y_j \in \pm 2^{\kappa + \varepsilon'} \bmod q$, and since $e_i \xleftarrow{\$} \pm 2^{\mathsf{pb}}$, the probability of this event is at most $2^{-\mathsf{pb}}$. □

4.2 Proof of Pedersen / Paillier-ElGamal Consistency (Π_{dlcom})

Definition 4.5 (Promise problem for Π_{dlcom})**.** The statement is a tuple $(g, h, t, u, U_1, \ldots, U_\tau, \mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u), B)$, parameterized by a chunk-magnitude bound $\kappa_0 \geq \kappa$.

- *Yes instances.* Tuples for which there exist $(y_j)_{j=0}^{\tau-1} \in (\pm 2^{\kappa_0})^\tau$, $\gamma \in \pm 2^{2\ell}$, $r \in \pm 2^{2\ell}$, and $(s_j)_{j=0}^{\tau-1} \in \mathbb{F}_q^\tau$ such that

$$\begin{aligned} U_j &= g^{y_{j-1}}\, h^{s_{j-1}} \;\; (j = 1, \ldots, \tau), \quad \mathcal{E}_u = u^\gamma \bmod N^2, \\ \mathcal{E}_t &= (1 + y'N)\, t^\gamma \bmod N^2, \qquad\quad B = (1 + \gamma N)\, u^r \bmod N^2. \end{aligned}$$

  where $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$.

- *Slack instances.* Tuples for which N admits small factors, or there exist $(y_j)_{j=0}^{\tau-1} \in (\pm 2^{\kappa_0+\varepsilon+1})^\tau$, $d_0, d \in \pm 2^{\mathsf{s}}$, $\gamma \in \pm 2^{2\ell+\mathsf{s}_0+\nu}$, $\rho \in \mathbb{Z}_{N^2}^*$, and $(s_j)_{j=0}^{\tau-1} \in \mathbb{F}_q^\tau$ such that

$$\begin{aligned} U_j &= g^{y_{j-1} \cdot d^{-1}}\, h^{s_{j-1}} \;\; (j = 1, \ldots, \tau), \qquad \mathcal{E}_u^{d_0} = u^{\gamma^*} \bmod N^2, \\ \mathcal{E}_t^{d_0} &= (1 + d_0 \cdot y' \cdot d^{-1} N)\, t^{\gamma^*} \bmod N^2, \qquad B = (1 + \gamma \cdot d^{-1} N) \cdot \rho^N \bmod N^2. \end{aligned}$$

  where $d_0 \mid d$, $\gamma^* = \gamma \cdot d^* \bmod \varphi(N)$ and $d^* = (d/d_0)^{-1} \bmod \varphi(N)/d_0$, and $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$.

Protocol Pedersen / Paillier-ElGamal Consistency

Common Input. Modulus N, precomputed bases $t, u \in \mathbb{Z}_{N^2}^*$, $g, h \in \mathbb{G}$, $U_1, \ldots, U_\tau \in \mathbb{G}$, and $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u) \in (\mathbb{Z}_{N^2}^*)^2$ such that $U_j = g^{y_{j-1}} \cdot h^{s_{j-1}}$, $B = (1 + \gamma N) \cdot u^r \bmod N^2$,

$$\mathcal{E}_t = (1 + y'N) \cdot t^\gamma \bmod N^2, \qquad \mathcal{E}_u = u^\gamma \bmod N^2,$$

for $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$, $\gamma \in \pm 2^{2\ell}$, and $r \in \pm 2^{2\ell}$.

Operations. (Recall $\varepsilon = \mathsf{s}_0 + \nu$. Repeated $\lceil \mathsf{s}/\mathsf{s}_0 \rceil$ times for soundness $\approx 2^{-\mathsf{s}}$.)

1. P samples $(\beta_0, \ldots, \beta_{\tau-1}) \xleftarrow{\$} (\pm 2^{\kappa_0+\varepsilon})^\tau$, $(\sigma_0, \ldots, \sigma_{\tau-1}) \xleftarrow{\$} \mathbb{F}_q^\tau$, $\delta \xleftarrow{\$} \pm 2^{2\ell+\varepsilon}$, $r' \xleftarrow{\$} \pm 2^{2\ell+\varepsilon}$, sets $\beta' = \sum_{j=1}^{\tau} \beta_{j-1} \cdot 2^{(j-1)\kappa}$, and sends

$$\begin{cases} \widetilde{A}_j \;=\; g^{\beta_{j-1}} \cdot h^{\sigma_{j-1}} \;\in\; \mathbb{G} \quad (j = 1, \ldots, \tau), \\ \widetilde{B} \;=\; (1 + \delta N) \cdot u^{r'} \bmod N^2, \\ \widetilde{D}_1 \;=\; (1 + \beta' N) \cdot t^\delta \bmod N^2, \\ \widetilde{D}_2 \;=\; u^\delta \bmod N^2. \end{cases}$$

2. V sends $e \xleftarrow{\$} \pm 2^{\mathsf{s}_0}$.
3. P returns $(z_j)_{j=0}^{\tau-1}$, $(w_j)_{j=0}^{\tau-1}$, and w_γ such that

$$\begin{aligned} z_j &= \beta_j + e \cdot y_j, \quad w_j = \sigma_j + e \cdot s_j \bmod q, \\ w_\gamma &= \delta + e \cdot \gamma, \qquad w_r = r' + e \cdot r. \end{aligned}$$

Verifier's Output. Let $\hat{z} = \sum_{j=1}^{\tau} z_{j-1} \cdot 2^{(j-1)\kappa}$. Accept if $z_j \in \pm 2^{\kappa_0+\varepsilon}$ for all j, $w_\gamma \in \pm 2^{2\ell+\varepsilon}$, and

$$\begin{cases} g^{z_{j-1}} \cdot h^{w_{j-1}} \;=\; \widetilde{A}_j \cdot U_j^e \;\in\; \mathbb{G} \quad (j = 1, \ldots, \tau), \\ (1 + \hat{z}\, N) \cdot t^{w_\gamma} \;=\; \widetilde{D}_1 \cdot \mathcal{E}_t^e \bmod N^2, \\ u^{w_\gamma} \;=\; \widetilde{D}_2 \cdot \mathcal{E}_u^e \bmod N^2, \\ (1 + w_\gamma N) \cdot u^{w_r} \;=\; \widetilde{B} \cdot B^e \bmod N^2. \end{cases}$$

Protocol 2: Proof of Pedersen / Paillier-ElGamal Consistency Π_{dlcom}.

Proposition 4.6 (Special Soundness for Π_{dlcom})**.** *Let*

$$\mathsf{trans} = \big((\widetilde{A}_j)_{j \le \tau},\, \widetilde{D}_1, \widetilde{D}_2,\, e,\, (z_j)_{j<\tau},\, (w_j)_{j<\tau},\, w_\gamma\big)$$

and

$$\mathsf{trans}' = \big((\widetilde{A}_j)_{j \le \tau},\, \widetilde{D}_1, \widetilde{D}_2,\, e',\, (z'_j)_{j<\tau},\, (w'_j)_{j<\tau},\, w'_\gamma\big)$$

be two accepting transcripts of Π_{dlcom} *with* $0 < |e - e'| < 2^{\mathfrak{s}_0}$*. Then there exists an extractor* E *given auxiliary input* $\varphi(N), t^{N^{-1}}, u^{N^{-1}}$ *such that at least one of the following holds:*

1. $k \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *with* $k \notin \{1, N\}$ *and* $k \mid N$*.*
2. $((y_j)_j, d, \gamma, (s_j)_j, v_s, v_t) \leftarrow \mathsf{E}(\mathsf{trans}, \mathsf{trans}')$ *where the tuple is a slack witness for* Π_{dlcom}*.*

Proof. If $e - e' \mid N$ or $\gcd(N, \varphi(N)) \neq 1$, then the extractor returns the relevant value as per item 1. Else, let $d = e - e'$, $\gamma = w_\gamma - w'_\gamma$, $y_j = z_j - z'_j$, $y = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$, $s_j = (w_j - w'_j) \cdot d^{-1} \bmod q$, and $\gamma' = \gamma \cdot d^* \bmod \varphi(N)$, where $d^* = (d/d_0)^{-1} \bmod \varphi'$ for $d_0 = \gcd(\varphi(N), d)$ and $\varphi' = \varphi(N)/d_0$. Additionally, let $v_t = u^{-\gamma'} \cdot \mathcal{E}_u^{d_0} \bmod N^2$ and $v_s = t^{-\gamma'} \cdot (1 - d_0 y d^{-1} N) \cdot \mathcal{E}_t^{d_0} \bmod N^2$. Observe that:

$$\begin{cases} g^{y_j \cdot d^{-1}} \cdot h^{s_j} \;=\; U_j \;\in\; \mathbb{G} \quad (j = 1, \ldots, \tau), \\ (1 + d_0 y \cdot d^{-1}\, N) \cdot v_s \cdot t^{\gamma'} \;=\; \mathcal{E}_t^{d_0} \bmod N^2, \\ v_t \cdot u^{\gamma'} \;=\; \mathcal{E}_u^{d_0} \bmod N^2. \end{cases}$$

Since $u^{\gamma'} = u^{d^*\gamma} = \mathcal{E}_u^{d^* d} = \mathcal{E}_u^{d_0}$. Therefore, $v_t = 1$, and similarly $v_s = ((1 - y d^{-1} N) \cdot \mathcal{E}_t)^{-\beta\varphi'}$. This concludes the proof. □

4.3 Combined Proof of Encrypted Discrete Log (Π_{encdl})

We combine Π_{small} and Π_{dlcom} into a single proof system Π_{encdl} for the joint statement that a group element $X = g^x$ and a Paillier-ElGamal ciphertext $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u)$ are consistent, in the sense that $\mathcal{E}$ encrypts an integer x' congruent to x modulo q with x' in a bounded range. The construction takes an additional public scalar $\alpha \in \mathbb{N}$, agreed upon as a product of small prime powers coprime to q; α is chosen divisible by every small d_i that Π_{small}'s slack extractor can produce, so raising to α converts any slack chunk $Y_i = g^{c_i/d_i}\, h^{r_i}$ back into an integer-exponent commitment.

Definition 4.7 (Promise problem for Π_{encdl})**.** The statement is a tuple $(g, h, t, u, X, \mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u), B, \alpha)$, where α is a product of the agreed prime powers, coprime to q.

- *Yes instances.* Tuples for which there exist $x \in [0, q)$, $\gamma \in \pm 2^{2\ell}$, and $r \in \pm 2^{2\ell}$ such that

$$\begin{aligned} X &= g^x, & x' &= \alpha \cdot (\alpha^{-1} x \bmod q), \\ \mathcal{E}_t &= (1 + x'N)\, t^\gamma \bmod N^2, & \mathcal{E}_u &= u^\gamma \bmod N^2, \\ B &= (1 + \gamma N)\, u^r \bmod N^2. \end{aligned}$$

- *Slack instances.* Tuples for which N admits small factors, or there exist an integer $x' \in \pm(\alpha\, \tau \cdot 2^{\kappa\tau + \varepsilon + 1})$, $d_0, d \in \pm 2^{\mathsf{s}}$, $\gamma \in \pm 2^{2\ell + \mathsf{s}_0 + \nu}$, and $\rho \in \mathbb{Z}^*_{N^2}$ such that

$$\begin{aligned} X &= g^{x'}, & \mathcal{E}_u^{d_0} &= u^{\gamma^*} \bmod N^2, \\ \mathcal{E}_t^{d_0} &= (1 + d_0 \cdot x' N)\, t^{\gamma^*} \bmod N^2, & B &= (1 + \gamma \cdot d^{-1} N) \cdot \rho^N \bmod N^2, \end{aligned}$$

  where $d_0 \mid d$, $\gamma^* = \gamma \cdot d^* \bmod \varphi(N)$ and $d^* = (d/d_0)^{-1} \bmod \varphi(N)/d_0$.

Construction. The prover reduces the joint statement to chunks of $\tilde{x} \coloneqq x \cdot \alpha^{-1} \bmod q$, which it Pedersen-commits to in $\mathbb{G}$; multiplying by α in the exponent of each commitment yields the "raised" commitments that play the role of the U_j's of Π_{dlcom}. The raised commitments are not transmitted: the verifier reconstructs them from the Pedersen commitments and the public α. The Paillier-style commitment $B = (1 + \gamma N)\, u^r$ to the Paillier-ElGamal randomness γ plays the same role as in Π_{dlcom} and is passed through directly. Two sub-proofs are then run in lockstep — Π_{small} on the Pedersen commitments and Π_{dlcom} on the raised commitments, $\mathcal{E}$, and B — with their challenges drawn independently but bundled together (sent in a single round-trip in the interactive case, or derived from a single Fiat-Shamir hash over the concatenated first messages of both sub-proofs in the non-interactive case). To anchor the joint proof to the public X, the prover picks the Pedersen randomness subject to a single linear constraint over $\mathbb{F}_q$, which forces the aggregated raised commitment to coincide with X.

Protocol Encrypted Discrete Log Π_{encdl}

Common Input. Pedersen bases $g, h \in \mathbb{G}$, Paillier bases $t, u \in \mathbb{Z}^*_{N^2}$ over a modulus N, a public scalar $\alpha \in \mathbb{N}$, a group element $X \in \mathbb{G}$, a ciphertext $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u) \in (\mathbb{Z}^*_{N^2})^2$, and a Paillier-style commitment $B \in \mathbb{Z}^*_{N^2}$.

Witness. $x \in \mathbb{F}_q$ with $X = g^x$, a bounded integer $x' \equiv x \bmod q$, Paillier-ElGamal randomness $\gamma \in \pm 2^{2\ell}$, and randomness $r \in \pm 2^{2\ell}$ such that $\mathcal{E}_t = (1 + x'N)\, t^\gamma \bmod N^2$, $\mathcal{E}_u = u^\gamma \bmod N^2$, and $B = (1 + \gamma N)\, u^r \bmod N^2$.

Operations.

1. *Chunking and commitment.* P computes $\tilde{x} = x \cdot \alpha^{-1} \bmod q$ and writes it as $\tilde{x} = \sum_{j=1}^{\tau} y_{j-1}\, 2^{(j-1)\kappa}$ with $y_{j-1} \in \pm 2^\kappa$. P samples randomness $(r_{j-1})_{j=1}^{\tau} \in \mathbb{F}_q^\tau$ subject

to the linear constraint

$$\sum_{j=1}^{\tau} r_{j-1}\, 2^{(j-1)\kappa} \;\equiv\; 0 \mod q,$$

and sends $Y_j = g^{y_{j-1}}\, h^{r_{j-1}} \in \mathbb{G}$ for $j = 1, \ldots, \tau$.

2. *Joint sub-proofs.* P and V run Π_{small} (Protocol 1) on $(Y_j)_{j=1}^{\tau}$ and Π_{dlcom} (Protocol 2) on the raised commitments $X_j := Y_j^{\alpha}$, $\mathcal{E}$, and B with Π_{dlcom}'s chunk-magnitude parameter set to $\kappa_0 := \kappa + \log \alpha$, with each sub-proof drawing its own challenges, bundled together as described above.

Verifier's Output. Accept if (i) α is well-formed (a product of the agreed prime powers, coprime to q), (ii) $X = \prod_{j=1}^{\tau} X_j^{2^{(j-1)\kappa}}$ for $X_j = Y_j^{\alpha}$, and (iii) both the Π_{small} and Π_{dlcom} sub-verifiers accept.

Protocol 3: Combined proof of encrypted discrete log Π_{encdl}.

Proposition 4.8 (HVZK of Π_{encdl})**.** Π_{encdl} *is honest-verifier zero-knowledge with statistical simulation distance at most* $\rho_0 \cdot (\tau + 1) \cdot 2^{-\nu}$.

Proof. The ZK simulator samples the X_j's subject to $\prod_{j=1}^{\tau} X_j^{2^{(j-1)\kappa}} = X$ and sets $Y_j := X_j^{\alpha^{-1}}$ for $j = 1, \ldots, \tau$, then samples the challenges and runs the ZK simulators of the sub-protocols. It outputs the simulated transcripts for both sub-protocols. To see why the output distribution is indistinguishable, first note that the X_j's are identically distributed in the real and simulated experiments; second, the first sub-proof (Π_{small}) is perfect HVZK and the second is $\rho_0 \cdot (\tau + 1) \cdot 2^{-\nu}$-HVZK, since the only distinguishable values are z_j and w_γ. It follows that the distinguishing advantage between real and ideal is also at most $\rho_0 \cdot (\tau + 1) \cdot 2^{-\nu}$. □

4.4 Cost Summary

Let $r_0 := \lfloor \mathsf{s}/\mathsf{pb} \rfloor$ and $\rho_0 := \lceil \mathsf{s}/\mathsf{s}_0 \rceil$. We count multi-scalar multiplications $\mathsf{MSM}_{\mathbb{G}}(k)$ in $\mathbb{G}$ and exponentiations Exp_{N^2} in $\mathbb{Z}_{N^2}^*$. Communication is measured in bits: $\mathbb{G}$ and $\mathbb{F}_q$ elements contribute $\log q$ bits each; $\mathbb{Z}_{N^2}^*$ elements contribute $2|N|$ bits each; integers in $\pm 2^b$ contribute b bits. We account for the prover's first message only when batch verification is enabled (the A values are needed to form the batched check). In the interactive setting we additionally account for the verifier's challenges; in the Fiat-Shamir setting challenges are hash-derived and excluded.

Concrete Parameters. We require that $\kappa + \log(\alpha) + \varepsilon + 1 < \log(q)$ (to make sure that the response doesn't overflow) and $\kappa + \log(\alpha) + \varepsilon' + \mathsf{s}_0 + 1 < \log(q)$ (to make sure that the witness multiplied by the challenge doesn't overflow). We use $\nu = 40$, and $\mathsf{s} = 128$ for the Fiat-Shamir and $\mathsf{s} = 40$ for the interactive setting. We set $\mathsf{pb} = 5$ which yields $\lceil \log(\alpha) \rceil = 48$.
We further fix $\log q = 256$, $\ell = 128$, and $|N| = 3072$. In each regime we maximize κ subject to the constraint, then set $\tau = \lceil \log q/\kappa \rceil$, $\varepsilon' = \lceil \log_2 \tau \rceil + \mathsf{pb} + \nu$, $\varepsilon = \mathsf{s}_0 + \nu$, $r_0 = \lfloor \mathsf{s}/\mathsf{pb} \rfloor$, and $\rho_0 = \lceil \mathsf{s}/\mathsf{s}_0 \rceil$. The resulting values are given in Table 5.

Quantity	No batch	Batch
Comm. (interactive)	$r_0 \cdot (\tau \cdot \mathsf{pb} + \kappa + \varepsilon' + \log q)$	$r_0 \cdot (2\log q + \tau \cdot \mathsf{pb} + \kappa + \varepsilon')$
Comm. (Fiat-Shamir)	$r_0 \cdot (\kappa + \varepsilon' + \log q)$	$r_0 \cdot (2\log q + \kappa + \varepsilon')$
Comp. prover	$r_0 \cdot \mathsf{MSM}_{\mathbb{G}}(2)$	$r_0 \cdot \mathsf{MSM}_{\mathbb{G}}(2)$
Comp. verifier	$r_0 \cdot \mathsf{MSM}_{\mathbb{G}}(\tau + 2)$	$\mathsf{MSM}_{\mathbb{G}}(r_0 + \tau + 2)$

Table 3: Costs for Π_{small}, where $r_0 = \lfloor \mathsf{s}/\mathsf{pb} \rfloor$. Interactive challenges: τ values in $\pm 2^{\mathsf{pb}}$ per repetition. First message (batch only): $A \in \mathbb{G}$ per repetition. Response: $z \in \pm 2^{\kappa+\varepsilon'}$, $w \in \mathbb{F}_q$ per repetition.

Quantity	Cost
Comm. (interactive)	$\rho_0 \cdot (\mathsf{s}_0 + 2(2\ell + \varepsilon) + \tau(\kappa_0 + \varepsilon + \log q))$
Comm. (Fiat-Shamir)	$\rho_0 \cdot (2(2\ell + \varepsilon) + \tau(\kappa_0 + \varepsilon + \log q))$
Comp. prover	$\rho_0 \cdot (\tau \cdot \mathsf{MSM}_{\mathbb{G}}(2) + 3\,\mathsf{Exp}_{N^2})$
Comp. verifier	$\rho_0 \cdot (\tau \cdot \mathsf{MSM}_{\mathbb{G}}(3) + 6\,\mathsf{Exp}_{N^2})$

Table 4: Costs for Π_{dlcom} (no batch verification), where $\rho_0 = \lceil \mathsf{s}/\mathsf{s}_0 \rceil$. Interactive challenge per repetition: $e \in \pm 2^{\mathsf{s}_0}$ (s_0 bits). Response per repetition: $(z_j)_{j<\tau} \in (\pm 2^{\kappa_0+\varepsilon})^\tau$, $(w_j)_{j<\tau} \in \mathbb{F}_q^\tau$, $w_\gamma \in \pm 2^{2\ell+\varepsilon}$, $w_r \in \pm 2^{2\ell+\varepsilon}$.

Concrete Communication. Substituting these parameters into Tables 3 and 4 gives the kilobyte counts also reported in Table 5. Both Π_{small} and Π_{dlcom} costs depend on s_0 through κ, τ, and ε; the Π_{dlcom} figures additionally account for $\rho_0 = \lceil \mathsf{s}/\mathsf{s}_0 \rceil$ repetitions.

5 Registered Non-Interactive VOLE from Non-Interactive Multiplication

Non-interactive secure two-party computation (NISC) is a class of two-party protocols between a receiver with an input x and a sender, where the receiver initially publishes an encryption of her input x. Afterwards, the sender holding a function f can, using a single message, reveal $f(x)$ to the receiver (and nothing more). In a reusable NISC [CDIKLOV19], the receiver's first message can be reused for an arbitrary number of functions. Non-interactive vector-OLE (NIVOLE) is a reusable NISC for linear functions: given the reusable receiver encryption of x, the sender can reveal $a_i x + b_i$ to the receiver for any number of (a_i, b_i) using a single message. The roots of the notion date back to [BM90]. In the semi-honest setting, (rerandomizable) additively homomorphic encryption naturally yields a NIVOLE: the receiver broadcasts an encryption $\mathcal{E}(x)$ of x under her own public key, and the sender replies with $\mathcal{E}(ax + b)$, which the receiver decrypts. Maliciously secure protocols for NIVOLE have been introduced in [CDIKLOV19]. More recently, in [OSY21], a new approach for NIVOLE has been introduced which combines distributed discrete logarithm [BGI16] with a technique inspired from [BM90] which has later been referred to as non-interactive multiplication (NIM) [BDSS25].

In this section, we introduce a new protocol for non-interactive vector-OLE (NIVOLE) with

Regime	κ	τ	ε	ε'	r_0	ρ_0	Π_{small} (KB)	Π_{dlcom} (KB)	Π_{encdl} (KB)
F-S ($\mathsf{s}_0 = 128$)	29	9	168	49	25	1	1.02	0.65	1.95
F-S ($\mathsf{s}_0 = 64$)	95	3	104	47	25	2	1.21	0.54	1.84
F-S ($\mathsf{s}_0 = 32$)	128	2	72	46	25	4	1.31	0.81	2.18
F-S ($\mathsf{s}_0 = 16$)	144	2	56	46	25	8	1.36	1.59	3.01
F-S ($\mathsf{s}_0 = 8$)	152	2	48	46	25	16	1.39	3.16	4.61
Inter. ($\mathsf{s}_0 = 40$)	119	3	80	47	8	1	0.43	0.27	0.79
Inter. ($\mathsf{s}_0 = 32$)	128	2	72	46	8	2	0.43	0.41	0.90
Inter. ($\mathsf{s}_0 = 16$)	144	2	56	46	8	3	0.45	0.60	1.11
Inter. ($\mathsf{s}_0 = 8$)	152	2	48	46	8	5	0.45	0.99	1.50

Table 5: Derived parameters and concrete communication per regime ($\nu = 40$, $\mathsf{pb} = 5$, $\lceil \log \alpha \rceil = 48$, $\log q = 256$, $\varepsilon = \mathsf{s}_0 + \nu$, $\ell = 128$, $|N| = 3072$). Here $r_0 = \lfloor \mathsf{s}/\mathsf{pb} \rfloor$ (Π_{small}) and $\rho_0 = \lceil \mathsf{s}/\mathsf{s}_0 \rceil$ (Π_{dlcom}). The constraint $\kappa + \log \alpha + \varepsilon' + \mathsf{s}_0 + 1 < 256$ has slack 1 in all rows; κ, τ, ε', r_0 are determined jointly by the constraints. Communication is in kilobytes (1 KB = 1024 bytes); Fiat-Shamir uses $\mathsf{s} = 128$ and Interactive uses $\mathsf{s} = 40$. Batch verification for Π_{small} adds 0.78 KB (FS) and 0.25 KB (interactive); Π_{dlcom} does not support batch verification. Π_{encdl} adds the τ Pedersen commitments $Y_j \in \mathbb{G}$ ($\tau \cdot \log q/8192$ KB) on top of $\Pi_{\mathsf{small}} + \Pi_{\mathsf{dlcom}}$.

unmatched computation and communication performances. Our protocol follows the NIM-based paradigm laid out in [OSY21] and integrates several optimizations from previous works, such as the use of short exponents [ABCGJM26]. Our core technical contribution lies in enhancing the security of our NIVOLE to the malicious setting using tailored new highly-optimized NIZKs. To facilitate applications, we chose to directly describe a new variant of NIVOLE, that we call *registered* NIVOLE, which integrates two tweaks:

- Two-phase structure: we separate the encryption of x in two phases, an input-independent initialization phase that sets up public parameters, and an input-dependent registration phase. This lets us push the bulk of the computation and communication to the input-independent initialization phase, which can be reused for an arbitrary number of NIVOLE instances (with different receiver inputs x).
- Tying x to a public key: our registered NIVOLE lets the client encrypt a value x and prove that the encrypted value is also the secret key corresponding to a public element g^x. This is an optional feature that has essentially no impact on performance (it can be dropped if one wants a "plain" NIVOLE) but is useful in applications (such as our example application to threshold ECDSA in Appendix B).

Registered NIVOLE can be seen as a non-interactive, reusable analogue of the committed-receiver OLE of [ABCGJM26].

5.1 Defining Registered NIVOLE

We adopt an operational definition of registered NIVOLE by defining security directly through the security of the functionality it aims to implement non-interactively.

Definition 5.1 (Registered NIVOLE algorithms)**.** Let $\mathbb{F}_q$ be a prime order field, and let $\mathbb{G}$ denote a group of order q. A registered non-interactive vector oblivious linear evaluation (RNIVOLE) over $(\mathbb{F}_q, \mathbb{G})$ is a six-tuple of algorithms $\mathsf{VOLE} = (\mathsf{VOLE.Setup}, \mathsf{VOLE.VerifySetup}, \mathsf{VOLE.Register}, \mathsf{VOLE.VerifyRegister}, \mathsf{VOLE.Send}, \mathsf{VOLE.Output})$, where:

- $\mathsf{VOLE.Setup}(1^\ell)$: given the security parameter in unary, outputs the public setup data $\mathsf{SetupData}$ and a private key sk.
- $\mathsf{VOLE.VerifySetup}(\mathsf{SetupData})$: given the setup data $\mathsf{SetupData}$, outputs a bit σ.
- $\mathsf{VOLE.Register}(\mathsf{SetupData}, x)$: given the setup data $\mathsf{SetupData}$ and an input $x \in \mathbb{F}_q$, outputs a pair $(\mathsf{e}_x, \mathsf{d}_x)$ where e_x is the (public) encryption of x, and d_x is the (private) decoding information.
- $\mathsf{VOLE.VerifyRegister}(\mathsf{SetupData}, X, \mathsf{e}_x)$: given the setup data $\mathsf{SetupData}$, a group element $X \in \mathbb{G}$, and the ciphertext e_x, output a bit σ.
- $\mathsf{VOLE.Send}(\mathsf{SetupData}, \mathsf{e}_x, a, b)$: given the setup data $\mathsf{SetupData}$, the ciphertext e_x, and a pair $(a, b) \in \mathbb{F}_q^2$, return a message m.
- $\mathsf{VOLE.Output}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_x, \mathsf{d}_x, m)$: given the setup data $\mathsf{SetupData}$, secret key sk, ciphertext e_x, decoding information d_x, and message m, output $w \in \mathbb{F}_q \cup \{\perp\}$.

Before we move on to the security requirements, we make a few comments. First, the algorithms $\mathsf{VOLE.VerifySetup}$ and $\mathsf{VOLE.VerifyRegister}$ are only included to simplify the description of protocols built on top of RNIVOLE. In the semi-honest setting, they can be ignored, and in the malicious setting, both $\mathsf{VOLE.VerifySetup}$ and $\mathsf{VOLE.VerifyRegister}$ could be merged inside $\mathsf{VOLE.Send}$ (this corresponds to letting the sender check that the setup data and ciphertext are well-formed before sending his message m). However, describing them as separate algorithms enables a cleaner isolation of each of the protocol's components, and allows clarifying which computational steps can be executed ahead of time, only once per setup $\mathsf{SetupData}$ or once per ciphertext e_x. Second, our constructions will make use of hash functions, modeled as random oracles. An algorithm $\mathcal{A}$ using an oracle O will be denoted $\mathcal{A}^{\mathsf{O}}$ to make the reliance on an oracle explicit.

