Two-Round Stateless Deterministic Two-Party Schnorr Signatures From Pseudorandom Correlation Functions

Yashvanth Kondi, Claudio Orlandi, and Lawrence Roy

Aarhus University, Aarhus, Denmark
yash@ykondi.net, orlandi@cs.au.dk, ldr709@gmail.com

Abstract. Schnorr signatures are a popular choice due to their simplicity, provable security, and linear structure that enables relatively easy threshold signing protocols. The deterministic variant of Schnorr (where the nonce is derived in a stateless manner using a PRF from the message and a long term secret) is widely used in practice since it mitigates the threats of a faulty or poor randomness generator (which in Schnorr leads to catastrophic breaches of security). Unfortunately, threshold protocols for the deterministic variant of Schnorr have so far been quite inefficient, as they make non black-box use of the PRF involved in the nonce generation.

In this paper, we present the first two-party threshold protocol for Schnorr signatures, where signing is stateless and deterministic, and only makes black-box use of the underlying cryptographic algorithms. We present a protocol from general assumptions which achieves covert security, and a protocol that achieves full active security under standard factoring-like assumptions. Our protocols make crucial use of recent advances within the field of *pseudorandom correlation functions (PCFs)*.

As an additional benefit, only two-rounds are needed to perform distributed signing in our protocol, connecting our work to a recent line of research on the trade-offs between round complexity and cryptographic assumptions for threshold Schnorr signatures.

Table of Contents

1 Introduction

Schnorr [Sch91] provides a simple method to leverage the hardness of computing discrete logarithms in some group of known prime order, to construct a provably unforgable signature scheme [PS96]. Although the adoption of Schnorr signatures was initially hampered by a patent, following the patent's expiration in 2008 its footprint in real-world use has been rapidly growing. For instance EdDSA [BDL+11] (a carefully parameterized instantion of Schnorr) already enjoys wide support in several standard libraries, and even more recently Schnorr signatures have seen interest from the Bitcoin community [WNR]. Along with their increasing deployment, interest in mechanisms for *key