5.1.1 Security

We consider the three-phase protocol Π_{VOLE} of Protocol 4 between a client C (the *sender*) with any number of input pairs $(a_i, b_i) \in \mathbb{F}_q^2$ and a server S (the *receiver*) with inputs $x_i \in \mathbb{F}_q$. We say that VOLE is a secure RNIVOLE if Π_{VOLE} securely instantiates the registered VOLE game represented in Game 1 in the malicious setting.

Protocol Registered NIVOLE protocol Π_{VOLE}

The parties have pre-specified inputs, matching the interface of Game 1: $x_i \in \mathbb{F}_q$ for S ($i \in I$, the set of registrations), and $(a_{i,j}, b_{i,j}, X_{i,j}) \in \mathbb{F}_q^2 \times \mathbb{G}$ for C ($(i,j) \in I \times J$, where J indexes the extends per registration). At the beginning of each round the attacker supplies a public input (init), $(\mathsf{register}, i)$, or (extend, i, j). For that round, the parties run (init), $(\mathsf{register}, x_i)$, or $(\mathsf{extend}, a_{i,j}, b_{i,j}, X_{i,j})$ respectively, under the restriction that (init) must occur before any

$(\mathsf{register}, \cdot)$ and $(\mathsf{register}, i)$ must occur before any $(\mathsf{extend}, i, \cdot)$, and each input is consumed after it is used. The parties have access to the setup functionality $\mathcal{F}_{\mathsf{setup}}$ and the random oracles H and H'. The phases of the protocol are specified as follows:

Initialize. S and C invoke $\mathcal{F}_{\mathsf{setup}}$ (Functionality 1) and receive the resulting setup data $\mathsf{SetupData}$; if $\mathcal{F}_{\mathsf{setup}}$ aborts, the protocol aborts.

Register. S runs $(\mathsf{e}_{x_i}, \mathsf{d}_{x_i}) \xleftarrow{\$} \mathsf{VOLE.Register}(\mathsf{SetupData}, x_i)$, sets $X_i \leftarrow g^{x_i}$, and sends (e_{x_i}, X_i) to C. C runs $\sigma \leftarrow \mathsf{VOLE.VerifyRegister}(\mathsf{SetupData}, X_i, \mathsf{e}_{x_i})$. If $\sigma = 0$, C aborts the protocol.

Extend. C checks that $X_{i,j}$ equals the registered X_i (aborting otherwise), runs $m \xleftarrow{\$} \mathsf{VOLE.Send}(\mathsf{SetupData}, \mathsf{e}_{x_i}, a_{i,j}, b_{i,j})$, and sends m to S. S outputs $w \leftarrow \mathsf{VOLE.Output}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_{x_i}, \mathsf{d}_{x_i}, m)$.

Protocol 4: Registered NIVOLE protocol Π_{VOLE}.

Functionality Registered VOLE Game

Parties. The game interacts with two players, a server S (the *receiver*) and a client C (the *sender*), and an adversary $\mathcal{A}$.

Initialize. When obtaining $(\mathsf{init}, \mathsf{sid}, (x_i)_{i \in I})$ from S and $(\mathsf{init}, \mathsf{sid}, (a_{i,j}, b_{i,j}, X_{i,j})_{(i,j) \in I \times J})$ from C, where sid encodes an elliptic curve $(\mathbb{G}, g, q)$ of prime order q generated by g, the index sets I, J, and the identities S, C of the server and the client, and where $x_i \in \mathbb{Z}_q$ and $(a_{i,j}, b_{i,j}, X_{i,j}) \in \mathbb{Z}_q^2 \times \mathbb{G}$: record sid and all inputs, and treat the game as activated. (All inputs are thus specified at initialization; the adversary schedules the operations below.)

Register. When obtaining $(\mathsf{register}, i)$ from $\mathcal{A}$ with $i \in I$, at most once per i: if S is corrupt, $\mathcal{A}$ supplies $x_i \in \mathbb{Z}_q$ along with the trigger. Mark i as registered and send $(\mathsf{register}, i, g^{x_i})$ to C.

Extend. When obtaining (extend, i, j) from $\mathcal{A}$ with $(i, j) \in I \times J$, at most once per (i, j): if C is corrupt, $\mathcal{A}$ supplies $(a_{i,j}, b_{i,j}, X_{i,j})$ along with the trigger.

1. If i is registered and $X_{i,j} = g^{x_i}$, return $a_{i,j} \cdot x_i + b_{i,j} \bmod q$ to S.
2. Otherwise, return $\perp$ to S.

Game 1: Registered VOLE game.

5.2 Extended Paillier-ElGamal

We consider below a variant of the Paillier-ElGamal encryption scheme where, in addition to the key $(u, t = u^d)$, a term $s = t^d = u^{d^2}$ is appended to the public key. We further consider, as

in [ABCGJM26], a variant where the secret exponent d is short.

Definition 5.2. The Extended Paillier-ElGamal encryption with short exponents, denoted $\mathsf{PE}^* = (\mathsf{PE}^*.\mathsf{KeyGen}, \mathsf{PE}^*.\mathsf{Enc}, \mathsf{PE}^*.\mathsf{Dec})$, is defined as the Paillier-ElGamal encryption (Definition 3.15) with the following modifications:

1. $\mathsf{PE}^*.\mathsf{KeyGen}(1^\ell)$: sample $d \xleftarrow{\$} \{0,1\}^{2\ell}$ instead of $d \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$. Additionally, set $s_0 = t_0^d \bmod N$ and $s = s_0^{2N} \bmod N^2$. Return an extended public key $\mathsf{pk}_{\mathsf{PE}} = (N, u_0, u, t_0, t, s_0, s)$.

2. $\mathsf{PE}^*.\mathsf{Enc}_{\mathsf{pk}_{\mathsf{PE}}}(m;\gamma)$: sample $\gamma \xleftarrow{\$} \{0,1\}^{2\ell}$ instead of $\gamma \xleftarrow{\$} \mathbb{Z}_N^*$ (note that $\mathsf{PE}^*.\mathsf{Enc}$ ignores the additional terms (s_0, s) from $\mathsf{pk}_{\mathsf{PE}}$).

Claim 5.3. *Assume the 2-power-SEDL assumption and the DCR assumption. Then the Extended Paillier-ElGamal encryption scheme with short exponents is IND-CPA secure.*

We omit the straightforward proof.

5.3 A RNIVOLE from Paillier-Based NIM

In this section, we describe an optimized RNIVOLE protocol using the Paillier-based non-interactive multiplication procedure from [OSY21], enhanced to the malicious setting with new NIZKs tailored to our construction. We represent all algorithms in Algorithm 7 (representing the two algorithms of the initialization phase), Algorithm 2 (representing the two algorithms of the registration phase), and Algorithm 3 (representing the two algorithms of the multiplication phase).

Functionality Setup Functionality $\mathcal{F}_{\mathsf{setup}}$

Parties. The functionality interacts with two players, a server S (the receiver) and a client C (the sender), and an adversary $\mathcal{A}$.

Operation. Upon receiving $(\mathsf{init}, \mathsf{sid})$ from S and $(\mathsf{init}, \mathsf{sid})$ from C, where sid encodes a group $(\mathbb{G}, g, q)$ of prime order q generated by g, parameters $(n_{\mathrm{rsa}}, \ell, \nu)$, and the identities S, C:

- Sample a Pedersen base $h \xleftarrow{\$} \mathbb{G}$ and send h to the adversary $\mathcal{A}$.
- *If* S *is honest:* sample $(N; p_1, p_2) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$; sample $d \xleftarrow{\$} [2^{2\ell}]$; sample u_0 uniformly from the quadratic residues modulo N and set $t_0 := u_0^d \bmod N$ and $s_0 := t_0^d \bmod N$ (so $s_0 = u_0^{d^2} \bmod N$). Send (N, u_0, t_0, s_0, h) to both parties and the trapdoor (p_1, p_2, d) to S.
- *If* S *is corrupt:* receive from $\mathcal{A}$ either $\perp$ (in which case, abort) or a tuple $(p_1, p_2, u_0, t_0, s_0, d)$; set $N := p_1 p_2$, and abort unless

  1. p_1, p_2 are primes (not necessarily safe) and $p_1, p_2 \in \pm 2^{n_{\mathrm{rsa}}/2+\ell+\nu}$;
  2. $u_0, t_0, s_0 \in \mathbb{Z}_N^*$ with $s_0 = t_0^d = u_0^{d^2} \bmod N$;
  3. $d \in \pm 2^{2\ell+\nu+1}$.

  If all checks pass, send (N, u_0, t_0, s_0, h) to both parties.

Functionality 1: Setup functionality $\mathcal{F}_{\mathsf{setup}}$.

The bound $d \in \pm 2^{2\ell+\nu+1}$ in the corrupt-S branch matches the slack set of the KZG promise problem (Definition 3.22): it is the bound guaranteed by the knowledge extractor of $\Pi^{\mathrm{FS}}_{\mathsf{par}}$, which recovers d as the difference of two responses in $\pm 2^{2\ell+\nu}$ (cf. Theorem C.1). The parameters n_0, n_1 of the multiplication phase below are chosen with the corresponding headroom, so that the masking argument (Claim 6.11) covers every d that the functionality may accept.
The following algorithm is used to register the VOLE scalar x, defined as the plaintext of a Paillier-ElGamal ciphertext $\mathcal{E}$ and the discrete logarithm of X. Rather than encrypting x itself, the registration encrypts the α-*lifted* representative $x' := \alpha \cdot (\alpha^{-1} \cdot x \bmod q)$, where $\alpha \in \mathbb{N}$ is the public scalar of Π_{encdl} (a product of the agreed small prime powers, coprime to q; cf. Protocol 3). This is required for the honest prover to satisfy the yes-instances of the promise problem of Definition 4.7: the prover of Π_{encdl} chunks $\tilde{x} = \alpha^{-1}x \bmod q$ and the verifier reconstructs commitments to $\alpha \cdot \tilde{x}$-chunks, so the encrypted integer must be $x' = \alpha\tilde{x}$. Since $x' \equiv x \bmod q$ and the receiver's output in Algorithm 3 is reduced modulo q, the correctness of the multiplication phase ($w = ax + b \bmod q$) is unaffected; only the magnitude of the plaintext changes, from $x < q$ to $x' < \alpha \cdot q$, which is accounted for by the parameter constraints of Algorithm 3.

Algorithm VOLE Registration

Parameters. $(\mathbb{G}, g, q)$, $\mathsf{SetupData}$, $d \in \mathbb{N}$, and the public scalar $\alpha \in \mathbb{N}$ (a product of the agreed small prime powers, coprime to q; cf. Protocol 3). $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ is modeled as a random oracle.

Input. $x \in \mathbb{F}_q$

$\mathsf{VOLE.Register}^{\mathsf{H}}(\mathsf{SetupData}, x)$: sample $\gamma \xleftarrow{\$} \pm 2^{2\ell}$, set $X = g^x$, and do:

- Set $x' \leftarrow \alpha \cdot (\alpha^{-1} \cdot x \bmod q)$, an integer satisfying $x' \equiv x \bmod q$ and $0 \le x' < \alpha \cdot q$.
- Set $\mathcal{E} \leftarrow \mathsf{PE}^* \,.\, \mathsf{Enc}_{\mathsf{pk}_{\mathsf{PE}}}(x'; \gamma)$ and generate proof

$$\psi_{\mathsf{encdl}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{encdl}}(\mathbb{G}, g, q, X, \mathcal{E}, \mathsf{pk}_{\mathsf{PE}}, \alpha, \log q, 2\ell; x, x', \gamma)$$

- Output $(\mathsf{e}_x = (\mathcal{E}, \psi_{\mathsf{encdl}}), \mathsf{d}_x = \gamma)$.

$\mathsf{VOLE.VerifyRegister}^{\mathsf{H}}(\mathsf{SetupData}, X, \mathsf{e}_x = (\mathcal{E}, \psi_{\mathsf{encdl}}))$:

- Check that $\mathcal{E} \in (\mathbb{Z}^*_{N^2})^2$.
- Verify proof $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{encdl}}}(\mathbb{G}, g, q, X, \mathcal{E}, \mathsf{pk}_{\mathsf{PE}}, \alpha, \log q, 2\ell, \psi_{\mathsf{encdl}}) = 1$.

Algorithm 2: VOLE registration algorithms.

Algorithm VOLE Multiplication

Parameters. $\mathsf{pp} = (\ell, \nu, \kappa, \tau, \mathsf{s})$. Define $n_r := \mathsf{s}+\nu$, $n_0 := 4\ell+\mathsf{s}+3\nu+1$, and $n_1 := 5\ell+\mathsf{s}+3\nu+1$. We require $\log(q+1) \leq 2\ell$, $n_1+3\ell+\lceil\log\alpha\rceil \leq \log N$, and $n_x+n_r+\log(q+1)+\ell < \log N$, where α is the public scalar of Π_{encdl} (cf. Algorithm 2) and n_x is the bound on x' in the slack instance from Definition 4.7. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathsf{H}' : \{0,1\}^\ell \to \mathbb{Z}_N$ are modeled as random oracles.

Common Input. $(\mathbb{G}, g, q)$, $\mathsf{SetupData}$, $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{N^2}^*$.

Private Inputs. $d \in [2^{2\ell}]$, $x \in \mathbb{F}_q$, $a, b \in \mathbb{F}_q$.

$\mathsf{VOLE.Send}^{\mathsf{H},\mathsf{H}'}(\mathsf{SetupData}, \mathsf{e}_x, a, b)$: parse $\mathcal{E}$ from e_x. Sample $\mathsf{salt} \xleftarrow{\$} [2^\ell]$, $r \xleftarrow{\$} [2^{n_r}]$, $\rho_0 \xleftarrow{\$} [2^{n_0}]$, $\rho_1 \xleftarrow{\$} [2^{n_1}]$, and do:

- Set $a' \leftarrow a + q \cdot r$ and $\mathcal{C} \leftarrow s_0^{a'} t_0^{\rho_0} u_0^{\rho_1} \bmod N$
- Compute $e \leftarrow \mathsf{H}(\mathcal{C}, \mathsf{salt})$, $\pi \leftarrow t_0^{a'} u_0^{a' \cdot e + \rho_0} \bmod N$, and $c = a' \cdot e^2 + \rho_0 \cdot e + \rho_1$.
- Set $V = \mathcal{E}_t^{a'} \mathcal{E}_u^{a' \cdot e + \rho_0} \bmod N^2$ and $z = [\mathsf{DDLOG}_N(V) + \mathsf{H}'(e)]_N + b \bmod q$.
- Return $m = (\mathsf{salt}, e, \pi, c, z)$.

$\mathsf{VOLE.Output}^{\mathsf{H},\mathsf{H}'}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_x, \mathsf{d}_x, m)$: parse $d := \mathsf{sk}$, $\mathcal{E}$ from e_x, $\gamma := \mathsf{d}_x$, and $m = (\mathsf{salt}, e, \pi, c, z)$.

- If $c \notin [2^{n_1}]$, return $\perp$.
- Set $\mathcal{C} \leftarrow \pi^{d-e} \cdot u_0^c \bmod N$. If $e \neq \mathsf{H}(\mathcal{C}, \mathsf{salt})$, return $\perp$.
- Return $w \leftarrow z - [\mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) + \mathsf{H}'(e)]_N \bmod q$.

Algorithm 3: VOLE multiplication algorithms.

6 Security Analysis

6.1 A variant of the power knowledge of exponent assumption

Knowledge of exponent assumptions have been originally introduced in [Dam92; BP04]. Many SNARKs rely on variants of the power knowledge of exponent assumption from [Gro10]. In this work, we introduce a variant of the power knowledge of exponent assumption. In the original k-PKE assumption, the adversary is given $(g^{c^i})_{i \leq k}$ and a "shifted" tuple $(g^{\alpha c^i})_{i \leq k}$, where α, c are random. In the variant below, we set $\alpha = c$. Our variant is essentially equivalent (up to minor differences which we discuss afterwards) to the k-KZG assumption introduced in [BFHK24] for the same purpose. For simplicity, we reuse the same assumption name.

Assumption 6.1 (k-KZG [BFHK24])**.** Let $\mathsf{GroupGen}$ denote a PPTM that, on input 1^ℓ, returns a triple $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is the description of a cyclic group of order $q(\ell)$ and g is a generator of $\mathbb{G}$. We say that the k-KZG assumption holds over $\mathsf{GroupGen}$ if for every non-uniform PPT algorithm $\mathcal{A}$,

there exists another non-uniform PPT algorithm Ext and a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr[X^c = Y \wedge X \neq \prod_{i=0}^{k} g^{a_i \cdot c^i}] \leq \mu(\ell),$$

where the probability is over the random choices of $(\mathbb{G}, g, q) \xleftarrow{\$} \mathsf{GroupGen}(1^\ell)$, $c \xleftarrow{\$} \mathbb{Z}_q$, $r \xleftarrow{\$} \{0,1\}^*$, $(X, Y) \xleftarrow{\$} \mathcal{A}(\mathbb{G}, q, (g^{c^i})_{i\in[k+1]}; r)$, and $(a_i)_{i\in[k]} \xleftarrow{\$} \mathsf{Ext}(\mathbb{G}, q, (g^{c^i})_{i\in[k+1]}; r)$.

The k-KZG assumption was introduced recently in [BFHK24], as a particular instance of their Uber-Knowledge assumption. The original k-KZG assumption from [BFHK24] is almost identical to our variant, with two differences: first, [BFHK24] does not explicitly specify whether $\mathcal{A}$ is uniform or non-uniform ($\mathcal{A}$ is only defined as "a PPT algorithm"), while we explicitly consider non-uniform algorithms. Second, the k-KZG variant from [BFHK24] is defined over an asymmetric pairing group, and the algorithm $\mathcal{A}$ is additionally given g_2^s as input (where g_2 is a generator of the second group). Our variant does not require pairings and does not use this additional element.
Despite these slight differences, the analysis of [BFHK24] applies directly to our variant as well. In particular, this shows that our variant is another particular instance of the Uber-Knowledge assumption, and that it holds in the generic group model.

Remark 6.2 (On auxiliary inputs). Our assumption is stated with respect to non-uniform algorithms. This is required for the following reason: in our hybrids, the simulator against a corrupted sender will define an algorithm that has the honest receiver input x hardcoded, run the (simulated) protocol, and obtain a relation (X, Y) as required for Assumption 6.1 using the code of the corrupted sender. In this step, the honest receiver input x is viewed as an arbitrary advice string (that depends only on the security parameter ℓ), hence the resulting (bounded) non-uniformity of the algorithm. We note that this use of a non-uniform advice does not run into the impossibility result of [BCPR14], as discussed in their paper.
In our application to registered VOLE, the non-uniform formulation captures the setting of a VOLE where the receiver's input is defined ahead of time. A useful alternative is to let the receiver inputs to the registered VOLE be defined via an efficient sampling procedure that depends on adversarially-provided inputs. This is required for example in our application to threshold ECDSA, where the VOLE inputs are computed in particular from an adversarially-chosen nonce. For this generalization, it is necessary to consider a strengthening of k-KZG with auxiliary inputs sampled from a fixed "benign" distribution. As discussed in [BCPR14, Remark 2.2], knowledge of exponent assumptions with respect to benign auxiliary inputs are not ruled out by their impossibility result, and are used in countless works on NIZKs, SNARKs, and targeted malleability (see [CHMMVW20] for one example).

6.2 Security of the Registered VOLE

We first prove that the protocol Π_{VOLE} (Protocol 4) securely instantiates the registered VOLE game $\mathcal{G}_{\mathsf{VOLE}}$ (Game 1) according to Definition 3.4.
The proof proceeds via two sequences of hybrids, one for each corruption case. In each hybrid, we consider a distinguisher that is given the full view of the adversary and outputs a bit b at the end of the protocol. We write $\Pr[\mathsf{Hybrid}_i]$ to denote the probability that the distinguisher outputs 1 at the end of the protocol in the i-th hybrid.

Theorem 6.3. *In the random oracle model, assuming the k-KZG assumption (Assumption 6.1), the 2-power-SEDL assumption, the DCR assumption, the hardness of factoring strong RSA moduli, and the hardness of the discrete logarithm problem in $\mathbb{G}$, protocol Π_{VOLE} (Protocol 4) securely realizes the Registered VOLE game (Game 1) in the $\mathcal{F}_{\mathsf{setup}}$-hybrid model (Functionality 1).*

Remark 6.4 (Composition with the concrete setup). By Theorem C.1, the concrete setup protocol Π_{setup} (Algorithm 7) securely realizes $\mathcal{F}_{\mathsf{setup}}$; this is where the quadratic residuosity assumption and the security properties of the proofs $\Pi^{\mathrm{FS}}_{\mathsf{mod}}$, $\Pi^{\mathrm{FS}}_{\mathsf{fac}}$, and $\Pi^{\mathrm{FS}}_{\mathsf{par}}$ are used. By modular composition (Remark 3.6), the composed protocol $\Pi^{\Pi_{\mathsf{setup}}}_{\mathsf{VOLE}}$ securely realizes the Registered VOLE game under the union of the assumptions of Theorem 6.3 and Theorem C.1.

In each sequence, we construct the simulator step by step. We start with a simulator Sim that simply runs the code of the honest party and knows its secrets, and through a sequence of hybrids we progressively strip away the "real" layers of the interaction until Sim fully emulates the random oracle and the ideal experiment with oracle access to $\mathcal{G}_{\mathsf{VOLE}}$. After each hybrid we state and prove a claim that it is indistinguishable from the previous one. The simulator is straight-line and does not rewind the adversary at any point, so it suffices to simulate the protocol for a single $(\mathsf{register}, x)$ call followed by a single (extend, a, b) call.

From a single call to many. This single-call analysis suffices even though the Registered VOLE game is designed to be invoked repeatedly—our threshold-ECDSA application, for instance, issues one extension per signature (and only a single registration). No separate composition theorem is needed to cover this. The functionality $\mathcal{G}_{\mathsf{VOLE}}$ already allows for arbitrarily many register/extend calls, and in the augmented standalone model the adversary decides, through the public-input tape (Remark 3.1), how many registrations and extensions are issued and in what order. The single simulator we construct already accounts for this more general case, where each additional registration and extension is handled by the same simulator and the order of the calls does not matter. We remark that this setting suffices to capture the scenario where any number L of pairs of parties can generate shared ECDSA keys at the cost of a factor-L security loss (the standard security degradation in the setting of multi-target attacks, where the adversary succeeds if it obtains a forgery for *any* key pair): the simulator in the L-key setting simply guesses the key for which the forgery will happen (with success probability $1/L$), emulates all signing sessions for this ECDSA key using the single-call simulator (interacting with the ECDSA functionality), and honestly plays the role of the uncorrupted parties for all other keys. See Section 6.3 in the full version of [ABCGJM26] for more details on this standard strategy.

6.2.1 Proof of Theorem 6.3: corrupted sender.

We consider a corrupted sender C running in polynomial time and making at most $2^{\ell/2}$ queries to the random oracle. Sim plays the role of the honest receiver; the transition to the functionality happens at Hybrid_9. Recall that in the protocol of Protocol 4, the parties obtain the setup data from the setup functionality $\mathcal{F}_{\mathsf{setup}}$; since the receiver (the server of $\mathcal{F}_{\mathsf{setup}}$) is honest in this corruption case, the honest branch of $\mathcal{F}_{\mathsf{setup}}$ is executed, and in particular no zero-knowledge proofs are exchanged during initialization.

Hybrid_1. This hybrid is identical to the original game, except that the simulator locally emulates (honestly) the answer of the random oracle H, maintaining a list L of pairs $(\mathsf{query}, \mathsf{answer})$ to the

oracle, and locally emulates the setup functionality $\mathcal{F}_{\mathsf{setup}}$ (Functionality 1). Since the receiver is honest, this amounts to the following: during initialization, Sim samples $(N; p_1, p_2) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $d \xleftarrow{\$} [2^{2\ell}]$, $u_0 \xleftarrow{\$} \mathbb{QR}_N$, and $h \xleftarrow{\$} \mathbb{G}$, sets $t_0 := u_0^d \bmod N$ and $s_0 := t_0^d \bmod N$, and delivers (N, u_0, t_0, s_0, h) to the corrupted sender (and to the emulated honest receiver, which keeps the trapdoor (p_1, p_2, d)). In the sequel, we denote by $\mathsf{SetupData}$ the setup data derived from (N, u_0, t_0, s_0, h), with associated public key $\mathsf{pk_{PE}} := (N, u_0, u, t_0, t, s_0, s)$ where $(u, t, s) := (u_0^{2N}, t_0^{2N}, s_0^{2N}) \bmod N^2$. During registration, the receiver runs $(\mathsf{e}_x, \mathsf{d}_x) \xleftarrow{\$} \mathsf{VOLE.Register}^{\mathsf{H}}(\mathsf{SetupData}, x)$ and sends (e_x, X) to the sender. For each Send session, upon receiving $m = (\mathsf{salt}, \pi, e, c, z)$ from the sender, the receiver returns $\mathsf{VOLEOutput}^{\mathsf{H},\mathsf{H}'}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_x, \mathsf{d}_x, m)$. Because the change is only conceptual, Hybrid_1 is perfectly indistinguishable from the original game.

Hybrid_2. In this hybrid, we modify the emulation of $\mathsf{VOLEOutput}^{\mathsf{H},\mathsf{H}'}$. First, Sim parses $\mathsf{pk_{PE}}$ from $\mathsf{SetupData}$ and N, u_0 from $\mathsf{pk_{PE}}$. Then, upon receiving $m = (\mathsf{salt}, \pi, e, c, z)$ from the sender, Sim does the following:

1. (same as before) Sim checks whether $c \in [2^{n_1}]$. If not, Sim returns $\perp$.
2. Sim searches through its list L of random oracle query-answer pairs for all pairs of the form $((\mathcal{C}_i, \mathsf{salt}), e)$ for some $\mathcal{C}_i \in \mathbb{Z}_N^*$. If the list contain no such pair, Sim returns $\perp$.
3. Else, Sim checks whether $\mathcal{C}_i = \pi^{d-e} \cdot u_0^c \bmod N$ for some i; if no check passes, Sim returns $\perp$.
4. Eventually, Sim computes w as in Hybrid_1.

Claim 6.5. $|\Pr[\mathsf{Hybrid}_2] - \Pr[\mathsf{Hybrid}_1]| \leq 2^{-\ell}$.

Proof. Observe that Sim behaves differently in both games only to determine whether or not to output $\perp$: its behavior conditioned on not outputting $\perp$ is identical. If Sim does not return $\perp$ in Hybrid_2, then it holds in particular that $e = \mathsf{H}(\mathcal{C}_i, \mathsf{salt})$ and $\mathcal{C}_i = \pi^{d-e} \cdot u_0^c \bmod N$ for some i, hence Sim does not output $\perp$ in Hybrid_1 either. Conversely, if Sim does not output $\perp$ in Hybrid_1, then it entails that $c \in [2^{n_1}]$ and $e = \mathsf{H}(\mathcal{C}, \mathsf{salt})$ with $\mathcal{C} = \pi^{d-e} \cdot u_0^c \bmod N$. Two cases can occur: either $(\pi^{d-e} \cdot u_0^c, \mathsf{salt})$ occurs as a query in L, in which case Sim does not output $\perp$ either in Hybrid_2. Or $(\pi^{d-e} \cdot u_0^c, \mathsf{salt})$ does not occur as a query in L, in which case Sim samples the value of $\mathsf{H}(\pi^{d-e} \cdot u_0^c, \mathsf{salt})$ uniformly at random (independently of all previous queries) from $\{0,1\}^\ell$, and $\Pr[\mathsf{H}(\pi^{d-e} \cdot u_0^c, \mathsf{salt}) = e] = 2^{-\ell}$. This concludes the proof. □

Hybrid_3. In this hybrid, we let Sim rely on the zero-knowledge simulator of $\Pi^{\mathrm{FS}}_{\mathsf{encdl}}$ to emulate the proof ψ_{encdl} during the emulation of $\mathsf{VOLE.Register}^{\mathsf{H}}$, programming the random oracle as required by the simulator. Namely, Sim sets $\psi_{\mathsf{encdl}} \xleftarrow{\$} \mathsf{Sim}_{\mathsf{encdl}}(\mathbb{G}, g, q, X, \mathcal{E}, \mathsf{pk_{PE}}, \log q, 2\ell)$. Under the zero-knowledge property of $\Pi^{\mathrm{FS}}_{\mathsf{encdl}}$, Hybrid_3 is indistinguishable from Hybrid_2.

Hybrid_4. In this hybrid, we modify the emulation of $\mathcal{F}_{\mathsf{setup}}$ to sample $d \xleftarrow{\$} \mathbb{Z}_{\phi(N)/4}$ instead of $d \xleftarrow{\$} [2^{2\ell}]$. Recall that u_0 is sampled from $\mathbb{QR}_N$, a cyclic group of order $\phi(N)/4$, so that $d \xleftarrow{\$} \mathbb{Z}_{\phi(N)/4}$ induces the uniform distribution over the exponents of u_0. Since the adversary's view about d is exactly $(u_0, t_0 = u_0^d, s_0 = u_0^{d^2})$ with $u_0 \xleftarrow{\$} \mathbb{QR}_N$, distinguishing Hybrid_4 from Hybrid_3 is equivalent to breaking the 2-power-SEI assumption (Assumption 3.11), whose challenge is precisely of this form (up to the exponent domains — $[2^{2\ell}]$ and $\mathbb{Z}_{\phi(N)/4}$ here, versus $\pm 2^{2\ell}$ and $\pm N$ in Assumption 3.11

— a variation covered by Remark E.4), and which is equivalent to the 2-power-SEDL assumption (Assumption 3.10) by Corollary E.3.

Hybrid_5. In this hybrid, we bound the number of solutions $\mathcal{C}_i$ to $\mathsf{H}(\mathcal{C}_i, \mathsf{salt}) = e$ for any e: if L contains at least 4 pairs $(\mathsf{query}_i, \mathsf{answer})$ with the same answer answer, Sim returns $\perp$. Let $Q \leq 2^{\ell/2}$ denote an upper bound on the number of queries of C to H. Since H maps each query to a uniformly random element of $\{0,1\}^\ell$, the probability of any 4-collision happening is at most

$$2^\ell \cdot \binom{Q}{4} \cdot \left(\frac{1}{2^\ell}\right)^4 \leq 2^\ell \cdot \frac{(2^{\ell/2})^4}{24 \cdot 2^{4\ell}} \leq \frac{1}{24 \cdot 2^\ell},$$

which also bounds the probability of distinguishing Hybrid_5 from Hybrid_4.

Hybrid_6. In this hybrid, we further alter the emulation of $\mathsf{VOLEOutput}^{\mathsf{H},\mathsf{H}'}$. Specifically, we change the way Sim checks $\mathcal{C}_i$ in step 3 of Hybrid_2. Let r denote the random tape of the corrupted sender C. For the purpose of the proof, we view C as an algorithm receiving as input $(\mathsf{SetupData}, X, \mathsf{e}_x)$ and returning either $\perp$ or a tuple $m = (\mathsf{salt}, \pi, e, c, z)$. W.l.o.g., we assume that C always returns a tuple $m = (\mathsf{salt}, \pi, e, c, z)$ (since we can replace $\perp$ by any dummy tuple).
Sim starts by emulating $\mathcal{F}_{\mathsf{setup}}$ and $\mathsf{VOLE.Register}^{\mathsf{H}}$ as in Hybrid_5, using a random tape $\tilde{r}$ to sample the Pedersen base h and to simulate the proof ψ_{encdl}, and randomness γ for $\mathcal{E}$. Then, Sim samples $d \xleftarrow{\$} \mathbb{Z}_{\phi(N)/4}$ and defines $\tilde{\mathbb{G}} := \mathbb{QR}_N$, $\tilde{q} := \phi(N)/4$, and $\tilde{g} := u_0$. Now, for $i = 1$ to 3, Sim defines the following algorithm $\mathcal{A}_i$, with x hardcoded in its description, which takes as input $(\tilde{\mathbb{G}}, \tilde{q}, (\tilde{g}^{d^i})_{i\leq 2})$ and a random tape $r||\tilde{r}||\gamma$: (the first three steps below are identical for all $\mathcal{A}_i$'s, the last one differs)

- **Setup.** $\mathcal{A}_i(\tilde{\mathbb{G}}, \tilde{q}, (\tilde{g}^{d^i})_{i\leq 2}; r||\tilde{r}||\gamma)$ reconstructs N from $\phi(N)$, sets $(u, t, s) = (u_0^{2N}, t_0^{2N}, s_0^{2N}) \bmod N^2$, and uses $\tilde{r}$ to sample the Pedersen base $h \xleftarrow{\$} \mathbb{G}$, obtaining the setup data (N, u_0, t_0, s_0, h) and the derived $\mathsf{SetupData}$ exactly as in the emulation of $\mathcal{F}_{\mathsf{setup}}$.
- **Registration.** $\mathcal{A}_i$ uses x and γ to reconstruct $X = g^x$, $x' = \alpha \cdot (\alpha^{-1} \cdot x \bmod q)$, and $\mathcal{E} = \mathsf{PE}^*.\mathsf{Enc}_{\mathsf{pk}_{\mathsf{PE}}}(x'; \gamma)$ (cf. Algorithm 2), and $\tilde{r}$ to reconstruct the simulated proof ψ_{encdl}.
- **Extension.** $\mathcal{A}_i$ runs the corrupted sender algorithm C on input $(\mathsf{SetupData}, X, \mathsf{e}_x)$ with random tape r and receives $m = (\mathsf{salt}, \pi, e, c, z)$.
- **Output.** $\mathcal{A}_i$ searches through the list L of random oracle queries for the i-th pair $((\mathcal{C}_i, \mathsf{salt}), e)$. If there are less than i such pairs, it returns $\perp$. Else, it returns $(\pi, \mathcal{C}_i \cdot u_0^{-c} \cdot \pi^e \bmod N)$.

We now modify Sim's check in step 3 of Hybrid_5 (which is the same as in Hybrid_2) as follows:

3.1 For each algorithm Adv_i, Sim runs $(a_i, \theta_i) \leftarrow \mathsf{Ext}_i(\tilde{\mathbb{G}}, \tilde{q}, (\tilde{g}^{d^i})_{i\leq 2}; r||\tilde{r}||\gamma)$, where Ext_i is the extractor whose existence is guaranteed by Assumption 6.1.

3.2 Sim checks if there exists i such that $\mathcal{C}_i \cdot u_0^{-c} \cdot \pi^e = s_0^{a_i} t_0^{\theta_i} \bmod N$ and $\pi = t_0^{a_i} u_0^{\theta_i} \bmod N$. If no pair satisfies both conditions, Sim returns $\perp$.

Claim 6.6. *If Assumption 6.1 holds,* $|\Pr[\mathsf{Hybrid}_6] - \Pr[\mathsf{Hybrid}_5]| \leq \mathrm{neg}(\ell)$.

Proof. If Sim does not return $\perp$ in Hybrid_6, then it holds that there exists $i \leq 3$ such that $e = \mathsf{H}(\mathcal{C}_i, \mathsf{salt})$, $\mathcal{C}_i \cdot u_0^{-c} \cdot \pi^e = s_0^{a_i} t_0^{\theta_i} \bmod N$, and $\pi = t_0^{a_i} u_0^{\theta_i} \bmod N$. If this is the case, then since $u_0^d = t_0 \bmod N$ and $t_0^d = s_0 \bmod N$, we get $\pi^d = s_0^{a_i} t_0^{\theta_i} = \mathcal{C}_i \cdot u_0^{-c} \cdot \pi^e \bmod N$, hence $\mathcal{C}_i =$

$\pi^{d-e} \cdot u_0^c \bmod N$ and Sim does not output $\perp$ in step 3 of Hybrid_5. Conversely, if Sim does not output $\perp$ in Hybrid_5, then there exists $i \leq 3$ such that $\mathcal{C}_i = \pi^{d-e} \cdot u_0^c \bmod N$, then $\pi^d = \mathcal{C}_i \cdot u_0^{-c} \cdot \pi^e \bmod N$, hence A_i outputs a pair (π, X) such that $\pi^d = X$. By Assumption 6.1, it therefore holds with overwhelming probability that Ext_i outputs (a_i, θ_i) such that $X = \tilde{g}^{a_i + d\cdot\theta_i} = s_0^{a_i} t_0^{\theta_i} \bmod N$, which entails that $\pi = X^{d^{-1}} = t_0^{a_i} u_0^{\theta_i} \bmod N$, and Sim does not output $\perp$ in Hybrid_6. This concludes the proof. □

Hybrid_7. In this hybrid, Sim behaves as in Hybrid_6, but additionally abort if the number of distinct pairs (a_i, θ_i) passing the check in step 3.2 is not exactly one.

Claim 6.7. *Under the hardness of factoring,* $|\Pr[\mathsf{Hybrid}_7] - \Pr[\mathsf{Hybrid}_6]| \leq \mathrm{neg}(\ell)$.

Proof. Assume that two distinct pairs (a_i, θ_i) satisfy $\pi = t_0^{a_i} u_0^{\theta_i} \bmod N$, and let $(\Delta_a, \Delta_\theta)$ denote their difference. Then we have $u_0^{d\cdot\Delta_a + \Delta_\theta} = 1 \bmod N$, which entails that $d \cdot \Delta_a + \Delta_\theta$ is a multiple of $\phi(N)/4$ (since u_0 is a generator of $\mathbb{QR}_N$ with overwhelming probability), and finding such a multiple enables factoring N efficiently (see e.g. Fact 4 from [CPP17]). This concludes the proof. □

We note that the corresponding binding property of the commitment is derived in [ABCGJM26] from the strong RSA assumption; the argument above only uses the hardness of factoring, a strictly weaker assumption.

Note that from Hybrid_7, Sim does not need to know d anymore in its emulation. Conditioned on not aborting, Sim obtains a unique pair $(\tilde{a}, \theta)$ such that $\pi = t_0^{\tilde{a}} u_0^{\theta} \bmod N$ and $\mathcal{C} = s_0^{\tilde{a}} t_0^{\theta} u_0^c \pi^{-e} = s_0^{\tilde{a}} \cdot t_0^{\theta - e\tilde{a}} \cdot u_0^{c-e\theta} \bmod N$. Let $\tilde{\rho}_0 \leftarrow \theta - e\tilde{a}$ and $\tilde{\rho}_1 \leftarrow c - e\theta$. Note that by construction, $c = e^2\tilde{a} + e\tilde{\rho}_0 + \rho_1$.

Hybrid_8. In this hybrid, Sim behaves as in Hybrid_7 but additionally checks that $|\tilde{a}| \leq 2^{n_1}$. If any check fails, Sim returns $\perp$.

Claim 6.8. *Under the hardness of factoring,* $|\Pr[\mathsf{Hybrid}_8] - \Pr[\mathsf{Hybrid}_7]| \leq \mathrm{neg}(\ell)$.

Proof. We show that the probability of Sim returning $\perp$ in Hybrid_8 but not in Hybrid_7 (i.e., returning $\perp$ because of a failed range check for $|\tilde{a}|$) is negligible. Let δ denote the probability that $|\tilde{a}| > 2^{n_1}$. Let $\mathcal{A}$ denote the following algorithm: on input x and a Q-tuple of uniformly random elements $(h_1, \ldots, h_Q) \in (\{0,1\}^\ell)^Q$, it emulates the interaction between Sim and the corrupted sender C in Hybrid_8, answering the j-th oracle (distinct) query of C with h_j. After receiving (π, e, c, z) from C, recovering the query-answer pairs $((\mathcal{C}_i, \mathsf{salt}), e)_{i\leq 3}$, and running the three extractors Ext_i (getting a unique pair $(\tilde{a}, \theta)$; let us denote $\mathcal{C}$ the lexicographically first $\mathcal{C}_i$ that produced the unique pair correctly), if Sim's tuple $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ computed from the output of Ext_i (for some $i \leq 3$) is such that $|\tilde{a}| > 2^{n_1}$, it retrieves the corresponding query-answer pair $((\mathcal{C}, \mathsf{salt}), e)$ (from which the output of $\mathcal{A}_i$ was computed) and outputs (j^*, so), where j^* is the index of the query $(\mathcal{C}, \mathsf{salt})$ and $\mathsf{so} = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e, \pi, c, \tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$. Else, $\mathcal{A}$ outputs $(0, \perp)$.

By the Generalized Forking Lemma (Lemma 3.8), denoting D_{IG} the distribution that always output x, 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

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

and when $b = 1$,

$$\mathsf{so} = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e, \pi, c, \tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1),$$
$$\mathsf{so}' = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e', \pi', c', \tilde{a}', \tilde{\rho}_0', \tilde{\rho}_1'),$$

where:

- $\mathcal{C} = s_0^{\tilde{a}} \cdot t_0^{\tilde{\rho}_0} \cdot u_0^{\tilde{\rho}_1} = s_0^{\tilde{a}'} \cdot t_0^{\tilde{\rho}'_0} \cdot u_0^{\tilde{\rho}'_1} \bmod N$,
- $|\tilde{a}| > 2^{n_1}$,
- $c = e^2\tilde{a} + e\tilde{\rho}_0 + \tilde{\rho}_1 \in [0, 2^{n_1})$ and $c' = (e')^2\tilde{a}' + e'\tilde{\rho}'_0 + \tilde{\rho}'_1 \in [0, 2^{n_1})$.

By the same argument as for Claim 6.8, under the hardness of factoring, it must hold with overwhelming probability that $\tilde{a} = \tilde{a}'$, $\tilde{\rho}_0 = \tilde{\rho}'_0$, $\tilde{\rho}_1 = \tilde{\rho}'_1$. This means that the value of the tuple $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ is fixed after the first run of $\mathcal{A}$. However, $\mathsf{F}_{\mathcal{A}}$ samples e' as a uniformly random element of $\{0,1\}^\ell$ (independently of the tuple $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ extracted in the first run). Over the choice of a random e', it holds that

$$\mathsf{p}_{\mathsf{fork}} \leq \Pr_{e' \xleftarrow{\$} \{0,1\}^\ell}[(e')^2\tilde{a}' + e'\tilde{\rho}'_0 + \tilde{\rho}'_1 \in [0, 2^{n_1})] \leq \frac{2}{2^\ell},$$

which implies that δ is negligible. This concludes the proof. □

Hybrid_9. In this game, we replace the real output w of the corrupted sender with an ideal output $\tilde{w}$ by letting Sim query the Registered VOLE game on behalf of C. Namely, Sim computes $a \leftarrow \tilde{a} \bmod q$, $\tilde{V} = \mathcal{E}_t^{\tilde{a}} \cdot \mathcal{E}_u^{\theta} \bmod N^2$, and $b = z - [\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N \bmod q$, and sends (a, b) to the Registered VOLE game.

Claim 6.9. $|\Pr[\mathsf{Hybrid}_9] - \Pr[\mathsf{Hybrid}_8]| \leq \mathrm{neg}(\ell)$.

Proof. In Hybrid_8, $\pi = t_0^{\tilde{a}} u_0^{\theta} \bmod N$, hence $\pi^{2N\gamma} = t^{\gamma\tilde{a}} u^{\gamma\theta} \bmod N^2$. Let $x' := \alpha \cdot (\alpha^{-1} \cdot x \bmod q)$ denote the plaintext of $\mathcal{E}$ (cf. Algorithm 2); recall $x' \equiv x \bmod q$ and $0 \leq x' < \alpha \cdot q$. Denoting $\tilde{V} = \mathcal{E}_t^{\tilde{a}} \mathcal{E}_u^{\theta}$, we have by construction of $\mathcal{E}$ that $\tilde{V} = ((1+N)^{x'} t^{\gamma})^{\tilde{a}} \cdot (u^{\gamma})^{\theta} = \pi^{2N\gamma} \cdot (1+N)^{x'\tilde{a}} \bmod N^2$. This implies that $(\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)) - (\mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) + \mathsf{H}'(e)) = \mathsf{DDLOG}_N(\tilde{V}) - \mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) = x'\tilde{a} \bmod N$. We rely on the following claim:

Claim 6.10. $\Pr[[|\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}] \leq \mathrm{neg}(\ell)$, *where the probability is over the randomness of the adversary and the random oracle.*

Proof. The proof uses the Generalized Forking Lemma (Lemma 3.8). Let δ denote the probability that $|[\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}$. Let $\mathcal{A}$ denote the following algorithm: on input x and a Q-tuple of uniformly random elements $(h_1, \ldots, h_Q) \in (\{0,1\}^\ell)^Q$, it emulates the interaction between Sim and the corrupted sender C in Hybrid_9, answering the j-th oracle (distinct) query of C with h_j. After receiving (π, e, c, z) from C, recovering the query-answer pairs $((\mathcal{C}_i, \mathsf{salt}), e)_{i \leq 3}$, and running the three extractors Ext_i (getting a unique pair $(\tilde{a}, \theta)$; let us denote $\mathcal{C}$ the lexicographically first $\mathcal{C}_i$ that produced the unique pair correctly) and let $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ be the triple computed from the output of Ext_i (for some $i \leq 3$). Let $\tilde{V} = \mathcal{E}_t^{\tilde{a}} \mathcal{E}_u^{\tilde{a} \cdot e + \tilde{\rho}_0}$. If $|[\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}$, $\mathcal{A}$ retrieves the corresponding query-answer pair $((\mathcal{C}, \mathsf{salt}), e)$ (from which the output of $\mathcal{A}_i$ was computed) and outputs (j^*, so), where j^* is the index of the query $(\mathcal{C}, \mathsf{salt})$ and $\mathsf{so} = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e, \tilde{V}, \tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$. Else, $\mathcal{A}$ outputs $(0, \perp)$.
By the Generalized Forking Lemma (Lemma 3.8), denoting D_{IG} the distribution that always output x, 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

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

and when $b = 1$,

$$\mathsf{so} = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e, \tilde{V}, \tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1),$$
$$\mathsf{so}' = (\mathsf{SetupData}, \mathcal{E}, \mathcal{C}, e', \tilde{V}', \tilde{a}', \tilde{\rho}'_0, \tilde{\rho}'_1),$$

where:

- $\mathcal{C} = s_0^{\tilde{a}} \cdot t_0^{\tilde{\rho}_0} \cdot u_0^{\tilde{\rho}_1} = s_0^{\tilde{a}'} \cdot t_0^{\tilde{\rho}'_0} \cdot u_0^{\tilde{\rho}'_1} \bmod N$,
- $\tilde{V}' = \mathcal{E}_t^{\tilde{a}'} \mathcal{E}_u^{\tilde{a}' \cdot e' + \tilde{\rho}'_0}$,
- $|[\mathsf{DDLOG}_N(\tilde{V}') + \mathsf{H}'(e')]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}$.

By the same argument as for Claim 6.8, under the hardness of factoring, it must hold with overwhelming probability that $\tilde{a} = \tilde{a}'$, $\tilde{\rho}_0 = \tilde{\rho}'_0$, $\tilde{\rho}_1 = \tilde{\rho}'_1$. This means that the value of the tuple $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ is fixed after the first run of $\mathcal{A}$. However, $\mathsf{F}_{\mathcal{A}}$ samples e' as a uniformly random element of $\{0,1\}^\ell$ (independently of the tuple $(\tilde{a}, \tilde{\rho}_0, \tilde{\rho}_1)$ extracted in the first run). Over the choice of a random e' and for fixed $\mathcal{E}_t, \mathcal{E}_u, \tilde{a}, \tilde{\rho}_0$, it holds that

$$\begin{aligned} \mathsf{p}_{\mathsf{fork}} \leq p' &= \Pr_{e' \xleftarrow{\$} \{0,1\}^\ell}[|[\mathsf{DDLOG}_N(\mathcal{E}_t^{\tilde{a}} \mathcal{E}_u^{\tilde{a} \cdot e' + \tilde{\rho}_0}) + \mathsf{H}'(e')]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}] \\ &\leq \mathrm{neg}(\ell), \end{aligned}$$

where the last inequality uses the following facts:

- $\Pr_{v \xleftarrow{\$} \mathbb{Z}_N}[|[\mathsf{DDLOG}_N(\mathcal{E}_t^{\tilde{a}} \mathcal{E}_u^{\tilde{a} \cdot e' + \tilde{\rho}_0}) + v]_N| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}] = \Pr_{v \xleftarrow{\$} \mathbb{Z}_N}[|v| < 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}] = \frac{2^{n_1 + 2\ell + \lceil \log \alpha \rceil + 1}}{N} < \frac{2}{2^\ell}$ (by our choice of N, n_1, and α, using $n_1 + 3\ell + \lceil \log \alpha \rceil \leq \log N$).
- $\mathsf{H}'(e')$ is sampled as a random element of $\mathbb{Z}_N$, independently of e'.

which implies that δ is negligible. This concludes the proof. □

Using Claim 6.10 together with the fact that $|\tilde{a}x'| \leq 2^{n_1} \cdot \alpha q \leq 2^{n_1 + 2\ell + \lceil \log \alpha \rceil}$, it follows that except with negligible probability, $[\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N - [\mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) + \mathsf{H}'(e)]_N = \tilde{a}x'$ holds over the integers, and therefore

$$\begin{aligned} w &= z - [\mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) + \mathsf{H}'(e)]_N \\ &= b + [\mathsf{DDLOG}_N(\tilde{V}) + \mathsf{H}'(e)]_N - [\mathsf{DDLOG}_N(\pi^{2N\gamma} \bmod N^2) + \mathsf{H}'(e)]_N \\ &= b + \tilde{a}x' = b + ax = \tilde{w} \bmod q, \end{aligned}$$

where the last equality uses $\tilde{a} = a \bmod q$ and $x' = x \bmod q$, which concludes the proof. □

Hybrid_{10}. Observe that in Hybrid_9, neither the factorization of N, d, or γ are used in the simulation. In this hybrid, we replace $\mathcal{E}$ with $\tilde{\mathcal{E}} \leftarrow \mathsf{PE}^* \,.\, \mathsf{Enc}_{\mathsf{pk}_{\mathsf{PE}}}(0; \gamma)$. Observe that Sim does not need anymore to know S's private input x to simulate. Under the IND-CPA security of the extended Paillier-ElGamal encryption scheme — which reduces to the power-SEDL assumption (Assumption 3.10) and the DCR assumption, and whose proof applies verbatim to the key distribution of the present hybrid, where $u_0 \xleftarrow{\$} \mathbb{QR}_N$ (as in Assumption 3.10) and d is sampled from the full range $\mathbb{Z}_{\phi(N)/4}$ — Hybrid_{10} is indistinguishable from Hybrid_9. This concludes the proof. □

6.2.2 Proof of Theorem 6.3: corrupted receiver.

We consider a corrupted receiver S running in polynomial time and making at most $2^{\ell/2}$ queries to the random oracle. Sim plays the role of the honest sender; the transition to the functionality happens at Hybrid_2.
We first state and prove a useful claim which will let us show that the view of the corrupted receiver is statistically independent of the sender input a:

Claim 6.11. *Fix $f \leq 2^{\mathsf{s}}$. Set $n_r := \mathsf{s} + \nu$, $n_0 := 4\ell + \mathsf{s} + 3\nu + 1$, and $n_1 := 5\ell + \mathsf{s} + 3\nu + 1$. Fix $d \in \pm 2^{2\ell+\nu+1}$ and $e \in [2^\ell]$. Consider the following distributions, where $\mathcal{D}_0[a]$ is parametrized by $a \in \mathbb{Z}_q$ for a prime q with $2^{\mathsf{s}} < q$ and $\log(q+1) \leq 2\ell$:*

$\mathcal{D}_0(a)$: *returns $(A, c, [a']_f, [\theta]_f)$, defined as*

- $r \xleftarrow{\$} [2^{n_r}]$, $\rho_0 \xleftarrow{\$} [2^{n_0}]$, $\rho_1 \xleftarrow{\$} [2^{n_1}]$
- $a' \leftarrow a + q \cdot r$
- $A \leftarrow (d+e) \cdot a' + \rho_0$
- $\theta \leftarrow e \cdot a' + \rho_0$
- $c \leftarrow e^2 \cdot a' + e \cdot \rho_0 + \rho_1$

$\mathcal{D}_1$: *returns $(A, c, [a']_f, [\theta]_f)$, defined as*

- $r \xleftarrow{\$} [2^{n_r}]$, $A' \xleftarrow{\$} [2^{n_1}]$
- $a' \leftarrow q \cdot r$
- $A \xleftarrow{\$} [2^{n_0}]$
- $\theta \leftarrow A - d \cdot a'$
- $c \leftarrow e \cdot A + A'$

Then for every $a \in \mathbb{Z}_q$, the statistical distance between $\mathcal{D}_0(a)$ and $\mathcal{D}_1$ is at most $2^{-\nu+2} + 2^{-\nu}$.

Proof. In $\mathcal{D}_0(a)$, let $A' \leftarrow \rho_1 - a' \cdot d \cdot e$. Rewrite $\theta = A - d \cdot a'$ and $c = e \cdot A + A'$. With this rewriting, observe that $[\theta]_f$ can be directly computed from $[a']_f$, A, and d. Now, consider the following distribution $\mathcal{D}'_0(a)$: replace $A = (d+e) \cdot a' + \rho_0$ with $A \leftarrow \rho_0$, $A' = -d \cdot e \cdot a' + \rho_1$ with $A' \leftarrow [2^{n_1}]$, and compute all remaining values $(c, [\theta]_f)$ from $[a']_f, A$, and the public values (d, e). We claim that the statistical distance between $\mathcal{D}_0(a)$ and $\mathcal{D}'_0(a)$ is at most $2^{-\nu+2}$. This follows from the fact that (recalling that d may be negative)

$$|(d+e) \cdot a'| = |d+e| \cdot (a + q \cdot r) \leq (2^{2\ell+\nu+1} + 2^\ell) \cdot (q+1) \cdot 2^{n_r} \leq 2^{4\ell+\nu+2} \cdot 2^{n_r} = 2^{n_0-\nu+1},$$

using $q + 1 \leq 2^{2\ell} + 1$ and $(2^{2\ell+\nu+1} + 2^\ell)(2^{2\ell} + 1) = 2^{4\ell+\nu+1} + 2^{2\ell+\nu+1} + 2^{3\ell} + 2^\ell \leq 2^{4\ell+\nu+2}$. Hence, since $\rho_0 \xleftarrow{\$} [2^{n_0}]$, it smudges the (signed) term $(d+e) \cdot a'$ up to statistical distance $2^{-\nu+1}$. Similarly, assuming that A is already sampled from $[2^{n_0}]$, we have $|d \cdot e \cdot a'| \leq 2^{2\ell+\nu+1} \cdot 2^\ell \cdot (q+1) \cdot 2^{n_r} \leq 2^{5\ell+\nu+2} \cdot 2^{n_r} = 2^{n_1-\nu+1}$, i.e., $-d \cdot e \cdot a' \in \pm 2^{n_1-\nu+1}$, and ρ_1 is sampled from $[2^{n_1}]$, hence ρ_1 smudges the term $-d \cdot e \cdot a'$ up to statistical distance $2^{-\nu+1}$. By a union bound, the total distance is $2^{-\nu+1} + 2^{-\nu+1} = 2^{-\nu+2}$.
Now, observe that in $\mathcal{D}'_0(a)$, the tuple $(A, c = eA + A', [a']_f, [\theta]_f = [A - da']_f)$ depends solely on $[a']_f$ and values A, A', d, e independent of a'. Because $q > f$ is prime and r is sampled randomly

from $[2^{n_r}]$ with $n_r = \mathsf{s} + \nu$, the distribution of $[q^{-1} \cdot a + r]_f$ is $2^{-\nu}$-close to the distribution of $[r]_f$. Replacing $q^{-1} \cdot a + r$ with r in $[a + qr]_f$ yields $[qr]_f$, which is exactly the distribution $\mathcal{D}_1$. This concludes the proof, yielding a total statistical distance of $2^{-\nu+2} + 2^{-\nu}$. □

Equipped with the claim, we now proceed with the hybrids:

Hybrid_1. This is the original game, where Sim plays as a honest sender, except that the simulator locally emulates the answer of the random oracle H, maintains a list L of pairs $(\mathsf{query}, \mathsf{answer})$ to the oracle, and locally emulates the setup functionality $\mathcal{F}_{\mathsf{setup}}$: since the receiver (the server of $\mathcal{F}_{\mathsf{setup}}$) is corrupted, Sim runs the corrupt-S branch of Functionality 1, receiving $(p_1, p_2, u_0, t_0, s_0, d)$ from the adversary and performing the functionality's checks. In particular, this lets Sim obtain the factorization of N and the Paillier-ElGamal secret key d from the adversary.

Hybrid_2. In this hybrid, we change the way Sim constructs z. During registration, the simulator Sim uses Π_{encdl}'s straight-line extractor to extract from the corrupted receiver S a slack instance witness. Sim obtains $(x, \gamma, f) \in [2^{n_x}] \times \mathbb{Z} \times [2^\ell]$ such that

- $\mathcal{E}_t^f = t^\gamma \cdot (1+N)^{fx} \bmod N^2$,
- $\mathcal{E}_u^f = u^\gamma \bmod N^2$,
- $X = g^x$.

Upon extracting x, Sim sends $[x]_q$ to the Registered VOLE game on behalf of the corrupted receiver S and receives an output w. Sim samples (r, ρ_0, ρ_1) and computes $a', \mathcal{C}, e$, and c as the honest sender. Then, Sim sets $a' = Q_a \cdot f + [a']_f$ where $(Q_a, [a']_f)$ are the quotient and remainder of the integer division of a' by f. Similarly, Sim computes $\theta \leftarrow a'e + \rho_0$ and sets $\theta = Q_\theta f + [\theta]_f$. Eventually, Sim sets $\pi_q \leftarrow t_0^{Q_a} u_0^{Q_\theta} \bmod N$ and $\pi_r \leftarrow t_0^{[a']_f} u_0^{[\theta]_f} \bmod N$ (note that $\pi_q^f \pi_r = t_0^{a'} u_0^\theta = \pi \bmod N$). Sim defines

$$W \leftarrow \pi_q^{2N\gamma} \cdot ((1 - Nx) \cdot \mathcal{E}_t)^{[a']_f} \cdot \mathcal{E}_u^{[\theta]_f} \bmod N^2,$$

and sets $\tilde{z} \leftarrow w + [\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N \bmod q$. Sim sends $(\pi, e, c, \tilde{z})$ to S.

Claim 6.12. $|\Pr[\mathsf{Hybrid}_2] - \Pr[\mathsf{Hybrid}_1]| \le 1/2^\ell$.

Proof. Plugging the slack instance equations for $\mathcal{E}_t, \mathcal{E}_u$ in the equation $V = \mathcal{E}_t^{a'} \cdot \mathcal{E}_u^\theta \bmod N^2$ yields

$$\begin{aligned} V &= (\mathcal{E}_t^{Q_a} \mathcal{E}_u^{Q_\theta})^f \cdot (\mathcal{E}_t^{[a']_f} \mathcal{E}_u^{[\theta]_f}) \bmod N^2 \\ &= (t^\gamma \cdot (1+N)^{fx})^{Q_a} \cdot u^{\gamma Q_\theta} \cdot (\mathcal{E}_t^{[a']_f} \mathcal{E}_u^{[\theta]_f}) \bmod N^2 \\ &= (t^{Q_a} u^{Q_\theta})^\gamma \cdot (1+N)^{a'x - [a']_f x} \cdot (\mathcal{E}_t^{[a']_f} \mathcal{E}_u^{[\theta]_f}) \bmod N^2 \\ &= W \cdot (1+N)^{a'x} \bmod N^2. \end{aligned}$$

Now, by correctness of DDLOG (Lemma 3.17), we get

$$\begin{aligned} &\mathsf{DDLOG}_N(V) - \mathsf{DDLOG}_N(W) = x \cdot a' \bmod N \\ \implies &[\mathsf{DDLOG}_N(V) + \mathsf{H}'(e)]_N = [\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N + x \cdot a' \bmod N. \end{aligned}$$

Because H' is a random oracle, the value of $\mathsf{H}'(e)$ is a uniformly random element of $\mathbb{Z}_N$ sampled independently of $\mathsf{DDLOG}_N(W)$. Since x is part of a slack instance witness, we have $x \le 2^{n_x}$,

and by construction $a' \leq (q+1) \cdot 2^{n_r}$. It follows that $xa' \leq (q+1) \cdot 2^{n_x+n_r}$. Therefore, for $[\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N + x \cdot a'$ to cause a wraparound modulo N, it must be the case that $[\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N \geq N - (q+1) \cdot 2^{n_x+n_r}$, which happens with probability at most $(q+1) \cdot 2^{n_x+n_r}/N < 1/2^\ell$ over the randomness of H'. Hence, except with probability $1/2^\ell$ and since $a' = a \bmod q$,

$$\begin{aligned} & [\mathsf{DDLOG}_N(V) + \mathsf{H}'(e)]_N = [\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N + x \cdot a \bmod q \\ \implies & [\mathsf{DDLOG}_N(V) + \mathsf{H}'(e)]_N + b = [\mathsf{DDLOG}_N(W) + \mathsf{H}'(e)]_N + x \cdot a + b \bmod q \\ \implies & z = \tilde{z}, \end{aligned}$$

where the last equation uses the fact that $w = a \cdot x + b \bmod q$. This concludes the proof. □

Hybrid_3. In this hybrid, we let Sim sample $e \xleftarrow{\$} \{0,1\}^\ell$ ahead of time. Sim computes a', ρ_0, ρ_1 and $\mathcal{C}$ as in Hybrid_2. If L already contains a pair $(\mathsf{query}, \mathsf{answer})$ with $\mathsf{query} = (\mathcal{C}, \mathsf{salt})$, Sim returns $\perp$. Else, Sim programs H to $\mathsf{H}(\mathcal{C}, \mathsf{salt}) = e$. This game differs from Hybrid_2 only if $(\mathcal{C}, \mathsf{salt})$ was already queried, which happens with probability at most $Q/2^\ell$ since salt was sampled randomly from $[2^\ell]$.

Hybrid_4. In this hybrid, we change the way Sim computes $\mathcal{C}$ and π by relying on the challenge e (known ahead of time since Hybrid_3) and on the Paillier-ElGamal secret key d. Then, Sim sets

- $r \xleftarrow{\$} [2^{n_r}]$, $\rho_0 \xleftarrow{\$} [2^{n_0}]$, $\rho_1 \xleftarrow{\$} [2^{n_1}]$.
- $a' \leftarrow a + q \cdot r$
- $A \leftarrow a' \cdot (d+e) + \rho_0$

and defines

- $c \leftarrow e^2 \cdot a' + e \cdot \rho_0 + \rho_1$,
- $\pi_r \leftarrow u_0^{[A - a'd]_f + d \cdot [a']_f} \bmod N$,
- $\pi_q \leftarrow u_0^{(A - [A-a'd]_f - d \cdot [a']_f)/f} \bmod N$,
- $\pi \leftarrow \pi_q^f \cdot \pi_r \bmod N$,
- $\mathcal{C} \leftarrow u_0^c \cdot \pi^{d-e} \bmod N$.

Afterwards, W is computed from $\pi_q, \gamma, x, [a']_f, [\theta]_f$ and z is computed from W, w, and e as in Hybrid_3.

Claim 6.13. $|\Pr[\mathsf{Hybrid}_4] = \Pr[\mathsf{Hybrid}_3]|$.

Proof. The changes are purely syntactic and the view of S in Hybrid_4 remains identical to Hybrid_3: observe that in Hybrid_4, by construction, $\pi = u_0^A = t_0^{a'} \cdot u_0^{a'e+\rho_0} \bmod N$ and $\mathcal{C} = u_0^c \cdot \pi^{d-e}$, exactly as in Hybrid_3. □

Hybrid_5. In this hybrid, we change the way Sim computes a', A, θ, and c. Namely, Sim sets

- $r \xleftarrow{\$} [2^{n_r}]$, $A' \xleftarrow{\$} [2^{n_1}]$
- $a' \leftarrow q \cdot r$
- $A \xleftarrow{\$} [2^{n_0}]$
- $\theta \leftarrow A - d \cdot a'$
- $c \leftarrow e \cdot A + A'$

Then, Sim defines

- $\pi_r \leftarrow u_0^{[\theta]_f + d \cdot [a']_f} \bmod N$,
- $\pi_q \leftarrow u_0^{(A - [\theta]_f - d \cdot [a']_f)/f} \bmod N$,
- $\pi \leftarrow \pi_q^f \cdot \pi_r \bmod N$,
- $\mathcal{C} \leftarrow u_0^c \cdot \pi^{d-e} \bmod N$.

Claim 6.14. $|\Pr[\mathsf{Hybrid}_5] - \Pr[\mathsf{Hybrid}_4]| \leq 2^{-\nu+2} + 2^{-\nu} + \mathrm{neg}(\ell)$.

Proof. The proof is by immediate reduction to Claim 6.11: the values $\pi_r, \pi_q, \pi, \mathcal{C}, W$ are computed solely from $A, c, [a']_f, [\theta]_f$ and the values d, e. Replacing $(A, c, [a']_f, [\theta]_f) \xleftarrow{\$} \mathcal{D}_0(a)$ (as in Hybrid_4) with $(A, c, [a']_f, [\theta]_f) \xleftarrow{\$} \mathcal{D}_1$ yields Hybrid_5. □

7 Batch Registered Non-Interactive VOLE

7.1 Generalized Paillier-ElGamal Encryption

We introduce a generalization of the extended Paillier-ElGamal encryption where the public key contains additional powers of d in the exponent:

Definition 7.1. The k-Generalized Paillier-ElGamal encryption with short exponents, denoted $\mathsf{PE}^k = (\mathsf{PE}^k.\mathsf{KeyGen}, \mathsf{PE}^k.\mathsf{Enc}, \mathsf{PE}^k.\mathsf{Dec})$, is defined as follows:

1. $\mathsf{PE}^k.\mathsf{KeyGen}(1^\ell)$: Sample $(N, n_{\mathrm{rsa}}; p, q) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$. Sample $d \xleftarrow{\$} \{0,1\}^{2\ell}$. Sample $u_{-(k-1)} \xleftarrow{\$} \mathbb{J}_N$ and define $u_{-(k-1)+i} = u_{-(k-1)}^{d^i} \bmod N$ for $i = 1$ to $2k-1$. Define $\bar{u}_j = u_j^{2N} \bmod N^2$ for $j = -(k-1)$ to k. Return an extended public key $\mathsf{pk}_{\mathsf{PE}} = (N, (u_i, \bar{u}_i)_{-(k-1) \leq i \leq k})$ and a secret key $\mathsf{sk}_{\mathsf{PE}} = (p, q, d)$.
2. $\mathsf{PE}^k.\mathsf{Enc}_{\mathsf{pk}_{\mathsf{PE}}}(m; \gamma)$: on input $m \in \mathbb{Z}_N$, $\mathsf{pk}_{\mathsf{PE}} = (N, (u_i, \bar{u}_i)_{-(k-1) \leq i \leq k})$, and random coins $\gamma \xleftarrow{\$} \{0,1\}^{2\ell}$, output $(\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2 \ldots, \mathcal{E}_k) = (\bar{u}_{-(k-1)}^\gamma, \ldots, \bar{u}_0^\gamma, \bar{u}_1^\gamma \cdot (1+N)^m, \bar{u}_2^\gamma, \ldots, \bar{u}_k^\gamma) \bmod N^2$.
3. $\mathsf{PE}.\mathsf{Dec}_{\mathsf{sk}_{\mathsf{PE}}}(\mathcal{E})$. On input $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_k) \in (\mathbb{Z}_{N^2}^*)^{2k}$ and $\mathsf{sk}_{\mathsf{PE}} = (p, q, d)$, output $m = \mathsf{DLOG}_N(\mathcal{E}_1 / \mathcal{E}_0^d \bmod N^2)$, where $\mathsf{DLOG}_N(X)$ returns x if $X = (1 + xN) \bmod N^2$, and $\perp$ otherwise.

Claim 7.2. *Assume the $2k$-power-SEDL assumption and the DCR assumption. Then the k-Generalized Paillier-ElGamal encryption scheme with short exponents is IND-CPA secure.*

We omit the straightforward proof of this claim.

7.2 Generalized Pedersen / Paillier-ElGamal Consistency ($\Pi_{k\text{-dlcom}}$)

The proof Π_{dlcom} (Protocol 2) certifies that the two-component Paillier-ElGamal ciphertext $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u)$ is consistent with a tuple of Pedersen commitments. We now generalize it to the k-Generalized Paillier-ElGamal ciphertext $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \dots, \mathcal{E}_k)$ of Definition 7.1, whose $2k$ components all share a single randomness γ over the bases $(\bar{u}_i)_{-(k-1)\le i\le k}$, with the plaintext carried only in $\mathcal{E}_1 = (1+y'N)\,\bar{u}_1^{\gamma}$. Writing $I = \{-(k-1), \dots, k\} \setminus \{1\}$ for the $2k-1$ randomness-only indices, the single equation $\mathcal{E}_u = u^{\gamma}$ of Π_{dlcom} is replaced by the $2k-1$ equations $\mathcal{E}_i = \bar{u}_i^{\gamma}$ $(i \in I)$, and the plaintext equation moves from the base t to $\bar{u}_1$. The Paillier-style commitment $B = (1+\gamma N)\,\bar{u}_0^{r}$ to the randomness γ is retained, now under the designated base $\bar{u}_0$.

Definition 7.3 (Promise problem for $\Pi_{k\text{-dlcom}}$)**.** The statement is a tuple $(g, h, (\bar{u}_i)_{-(k-1)\le i\le k}, U_1, \dots, U_\tau, \mathcal{E} = (\mathcal{E}_{-(k-1)}, \dots, \mathcal{E}_k), B)$, parameterized by a chunk-magnitude bound $\kappa_0 \ge \kappa$.

- *Yes instances.* Tuples for which there exist $(y_j)_{j=0}^{\tau-1} \in (\pm 2^{\kappa_0})^{\tau}$, $\gamma \in \pm 2^{2\ell}$, $r \in \pm 2^{2\ell}$, and $(s_j)_{j=0}^{\tau-1} \in \mathbb{F}_q^{\tau}$ such that

$$U_j = g^{y_{j-1}}\, h^{s_{j-1}} \;\; (j = 1, \dots, \tau), \quad \mathcal{E}_1 = (1+y'N)\,\bar{u}_1^{\gamma} \bmod N^2,$$
$$\mathcal{E}_i = \bar{u}_i^{\gamma} \bmod N^2 \;\; (i \in I), \qquad\qquad B = (1+\gamma N)\,\bar{u}_0^{r} \bmod N^2.$$

  where $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$.

- *Slack instances.* Tuples for which N admits small factors, or there exist $(y_j)_{j=0}^{\tau-1} \in (\pm 2^{\kappa_0+\varepsilon+1})^{\tau}$, $d_0, d \in \pm 2^{\mathsf{s}}$, $\gamma \in \pm 2^{2\ell+\mathsf{s}_0+\nu}$, $\rho \in \mathbb{Z}_{N^2}^*$, and $(s_j)_{j=0}^{\tau-1} \in \mathbb{F}_q^{\tau}$ such that

$$U_j = g^{y_{j-1}\cdot d^{-1}}\, h^{s_{j-1}} \;\; (j = 1, \dots, \tau), \; \mathcal{E}_1^{d_0} = (1 + d_0 \cdot y' \cdot d^{-1} N)\,\bar{u}_1^{\gamma^*} \bmod N^2,$$
$$\mathcal{E}_i^{d_0} = \bar{u}_i^{\gamma^*} \bmod N^2 \;\; (i \in I), \qquad\qquad B = (1 + \gamma \cdot d^{-1} N) \cdot \rho^{N} \bmod N^2.$$

  where $d_0 \mid d$, $\gamma^* = \gamma \cdot d^* \bmod \varphi(N)$ and $d^* = (d/d_0)^{-1} \bmod \varphi(N)/d_0$, and $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$.

Protocol Generalized Pedersen / Paillier-ElGamal Consistency

Common Input. Modulus N, precomputed bases $(\bar{u}_i)_{-(k-1)\le i\le k} \subset \mathbb{Z}_{N^2}^*$, $g, h \in \mathbb{G}$, $U_1, \dots, U_\tau \in \mathbb{G}$, and $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \dots, \mathcal{E}_k) \in (\mathbb{Z}_{N^2}^*)^{2k}$ such that $U_j = g^{y_{j-1}} \cdot h^{s_{j-1}}$, $B = (1+\gamma N) \cdot \bar{u}_0^{r} \bmod N^2$,

$$\mathcal{E}_1 = (1+y'N) \cdot \bar{u}_1^{\gamma} \bmod N^2, \qquad \mathcal{E}_i = \bar{u}_i^{\gamma} \bmod N^2 \;\; (i \in I),$$

for $y' = \sum_{j=1}^{\tau} y_{j-1} \cdot 2^{(j-1)\kappa}$, $\gamma \in \pm 2^{2\ell}$, and $r \in \pm 2^{2\ell}$, where $I = \{-(k-1), \dots, k\} \setminus \{1\}$.

Operations. (Recall $\varepsilon = \mathsf{s}_0 + \nu$. Repeated $\lceil \mathsf{s}/\mathsf{s}_0 \rceil$ times for soundness $\approx 2^{-\mathsf{s}}$.)

1. P samples $(\beta_0, \dots, \beta_{\tau-1}) \xleftarrow{\$} (\pm 2^{\kappa_0+\varepsilon})^{\tau}$, $(\sigma_0, \dots, \sigma_{\tau-1}) \xleftarrow{\$} \mathbb{F}_q^{\tau}$, $\delta \xleftarrow{\$} \pm 2^{2\ell+\varepsilon}$, $r' \xleftarrow{\$}$

$\pm 2^{2\ell+\varepsilon}$, sets $\beta' = \sum_{j=1}^{\tau} \beta_{j-1} \cdot 2^{(j-1)\kappa}$, and sends

$$\begin{cases} \widetilde{A}_j \;=\; g^{\beta_{j-1}} \cdot h^{\sigma_{j-1}} \;\in\; \mathbb{G} \quad (j = 1, \ldots, \tau), \\ \widetilde{B} \;=\; (1 + \delta N) \cdot \bar{u}_0^{r'} \bmod N^2, \\ \widetilde{D}_1 \;=\; (1 + \beta' N) \cdot \bar{u}_1^{\delta} \bmod N^2, \\ \widetilde{D}_2^{(i)} \;=\; \bar{u}_i^{\delta} \bmod N^2 \quad (i \in I). \end{cases}$$

2. V sends $e \xleftarrow{\$} \pm 2^{\mathsf{s}_0}$.
3. P returns $(z_j)_{j=0}^{\tau-1}$, $(w_j)_{j=0}^{\tau-1}$, w_γ, and w_r such that

$$\begin{aligned} z_j \;&=\; \beta_j + e \cdot y_j, \quad & w_j \;&=\; \sigma_j + e \cdot s_j \bmod q, \\ w_\gamma \;&=\; \delta + e \cdot \gamma, & w_r \;&=\; r' + e \cdot r. \end{aligned}$$

Verifier's Output. Let $\hat{z} = \sum_{j=1}^{\tau} z_{j-1} \cdot 2^{(j-1)\kappa}$. Accept if $z_j \in \pm 2^{\kappa_0+\varepsilon}$ for all j, $w_\gamma \in \pm 2^{2\ell+\varepsilon}$, and

$$\begin{cases} g^{z_{j-1}} \cdot h^{w_{j-1}} \;=\; \widetilde{A}_j \cdot U_j^{e} \;\in\; \mathbb{G} \quad (j = 1, \ldots, \tau), \\ (1 + \hat{z}\, N) \cdot \bar{u}_1^{w_\gamma} \;=\; \widetilde{D}_1 \cdot \mathcal{E}_1^{e} \bmod N^2, \\ \bar{u}_i^{w_\gamma} \;=\; \widetilde{D}_2^{(i)} \cdot \mathcal{E}_i^{e} \bmod N^2 \quad (i \in I), \\ (1 + w_\gamma N) \cdot \bar{u}_0^{w_r} \;=\; \widetilde{B} \cdot B^{e} \bmod N^2. \end{cases}$$

Protocol 5: Generalized proof of Pedersen / Paillier-ElGamal consistency $\Pi_{k\text{-}\mathsf{dlcom}}$.

The $2k-1$ randomness-base checks $\bar{u}_i^{w_\gamma} = \widetilde{D}_2^{(i)} \cdot \mathcal{E}_i^{e}$ all share the single response w_γ, and can be collapsed into a single check on a random linear combination of the bases, trading $2k-1$ verifier exponentiations for one at the cost of a $2^{-\mathsf{s}_0}$ soundness loss.

7.3 Combined Generalized Proof of Encrypted Discrete Log ($\Pi_{k\text{-}\mathsf{encdl}}$)

As in the non-batch setting (Protocol 3), we combine Π_{small} and $\Pi_{k\text{-}\mathsf{dlcom}}$ into a single proof system $\Pi_{k\text{-}\mathsf{encdl}}$ for the joint statement that a group element $X = g^x$ and a k-Generalized Paillier-ElGamal ciphertext $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_k)$ are consistent, in the sense that $\mathcal{E}$ encrypts an integer x' congruent to x modulo q with x' in a bounded range. The construction takes the same public scalar $\alpha \in \mathbb{N}$ as Π_{encdl}: a product of small prime powers coprime to q, chosen divisible by every small d_i that Π_{small}'s slack extractor can produce.

Definition 7.4 (Promise problem for $\Pi_{k\text{-}\mathsf{encdl}}$)**.** The statement is a tuple $(g, h, (\bar{u}_i)_{-(k-1)\le i\le k}, X, \mathcal{E} = (\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_k), B, \alpha)$, where α is a product of the agreed prime powers, coprime to q, and $I = \{-(k-1), \ldots, k\} \setminus \{1\}$.

- *Yes instances.* Tuples for which there exist $x \in [0, q)$, $\gamma \in \pm 2^{2\ell}$, and $r \in \pm 2^{2\ell}$ such that

$$\begin{aligned} &X = g^x, && x' = \alpha \cdot (\alpha^{-1} x \bmod q), \\ &\mathcal{E}_1 = (1 + x'N)\, \bar{u}_1^{\gamma} \bmod N^2, && \mathcal{E}_i = \bar{u}_i^{\gamma} \bmod N^2 \ \ (i \in I), \\ &B = (1 + \gamma N)\, \bar{u}_0^{r} \bmod N^2. \end{aligned}$$

- *Slack instances.* Tuples for which N admits small factors, or there exist an integer $x' \in \pm(\alpha\, \tau \cdot 2^{\kappa\tau+\varepsilon+1})$, $d_0, d \in \pm 2^{\mathsf{s}}$, $\gamma \in \pm 2^{2\ell+\mathsf{s}_0+\nu}$, and $\rho \in \mathbb{Z}_{N^2}^*$ such that

$$\begin{aligned} &X = g^{x'}, && \mathcal{E}_1^{d_0} = (1 + d_0 \cdot x'N)\, \bar{u}_1^{\gamma^*} \bmod N^2, \\ &\mathcal{E}_i^{d_0} = \bar{u}_i^{\gamma^*} \bmod N^2 \ \ (i \in I), && B = (1 + \gamma \cdot d^{-1} N) \cdot \rho^N \bmod N^2, \end{aligned}$$

  where $d_0 \mid d$, $\gamma^* = \gamma \cdot d^* \bmod \varphi(N)$ and $d^* = (d/d_0)^{-1} \bmod \varphi(N)/d_0$.

Construction. The construction is identical to that of Π_{encdl}, with Π_{dlcom} replaced by $\Pi_{k\text{-}\mathsf{dlcom}}$. The prover reduces the joint statement to chunks of $\tilde{x} \coloneqq x \cdot \alpha^{-1} \bmod q$, which it Pedersen-commits to in $\mathbb{G}$; multiplying by α in the exponent of each commitment yields the "raised" commitments that play the role of the U_j's of $\Pi_{k\text{-}\mathsf{dlcom}}$. The raised commitments are not transmitted: the verifier reconstructs them from the Pedersen commitments and the public α. The Paillier-style commitment $B = (1 + \gamma N)\, \bar{u}_0^{r}$ to the ciphertext randomness γ plays the same role as in $\Pi_{k\text{-}\mathsf{dlcom}}$ and is passed through directly. Two sub-proofs are then run in lockstep — Π_{small} on the Pedersen commitments and $\Pi_{k\text{-}\mathsf{dlcom}}$ on the raised commitments, $\mathcal{E}$, and B — with their challenges drawn independently but bundled together (sent in a single round-trip in the interactive case, or derived from a single Fiat-Shamir hash over the concatenated first messages of both sub-proofs in the non-interactive case). To anchor the joint proof to the public X, the prover picks the Pedersen randomness subject to a single linear constraint over $\mathbb{F}_q$, which forces the aggregated raised commitment to coincide with X.

Protocol Generalized Encrypted Discrete Log $\Pi_{k\text{-}\mathsf{encdl}}$

Common Input. Pedersen bases $g, h \in \mathbb{G}$, Paillier bases $(\bar{u}_i)_{-(k-1) \le i \le k} \subset \mathbb{Z}_{N^2}^*$ over a modulus N, a public scalar $\alpha \in \mathbb{N}$, a group element $X \in \mathbb{G}$, a ciphertext $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_k) \in (\mathbb{Z}_{N^2}^*)^{2k}$, and a Paillier-style commitment $B \in \mathbb{Z}_{N^2}^*$.

Witness. $x \in \mathbb{F}_q$ with $X = g^x$, a bounded integer $x' \equiv x \bmod q$, ciphertext randomness $\gamma \in \pm 2^{2\ell}$, and randomness $r \in \pm 2^{2\ell}$ such that $\mathcal{E}_1 = (1 + x'N)\, \bar{u}_1^{\gamma} \bmod N^2$, $\mathcal{E}_i = \bar{u}_i^{\gamma} \bmod N^2$ for $i \in I$, and $B = (1 + \gamma N)\, \bar{u}_0^{r} \bmod N^2$.

Operations.

1. *Chunking and commitment.* P computes $\tilde{x} = x \cdot \alpha^{-1} \bmod q$ and writes it as $\tilde{x} = \sum_{j=1}^{\tau} y_{j-1}\, 2^{(j-1)\kappa}$ with $y_{j-1} \in \pm 2^{\kappa}$. P samples randomness $(r_{j-1})_{j=1}^{\tau} \in \mathbb{F}_q^{\tau}$ subject to the linear constraint

$$\sum_{j=1}^{\tau} r_{j-1}\, 2^{(j-1)\kappa} \ \equiv \ 0 \ \bmod q,$$

and sends $Y_j = g^{y_{j-1}} h^{r_{j-1}} \in \mathbb{G}$ for $j = 1, \ldots, \tau$.

2. *Joint sub-proofs.* P and V run Π_{small} (Protocol 1) on $(Y_j)_{j=1}^{\tau}$ and $\Pi_{k\text{-dlcom}}$ (Protocol 5) on the raised commitments $X_j := Y_j^{\alpha}$, $\mathcal{E}$, and B with $\Pi_{k\text{-dlcom}}$'s chunk-magnitude parameter set to $\kappa_0 := \kappa + \log \alpha$, with each sub-proof drawing its own challenges, bundled together as described above.

Verifier's Output. Accept if (i) α is well-formed (a product of the agreed prime powers, coprime to q), (ii) $X = \prod_{j=1}^{\tau} X_j^{2^{(j-1)\kappa}}$ for $X_j = Y_j^{\alpha}$, and (iii) both the Π_{small} and $\Pi_{k\text{-dlcom}}$ sub-verifiers accept.

Protocol 6: Combined generalized proof of encrypted discrete log $\Pi_{k\text{-encdl}}$.

7.4 A Batch-RNIVOLE from Batch-NIM

In this section, we introduce a variant of the RNIVOLE from the Section 5 where the sender transmits a batch of k VOLE evaluations $(a_i \cdot x + b_i \bmod q)_{i \le k}$ at once. This variant leverages the k-Generalized Paillier-ElGamal encryption scheme, which we introduce in Definition 7.1, to significantly reduce the per-VOLE communication (at the cost of a slightly increased computation). At the heart of our approach is a technique introduced concurrently in [BCMPR24] and [ARS24], that lets a sender execute a batch of k NIMs by sending a *single* group element. This comes at the cost of increasing the size of the Paillier-ElGamal encryption of x by k elements of $\mathbb{Z}_{N^2}$ in the (one-time) key generation phase.

We extend our KZG-based methodology to enhance this batch-NIM to the malicious setting, where the proof consists again of a single tuple $(\mathsf{salt}, e, \pi, c)$ (as before, the DFO commitment $\mathcal{C}$ needs not be sent). However, as c must be computed as a polynomial in the challenge e (over the integers), its size grows linearly with k: concretely, $|c| = 2\nu + \mathsf{s} + (k+4) \cdot \ell$. For security, recall that we require $|c| \le \log N - 3\ell$. This puts a limit to the maximal batch size as the largest k such that $2\nu + \mathsf{s} + (k+7) \cdot \ell \le \log N$. Using $\nu = 40$, $\mathsf{s} = 64$, $\ell = 128$, and $\log N = 3072$, this translates to $k \le 15$.

Beyond c, the NIM shares $(z_i)_{i \le k}$ are the only other part of the VOLE multiplication algorithm that grow with k: every additional multiplication adds only $\ell + \log q \approx 3\ell$ bits of communication. This translates to significant savings in the per-VOLE communication, from 535 bytes when $k = 1$ down to 305 when $k = 2$, all the way to 82 bytes per VOLE when $k = 15$. Another unexpected byproduct is that the batch setting also reduces significantly the *receiver computation*: intuitively, in Algorithm 3, the receiver computation was dominated by the calculation of $\mathcal{C}^{2N\gamma} \bmod N^2$, as it involves a full-sized exponentiations. In the batch setting with batch size k, computing all outputs still requires a single full-sized exponentiation, followed only by exponentiations by d, a 2ℓ-bit value. On the other hand, the sender computation increases, due to the need to compute k size-k multiexponentiations.

Setup. The setup remains essentially identical to the non-batch setting, with a single modification: the KZG parameters must be extended to handle polynomials of degree up to $2k$, *i.e.*, the key $\mathsf{pk}_{\mathsf{PE}}$ now contains terms $u_{-k}^{d^i} \bmod N$ for $i = 0$ to $2k$. Correspondingly, the proof of correctness of the KZG parameters is generalized to prove correctness of the extended parameters. We

let $\mathsf{PE}^k = (\mathsf{PE}^k.\mathsf{Setup}, \mathsf{PE}^k.\mathsf{Enc}, \mathsf{PE}^k.\mathsf{Dec})$ denote the k-generalized Paillier-ElGamal encryption scheme (Definition 7.1).
Our setup protocol assumes a proof $\Pi_{\mathsf{k\text{-}par}}$ that proves well-formedness of the KZG parameters $\mathsf{pk}_{\mathsf{PE}}$. This proof is an immediate extension of the KZG parameter proof (Protocol 10) using a standard Σ-protocol for discrete logarithm equality, and we omit it here.

Functionality Setup Functionality $\mathcal{F}_{\mathsf{b\text{-}setup}}$

Parties. The functionality interacts with two players, a server S (the receiver) and a client C (the sender), and an adversary $\mathcal{A}$.

Operation. Upon receiving $(\mathsf{init}, \mathsf{sid})$ from S and $(\mathsf{init}, \mathsf{sid})$ from C, where sid encodes a group $(\mathbb{G}, g, q)$ of prime order q generated by g, parameters $(n_{\mathrm{rsa}}, \ell, \nu)$, and the identities S, C:

- Sample a Pedersen base $h \xleftarrow{\$} \mathbb{G}$.
- *If* S *is honest:* sample two safe $n_{\mathrm{rsa}}/2$-bit primes p_1, p_2 and set $N := p_1 p_2$; sample $d \xleftarrow{\$} [2^{2\ell}]$; sample u_{-k} uniformly from the quadratic residues modulo N and set $u_{-k+i} := u_{-k}^{d^i} \bmod N$ for $i = 1$ to $2k$. Set $\bar{u}_i = u_i^N \bmod N^2$ for $i = -k$ to k. Send $(N, (u_i, \bar{u}_i)_{-k \le i \le k}, h)$ to both parties and the trapdoor (p_1, p_2, d) to S.
- *If* S *is corrupt:* receive $(p_1, p_2, (u_i)_{-k \le i \le k}, d)$ from $\mathcal{A}$, set $N := p_1 p_2$, and abort unless
  1. p_1, p_2 are primes (not necessarily safe) and $p_1, p_2 \in \pm 2^{n_{\mathrm{rsa}} + 2\ell + \nu}$;
  2. $u_i \in \mathbb{Z}_N^*$ with $u_i = u_{-k}^{i-k} \bmod N$ for $i = -k$ to k;
  3. $d \in [2^{2\ell+\nu}]$.

  If all checks pass, send $(N, (u_i, \bar{u}_i)_{-k \le i \le k}, h)$ to both parties.

Functionality 2: Batch setup functionality $\mathcal{F}_{\mathsf{b\text{-}setup}}$.

Algorithm Batch VOLE Setup

Parameters. $\mathsf{pp} = (\ell, \nu, \kappa, \tau)$. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ is modeled as a random oracle.

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

Input. Security parameter ℓ.

$\mathsf{VOLE.Setup}^{\mathsf{H}}(1^\ell)$.

1. Sample $(\mathsf{pk}_{\mathsf{PE}}, \mathsf{sk}_{\mathsf{PE}}) \xleftarrow{\$} \mathsf{PE}^k.\mathsf{KeyGen}(1^\ell)$. Parse $\mathsf{pk}_{\mathsf{PE}} = (N, (u_i, \bar{u}_i)_{-(k-1) \le i \le k})$.
   - Generate Paillier-Blum well-formedness proof $\pi_{\mathsf{mod}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{mod}}(\mathrm{aux}, N; p_1, p_2)$.
   - Generate small factors proof $\pi_{\mathsf{fac}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{fac}}(\mathrm{aux}, N; p_1, p_2)$

– Generate DFO-KZG^k params proof $\pi_{\mathsf{par}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\text{k-par}}((N, (u_{0,i}, u_i)_{-(k-1)\le i\le k})$.

2. Generate $g_1, h \in \mathbb{G}'$ as nothing-up-my-sleeve numbers using $(\mathbb{G}', g_0, q')$ and aux as a seed.

   Public Output. $\mathsf{SetupData} = (\mathsf{pk}_{\mathsf{PE}}, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{par}}, g_0, g_1, h)$.

   Secret Output. $\mathsf{sk} = d$

$\mathsf{VOLE.VerifySetup}^{\mathsf{H}}(\mathsf{SetupData})$: output 1 iff all verifications below succeed.

1. Parse $\mathsf{SetupData} = (\mathsf{pk}_{\mathsf{PE}}, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{par}}, g_0, g_1, h)$ and $\mathsf{pk}_{\mathsf{PE}} = (N, (u_{0,i}, u_i)_{-(k-1)\le i\le k})$.

   Let $n_{\mathrm{rsa}} \leftarrow \mathsf{SampleRSA}(1^\ell)$.

2. Verify that $N \ge 2^{n_{\mathrm{rsa}}}$, $u_{0,i} \in \mathbb{J}_N$, and $u_i = u_{0,i}^{2N} \bmod N^2$ for $i = -(k-1)$ to k.
3. Verify that $g_1, h \in \mathbb{G}'$ can be reconstructed according to Item 2 of Algorithm 7.
4. Verify the ZK proofs:
   (a) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{mod}}}(\mathrm{aux}, N, \pi_{\mathsf{mod}}) = 1$
   (b) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{fac}}}(\mathrm{aux}, N, \pi_{\mathsf{fac}}) = 1$
   (c) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\text{k-par}}}(N, (u_{0,i}, u_i)_{-(k-1)\le i\le k}, \pi_{\mathsf{par}}) = 1$

Algorithm 4: Non-interactive batch setup and setup verification algorithms

Registration. The VOLE scalar x is registered as the plaintext of a k-Generalized Paillier-ElGamal encryption $\mathcal{E}$. The proof Π_k-encdl generalizes Π_{encdl} to proving that $\mathcal{E}$ correctly encrypts the discrete logarithm of X.
Our registration protocol uses the proof $\Pi_{k\text{-encdl}}$ (Protocol 6), which proves that the k-Generalized Paillier-ElGamal ciphertext $\mathcal{E}$ encrypts the discrete logarithm of X. It is the generalization of Π_{encdl} (Protocol 3) to the $2k$-component ciphertext, obtained by replacing the consistency sub-proof Π_{dlcom} with Π_k-dlcom (Protocol 5).

Algorithm Batch VOLE Registration

Parameters. $(\mathbb{G}, g, q)$, $\mathsf{SetupData}$, and $d \in \mathbb{N}$. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ is modeled as a random oracle.

Input. $x \in \mathbb{F}_q$

$\mathsf{VOLE.Register}^{\mathsf{H}}(\mathsf{SetupData}, x)$: sample $\gamma \xleftarrow{\$} \pm\{0,1\}^{2\lambda}$, set $X = g^x$, and do:

– Set $\mathcal{E} \leftarrow \mathsf{PE}.\mathsf{Enc}^k_{\mathsf{pk}_{\mathsf{PE}}}(x; \gamma)$ and generate proof

$$\psi_{\mathsf{encdl}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{k\text{-encdl}}(\mathbb{G}, g, q, X, \mathcal{E}, \mathsf{pk}_{\mathsf{PE}}, \log q, 2\ell; x, \gamma)$$

– Output $(\mathsf{e}_x = (\mathcal{E}, \psi_{\mathsf{encdl}}), \mathsf{d}_x = \gamma)$.

$\mathsf{VOLE.VerifyRegister}^{\mathsf{H}}(\mathsf{SetupData}, X, \mathsf{e}_x = (\mathcal{E}, \psi_{\mathsf{encdl}}))$:

– Check that $\mathcal{E} \in (\mathbb{Z}_{N^2}^*)^{2k}$.

– Verify proof $\mathsf{vrfy}_{\Pi_k^{\mathrm{FS}}\text{-}\mathsf{encdl}}(\mathbb{G}, g, q, X, \mathcal{E}, \mathsf{pk}_{\mathsf{PE}}, \log q, 2\ell, \psi_{\mathsf{encdl}}) = 1$.

Algorithm 5: VOLE registration algorithms.

Multiplication. The batch VOLE multiplication procedure replaces the NIM with a batch-NIM. We use the batch-NIM from [ARS24], with minor modifications. We adapt our KZG-based proof technique to the batch setting. An important difference is that we must add an additional "low-degree proof" to guarantee that the extracted representation is valid (i.e., is only defined over a "width-k" window of the basis elements $(u_{-k}, \ldots, u_k)$).

Algorithm Batch VOLE Multiplication

Parameters. $\mathsf{pp} = (\ell, \nu, \kappa, \tau, \mathsf{s})$. Define $n_r := \mathsf{s} + \nu$, $n_0 := (2k+3)\ell + \mathsf{s} + 2\nu$, and $n_\rho := (2k+4)\ell + \mathsf{s} + 2\nu$. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathsf{H}' : \{0,1\}^\ell \to \mathbb{Z}_N$ are modeled as random oracles.

Common Input. $(\mathbb{G}, g, q)$, $\mathsf{SetupData}$, $\mathcal{E} = (\mathcal{E}_{-(k-1)}, \ldots, \mathcal{E}_k) \in (\mathbb{Z}_{N^2}^*)^{2k}$.

Private Inputs. $d \in [2^{2\ell}]$, $x \in \mathbb{F}_q$, $(\vec{a}, \vec{b}) = (a_i, b_i)_{i \le k} \in \mathbb{F}_q^{2k}$.

$\mathsf{VOLE.Send}^{\mathsf{H},\mathsf{H}'}(\mathsf{SetupData}, \mathsf{e}_x, \vec{a}, \vec{b})$: parse $\mathcal{E}$ from e_x. Sample $\mathsf{salt} \xleftarrow{\$} [2^\ell]$, $r_1, \ldots, r_k \xleftarrow{\$} [2^{n_r}]$, $\rho \xleftarrow{\$} [2^{n_\rho}]$, $\rho_0 \xleftarrow{\$} [2^{n_0}]$, and do:

– Set $a'_i \leftarrow a_i + q \cdot r_i$ for $i = 1$ to k and $\mathcal{C} \leftarrow u_{-1}^{\rho} \cdot u_0^{\rho_0} \cdot \prod_{i=1}^k u_i^{a'_i} \bmod N$.

– Compute $e \leftarrow \mathsf{H}(\mathcal{C}, \mathsf{salt})$ and $c = \sum_{i=1}^k a'_i \cdot e^{i+1} + \rho_0 \cdot e + \rho$.

– For $j = k$ to 1, define $\alpha_j \leftarrow \sum_{i=j}^k a'_i e^{i-j}$ and set

$$\pi \leftarrow u_{-k}^{\rho_0 + e\cdot\alpha_1} \cdot \prod_{j=1}^k u_{j-k}^{\alpha_j} \bmod N,$$

$$\pi' \leftarrow \mathsf{H}\left(u_{-1}^{\rho_0 + e\cdot\alpha_1} \cdot \prod_{j=1}^k u_{j-1}^{\alpha_j} \bmod N \right).$$

– For $j = k$ to 1, set $V_j = \mathcal{E}_{-j}^{\rho_0 + e\cdot\alpha_1} \cdot \prod_{i=1}^k \mathcal{E}_{i-j}^{\alpha_i} \bmod N^2$ and $v_j = [\mathsf{DDLOG}_N(V_j) + \mathsf{H}'(e)]_N$. Let $v_{k+1} \leftarrow 0$.

– For $j = k$ to 1, set $z_j \leftarrow v_j - e \cdot v_{j+1} + b_j \bmod q$.

– Return $m = (\mathsf{salt}, e, \pi, \pi', c, z_1, \ldots, z_k)$.

$\mathsf{VOLE.Output}^{\mathsf{H},\mathsf{H}'}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_x, \mathsf{d}_x, m)$: parse $d := \mathsf{sk}$, $\mathcal{E}$ from e_x, $\gamma := \mathsf{d}_x$, and $m = (\mathsf{salt}, e, \pi, \pi', c, z_1, \ldots, z_k)$.

– If $c \notin [2^{n_\rho+2}]$ or $\pi' \neq \mathsf{H}(\pi^{d^{k-1}})$, return $\perp$. Else, set $\pi'' := \pi^{d^{k-1}} \bmod N$.
– Set $\mathcal{C} \leftarrow (\pi'')^{d-e} \cdot u_{-1}^{c} \bmod N$. If $e \neq \mathsf{H}(\mathcal{C}, \mathsf{salt})$, return $\perp$.
– For $j = k$ to 1, set $w'_j \leftarrow [\mathsf{DDLOG}_N(\mathcal{C}^{2N\gamma \cdot d^{-j}} \bmod N^2) + \mathsf{H}'(e)]_N$. Let $w'_{k+1} \leftarrow 0$.
– For $j = k$ to 1, set $w_j \leftarrow z_j - (w'_j - e \cdot w'_{j+1}) \bmod q$.
– Output $\vec{w} = (w_1, \ldots, w_k)$.

Algorithm 6: VOLE multiplication algorithms.

7.5 Efficiency

We report below on the communication, in bytes, of $\mathsf{VOLE.Send}$ for a batch of k VOLE multiplications.

Table 6: Communication of $\mathsf{VOLE.Send}$ for various batch sizes

Batch size (k)	2	3	4	5	10	15
Communication	305	219	176	151	99	82

7.6 A generalization of k-KZG

We introduce a generalization of the k-KZG assumption (Assumption 6.1). Roughly, the variant, which we denote (k, k')-KZG, says the following: if the adversary is given $(g^{c^i})_{0 \le i \le k+k'}$ manages to produce (X, Y) such that $X^{c^{k'}} = Y$, then it must know a representation $(a_0, \ldots, a_k)$ of X such that $X = \prod_{i=0}^{k} (g^{c^i})^{a_i}$. Note that k-KZG is the same as $(k, 1)$-KZG: the generalizations extends it to allow for higher powers of c in the relation between X and Y. Intuitively, this is required in our analysis to "force" the adversary to hold a representation of X that depends only on $(g^{c^i})_{i \le k}$ despite the fact that it also knows higher powers $(g^{c^i})_{i > k}$.

Assumption 7.5 ((k, k')-KZG)**.** Let $\mathsf{GroupGen}$ denote a PPTM that, on input 1^ℓ, returns a triple $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is the description of a cyclic group of order $q(\ell)$ and g is a generator of $\mathbb{G}$. We say that the (k, k')-KZG assumption holds over $\mathsf{GroupGen}$ if for every non-uniform PPT algorithm $\mathcal{A}$, there exists another non-uniform PPT algorithm Ext and a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr[X^{c^{k'}} = Y \wedge X \neq \prod_{i=0}^{k} g^{a_i \cdot c^i}] \le \mu(\ell),$$

where the probability is over the random choices of $(\mathbb{G}, g, q) \xleftarrow{\$} \mathsf{GroupGen}(1^\ell)$, $c \xleftarrow{\$} \mathbb{Z}_q$, $r \xleftarrow{\$} \{0,1\}^*$, $(X, Y) \xleftarrow{\$} \mathcal{A}(\mathbb{G}, q, (g^{c^i})_{i \in [k+k']}; r)$, and $(a_i)_{i \in [k]} \xleftarrow{\$} \mathsf{Ext}(\mathbb{G}, q, (g^{c^i})_{i \in [k+k']}; r)$.

As for the k-KZG assumption, which it generalizes, (k,k')-KZG is a special case of the Uber-Knowledge assumption from [BFHK24] and it holds unconditionally in the generic group model.

7.7 Security

The security analysis of the batch VOLE is essentially identical to the analysis in Section 6. We omit a fully detailed adaptation of the analysis and confine ourselves to pointing out the parts where the analysis requires some careful adaptation to the batch setting.

Corrupted sender: extraction via the (k,k)-KZG assumption. Hybrid_1 to Hybrid_6 are identical to the same hybrids in the proof against a corrupted sender for the non-batch setting (Section 6). In Hybrid_7 of the original security proof against a corrupted sender, the simulator Sim defines an algorithm that returns $(\pi, \mathcal{C} \cdot u_0^{-c} \cdot \pi^e \bmod N)$ for a suitable $\mathcal{C}$ extracted from the random oracle. The generalization of this approach to our setting sets $\mathcal{C} \leftarrow u_{-1}^{\rho} \cdot u_0^{\rho_0} \cdot \prod_{i=1}^{k} u_i^{a'_i} \bmod N$ and crafts π as the standard proof of correct opening to c for the KZG commitment $\mathcal{C}$ at the evaluation point e. This ensures that $\pi^{d-e} = \mathcal{C} \cdot u_{-1}^{-c}$. However, recall that due to the use of the batch-NIM, the parties are given $(u_i)_{-k \le i \le k} = (u_{-k}^{d^i})_{0 \le i \le 2k}$: that is, they know $2k+1$ elements of the basis $(u_i)_i$, even though a honest commitment $\mathcal{C}$ should involve at most $k+2$ bases u_i. Now, running the extractor from Assumption 6.1 given $(\pi, \mathcal{C} \cdot u_{-1}^{-c} \cdot \pi^e \bmod N)$ would yield a representation $(\theta_i)_i$ of π as $\pi = \prod_{i=-k}^{k-1} u_i^{\theta_i}$ and of $\mathcal{C}$ as $\prod_{i=-k}^{k-1} u_{i+1}^{\theta_i}$. Unfortunately, this is insufficient to properly simulate: it is necessary to extract a correct representation of $\mathcal{C}$ in order to securely replace the outputs $w_j = z_j - [\mathsf{DDLOG}_N(\mathcal{C}^{2N\gamma \cdot d^{k-j}} \bmod N^2) + \mathsf{H}'(e)]_N \bmod q$ with the ideal outputs $\tilde{w}_j$ obtained from the (batch) RVOLE functionality. Indeed, the indistinguishability of this replacement relies on the correctness of the DDLOG algorithm, which is not guaranteed anymore if $\mathcal{C}$ can have an incorrect representation. To fix this, we let the sender append an additional term π' computed to satisfy $\pi' = \mathsf{H}(\pi'')$ with $\pi'' = \pi^{d^{k-1}}$. In "KZG terminology", π commits to some polynomial $P(X)$, and π'' is computed as a commitment to $X^{k-1} \cdot P(X)$. The verification guarantees that P can have degree at most $k+1$; combined with the proof of correct opening, which bounds the degree of $X^{k-1}P(X)$ to $2k-1$, we get a degree bound of k on P.

To translate this intuition into an extraction strategy, the simulator extracts π'' from π' and the queries to the random oracle (if no query was made with input π'', the check $\pi' = \mathsf{H}(\pi^{d^{k-1}})$ fails with probability $1 - 1/2^{\ell}$). Then, the simulator computes $(\pi, Y = \mathcal{C} \cdot u_{-1}^{-c} \cdot (\pi'')^e)$; observe that $\pi^{d^k} = Y$. We feed the pair (π, Y) to an extractor to obtain a representation of π in the basis $(u_{-k}, \ldots, u_0)$.

To formalize the required extraction, we introduce a generalization of the k-KZG assumption that accounts for adversaries producing relations of the form $X^{d^k} = Y$. Concretely, the security analysis requires the (k,k)-KZG assumption (Assumption 7.5) over the group QR_N of quadratic residues.

A part that we did not address is the use of π as a DFO/KZG commitment in the batch-NIM. Recall that $\pi = u_{-k}^{\rho_0+\alpha_0} \cdot \prod_{j=1}^{k} u_{j-k}^{\alpha_j} \bmod N$. In the batch-NIM, the parties obtain (via the DDLOG algorithm) subtractive shares of $v_j = x \cdot \alpha_j$ for $j = k$ to 1, where $\alpha_j = \sum_{i=j}^{k} a'_j \cdot e^{i-j}$. From there, they can directly compute shares of the $x \cdot a'_j$ using the relation $a'_j = \alpha_j - \alpha_{j+1}$ (setting $\alpha_{k+1} = 0$). Afterwards, the hybrids proceed as in the original proof, using factoring to ensure uniqueness of the extracted representation in Hybrid_8 and the forking lemma to ensure that all extracted terms are short in Hybrid_9, where "short" means that $\max(\tilde{a}_1, \ldots, \tilde{a}_k, \rho_0, \rho) \le 2^{n_\rho}$, Hybrid_{10} uses the forking lemma again to guarantee no wraparounds in the DDLOGs with overwhelming probability (this step requires enforcing $n_\rho + 3\ell \le \log N$), and Hybrid_{11} invokes the IND-CPA security of the k-Generalized

Paillier-ElGamal encryption (Definition 7.1) to replace $\mathcal{E}$ with an encryption of 0.

Corrupted receiver: general indistinguishability lemma. The analysis in the case of a corrupted receiver is essentially identical to the non-batch setting, but it requires using a generalization of Claim 6.11, which we outline below:

Claim 7.6. *Fix $f \le 2^{\mathsf{s}}$ and k with $k^2 \le 2^\ell$. Set $n_r := \mathsf{s} + \nu$, $n_0 := (2k+3)\ell + \mathsf{s} + 2\nu$, and $n_\rho := (2k+4)\ell + \mathsf{s} + 2\nu$. Fix $d \in [2^{2\ell}]$ and $e \in [2^\ell]$. Consider the following distributions, where $\mathcal{D}_0[a]$ is parametrized by $a = (a_1, \ldots, a_k) \in \mathbb{Z}_q^k$ for a prime $2^{\mathsf{s}} < q \le 2^{2\ell}$:*

$\mathcal{D}_0(a)$: *returns $(A, c, \{[\alpha_i]_f\}_{i=1}^k, [\theta]_f)$, defined as*

- $(r_1, \ldots, r_k) \xleftarrow{\$} [2^{n_r}]^k$, $\rho \xleftarrow{\$} [2^{n_\rho}]$, $\rho_0 \xleftarrow{\$} [2^{n_0}]$
- *For $i = 1$ to k, $a'_i \leftarrow a_i + q \cdot r_i$ and $\alpha_i \leftarrow \sum_{j=i}^k a'_j \cdot e^{j-i}$*
- $A \leftarrow \rho_0 + e \cdot \alpha_1 + \sum_{i=1}^k \alpha_i \cdot d^i$
- $\theta \leftarrow e \cdot \alpha_1 + \rho_0$
- $c \leftarrow \sum_{i=1}^k a'_i \cdot e^{i+1} + e \cdot \rho_0 + \rho$

$\mathcal{D}_1$: *returns $(A, c, \{[\alpha_i]_f\}_{i=1}^k, [\theta]_f)$, defined as*

- $(r_1, \ldots, r_k) \xleftarrow{\$} [2^{n_r}]^k$, $A' \xleftarrow{\$} [2^{n_\rho}]$
- *For $i = 1$ to k, $a'_i \leftarrow q \cdot r_i$ and $\alpha_i \leftarrow \sum_{j=i}^k a'_j \cdot e^{j-i}$*
- $A \xleftarrow{\$} [2^{n_0}]$
- $\theta \leftarrow A - \sum_{i=1}^k \alpha_i \cdot d^i$
- $c \leftarrow e \cdot A + A'$

Then for every $a \in \mathbb{Z}_q^k$, the statistical distance between $\mathcal{D}_0(a)$ and $\mathcal{D}_1$ is at most $2^{-\nu+2} + k \cdot 2^{-\nu}$.

Proof. In $\mathcal{D}_0(a)$, let $A' \leftarrow \rho - \sum_{i=1}^k \alpha_i \cdot e \cdot d^i$. Since $A = \rho_0 + e \cdot \alpha_1 + \sum_{i=1}^k \alpha_i d^i$ and $\theta = e \cdot \alpha_1 + \rho_0$, we have $\theta = A - \sum_{i=1}^k \alpha_i d^i$, so $[\theta]_f$ can be computed from A, $\{[\alpha_i]_f\}_{i=1}^k$, and d. Moreover $c = e \cdot A + A'$: indeed $e \cdot A = e\rho_0 + e^2\alpha_1 + \sum_{i=1}^k \alpha_i e d^i$, and $e^2\alpha_1 = e^2 \sum_{j=1}^k a'_j e^{j-1} = \sum_{j=1}^k a'_j e^{j+1}$, hence $e \cdot A + A' = \sum_{j=1}^k a'_j e^{j+1} + e\rho_0 + \rho = c$.
Consider the distribution $\mathcal{D}'_0(a)$: replace $A = \rho_0 + e\alpha_1 + \sum_i \alpha_i d^i$ with $A \leftarrow [2^{n_0}]$, replace $A' = \rho - \sum_i \alpha_i e d^i$ with $A' \leftarrow [2^{n_\rho}]$, and compute all remaining values $(c, [\theta]_f)$ from $\{[\alpha_i]_f\}_{i=1}^k$, A, and the public values (d, e). We claim $\mathrm{SD}(\mathcal{D}_0(a), \mathcal{D}'_0(a)) \le 2^{-\nu+2}$.
Since $q \le 2^{2\ell}$, each $a'_j = a_j + q \cdot r_j < q \cdot 2^{n_r} \le 2^{2\ell+n_r}$, so every monomial of $T_A := e\alpha_1 + \sum_{i=1}^k \alpha_i d^i$ is at most $a'_k d^k < 2^{2\ell+n_r} \cdot 2^{2\ell k} = 2^{2(k+1)\ell+n_r}$. As T_A has at most $\frac{k(k+3)}{2} \le 2k^2$ monomials and $k^2 \le 2^\ell$,

$$T_A \;<\; 2k^2 \cdot 2^{2(k+1)\ell+n_r} \;\le\; 2^{(2k+3)\ell+n_r+1} \;=\; 2^{n_0-\nu+1}.$$

Since $\rho_0 \xleftarrow{\$} [2^{n_0}]$, it smudges T_A up to statistical distance $2^{-\nu+1}$. Similarly, assuming A is already uniform on $[2^{n_0}]$, the shift of A' is $e\sum_{i=1}^k \alpha_i d^i < 2^\ell \cdot 2k^2 \cdot 2^{2(k+1)\ell+n_r} \le 2^{(2k+4)\ell+n_r+1} = 2^{n_\rho-\nu+1}$, so $\rho \xleftarrow{\$} [2^{n_\rho}]$ smudges it up to statistical distance $2^{-\nu+1}$. By a union bound over the two independent noises ρ_0, ρ, we get $\mathrm{SD}(\mathcal{D}_0(a), \mathcal{D}'_0(a)) \le 2^{-\nu+2}$.

Finally, in $\mathcal{D}'_0(a)$ the tuple $(A, c = eA + A', \{[\alpha_i]_f\}_{i=1}^k, [\theta]_f = [A - \sum_i \alpha_i d^i]_f)$ is a deterministic function of A, A' (now uniform and independent of the a'_i) and $\{[\alpha_i]_f\}_{i=1}^k$. It therefore remains to bound the distance between the distributions of $\{[\alpha_i]_f\}_{i=1}^k$ under $a'_i = a_i + qr_i$ and under $a'_i = qr_i$. Switch the coordinates one at a time. For each j, conditioned on $(r_{j'})_{j' \neq j}$, the vector $\{[\alpha_i]_f\}_i$ is a deterministic function of $[a'_j]_f$. Because $q > f$ is prime, q is invertible mod f, and with $r_j \xleftarrow{\$} [2^{n_r}]$, $n_r = \mathsf{s} + \nu$, the distribution of $[a_j + qr_j]_f = [q]_f\,[q^{-1}a_j + r_j]_f$ is $2^{-\nu}$-close to that of $[qr_j]_f$. By data processing, each coordinate swap costs at most $2^{-\nu}$, so by a union bound over the k coordinates the total cost is $k \cdot 2^{-\nu}$; the result is exactly $\mathcal{D}_1$. Combining, $\mathrm{SD}(\mathcal{D}_0(a), \mathcal{D}_1) \leq 2^{-\nu+2} + k \cdot 2^{-\nu}$. □

Equipped with this claim, the analysis of the corrupted receiver case proceeds in the same fashion as the non-batch setting.

AI Disclosure. Generative AI tools were used to produce preliminary drafts of some material, to proofread and edit the text, and to review technical claims for correctness. The final version was written by the authors, who take full responsibility for its content.

References

[ABCGJM26] Michael Adjedj, Constantin Blokh, Geoffroy Couteau, Arik Galansky, Antoine Joux, and Nikolaos Makriyannis. "Two-Round 2PC ECDSA at the Cost of 1 OLE". In: *PKC*. Full version: ePrint 2024/1950. 2026 (cit. on pp. 2–4, 6, 7, 10, 12, 17, 19, 20, 29, 32, 36, 39, 63–65, 67–70, 73–75).

[AM09] Divesh Aggarwal and Ueli Maurer. "Breaking RSA Generically Is Equivalent to Factoring". In: *EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, Berlin, Heidelberg, Apr. 2009, pp. 36–53. DOI: 10.1007/978-3-642-01001-9_2 (cit. on p. 64).

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. "Succinct Homomorphic Secret Sharing". In: *EUROCRYPT 2024, Part VI*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 301–330. DOI: 10.1007/978-3-031-58751-1_11 (cit. on pp. 9, 11, 49, 52).

[Ash26] Gilad Asharov. *Revisiting DKLs Threshold ECDSA: Enhanced OT-based VOLE and Two-Party Signing*. Cryptology ePrint Archive, Paper 2026/976. 2026. URL: https://eprint.iacr.org/2026/976 (cit. on p. 3).

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. "Constant-Size Dynamic k-TAA". In: *SCN 06*. Ed. by Roberto De Prisco and Moti Yung. Vol. 4116. LNCS. Springer, Berlin, Heidelberg, Sept. 2006, pp. 111–125. DOI: 10.1007/11832072_8 (cit. on p. 4).

[BB04] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Berlin, Heidelberg, May 2004, pp. 56–73. DOI: 10.1007/978-3-540-24676-3_4 (cit. on p. 4).

[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. 4).

[BCDLMOT24] Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker. "An Improved Threshold Homomorphic Cryptosystem Based on Class Groups". In: LNCS. 2024, pp. 24–46. DOI: 10.1007/978-3-031-71073-5_2 (cit. on p. 2).

[BCKT26] Jan Bobolz, Elizabeth C. Crites, Markulf Kohlweiss, and Akira Takahashi. "UC4Free! Existing Threshold Signatures are UC Secure". In: *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part I*. Ed. by Joan Daemen and Emmanuel Thomé. Lecture Notes in Computer Science. Springer, 2026, pp. 63–93. DOI: 10.1007/978-3-032-25291-3_3. URL: https://doi.org/10.1007/978-3-032-25291-3_3 (cit. on p. 68).

[BCMPR24] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. "Fast Public-Key Silent OT and More from Constrained Naor-Reingold". In: *EUROCRYPT 2024, Part VI*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 88–118. DOI: 10.1007/978-3-031-58751-1_4 (cit. on pp. 11, 49).

[BCMRS26] Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. "Succinct Two-Round Two-Party Signing from PCFs". In: *Advances in Cryptology – CRYPTO 2026*. Lecture Notes in Computer Science. Springer, 2026 (cit. on pp. 3, 4, 12).

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the existence of extractable one-way functions". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 505–514. DOI: 10.1145/2591796.2591859 (cit. on p. 35).

[BDSS25] Elette Boyle, Lalita Devadas, and Sacha Servan-Schreiber. "Non-Interactive Distributed Point Functions". In: *PKC 2025, Part I*. LNCS. Springer, Cham, May 2025, pp. 3–35. DOI: 10.1007/978-3-031-91820-9_1 (cit. on pp. 1, 28).

[BFHK24] Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Markulf Kohlweiss. "The Uber-Knowledge Assumption: A Bridge to the AGM". In: *CiC* 1.3 (2024), p. 31. DOI: 10.62056/anr-zoja5 (cit. on pp. 9, 11, 34, 35, 54, 64).

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Breaking the Circuit Size Barrier for Secure Computation Under DDH". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Berlin, Heidelberg, Aug. 2016, pp. 509–539. DOI: 10.1007/978-3-662-53018-4_19 (cit. on p. 28).

[BHLS25] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. "Exponent-VRFs and Their Applications". In: LNCS. Springer, Cham, June 2025, pp. 195–224. DOI: 10.1007/978-3-031-91098-2_8 (cit. on p. 4).

[BM90] Mihir Bellare and Silvio Micali. "Non-Interactive Oblivious Transfer and Applications". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 547–557. DOI: 10.1007/0-387-34805-0_48 (cit. on pp. 5, 28).

[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. 16).

[BP04] Mihir Bellare and Adriana Palacio. "The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Berlin, Heidelberg, Aug. 2004, pp. 273–289. DOI: 10.1007/978-3-540-28628-8_17 (cit. on pp. 34, 64, 68).

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *Journal of Cryptology* 13.1 (2000), pp. 143–202 (cit. on p. 16).

[CCLST20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. "Bandwidth-Efficient Threshold EC-DSA". In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Cham, May 2020, pp. 266–296. DOI: 10.1007/978-3-030-45388-6_10 (cit. on p. 3).

[CDIKLOV19] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. "Reusable Non-Interactive Secure Computation". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Cham, Aug. 2019, pp. 462–488. DOI: 10.1007/978-3-030-26954-8_15 (cit. on pp. 1, 2, 28).

[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. 3, 68).

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 738–768. DOI: 10.1007/978-3-030-45721-1_26 (cit. on p. 35).

[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 p. 64).

[CL04] Jan Camenisch and Anna Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Berlin, Heidelberg, Aug. 2004, pp. 56–72. DOI: 10.1007/978-3-540-28628-8_4 (cit. on p. 4).

[CL24] Yi-Hsiu Chen and Yehuda Lindell. "Optimizing and Implementing Fischlin's Transform for UC-Secure Zero Knowledge". In: *CiC* 1.2 (2024), p. 11. DOI: 10.62056/a66chey6b (cit. on p. 65).

[CMP20] Ran Canetti, Nikolaos Makriyannis, and Udi Peled. *UC Non-Interactive, Proactive, Threshold ECDSA*. Cryptology ePrint Archive, Report 2020/492. 2020. URL: https://eprint.iacr.org/2020/492 (cit. on p. 68).

[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. 39, 81).

[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 pp. 34, 64, 68).

[DD13] Vasily Dolmatov and Alexey Degtyarev. *Gost r 34.10-2012: Digital signature algorithm*. Tech. rep. 2013 (cit. on p. 4).

[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. 6, 9).

[DGNP25] Hila Dahari-Garbian, Ariel Nof, and Luke Parker. "Trout: Two-Round Threshold ECDSA from Class Groups". In: *ACM CCS 2025*. ACM Press, Nov. 2025, pp. 380–393. DOI: 10.1145/3719027.3765192 (cit. on pp. 2, 4, 12).

[DKLs18] 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 (SP)* (2018) (cit. on p. 3).

[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. 2–4, 12).

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. "A Verifiable Random Function with Short Proofs and Keys". In: *PKC 2005*. Ed. by Serge Vaudenay. Vol. 3386. LNCS. Springer, Berlin, Heidelberg, Jan. 2005, pp. 416–431. DOI: 10.1007/978-3-540-30580-4_28 (cit. on pp. 2, 4).

[DYY06] Yevgeniy Dodis, Aleksandr Yampolskiy, and Moti Yung. "Threshold and Proactive Pseudo-Random Permutations". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Berlin, Heidelberg, Mar. 2006, pp. 542–560. DOI: 10.1007/11681878_28 (cit. on p. 2).

[FHKRS25] Sebastian Faust, Carmit Hazay, David Kretzler, Leandro Rometsch, and Benjamin Schlosser. "Non-interactive Threshold BBS+ from Pseudorandom Correlations". In: *CT-RSA 2025*. LNCS. Springer, Cham, Feb. 2025, pp. 198–222. DOI: 10.1007/978-3-031-88661-4_9 (cit. on p. 4).

[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 pp. 7, 65).

[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. 6).

[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. 64, 75).

[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 p. 3).

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Vol. 2. Cambridge, UK: Cambridge University Press, 2004. ISBN: 978-0511721656. DOI: 10.1017/CBO9780511721656 (cit. on p. 14).

[Gro10] Jens Groth. "Short Pairing-Based Non-interactive Zero-Knowledge Arguments". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 321–340. DOI: 10.1007/978-3-642-17373-8_19 (cit. on pp. 9, 34, 64).

[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. 12, 67, 69).

[HLLXBA26] Yang Heng, Mengling Liu, Xingye Lu, Haiyang Xue, Zijian Bao, and Man Ho Au. "Efficient Construction of Threshold BBS+ Signatures and its Extensions". In: *Advances in Cryptology – PKC 2026*. Lecture Notes in Computer Science. Cryptology ePrint Archive, Report 2026/787. Springer, 2026 (cit. on p. 4).

[HSSCD26] Kaiwen He, Sacha Servan-Schreiber, Geoffroy Couteau, and Srinivas Devadas. "Concretely-Efficient Multi-Key Homomorphic Secret Sharing and Applications". In: *IEEE Symposium on Security and Privacy (S&P)* (2026) (cit. on pp. 2, 6).

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. "The elliptic curve digital signature algorithm (ECDSA)". In: *International journal of information security* 1.1 (2001), pp. 36–63 (cit. on p. 2).

[JS13] Tibor Jager and Jörg Schwenk. "On the Analysis of Cryptographic Assumptions in the Generic Ring Model". In: *Journal of Cryptology* 26.2 (Apr. 2013), pp. 225–245. DOI: 10.1007/s00145-012-9120-y (cit. on p. 64).

[KG20] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 34–65. DOI: 10.1007/978-3-030-81652-0_2 (cit. on p. 2).

[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. 64, 75).

[Kon26] Yashvanth Kondi. "Two-Party ECDSA Signing at Constant Communication Overhead". In: *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part I*. Ed. by Joan Daemen and Emmanuel Thomé. Lecture Notes in Computer Science. Springer, 2026, pp. 338–367. DOI: 10.1007/978-3-032-25291-3_12. URL: https://doi.org/10.1007/978-3-032-25291-3_12 (cit. on pp. 3, 4).

[KOR23] Yashvanth Kondi, Claudio Orlandi, and Lawrence Roy. "Two-Round Stateless Deterministic Two-Party Schnorr Signatures from Pseudorandom Correlation Functions". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, Aug. 2023, pp. 646–677. DOI: 10.1007/978-3-031-38557-5_21 (cit. on p. 2).

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11 (cit. on pp. 9, 64).

[LHD26] Simon Langowski, Kaiwen He, and Srinivas Devadas. *VROOM: Accelerating (Almost All) Number-Theoretic Cryptography Using Vectorization and the Residue Number System*. Cryptology ePrint Archive, Paper 2026/393. 2026. URL: https://eprint.iacr.org/2026/393 (cit. on pp. 2, 4).

[Lin16] Yehuda Lindell. *How To Simulate It - A Tutorial on the Simulation Proof Technique*. Cryptology ePrint Archive, Report 2016/046. 2016. URL: https://eprint.iacr.org/2016/046 (cit. on p. 14).

[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, 10, 12).

[LLZD25] Yingjie Lyu, Zengpeng Li, Hong-Sheng Zhou, and Xudong Deng. "Threshold ECDSA in Two Rounds". In: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*. Ed. by Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier. ACM, 2025, pp. 2937–2950. DOI: 10.1145/3719027.3765176. URL: https://doi.org/10.1145/3719027.3765176 (cit. on pp. 2, 4, 12).

[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. 7).

[Mil76] Gary L. Miller. "Riemann's hypothesis and tests for primality". In: *Journal of Computer and System Sciences* (1976), pp. 300–317 (cit. on p. 81).

[MYG24] Nikolaos Makriyannis, Oren Yomtov, and Arik Galansky. "Practical Key-Extraction Attacks in Leading MPC Wallets". In: *ACM CCS 2024*. ACM Press, Nov. 2024, pp. 3053–3064. DOI: 10.1145/3658644.3670359 (cit. on p. 3).

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. "Distributed Pseudo-random Functions and KDCs". In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Berlin, Heidelberg, May 1999, pp. 327–346. DOI: 10.1007/3-540-48910-X_23 (cit. on p. 2).

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. "The Rise of Paillier: Homomorphic Secret Sharing and Public-Key Silent OT". In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 678–708. DOI: 10.1007/978-3-030-77870-5_24 (cit. on pp. 5, 11, 18, 28, 29, 32).

[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. 64).

[Pol23] Joop van de Pol. *Don't overextend your oblivious transfer*. The Trail of Bits Blog. https://blog.trailofbits.com/2023/09/20/dont-overextend-your-oblivious-transfer/, accessed: May 2026. 2023 (cit. on p. 3).

[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. 64).

[PS00] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and Blind Signatures". In: *Journal of Cryptology* 13.3 (June 2000), pp. 361–396. DOI: 10.1007/s001450010003 (cit. on p. 16).

[PS96] David Pointcheval and Jacques Stern. "Security Proofs for Signature Schemes". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 387–398. DOI: 10.1007/3-540-68339-9_33 (cit. on p. 16).

[PUB00] FIPS PUB. "Digital signature standard (DSS)". In: *Fips pub* (2000), pp. 186–192 (cit. on p. 64).

[TZ23] Stefano Tessaro and Chenzhi Zhu. "Revisiting BBS Signatures". In: *EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. LNCS. Springer, Cham, Apr. 2023, pp. 691–721. DOI: 10.1007/978-3-031-30589-4_24 (cit. on p. 4).

[XLXADY24] Zhikang Xie, Mengling Liu, Haiyang Xue, Man Ho Au, Robert H. Deng, and Siu-Ming Yiu. "Direct Range Proofs for Paillier Cryptosystem and Their Applications". In: *ACM CCS 2024*. ACM Press, Nov. 2024, pp. 899–913. DOI: 10.1145/3658644.3690261 (cit. on p. 7).

[ZLLZH24] C Zhang, Y Liu, F Leng, Q Zhao, and Z He. *RFC 9563: SM2 Digital Signature Algorithm for DNSSEC*. 2024 (cit. on p. 4).

A Discussions

A.1 On the Use of Knowledge of Exponent Assumptions

Comparison with ABCGJM. Firebolt's security is proven in the random oracle model under standard falsifiable assumptions (short-exponent variants of DCR and discrete-logarithm assumptions) together with SNARK-type knowledge of exponent assumptions. The closest related work is that of Adjedj *et al.* (ABCGJM, [ABCGJM26]), from which we adapt several zero-knowledge proofs (both works must generate and prove well-formedness of a Paillier modulus in the setup) and which also provides, under the hood, a NIVOLE protocol; their analysis, however, is conducted directly in the generic group model (GGM), on top of standard falsifiable assumptions on Paillier groups similar to ours. While Firebolt's most immediate advantage over ABCGJM is its much better performance (in communication and computation), we believe that avoiding the GGM is a useful side feature, reducing the reliance on heuristics to the random oracle model alone.

How ABCGJM uses the GGM. ABCGJM relies on the GGM to obtain straight-line extraction from Fiat-Shamir-compiled Σ-protocols, so as to avoid Fischlin's transform and its significant performance cost (see Section A.3 below). Their approach proceeds in three steps, and the result is made non-interactive with Fiat-Shamir:

(i) run a standard Σ-protocol over an unknown-order group;

(ii) separately commit to the witness over an elliptic curve (modeled as a generic group);

(iii) augment the Σ-protocol with a proof that the committed values equal the witness.

In the analysis, the simulator extracts the witness from the commitment by observing the generic-group queries, and the forking lemma (Lemma 3.8) shows that the extracted witness satisfies the target relation — the key being that the forking lemma is used only in the analysis, never by the (straight-line) simulator.

The GGM as a knowledge assumption. Any such use of the GGM *can* be framed as a knowledge assumption: simply *assume* a straight-line extractor recovering the witness from the NIZK, and the GGM analysis becomes a proof that this assumption holds in the GGM. The resulting assumption, however, is a highly unconventional "cross-group, interactive" one: it concerns an interactive game (with the random oracle as challenger) involving secrets over both an RSA group and an elliptic curve, and asserts that whenever a certain condition on the game's output is met, the secrets can be extracted from their joint commitments over both groups. This essentially amounts

to assuming the security of the full construction, and isolating such an assumption is no more meaningful than analyzing security in the GGM directly.[2]

Our approach: KZG over an RSA group. Our approach departs sharply from this methodology: rather than designing cross-group Σ-protocols to port witnesses onto an elliptic curve where a generic extractor is available, we design new zero-knowledge proofs from scratch that avoid Σ-protocols altogether and instead leverage polynomial commitments, instantiated with the KZG scheme [KZG10] over an RSA group. Beyond much more compact and faster proofs, this removes any unconventional cross-group assumption: we rely instead on the k-KZG assumption of [BFHK24], a natural knowledge-of-exponent assumption closely related to Damgård's original one [Dam92; BP04] and in the spirit of the variants introduced by Groth [Gro10] and their many follow-ups — an assumption of the same nature as those already underpinning many SNARKs deployed in the real world.

Why not the GGM over RSA groups? There remains one difference: previous works typically make these assumptions over prime-order elliptic curves equipped with a bilinear map, whereas we use ours over an RSA group. To the best of our knowledge, this has no impact on the confidence one should place in the assumption, but it does mean that, unlike ABCGJM, we cannot sidestep it by working in the GGM: the GGM is not an appropriate heuristic for RSA groups, which are well known to exhibit non-generic behavior.[3] We refer the reader to Section 6.1 for additional discussion of our concrete knowledge assumption and to Remark 3.5 for how we maintain composability despite using knowledge-of-exponent assumptions.

A.2 On the Use of Short-Exponent Assumptions

As in ABCGJM [ABCGJM26], our protocols use short (i.e., 2ℓ-bit) secret exponents over the Paillier group, and their security relies on variants of the *short-exponent discrete logarithm* assumption (SEDL): computing discrete logarithms remains hard when the exponent is sampled from a short interval. SEDL is a standard and well-studied assumption, essentially as old as the discrete logarithm problem itself: the generic $O(\sqrt{T})$-time algorithm for T-bounded exponents goes back to Pollard's seminal work [Pol78], the use of short secret exponents is explicitly standardized in DSS for the DSA algorithm [PUB00], and the assumption was analyzed by van Oorschot and Wiener [OW96], who showed in particular that the order of the group should have no small factors. Moreover, over such groups, SEDL is known to be *equivalent* to its decisional counterpart, the short-exponent indistinguishability assumption (SEI, a name coined in [CKLR21]): the equivalence was proved by Gennaro [Gen00] over $\mathbb{Z}_p^*$, extended by Koshiba and Kurosawa [KK04] to arbitrary prime-order groups, and recently by ABCGJM to groups of unknown order such as the Paillier group; we refer to the full version of [ABCGJM26] (Appendix A) for a detailed overview of the history of the assumption and of the supporting literature. The exact variant used in this work is a *power* short-exponent indistinguishability assumption (power-SEI, Assumption 3.11), which states that a tuple $(t, t^x, \ldots, t^{x^k})$ computed from a short exponent x is indistinguishable from the same tuple computed from a full-range exponent. As a side contribution of independent interest, we generalize

[2] It says, in effect, that any GGM proof can be de-idealized under the trivial standard-model assumption that the protocol is secure.

[3] Adaptations of the GGM to RSA groups have been used in the past [AM09], but they are more ad hoc in nature and have known shortcomings [JS13].

the SEI-to-SEDL reduction of [ABCGJM26] to the k-power setting, proving that power-SEI and power-SEDL are equivalent (Section E).

A.3 Why Not Fischlin?

A natural alternative route to straight-line extraction is to compile a standard Σ-protocol for short DFO commitment opening with the Fischlin transform [Fis05], which a recent work of Chen and Lindell [CL24] showed to be surprisingly practical in the Schnorr setting: with optimized parameters ($\rho = 32$ parallel repetitions with $b = 4$-bit challenges, for $\rho \cdot b = 128$-bit soundness), proving knowledge of a discrete logarithm over secp256k1 costs 0.41ms on a modern laptop, a mere 15× overhead over plain Fiat-Shamir. This factor, however, is specific to the Schnorr setting, where the per-repetition cost (a single short-exponent elliptic-curve multiplication) is small enough to be comparable to the hashing effort: the transform inherently requires ρ independent first messages together with an expected $\rho \cdot 2^b$ hash evaluations. In our setting, each repetition instead requires a fresh multi-exponentiation over $\mathbb{Z}_N$ (even granting Fischlin the same short-exponent optimizations we use). On our machine, using VroomSSL, a short-exponent DFO commitment costs about 1ms; plugging this into the parameter-selection methodology of [CL24] yields optimal parameters $(\rho, b, t) = (16, 8, 13)$ (for 128-bit soundness), i.e., 16 multi-exponentiations over $\mathbb{Z}_N$ together with an expected 4,096 hash evaluations, for a prover cost of the order of 15ms—compared to a single additional multi-exponentiation ($\mathcal{C}$) for our KZG-based proof, on top of the commitment itself which both approaches must compute. This prover cost could likely be reduced further with precomputation, since g, h are fixed system parameters and standard fixed-base exponentiation techniques apply.

However, the real culprit for not using Fischlin is communication, not computation, and precomputation does not help here: the Fischlin proof must transmit all ρ first messages in full—i.e., 16 elements of $\mathbb{Z}_N$ (over 6kB for $\log N = 3072$, before counting responses), whereas our proof adds a single short integer to the sender message ($\mathcal{C}$ itself is reverse-sampled by the verifier), for a total of 535 bytes. We stress that this is not an artifact of suboptimal parameters: since communication scales with ρ while soundness requires $\rho \cdot b \geq 128$, shrinking the Fischlin proof means increasing b, and the prover's expected $\rho \cdot 2^b$ hash evaluations grow exponentially — already at $(\rho, b) = (8, 16)$, the prover performs an expected 2^{19} hash evaluations, which dwarfs the cost of the multi-exponentiations,[4] while the proof still transmits 8 elements of $\mathbb{Z}_N$ ($\approx$ 3kB).

B 2PC ECDSA from VOLE

B.1 The Protocol

In this subsection we design a two-party ECDSA protocol Π_{ECDSA} from the RNIVOLE algorithms of Definition 5.1, in the $(\mathcal{F}_{\mathsf{setup}}, \mathcal{F}_{\mathsf{keygen}})$-hybrid model.

[4] Using the (non-hardware-accelerated) SHA-256 timings measured in [CL24] (0.2–0.6μs per hash), 2^{19} hash evaluations cost 105–315ms, an order of magnitude more than the 8ms of 8 DFO commitments; even at best-case hardware-accelerated hash speeds ($\sim$ 30ns), the hash cost would already match the multi-exponentiation cost, before accounting for the extra input length of the DFO response, which spans more than one hash block.

Functionality Key Generation Functionality $\mathcal{F}_{\mathsf{keygen}}$

Parties. The functionality interacts with two players P_1 and P_2, and an adversary $\mathcal{A}$.

Operation. Upon receiving $(\mathsf{keygen}, \mathsf{sid}, \mathsf{SetupData})$ from P_1 and $(\mathsf{keygen}, \mathsf{sid}, \mathsf{SetupData})$ from P_2 with matching $\mathsf{SetupData}$, where sid encodes the curve $(\mathbb{G}, g, q)$ and the identities $\mathsf{P}_1, \mathsf{P}_2$, and where (N, t, u) are parsed from $\mathsf{SetupData}$:

- *If* P_1 *is honest:* sample $x_1 \xleftarrow{\$} \mathbb{Z}_q$ and run $(\mathsf{e}_{x_1}, \mathsf{d}_{x_1}) \xleftarrow{\$} \mathsf{VOLE.Register}^{\mathsf{H}}(\mathsf{SetupData}, x_1)$.
- *If* P_1 *is corrupt:* receive from $\mathcal{A}$ either $\perp$ (in which case, abort) or a tuple $(x_1, \mathcal{E}, \gamma, f)$ where $\mathcal{E} = (\mathcal{E}_t, \mathcal{E}_u) \in (\mathbb{Z}^*_{N^2})^2$; abort unless
  1. $x_1 \in [2^{n_x}]$, $\gamma \in \mathbb{Z}$, and $f \in [2^\ell]$;
  2. $\mathcal{E}_t^f = t^\gamma \cdot (1+N)^{f x_1} \bmod N^2$ and $\mathcal{E}_u^f = u^\gamma \bmod N^2$.

  Set $\mathsf{e}_{x_1} := \mathcal{E}$. (The checks match the slack instances of Definition 4.7: the attacker gets to pick a slack instance when P_1 is corrupt.)
- *If* P_2 *is honest,* sample $x_2 \xleftarrow{\$} \mathbb{Z}_q$; *if* P_2 *is corrupt,* receive $x_2 \in \mathbb{Z}_q$ from $\mathcal{A}$ (abort on malformed input).
- Set $X_1 := g^{x_1}$, $X_2 := g^{x_2}$, and $X := g^{x_1 + x_2}$. Send $(X, X_1, X_2, \mathsf{e}_{x_1})$ to both parties; additionally, if P_1 is honest, send (x_1, d_{x_1}) to P_1, and if P_2 is honest, send x_2 to P_2.

Functionality 3: Key generation functionality $\mathcal{F}_{\mathsf{keygen}}$.

The functionality idealizes both the distributed key generation and the registration phase of the RNIVOLE; the realization is straightforward: the parties run a simulatable DKG, P_1 registers x_1 via $\mathsf{VOLE.Register}$, and P_2 verifies via $\mathsf{VOLE.VerifyRegister}$ against the public share X_1.

Protocol. We give a two-round protocol Π_{ECDSA} realizing $\mathcal{G}_{\mathsf{psECDSA}}$ using the RNIVOLE algorithms directly, with a single VOLE message per signature.

Protocol Two-party ECDSA from registered VOLE Π_{ECDSA}

At the beginning of each round, the attacker writes a public input of the form (init), (keygen), or (sign, m) for $m \in \mathbb{Z}_q$ of his choosing. The parties run the corresponding phase that round. The inputs are subject to the constraint that (init) must occur first, followed by (keygen), followed by any number of (sign, m) inputs in any order; each input is consumed after it is used. The phases of the protocol are specified as follows:

Parameters. An elliptic curve $(\mathbb{G}, g, q)$ of prime order q and a random oracle $\mathsf{RO} : \mathbb{G}^3 \to \mathbb{Z}_q$, together with the random oracles H, H' of the RNIVOLE algorithms (domain-separated from RO).

Initialization. P_1 and P_2 invoke $\mathcal{F}_{\mathsf{setup}}$ (Functionality 1) with a session identifier sid encoding the curve $(\mathbb{G}, g, q)$, the parameters $(n_{\mathrm{rsa}}, \ell, \nu)$, and the parties $\mathsf{P}_1, \mathsf{P}_2$, with P_1 playing the server S and P_2 playing the client C. Both parties receive the setup data $\mathsf{SetupData}$; P_1 additionally receives the trapdoor (p_1, p_2, d) and sets $\mathsf{sk} := d$. If $\mathcal{F}_{\mathsf{setup}}$ aborts, the protocol aborts.

KeyGen. P_1 and P_2 invoke $\mathcal{F}_{\mathsf{keygen}}$ (Functionality 3) on $(\mathsf{keygen}, \mathsf{sid}, \mathsf{SetupData})$, obtaining shares $x_1, x_2 \in \mathbb{Z}_q$, the joint public key $X = g^{x_1+x_2}$ with public shares $X_1 := g^{x_1}$ and $X_2 := g^{x_2}$, and the registration e_{x_1} (with d_{x_1} delivered to P_1). If $\mathcal{F}_{\mathsf{keygen}}$ aborts, the protocol aborts.

Signing. On a message $m \in \mathbb{Z}_q$:

1. *Round 1* ($\mathsf{P}_1 \to \mathsf{P}_2$). Sample $k_1 \leftarrow \mathbb{Z}_q$ and send $(R_1, R_1') := (g^{k_1},\, X_2^{k_1})$ to P_2.
2. *Round 2* ($\mathsf{P}_2 \to \mathsf{P}_1$). Abort if $R_1^{x_2} \neq R_1'$. Otherwise:
   (a) Sample $k_2 \leftarrow \mathbb{Z}_q$, set $R_2 := g^{k_2}$, and compute the tweak $\delta := \mathsf{RO}(R_1, R_2, R_1^{k_2}) \in \mathbb{Z}_q$ and the nonce $R := (R_1 \cdot g^{\delta})^{k_2} \in \mathbb{G}$. Letting $r \in \mathbb{Z}_q$ denote the x-coordinate of R reduced modulo q, compute
   $$a := k_2^{-1}(m + r \cdot x_2) \bmod q, \qquad b := k_2^{-1} r \bmod q.$$
   (b) Run $v \xleftarrow{\$} \mathsf{VOLE.Send}(\mathsf{SetupData}, \mathsf{e}_{x_1}, b, a)$ and send (R_2, R_2', v) to P_1, where $(R_2, R_2') := (g^{k_2},\, X_1^{k_2})$.
3. *Output* (*local at* P_1). P_1 computes $w \leftarrow \mathsf{VOLE.Output}(\mathsf{SetupData}, \mathsf{sk}, \mathsf{e}_{x_1}, \mathsf{d}_{x_1}, v)$ and aborts unless $R_2^{x_1} = R_2'$ and $w \neq \perp$. Otherwise:
   – Compute $\delta := \mathsf{RO}(R_1, R_2, R_2^{k_1}) \in \mathbb{Z}_q$ (matches P_2's value since $R_2^{k_1} = R_1^{k_2}$).
   – Set $R := R_2^{k_1+\delta}$ and $s := (k_1 + \delta)^{-1} \cdot w \bmod q$.
   – Output (R, s) on m if it verifies under the joint public key X; otherwise abort.

Protocol 7: Two-party ECDSA protocol Π_{ECDSA} from registered VOLE.

B.2 Security Analysis

In this subsection we define the two-party ECDSA-with-presignatures game $\mathcal{G}_{\mathsf{psECDSA}}$ — which is known to be unforgeable against every attacker in the generic group model [ABCGJM26; GS22] — and show that Π_{ECDSA} securely realizes it.

Functionality Two-party ECDSA with Presignatures $\mathcal{G}_{\mathsf{psECDSA}}$

- **Initialization.** Upon receiving $(\mathsf{init}, \mathsf{sid})$ from P_1 and $(\mathsf{init}, \mathsf{sid})$ from P_2 with matching sid, where sid encodes an elliptic curve $(\mathbb{G}, g, q)$ of prime order q generated by g and the identities $\mathsf{P}_1, \mathsf{P}_2$: record sid and treat the game as activated. All commands below are ignored until the game is activated, and (keygen) is accepted at most once.

- **KeyGen.**
  1. Upon receiving (keygen) from both P_1 and P_2, sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{ECDSAGen}(\mathbb{G}, g, q)$, store it in memory, and send $(\mathsf{public\text{-}key}, \mathsf{pk})$ to $\mathcal{S}$.
- **Presigning.** Upon receiving $(\mathsf{presign})$ from $\mathcal{S}$, sample $k_0 \leftarrow \mathbb{Z}_q$, set $R_0 = g^{k_0}$, store (R_0, k_0) in memory, and send $(\mathsf{presign}, R_0)$ to $\mathcal{S}$.
- **Tweak.** Upon receiving $(\mathsf{tweak}, R_0, m, \tau)$ from $\mathcal{S}$ with $\tau \in \mathbb{Z}_q$:
  - If no tuple (R_0, k_0) is stored, ignore the command;
  - Else if a tuple (R_0, m, τ, μ) is already stored, return μ to $\mathcal{S}$;
  - Else sample $\mu \leftarrow \mathbb{Z}_q$, store (R_0, m, τ, μ), and send $(\mathsf{modifier}, R_0, m, \tau, \mu)$ to $\mathcal{S}$.
- **Signing.**
  1. On receipt of (sign, m) from $\mathcal{S}$, record (sign, m) as a pending entry.
  2. On receipt of $(\mathsf{sign}, R_0, \tau, \mu, m)$ from $\mathcal{S}$, if no pending (sign, m) entry exists, ignore the command. Otherwise, consume one pending (sign, m) entry and:
     - If a tuple (R_0, m, τ, μ) is recorded, retrieve (R_0, k_0), set $k = \tau \cdot k_0 + \mu$, and delete every memory entry whose first component is R_0. Set $R = g^k$, compute $s = \mathsf{ECDSASign}(\mathsf{sk}, m; k)$, record $(m, (R, s))$ as the output associated with nonce R, and send (nonce, R) to both P_1 and P_2.
     - Otherwise, send $(\mathsf{sign}, \mathsf{abort}, m)$ to both P_1 and P_2.
  3. At the end of the execution, output the public key and the sequence of message-signature pairs $(m, (R, s))$ for all m for which a signature was generated.

Game 2: Two-party ECDSA with presignatures game $\mathcal{G}_{\mathsf{psECDSA}}$.

Standalone security. Note that $\mathcal{G}_{\mathsf{psECDSA}}$ is a game in the sense of Definition 3.2: all reactive inputs — the messages to be signed — are public and supplied by the adversary. This definition already provides a lot of security, as unforgeability is baked into the ideal functionality: any attacker in the protocol that can forge signatures can be used to forge signatures of the underlying scheme, since the ideal functionality effectively acts as a signing oracle (more precisely, as the extended unforgeability oracle of [ABCGJM26]). In addition, lifting the security to the UC setting can be done via the threshold signature functionality of [CGGMP20; CMP20; BCKT26], whereby the protocol can be shown to realize the threshold signature functionality by reducing to the standalone security.
The simulation for a corrupted party extracts the attacker's nonce contribution under the following knowledge of exponent assumption.

Assumption B.1 (KEA1 [Dam92; BP04])**.** We say that KEA1 holds over $(\mathbb{G}, g, q)$ if for every PPT $\mathcal{A}$, there exists a PPT Ext and a negligible function $\mu : \mathbb{N} \to \mathbb{R}$ such that

$$\Pr[X^a = Y \wedge X \neq g^x] \leq \mu(\ell),$$

where the probability is over the random choices of $a \xleftarrow{\$} \mathbb{Z}_q$, $r \xleftarrow{\$} \{0,1\}^*$, $(X, Y) \xleftarrow{\$} \mathcal{A}(\mathbb{G}, g, q, g^a; r)$, and $x \xleftarrow{\$} \mathsf{Ext}(\mathbb{G}, g, q, g^a; r)$.

Theorem B.2. *Assume the assumptions of Theorem 6.3 and that KEA1 holds over $(\mathbb{G}, g, q)$. Then, modeling RO, H, and H' as (domain-separated) random oracles, the protocol Π_{ECDSA} securely realizes the game $\mathcal{G}_{\mathsf{psECDSA}}$ (Definition 3.4) in the $(\mathcal{F}_{\mathsf{setup}}, \mathcal{F}_{\mathsf{keygen}})$-hybrid model.*

Corollary B.3. *Assume further that no PPT attacker interacting with the game $\mathcal{G}_{\mathsf{psECDSA}}$ outputs a valid signature on a message m^* that was never submitted for signing, except with negligible probability (this holds in the generic group model [ABCGJM26; GS22]). Then the protocol inherits this security property: no PPT attacker corrupting a single party in Π_{ECDSA} — and scheduling, via the public-input tape, any (polynomial) number of concurrent signing sessions on messages of its choice — outputs a valid signature on a message m^* for which no (sign, m^*) public input was consumed, except with negligible probability.*

Proof sketch of Theorem B.2. The analysis is very close to the security analysis of the VOLE game from the main body (Theorem 6.3); we sketch the simulation for each corruption case and the hybrid sequence, without reproving the steps that are imported unchanged.

Corrupted P_1. This is the easy case. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{setup}}$ and $\mathcal{F}_{\mathsf{keygen}}$, thereby receiving from the attacker the setup trapdoor (p_1, p_2, d) (corrupt-S branch of Functionality 1) and the slack instance $(x_1, \mathcal{E}, \gamma, f)$ (corrupt-P_1 branch of Functionality 3), and embeds the game's public key by setting $X_2 := \mathsf{pk} \cdot g^{-x_1}$. Upon receiving a Round-1 message (R_1, R_1'), $\mathcal{S}$ runs the KEA1 extractor (with respect to (g, X_2)) to obtain k_1, and continues iff $R_1 = g^{k_1}$ and $R_1' = X_2^{k_1}$; by KEA1, this deviates from the honest check $R_1^{x_2} = R_1'$ with negligible probability. $\mathcal{S}$ then runs $\mathsf{presign}$, tweak, and sign on the attacker's chosen message m, and obtains a signature (R, s) from $\mathcal{G}_{\mathsf{psECDSA}}$. It then samples a random $\delta \xleftarrow{\$} \mathbb{Z}_q$, sets $R_2 := R^{(k_1+\delta)^{-1}}$ and $R_2' := R_2^{x_1}$, and programs the random oracle so that the query $(R_1, R_2, R_2^{k_1})$ lands on δ. Finally, $\mathcal{S}$ crafts the VOLE message v so that it decodes to $w := s \cdot (k_1 + \delta) \bmod q$, reusing the corrupted-receiver simulation of Theorem 6.3 (which uses the trapdoor d and the slack instance); P_1's output then equals the game's signature (R, s).

Corrupted P_2: sketch of the hybrid sequence. We follow the corrupted-sender analysis of Theorem 6.3, which we import wholesale; only the hybrids at the two ends are specific to the ECDSA layer.

1. *Original game.* $\mathcal{S}$ runs the honest P_1 and locally emulates the random oracles, $\mathcal{F}_{\mathsf{setup}}$ (honest branch, so $\mathcal{S}$ knows the trapdoor), and $\mathcal{F}_{\mathsf{keygen}}$ (corrupt-P_2 branch, so $\mathcal{S}$ receives x_2 from the attacker).

2. *VOLE block (imported).* The corrupted-sender hybrid sequence of Theorem 6.3 is replayed verbatim on the embedded VOLE: $\mathcal{S}$ extracts from each VOLE message v a pair (a, b) (via the knowledge extractor and the forking-based smallness claims) and replaces P_1's output $w \leftarrow \mathsf{VOLE.Output}(\dots)$ with the ideal value $b \cdot x_1 + a \bmod q$. The claims bounding the deviation are unchanged. Since the VOLE inputs are here computed from adversarially-chosen nonces (rather than pre-specified), the knowledge assumption is invoked with benign auxiliary inputs, as discussed in Remark 6.2.

3. *Presigning.* $\mathcal{S}$ replaces P_1's Round-1 message: it queries $\mathcal{G}_{\mathsf{psECDSA}}.\mathsf{presign}$, obtains R_0, and forwards $(R_1, R_1') := (R_0, R_0^{x_2})$ to $\mathcal{A}$ as P_1's message (computable since $\mathcal{S}$ holds x_2).

4. *Tweak.* When $\mathcal{A}$ queries the random oracle on $(R_1, R_2, R_1^{k_2})$, $\mathcal{S}$ extracts k_2 under KEA1 (with respect to (g, R_1)), calls $\mathcal{G}_{\mathsf{psECDSA}}.\mathsf{tweak}$ with $(R_0, m, \tau := k_2)$ to obtain μ, and programs the random oracle to return $\delta := \mu \cdot k_2^{-1} \in \mathbb{Z}_q$.

5. *Signing.* Upon a Round-2 message (R_2, R_2', v), $\mathcal{S}$ checks $R_2' = X_1^{k_2}$ using the extracted k_2 (in place of the honest check $R_2^{x_1} = R_2'$) and checks that the extracted (a, b) equal the honest values $(k_2^{-1}(m + r \cdot x_2),\ k_2^{-1} r)$, where r is computed from the nonce $R = (R_0 \cdot g^{\delta})^{k_2}$. If the checks pass, $\mathcal{S}$ submits $(\mathsf{sign}, R_0, k_2, \mu, m)$ to $\mathcal{G}_{\mathsf{psECDSA}}$, which assembles $k = k_2 \cdot k_0 + \mu = k_2(k_0 + \delta)$ and outputs the signature (R, s); the game's nonce matches the real-protocol nonce by construction. If any check fails, $\mathcal{S}$ aborts as the honest P_1 would (its candidate signature would fail verification); the bad event — a malformed (a, b) that nonetheless yields a verifying signature — amounts to guessing the share of the honest party, and is handled as in other analyses, e.g. [ABCGJM26].

6. *Removing x_1.* At this point x_1 is used only to define X_1 and e_{x_1}: $\mathcal{S}$ sets $X_1 := \mathsf{pk} \cdot g^{-x_2}$ (embedding the game's public key) and replaces e_{x_1} with an encryption of 0, exactly as in the last hybrid of Theorem 6.3 (under the IND-CPA security of extended Paillier-ElGamal).

□

C Realizing the Setup Functionality $\mathcal{F}_{\mathsf{setup}}$

We give the concrete non-interactive protocol Π_{setup} that realizes the setup functionality $\mathcal{F}_{\mathsf{setup}}$ (Functionality 1) in the random-oracle model. It makes use of the zero-knowledge proofs $\Pi_{\mathsf{mod}}, \Pi_{\mathsf{fac}}$ from [ABCGJM26] for proving well-formedness of an RSA modulus (Protocols 8 and 9; see Section 3.6 and Section D). The protocol is executed once, ahead of time, by the server; the resulting setup data is then verified by the client.

C.1 Proof of KZG Parameters ($\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$)

The setup relies on the proof $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$ of relaxed correctness of the KZG parameters, targeting the promise problem $\mathsf{kzg}_{\mathsf{PE}}$ (Definition 3.22). The protocol is described in Section D (Protocol 10), and its security guarantees — special soundness with extraction slack $\pm 2^{2\ell+\nu+1}$ (Lemma 3.26) and $(2^{-\nu})$-HVZK (Lemma 3.23) — are stated in Section 3.6.

C.2 Concrete Setup Protocol

Algorithm VOLE Setup

Parameters. $\mathsf{pp} = (\ell, \nu, \kappa, \tau)$. $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\ell}$ is modeled as a random oracle.

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

Input. Security parameter ℓ.

$\mathsf{VOLE.Setup}^{\mathsf{H}}(1^\ell)$.

1. Sample $(\mathsf{pk_{PE}}, \mathsf{sk_{PE}}) \xleftarrow{\$} \mathsf{PE}^* \,.\, \mathsf{KeyGen}(1^\ell)$. Parse $\mathsf{pk_{PE}} = (N, u_0, u, t_0, t, s_0, s)$, $\mathsf{sk_{PE}} = (p_1, p_2, d)$.
   - Generate Paillier-Blum well-formedness proof $\pi_{\mathsf{mod}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{mod}}(\mathrm{aux}, N; p_1, p_2)$.
   - Generate small factors proof $\pi_{\mathsf{fac}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{fac}}(\mathrm{aux}, N; p_1, p_2)$
   - Generate DFO-KZG params proof $\pi_{\mathsf{par}} \xleftarrow{\$} \Pi^{\mathrm{FS}}_{\mathsf{par}^*}(\mathrm{aux}, N, s_0, t_0, u_0; d)$.
2. Generate $g_1, h \in \mathbb{G}'$ as nothing-up-my-sleeve numbers using $(\mathbb{G}', g_0, q')$ and aux as a seed.

   Public Output. $\mathsf{SetupData} = (\mathsf{pk_{PE}}, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{par}}, g_0, g_1, h)$.

   Secret Output. $\mathsf{sk} = d$

$\mathsf{VOLE.VerifySetup}^{\mathsf{H}}(\mathsf{SetupData})$: output 1 iff all verifications below succeed.

1. Parse $\mathsf{SetupData} = (\mathsf{pk_{PE}}, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{par}}, g_0, g_1, h)$ and $\mathsf{pk_{PE}} = (N, u_0, u, t_0, t, s_0, s)$.

   Let $n_{\mathrm{rsa}} \leftarrow \mathsf{SampleRSA}(1^\ell)$.
2. Verify that $N \geq 2^{n_{\mathrm{rsa}}-1}$, $s_0, t_0, u_0 \in \mathbb{J}_N$, and $s = s_0^{2N}, t = t_0^{2N}, u = u_0^{2N} \bmod N^2$.
3. Verify that $g_1, h \in \mathbb{G}'$ can be reconstructed according to Item 2.
4. Verify the ZK proofs:
   (a) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{mod}}}(\mathrm{aux}, N, \pi_{\mathsf{mod}}) = 1$
   (b) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{fac}}}(\mathrm{aux}, N, \pi_{\mathsf{fac}}) = 1$
   (c) $\mathsf{vrfy}_{\Pi^{\mathrm{FS}}_{\mathsf{par}}}(\mathrm{aux}, N, s_0, t_0, u_0, \pi_{\mathsf{par}}) = 1$

Algorithm 7: Non-interactive setup algorithm executed ahead-of-time by the server.

C.3 Security of the Setup Protocol

We now state and prove that the concrete setup securely realizes $\mathcal{F}_{\mathsf{setup}}$. Let Π_{setup} denote the following non-interactive protocol between the server S and the client C: the server runs $(\mathsf{SetupData}, d) \xleftarrow{\$} \mathsf{VOLE.Setup}^{\mathsf{H}}(1^\ell)$ (Algorithm 7) and sends $\mathsf{SetupData}$ to the client; the client runs $\mathsf{VOLE.VerifySetup}^{\mathsf{H}}(\mathsf{SetupData})$ and aborts on output 0. Upon (non-aborting) termination, both parties output (N, u_0, t_0, s_0, h), parsed from $\mathsf{SetupData}$, and the server additionally outputs the trapdoor (p_1, p_2, d).

Theorem C.1. *In the random oracle model, assuming the quadratic residuosity assumption (Assumption 3.13) and the hardness of computing discrete logarithms between the auxiliary Pedersen bases g, h of Π_{fac} (cf. Protocol 9 and Lemma 3.25), protocol Π_{setup} securely realizes the setup functionality $\mathcal{F}_{\mathsf{setup}}$ (Functionality 1).*

The zero-knowledge and soundness properties used in the proof are supplied by the lemmas of Section 3.6: Lemma 3.23 for the (statistical) zero-knowledge of the three proofs, Lemma 3.24 for the

soundness of Π_{mod}, and the special-soundness Lemmas 3.25 and 3.26, combined with the Generalized Forking Lemma (Lemma 3.8), for the knowledge soundness of $\Pi^{\mathrm{FS}}_{\mathsf{fac}}$ and $\Pi^{\mathrm{FS}}_{\mathsf{par}}$ (the Fiat–Shamir compilation of the $r_{\mathsf{kzg}_{\mathsf{PE}}}$-fold repetition of Protocol 10).

Proof. We treat the two corruption cases separately. In both cases, the simulator $\mathcal{S}$ locally emulates the random oracle H, maintaining a list L of query-answer pairs, and receives the Pedersen base h from $\mathcal{F}_{\mathsf{setup}}$ at initialization. Before answering any oracle query of the adversary, $\mathcal{S}$ programs H so that the nothing-up-my-sleeve derivation of the Pedersen base (Item 2 of Algorithm 7) returns h. Since the derivation queries the oracle on a fixed public input and h is uniform over $\mathbb{G}$, this programming induces the same distribution on the oracle as honest emulation, and the derived Pedersen base coincides with the functionality's h in both corruption cases.

Corrupted client. We proceed through a short sequence of hybrids, starting from the real game.

Hybrid_1. This is the real game: $\mathcal{S}$ runs the honest server, i.e., runs $(\mathsf{SetupData}, d) \xleftarrow{\$} \mathsf{VOLE.Setup}^{\mathsf{H}}(1^\ell)$ and sends $\mathsf{SetupData} = (\mathsf{pk}_{\mathsf{PE}}, \pi_{\mathsf{mod}}, \pi_{\mathsf{fac}}, \pi_{\mathsf{par}}, g_0, g_1, h)$ to the corrupted client, emulating (and h-programming) the random oracle as described above. By the programming argument, the honestly derived Pedersen base equals the functionality's h and the oracle is distributed as an honest oracle, so this identification is purely syntactic.

Hybrid_2. In this hybrid, $\mathcal{S}$ relies on the zero-knowledge simulators to emulate all setup proofs, programming the random oracle as required by these simulators: it sets $\pi_{\mathsf{mod}} \xleftarrow{\$} \mathcal{S}_{\mathsf{mod}}(\mathrm{aux}, N)$, $\pi_{\mathsf{fac}} \xleftarrow{\$} \mathcal{S}_{\mathsf{fac}}(\mathrm{aux}, N)$, and $\pi_{\mathsf{par}} \xleftarrow{\$} \mathcal{S}_{\mathsf{par}}(\mathrm{aux}, N, s_0, t_0, u_0)$. By Lemma 3.23 ($\Pi_{\mathsf{mod}}$ is perfect HVZK, Π_{fac} is $(2^{-\nu+1})$-HVZK, and each of the $r_{\mathsf{kzg}_{\mathsf{PE}}}$ repetitions of Protocol 10 is $(2^{-\nu})$-HVZK), and since programming the Fiat–Shamir oracle on the simulated first messages fails only if one of them collides with a prior oracle query (an event of probability at most $Q \cdot 2^{-\ell}$ over the first messages' entropy, for Q oracle queries), Hybrid_2 is within statistical distance $2^{-\nu+1} + r_{\mathsf{kzg}_{\mathsf{PE}}} \cdot 2^{-\nu} + Q \cdot 2^{-\ell}$ of Hybrid_1. Note that from Hybrid_2, the simulation no longer uses the factorization (p_1, p_2), and uses the exponent d only to compute (t_0, s_0) from u_0.

Hybrid_3. In this hybrid, $\mathcal{S}$ samples $u_0 \xleftarrow{\$} \mathbb{QR}_N$ instead of $u_0 \xleftarrow{\$} \mathbb{J}_N$. Since all remaining components of the view of the corrupted client are computed from (N, u_0, d, h) and simulated proofs, an efficient distinguisher between Hybrid_3 and Hybrid_2 immediately yields a distinguisher between a random element of $\mathbb{J}_N$ and a random element of $\mathbb{QR}_N$, contradicting the quadratic residuosity assumption (Assumption 3.13).
Hybrid_3 defines the simulator for the corrupted-client case. $\mathcal{S}$ sends $(\mathsf{init}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{setup}}$ on behalf of the corrupted client and receives h (leaked at initialization) followed by the client output (N, u_0, t_0, s_0, h), where the functionality sampled $(N; p_1, p_2) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$, $d \xleftarrow{\$} [2^{2\ell}]$, $u_0 \xleftarrow{\$} \mathbb{QR}_N$, $t_0 = u_0^d \bmod N$, $s_0 = t_0^d \bmod N$, and $h \xleftarrow{\$} \mathbb{G}$ — exactly the distribution of these values in Hybrid_3. $\mathcal{S}$ computes $(u, t, s) := (u_0^{2N}, t_0^{2N}, s_0^{2N}) \bmod N^2$, assembles $\mathsf{pk}_{\mathsf{PE}} = (N, u_0, u, t_0, t, s_0, s)$, derives g_0, g_1 honestly, programs the oracle so that the NUMS derivation returns h, simulates the three proofs as in Hybrid_2, and sends the resulting $\mathsf{SetupData}$ to the corrupted client. The view of the corrupted client, jointly with the honest server's output $((N, u_0, t_0, s_0, h), (p_1, p_2, d))$ delivered by $\mathcal{F}_{\mathsf{setup}}$, is distributed exactly as in Hybrid_3, which concludes this case.

Corrupted server. $\mathcal{S}$ receives h from $\mathcal{F}_{\mathsf{setup}}$, programs the oracle as above, runs the corrupted server S^* with the emulated random oracle, and receives a message $\mathsf{SetupData}$. If $\mathsf{VOLE.VerifySetup}^{\mathsf{H}}(\mathsf{SetupData}) = 0$ (or no message is sent), $\mathcal{S}$ sends $\perp$ to $\mathcal{F}_{\mathsf{setup}}$, which aborts, matching the real-world abort of the honest client. Otherwise, $\mathcal{S}$ parses (N, u_0, t_0, s_0, h^*) from $\mathsf{SetupData}$ (verification guarantees $h^* = h$, since $\mathsf{VOLE.VerifySetup}$ checks that h^* is reconstructed by the NUMS derivation, which $\mathcal{S}$ programmed to return h), and extracts:

- by forking the corrupted server on the challenge of $\Pi^{\mathrm{FS}}_{\mathsf{par}}$ and applying Lemma 3.26, an exponent d in the slack set of Definition 3.22, i.e., $d \in \pm 2^{2\ell+\nu+1}$ with $t_0 = u_0^d \bmod N$ and $s_0 = u_0^{d^2} \bmod N$ (equivalently $s_0 = t_0^d \bmod N$): the two forked challenge strings differ in at least one of the $r_{\mathsf{kzg_{PE}}}$ repetitions of Protocol 10, and that repetition yields two accepting transcripts with $e \neq e'$, from which $d = z - z'$ is computed directly — with the *same* d across both relations, since Protocol 10 proves all links of the chain with a single shared response;
- by forking on the challenge of $\Pi^{\mathrm{FS}}_{\mathsf{fac}}$ and applying Lemma 3.25, either a factorization $N = p_1 p_2$ with $p_1, p_2 \in \pm 2^{n_{\mathrm{rsa}}/2+\ell+\nu}$ (a witness for a slack instance of Definition 3.21), or a relation $g^\alpha = h \bmod p_0$ between the auxiliary bases of Protocol 9 — the latter occurring with negligible probability under the discrete-logarithm assumption of Theorem C.1.

The rewinding involved in these extractions is permitted in the standalone model, as $\mathcal{S}$ controls the random oracle and the corrupted server's random tape. Quantitatively, by the Generalized Forking Lemma (Lemma 3.8), if the corrupted server makes at most Q oracle queries and outputs an accepting setup with probability $\varepsilon_{\mathsf{acc}}$, each forking extraction succeeds with probability at least $\varepsilon_{\mathsf{acc}}^2/Q - \varepsilon_{\mathsf{acc}} \cdot 2^{-r_{\mathsf{kzg_{PE}}}}$ (for $\Pi^{\mathrm{FS}}_{\mathsf{par}}$, whose challenge space has size $2^{r_{\mathsf{kzg_{PE}}}}$) and $\varepsilon_{\mathsf{acc}}^2/Q - \varepsilon_{\mathsf{acc}} \cdot 2^{-\ell}$ (for $\Pi^{\mathrm{FS}}_{\mathsf{fac}}$, whose challenges range over $\pm 2^\ell$). Moreover, by Lemma 3.24, a false statement $N \notin \mathsf{mod}$ passes the r_{mod}-fold repetition of $\Pi^{\mathrm{FS}}_{\mathsf{mod}}$ with probability at most $Q \cdot 4^{-r_{\mathsf{mod}}}$ (a union bound over the oracle queries), so except with this probability N is a product of exactly two primes p, q with $\gcd(N, \varphi(N)) = 1$ (Definition 3.20). Since $p_1 p_2 = N$ with $1 < |p_1|, |p_2| \leq 2^{n_{\mathrm{rsa}}/2+\ell+\nu} < 2^{n_{\mathrm{rsa}}-1} \leq N$ (so both factors are nontrivial; the inequality uses the parameter constraint $n_{\mathrm{rsa}}/2 > \ell + \nu + 1$ of Section 3), and N has exactly two prime factors, it follows that $\{|p_1|, |p_2|\} = \{p, q\}$: the extracted factors are, up to sign, the prime factors of N, and in particular $|p_1|, |p_2|$ are prime.
$\mathcal{S}$ then sends $(|p_1|, |p_2|, u_0, t_0, s_0, d)$ to $\mathcal{F}_{\mathsf{setup}}$. Except with negligible probability, all checks of the corrupt-S branch of Functionality 1 pass: $|p_1|, |p_2|$ are primes in $\pm 2^{n_{\mathrm{rsa}}/2+\ell+\nu}$; $u_0, t_0, s_0 \in \mathbb{Z}_N^*$ (indeed in $\mathbb{J}_N$, as checked by $\mathsf{VOLE.VerifySetup}$) with $s_0 = t_0^d = u_0^{d^2} \bmod N$; and $d \in \pm 2^{2\ell+\nu+1}$. The functionality thus delivers (N, u_0, t_0, s_0, h) to the honest client. In the real experiment, the honest client outputs the same tuple (N, u_0, t_0, s_0, h) parsed from $\mathsf{SetupData}$ whenever verification passes, and aborts otherwise. Therefore, the joint distribution of the corrupted server's view and the honest client's output is identical in the real and ideal experiments, up to the negligible failure probabilities of the extractors and of the soundness of $\Pi^{\mathrm{FS}}_{\mathsf{mod}}$. This concludes the proof. □

Remark C.2. The theorem above, combined with modular composition (Remark 3.6) and Theorem 6.3, yields security of the composed protocol $\Pi^{\Pi_{\mathsf{setup}}}_{\mathsf{VOLE}}$; see Remark 6.4.

D Description of the Standard Zero-Knowledge Protocols

This appendix describes the three standard zero-knowledge protocols used by the setup: the well-formed modulus proof Π_{mod} and the small factors proof Π_{fac}, both taken from [ABCGJM26], and

the KZG parameters proof $\Pi_{\mathsf{kzg}_{\mathsf{PE}}}$. The associated promise problems, notation, and security claims are given in Section 3.6.

Protocol Well-Formed Modulus Proof

Common Input. $N \in \mathbb{N}$ 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 $y_0 = (-1)^a \cdot 2^{Nb} \cdot y \in \mathbb{QR}_N$,
   - 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

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

Protocol 8: Well-formed modulus proof Π_{mod} [ABCGJM26].

Protocol Small Factors Proof

ZK Parameter: ν

Common Input. $N \in [2^{n_{\text{rsa}}-1}, 2^{n_{\text{rsa}}})$, soundness parameter ℓ, and aux $\in \{0,1\}^*$. A strong prime p_0 such that $p_0 > 2^{n_{\text{rsa}}+2\ell+2\nu+3}$, and $g, h \in \mathbb{QR}_{p_0}$.

Secret Input. (p, q) such that $N = pq$.

Preliminary Step. P sets $\widetilde{P} = g^p h^r \bmod p_0$ and $\widetilde{Q} = g^q h^s \bmod p_0$ for random $r, s \xleftarrow{\$} \mathbb{Z}_{(p_0-1)/2}$.

Operations.

1. P sends $(p_0, \widetilde{P}, \widetilde{Q})$ and $(\widetilde{A}, \widetilde{B}, \widetilde{C}) = (g^\alpha \cdot h^\rho, g^\beta \cdot h^\sigma, \widetilde{Q}^\alpha \cdot h^\mu) \bmod p_0$ for
$$(\alpha, \beta, \rho, \sigma, \mu) \xleftarrow{\$} (\pm 2^{n_{\text{rsa}}/2+\ell+\nu})^2 \times \mathbb{Z}^3_{(p_0-1)/2}.$$
2. V sends $e \xleftarrow{\$} \pm 2^\ell$.

3. P returns $(z_1, z_2, \lambda_1, \lambda_2, w)$ such that

$$\begin{cases} z_1 = \alpha + ep \\ z_2 = \beta + eq \\ \lambda_1 = \rho + er \bmod (p_0 - 1)/2 \\ \lambda_2 = \sigma + es \bmod (p_0 - 1)/2 \\ w = \mu + e \cdot (-sp) \bmod (p_0 - 1)/2 \end{cases}$$

Verifier's Output. Reconstruct g, h from (p_0, aux) and accept if $z_1, z_2 \in \pm 2^{n_{\text{rsa}}/2+\ell+\nu}$ and

$$\begin{cases} g^{z_1} h^{\lambda_1} = \widetilde{A} \cdot \widetilde{P}^e \bmod p_0 \\ g^{z_2} h^{\lambda_2} = \widetilde{B} \cdot \widetilde{Q}^e \bmod p_0 \\ \widetilde{Q}^{z_1} h^{w} = \widetilde{C} \cdot g^{Ne} \bmod p_0 \end{cases}$$

Protocol 9: Small factors proof Π_{fac} [ABCGJM26].

Protocol KZG Params Proof

ZK Parameter: ν

Common Input. $N \in \mathbb{N}$ for $(u_{-k+1+i})_{i=1}^{2k-1} \in \mathbb{Z}_N^*$ s.t. $u_{-k+1+i} = u_{-k+1}^{d^i} \bmod N$ and $d \in [2^{2\ell}]$.

Operations.

1. P sends $A_{-k+1+j} = u_{-k+j}^{\alpha} \bmod N$ for $\alpha \xleftarrow{\$} \pm 2^{2\ell+\nu}$ and $j \in \{1, \ldots, 2k-1\}$.
2. V replies with $e \xleftarrow{\$} \{0, 1\}$
3. P sends $z = \alpha + e \cdot d$.

Verifier's Output. Accept if $u_{-k+j}^{z} = A_{-k+j+1} \cdot u_{-k+1+j}^{e} \bmod N$, and $z \in \pm 2^{2\ell+\nu}$.

Protocol 10: Σ-protocol for proving relaxed correctness of the KZG parameters (s, t, u).

E On the Equivalence of Power-SEI and Power-SEDL

In this appendix we prove the equivalence between the k-power small-exponent indistinguishability assumption PSEI_k (Assumption 3.11) and the k-power small-exponent discrete logarithm assumption PSEDL_k (Assumption 3.10), generalizing the $k = 1$ equivalence of [ABCGJM26] (Appendix A), which follows the blueprint of Gennaro [Gen00] and Koshiba–Kurosawa [KK04].
Beyond handling $k \geq 2$, the proof below tightens the $k = 1$ argument of [ABCGJM26]: it works over the exact short domain $\pm 2^{2\ell}$ of Assumptions 3.10 and 3.11 (removing the restriction to "special bounds" of the form $\lfloor |G|/2^i \rfloor$).

E.1 Setting and statements

Throughout this appendix, $(N; p, q) \xleftarrow{\$} \mathsf{SampleRSA}(1^\ell)$ denotes a strong RSA modulus as in Section 3.3: $N = pq \in [2^{n_{\mathrm{rsa}}-1}, 2^{n_{\mathrm{rsa}}})$ with $p = 2p'+1$, $q = 2q'+1$ and p', q' prime. We assume $n_{\mathrm{rsa}} \geq 4\ell$ (satisfied by any standard instantiation, e.g. $n_{\mathrm{rsa}} = 2048$, $\ell = 128$). We write $\mathsf{QR}_N \leq \mathbb{Z}_N^*$ for the group of quadratic residues modulo N; it is cyclic of *odd* order

$$M := |\mathsf{QR}_N| = \varphi(N)/4 = p'q',$$

and a uniform $t \xleftarrow{\$} \mathsf{QR}_N$ generates QR_N except with probability at most $1/p' + 1/q' \leq 2^{-n_{\mathrm{rsa}}/2+3}$. Since M is odd, 2 is invertible modulo M and we write $\theta := 2^{-1} \bmod M$; the quantity θ is used *only in the analysis* and is never computed by any reduction. We set

$$T_0 := 2^{2\ell+1} + 1 = \left|\pm 2^{2\ell}\right|.$$

For $s \in \mathsf{QR}_N$ and $w \in \mathbb{Z}$ we write

$$\mathsf{pow}_s^k(w) := \left(s,\ [s^w]_N,\ [s^{w^2}]_N,\ \ldots,\ [s^{w^k}]_N\right)$$

for the *k-power tuple* of w in base s. Note that $\mathsf{pow}_s^k(w)$ depends only on $w \bmod \mathsf{ord}(s)$, that the challenge of PSEDL_k (Assumption 3.10) is exactly $(N, \mathsf{pow}_t^k(x))$ for $t \xleftarrow{\$} \mathsf{QR}_N$ and $x \xleftarrow{\$} \pm 2^{2\ell}$, and that the challenge of PSEI_k (Assumption 3.11) is $(N, \mathsf{pow}_t^k(\alpha x + (1-\alpha)y))$ with additionally $y \xleftarrow{\$} \pm N$ and $\alpha \xleftarrow{\$} \{0,1\}$. Throughout, we freely switch between the α-guessing formulation above and the equivalent distinguishing formulation: an algorithm A guessing α with probability $\frac{1}{2} + \frac{\varepsilon}{2}$ yields $|\Pr[\mathsf{A}(\mathsf{short}) = 1] - \Pr[\mathsf{A}(\mathsf{rand}) = 1]| = \varepsilon$ for the two conditional distributions, and conversely. The results of this appendix are the following (the factoring assumption FAC is Assumption 3.9; recall that it is implied by the DCR assumption, which this paper assumes anyway).

Theorem E.1 (search-to-decision)**.** *Let $k = k(\ell)$ be polynomially bounded. If PSEDL_k holds, then PSEI_k holds.*

Theorem E.2 (decision-to-search)**.** *Let $k = k(\ell)$ be polynomially bounded. If PSEI_k and FAC hold, then PSEDL_k holds.*

Corollary E.3. *PSEDL_k implies PSEI_k; under the factoring assumption, the two assumptions are equivalent, for every polynomially bounded k.*

Remark E.4 (general exponent domains). We use the domains $\pm 2^{2\ell}$ and $\pm N$ in Assumptions 3.10 and 3.11 for simplicity, but Theorems E.1 and E.2 holds verbatim for the variants where the short exponent is uniform over an arbitrary publicly known integer interval I with $2^\ell \leq |I| \leq 2^{n_{\mathrm{rsa}}-2\ell}$: the reduction first shifts the challenge to $[0, |I|)$ (Lemma E.6), and the hybrid chain below uses $T_0 := |I|$ with $c := \lceil \log_2(2^{2\ell+3} N / T_0) \rceil \leq n_{\mathrm{rsa}} + \ell + 3$ levels. Similarly, the full-range exponent can be drawn from any distribution whose reduction modulo $\varphi(N)/4$ is statistically close to uniform, e.g. $y \xleftarrow{\$} \mathbb{Z}_{\varphi(N)/4}$. In particular, this covers the exponent domain $[2^{2\ell}]$ used by $\mathcal{F}_{\mathsf{setup}}$ in Section 5.

E.2 Preliminary lemmas

We first record standard statistical facts.

Lemma E.5. *Let $M \geq 2$, $R \geq 1$ be integers.*

1. *If $Z \xleftarrow{\$} [0, R)$, then $\mathrm{SD}(Z \bmod M,\ U(\mathbb{Z}_M)) \leq \min\{r,\ M/4\}/R$, where $r := R \bmod M$. In particular the bound is at most M/R.*

2. *If $y \xleftarrow{\$} \pm N$ for a strong RSA modulus $N = (2p'+1)(2q'+1)$ and $M = p'q'$, then $\mathrm{SD}(y \bmod M,\ U(\mathbb{Z}_M)) \leq (p+q)/N \leq 2^{-n_{\mathrm{rsa}}/2+3}$.*

3. *If s generates QR_N and $\sigma \xleftarrow{\$} [0, N \cdot 2^{\ell})$, then $\mathrm{SD}(s^{\sigma},\ U(\mathsf{QR}_N)) \leq 2^{-\ell-1}$.*

Proof. (1) Write $R = aM + r$. Each residue receives either a or $a+1$ preimages, so the statistical distance equals $r(M-r)/(RM) \leq \min\{r, M/4\}/R$. (2) The range $\pm N$ has $R = 2N+1 = 8M + (2(p+q)-1)$ elements, so $r = 2(p+q)-1$ and the claim follows from (1) after translating the range to $[0, R)$ (a shift does not affect the distance to uniform). (3) The exponent σ mod M is within $M/(N2^{\ell}) \leq 2^{-\ell-1}$ of uniform by (1), and $z \mapsto s^z$ is a bijection from $\mathbb{Z}_M$ to QR_N. □

We now introduce some "algebraic toolbox": k-power tuples support the following operations, each computable in time $\mathrm{poly}(k, n_{\mathrm{rsa}}, \ell, \log|a|)$ from the tuple alone (without knowledge of the exponent, of M, or of the factorization of N). Below, $Z = \mathsf{pow}^k_s(w) = (z_0 := s, z_1, \ldots, z_k)$.

Lemma E.6 (tuple operations)**.** *1.* Shift. *For any $a \in \mathbb{Z}$:* $\mathsf{shift}_a(Z) := \left(s,\ \left(\prod_{n=0}^{m} z_n^{\binom{m}{n} a^{m-n}}\right)_{1 \leq m \leq k}\right) = \mathsf{pow}^k_s(w+a)$.

2. Scaling. *For any $e \in \mathbb{N}$:* $\mathsf{scale}_e(Z) := \left(s, \left(z_m^{e^m}\right)_m\right) = \mathsf{pow}^k_s(ew)$.

3. Rebasing. *For any $\gamma \in \mathbb{N}$:* $\mathsf{rebase}_\gamma(Z) := \left(s^\gamma, \left(z_m^{\gamma}\right)_m\right) = \mathsf{pow}^k_{s^\gamma}(w)$.

4. Halving. *If $w = 2v$ is even:* $\mathsf{halve}(Z) := \left(s^{2^k}, \left(z_m^{2^{k-m}}\right)_m\right) = \mathsf{pow}^k_{s^{2^k}}(v)$.

5. Division presentation. *For any $a \in \mathbb{N}$ and $\gamma \in \mathbb{N}$:* $\Psi_{a,\gamma}(Z) := \left(s^{2^{ak}\gamma},\ \left(z_m^{2^{a(k-m)}\gamma}\right)_m\right) = \mathsf{pow}^k_{s^{2^{ak}\gamma}}(\theta^a w \bmod M)$.

Proof. (1) follows from the binomial identity $(w+a)^m = \sum_{n \leq m} \binom{m}{n} a^{m-n} w^n$ in the exponent; (2) from $(ew)^m = e^m w^m$; (3) from $(s^\gamma)^{w^m} = (s^{w^m})^\gamma$. For (5), write $g := s^{2^{ak}\gamma}$; then

$$g^{(\theta^a w)^m} = s^{2^{ak}\,\gamma\,\theta^{am}\,w^m} = s^{2^{a(k-m)}\,\gamma\,w^m} = z_m^{2^{a(k-m)}\gamma},$$

using $2^a\theta^a \equiv 1 \pmod M$ and the fact that $\mathsf{pow}^k_{\cdot}(\cdot)$ depends on its exponent only modulo the group order. (4) follows from (5) by plugging $a = 1$, $\gamma = 1$, together with $\theta \cdot 2v \equiv v \pmod M$. All exponents involved have $\mathrm{poly}(k, n_{\mathrm{rsa}}, \ell, \log|a|)$ bits. □

Note that $\Psi_{a,\gamma}$ is defined on *every* tuple and divides the exponent by 2^a *modulo* M; when $2^a \mid w$ over the integers, this coincides with integer division.

E.3 Proof of Theorem E.1 (search-to-decision)

Fix a PPTM A against PSEI_k, and let $\varepsilon = \varepsilon(\ell)$ denote its distinguishing advantage. Fix a modulus N in the support of $\mathsf{SampleRSA}(1^\ell)$. All distributions in this subsection are conditioned on this N. Define the two *endpoint* distributions (after the affine renormalization $u := x + 2^{2\ell}$, which by Lemma E.6(1) is freely interconvertible with the original challenge):

$$\mathsf{S}_N := \mathsf{pow}_t^k(u),\ t \xleftarrow{\$} \mathsf{QR}_N,\ u \xleftarrow{\$} [0, T_0) \qquad \text{and}$$
$$\mathsf{U}_N := \mathsf{pow}_t^k(y),\ t \xleftarrow{\$} \mathsf{QR}_N,\ y \xleftarrow{\$} \pm N.$$

The hybrid chain. Let $c := n_{\mathrm{rsa}}$ and, for $0 \le j \le c$, let $W_j := T_0 \cdot 2^{c-j}$ and define

$$C_j := \mathsf{pow}_s^k(\,\theta^{\,c-j} x \bmod M\,), \qquad s \xleftarrow{\$} \mathsf{QR}_N,\ x \xleftarrow{\$} [0, W_j).$$

The chain has the following properties.

1. *Bottom endpoint (exact).* $C_c \equiv \mathsf{S}_N$, since $W_c = T_0$ and $\theta^0 = 1$.
2. *Top endpoint (statistical).* $\mathrm{SD}(C_0, \mathsf{U}_N) \le \eta_0 := 2^{-2\ell-3} + 2^{-n_{\mathrm{rsa}}/2+3}$. Indeed, $W_0 = T_0 2^{n_{\mathrm{rsa}}} \ge M \cdot 2^{2\ell+3}$, so by Lemma E.5(1) the exponent x mod M is $2^{-2\ell-3}$-close to uniform, multiplication by the unit θ^c preserves uniformity, and by Lemma E.5(2) the exponent of U_N is $2^{-n_{\mathrm{rsa}}/2+3}$-close to uniform (a k-power tuple depends only on its exponent modulo $\mathsf{ord}(s) \mid M$, and modulo $\mathsf{ord}(s)$, both exponent distributions above are at least as close to uniform, by projection).
3. *Exact parity splitting.* For $0 \le j \le c-1$, the width W_j is even and conditioning C_j on an even exponent source $x = 2v$ yields *exactly* C_{j+1}: indeed $v \xleftarrow{\$} [0, W_j/2) = [0, W_{j+1})$ and $\theta^{c-j} \cdot 2v = \theta^{c-j-1} v \bmod M$.
4. *Samplability.* Given N, a C_j-sample together with its exponent source x is samplable in $\mathrm{poly}(k, n_{\mathrm{rsa}}, \ell)$ time: pick $\rho \xleftarrow{\$} \mathbb{Z}_N^*$, $h := \rho^2$, $x \xleftarrow{\$} [0, W_j)$ and output $\Psi_{c-j,1}(\mathsf{pow}_h^k(x))$, where $\mathsf{pow}_h^k(x)$ is computed directly from (h, x). Since squaring is a permutation of QR_N, the resulting base $h^{2^{(c-j)k}}$ is uniform, and by Lemma E.6(5) the tuple is exactly $\mathsf{pow}^k_{h^{2^{(c-j)k}}}(\theta^{c-j} x)$. The same holds conditioned on x even or odd.

Write $b_j := \Pr[\mathsf{A}(N, C_j) = 1]$, and let $\varepsilon_N := b_c - \Pr[\mathsf{A}(N, \mathsf{U}_N) = 1]$ denote the (signed) conditional advantage of A on modulus N; without loss of generality orient A so that $\mathrm{E}_N[\varepsilon_N] = \varepsilon$. By the endpoint properties, $\sum_{j=0}^{c-1} |b_{j+1} - b_j| \ge |b_c - b_0| \ge \varepsilon_N - \eta_0$. By exact parity splitting, $b_j = \frac{1}{2} b_{j+1} + \frac{1}{2} o_j$ where $o_j := \Pr[\mathsf{A}(N, C_j \mid x \text{ odd}) = 1]$, whence

$$|b_{j+1} - o_j| = 2|b_{j+1} - b_j|. \tag{2}$$

The extraction lemma. The heart of the proof is the following per-modulus statement.

Lemma E.7 (amplified bit extraction)**.** *There is an oracle machine B^{A} with the following guarantee. Let N be in the support of $\mathsf{SampleRSA}(1^\ell)$ and let $\delta \in (2^{-\ell/2}, 1)$ be given to B as a parameter. If*

$$\varepsilon_N \ge 4c\,\delta + \eta_0,$$

then on input a PSEDL_k*-challenge* $(N, t, t_1, \ldots, t_k)$ *with modulus* N, B^{A} *outputs* $\hat{x}$ *with* $t^{\hat{x}} = t_1 \bmod N$ *with probability at least* $1 - 2^{-\ell+1}$*, and runs in time* $O\big((c + 2^{b^*}\ell) \cdot n \cdot (T_{\mathsf{A}} + \mathrm{poly}(k, n_{\mathrm{rsa}}, \ell))\big)$ *where* $b^* := \lceil \log_2(16/\delta) \rceil$*,* $n := \lceil 8\ell/\delta^2 \rceil$ *and* T_{A} *is the running time of* A*.*

Proof. We describe B in stages. Throughout, "negligible base error" refers to the event that t does not generate QR_N, which happens with probability at most $2^{-n_{\mathrm{rsa}}/2+3}$ and is absorbed in the final failure bound. We condition on t generating QR_N from now on.

Stage 1: level and orientation selection. Using the samplability of the C_j (and of their odd-conditioned versions), B estimates every b_j, $0 \le j \le c$, to within additive error $\delta/2$ with confidence $1 - 2^{-2\ell}/(4c)$ each, by a Chernoff–Hoeffding bound using $O((\ell + \log c)/\delta^2)$ samples per level. It selects $\hat{\jmath} := \arg\max_j \left|\hat{b}_{j+1} - \hat{b}_j\right|$ and the orientation bit $\beta^* := [\hat{b}_{\hat{\jmath}+1} > \hat{o}_{\hat{\jmath}}]$, where $\hat{o}_{\hat{\jmath}} := 2\hat{b}_{\hat{\jmath}} - \hat{b}_{\hat{\jmath}+1}$. Since $\max_j |b_{j+1} - b_j| \ge (\varepsilon_N - \eta_0)/c \ge 4\delta$, on the (overwhelming) event that all estimates are $\delta/2$-accurate we get

$$|b_{\hat{\jmath}+1} - b_{\hat{\jmath}}| \ge 4\delta - \delta = 3\delta, \qquad \text{hence by Equation (2)} \qquad |b_{\hat{\jmath}+1} - o_{\hat{\jmath}}| \ge 6\delta, \tag{3}$$

and the orientation β^* agrees with the sign of $b_{\hat{\jmath}+1} - o_{\hat{\jmath}}$ (the estimation error of $\hat{b}_{\hat{\jmath}+1} - \hat{o}_{\hat{\jmath}} = 2(\hat{b}_{\hat{\jmath}+1} - \hat{b}_{\hat{\jmath}})$ is at most $2\delta < 6\delta$). Write $W := W_{\hat{\jmath}} \ge 2T_0$ for the selected width.

Stage 2: the elementary predictor. Define the predictor $\mathsf{P}(Z)$, taking a k-power tuple Z over a base generating QR_N and one fresh coin string: sample $\sigma \xleftarrow{\$} [0, N2^{\ell})$, compute $\Phi := \Psi_{c-\hat{\jmath},\sigma}(Z)$ and output even if $\mathsf{A}(N, \Phi) = \beta^*$, else odd. By Lemma E.6(5) and Lemma E.5(3), if $Z = \mathsf{pow}^k_s(w)$ with $\langle s \rangle = \mathsf{QR}_N$ and any fixed $w \in [0, W)$, then Φ is within statistical distance $2^{-\ell-1}$ of a $C_{\hat{\jmath}}$-sample conditioned on exponent source $x = w$. Consequently, if $w \xleftarrow{\$} [0, W)$, then

$$\Pr\big[\mathsf{P}(\mathsf{pow}^k_s(w)) = \mathsf{parity}(w)\big] \ge \tfrac{1}{2} + \tfrac{1}{2}|b_{\hat{\jmath}+1} - o_{\hat{\jmath}}| - 2^{-\ell-1} \ge \tfrac{1}{2} + 3\delta - 2^{-\ell-1},$$

using that the even branch of $C_{\hat{\jmath}}$ is exactly $C_{\hat{\jmath}+1}$ and the odd branch has acceptance probability $o_{\hat{\jmath}}$. Moreover, this bound holds *for every fixed conditional distribution of* w *given its parity that is* ζ*-close to uniform on its parity class*, at an additive cost of ζ: the acceptance probabilities on each branch are changed by at most ζ.

Stage 3: guess, extract, verify. B normalizes the challenge into $Z_0 := \mathsf{shift}_{2^{2\ell}}(t, t_1, \ldots, t_k) = \mathsf{pow}^k_t(u)$ with $u := x^* + 2^{2\ell} \in [0, T_0)$, where x^* is the challenge exponent. For every guess $g \in [0, 2^{b^*})$ of the b^* low-order bits of u, it runs the following extraction, and finally verifies each obtained candidate.

Stripping. Compute $Z_g := \mathsf{halve}^{(b^*)}(\mathsf{shift}_{-g}(Z_0))$, i.e., shift by $-g$ and halve b^* times. If g equals $u \bmod 2^{b^*}$, then $Z_g = \mathsf{pow}^k_{s_0}(u^{(0)})$ with $s_0 := t^{2^{kb^*}}$ (which also generates QR_N) and $u^{(0)} := (u - g)/2^{b^*} \in [0, T')$, $T' := \lceil T_0/2^{b^*} \rceil$. (For wrong guesses the extraction still runs; its output is eliminated by the final verification.)

Bit-by-bit extraction. For $i = 0, 1, \ldots, \lceil \log_2 T' \rceil - 1$, given the invariant state $Z^{(i)} = \mathsf{pow}^k_{s_i}(u^{(i)})$ with $u^{(i)} < T_i := \lceil T'/2^i \rceil$ and $\langle s_i \rangle = \mathsf{QR}_N$, extract $\beta_i := \mathsf{lsb}(u^{(i)})$ by a majority vote over n independent trials of:

$$\rho \xleftarrow{\$} \{0,1\}, \quad r \xleftarrow{\$} [0, R_v),\ R_v := \lfloor (W - T_i - 1)/2 \rfloor, \quad v := \mathsf{P}\big(\mathsf{shift}_{\rho+2r}(Z^{(i)})\big),$$

recording the vote $v \oplus \rho$ (with $\mathsf{even} \equiv 0$). Then update $Z^{(i+1)} := \mathsf{halve}(\mathsf{shift}_{-\beta_i}(Z^{(i)}))$ and $s_{i+1} := s_i^{2^k}$. After the loop, $u^{(\cdot)} = 0$ and the candidate is $\hat{x} := 2^{b^*} \sum_i \beta_i 2^i + g - 2^{2\ell}$; *verify* $t^{\hat{x}} = t_1 \bmod N$ and output the first verified candidate.

Analysis of one vote. Let $w := u^{(i)} + \rho + 2r$, so that $\mathsf{shift}_{\rho+2r}(Z^{(i)}) = \mathsf{pow}^k_{s_i}(w)$ and $\mathsf{parity}(w) = \beta_i \oplus \rho$. Conditioned on $(u^{(i)}, \rho)$, the value w is uniform over $R_v \geq (W - T_0 - 1)/2 \geq W/4$ consecutive integers of fixed parity inside $[0, W)$, hence its conditional distribution is within $\zeta := (T_i + 3)/W \leq (T' + 3)/(2T_0) \leq 2^{-b^*} + 2^{-2\ell} \leq \delta/16 + 2^{-2\ell}$ of uniform on its parity class of $[0, W)$. By Stage 2, each vote equals β_i with probability at least $\frac{1}{2} + 3\delta - 2^{-\ell-1} - \zeta \geq \frac{1}{2} + 2\delta$, independently across trials (fresh ρ, r, σ and coins of A per trial, for fixed $Z^{(i)}$).
Error probability. Since each vote is correct with probability at least $\frac{1}{2} + 2\delta$ and votes are independent, by a Hoeffding bound one majority vote errs with probability at most $e^{-2n(2\delta)^2} = e^{-8n\delta^2} \leq e^{-64\ell} \leq 2^{-4\ell}$ with $n = \lceil 8\ell/\delta^2 \rceil$. A union bound over at most $(2\ell + 2)$ bits and $2^{b^*} \leq 32 \cdot 2^{\ell/2}$ guesses, together with the estimation failures of Stage 1 and the non-generating-t event, bounds the total failure probability by

$$(2\ell + 2) \cdot 2^{b^*} \cdot 2^{-4\ell} + 2^{-2\ell} + 2^{-n_{\mathrm{rsa}}/2+3} \leq 2^{-\ell+1}.$$

In case of success (for the correct guess g), every β_i is correct, so $\hat{x} \equiv x^* \pmod{\mathsf{ord}(t)}$ (in fact $\hat{x} = x^*$ over $\mathbb{Z}$) and verification passes. □

From Lemma E.7 to Theorem E.1. Suppose towards contradiction that some PPTM A has advantage $\varepsilon(\ell) \geq 1/P(\ell)$ against PSEI_k for a polynomial P and infinitely many ℓ. For such ℓ, set $\delta := (1/(2P(\ell)) - \eta_0)/(4c)$, which satisfies $\delta > 2^{-\ell/2}$ for all large ℓ. Since $\mathrm{E}_N[\varepsilon_N] \geq 1/P(\ell)$ and $\varepsilon_N \leq 1$, we have $\Pr_N[\varepsilon_N \geq 1/(2P(\ell))] \geq 1/(2P(\ell))$. For every such N the hypothesis of Lemma E.7 holds, so B^{A} (with parameter δ) solves PSEDL_k with probability at least

$$\frac{1}{2P(\ell)} \cdot \left(1 - 2^{-\ell+1}\right) \geq \frac{1}{4P(\ell)}$$

over all coins, in polynomial time, which contradicts PSEDL_k. □

E.4 Proof of Theorem E.2 (decision-to-search)

Let S_0 be a PPTM solving PSEDL_k with probability $\varepsilon = \varepsilon(\ell)$: on input $(N, \mathsf{pow}^k_t(x))$ with $x \xleftarrow{\$} \pm 2^{2\ell}$ it outputs $\hat{x}$ with $t^{\hat{x}} = t_1 \bmod N$ with probability ε. Split the success event according to whether the returned representative is *short*: let $\varepsilon_s := \Pr[t^{\hat{x}} = t_1 \wedge \hat{x} \in \pm 2^{2\ell}]$ and $\varepsilon_h := \varepsilon - \varepsilon_s$ (valid but non-short representative). We show that ε_s bounds an advantage against PSEI_k while ε_h bounds (up to constants) a success probability against FAC; hence if both assumptions hold, ε is negligible.

Short representatives break PSEI_k. Consider the distinguisher $\mathsf{D}(N, T)$: run $\hat{x} \xleftarrow{\$} \mathsf{S}_0(N, T)$; output 1 iff $T_1 = T_0^{\hat{x}} \bmod N$ and $\hat{x} \in \pm 2^{2\ell}$ (writing $T = (T_0, T_1, \ldots, T_k)$). On the $\alpha = 1$ branch, D outputs 1 with probability ε_s. On the $\alpha = 0$ branch, the exponent $y \bmod \mathsf{ord}(t)$ is $2^{-n_{\mathrm{rsa}}/2+3}$-close to uniform (Lemma E.5(2)); the set of elements of $\langle t \rangle$ admitting a representative in $\pm 2^{2\ell}$ has density at most $T_0/\mathsf{ord}(t) \leq 2^{2\ell+2}/M + 2^{-n_{\mathrm{rsa}}/2+3} \leq 2^{-n_{\mathrm{rsa}}+2\ell+5}$ (again absorbing the event that t generates a small subgroup). Hence D's advantage is at least $\varepsilon_s - 2^{-n_{\mathrm{rsa}}/2+4}$.

Long representatives break FAC. Consider the algorithm $\mathcal{R}(N)$: sample $\rho \xleftarrow{\$} \mathbb{Z}_N^*$, $t := \rho^2 \bmod N$, $x \xleftarrow{\$} \pm 2^{2\ell}$; compute $\mathsf{pow}^k_t(x)$ and run $\hat{x} \xleftarrow{\$} \mathsf{S}_0(N, \mathsf{pow}^k_t(x))$. If $t^{\hat{x}} = t^x$ and $\hat{x} \notin \pm 2^{2\ell}$, then $K := \hat{x} - x \neq 0$ and $\mathsf{ord}(t) \mid K$. Except with probability $2^{-n_{\mathrm{rsa}}/2+3}$, t generates QR_N, so K is a nonzero multiple of $M = p'q'$ and $2K$ is a nonzero multiple of $\lambda(N) = 2p'q'$. Given such a multiple,

N can be factored in expected polynomial time by the standard Miller-style procedure [Mil76] (see also Fact 4 in [CPP17] for the statement in this exact form): repeatedly pick a random base $a \xleftarrow{\$} \mathbb{Z}_N^*$, write $2K = 2^e K'$ with K' odd, and inspect the sequence $a^{K'}, a^{2K'}, \ldots$ for a nontrivial square root of 1; each iteration succeeds with constant probability. Overall $\mathcal{R}$ factors N with probability at least $\varepsilon_h/2 - 2^{-n_{\mathrm{rsa}}/2+3}$ after $O(1)$ iterations, and with probability arbitrarily close to $\varepsilon_h - 2^{-n_{\mathrm{rsa}}/2+3}$ after polynomially many.

Combining the two paragraphs: $\varepsilon = \varepsilon_s + \varepsilon_h$ is bounded by the sum of an advantage against PSEI_k and a success probability against FAC, up to additive $2^{-n_{\mathrm{rsa}}/2+5}$. Under the two assumptions, ε is negligible. □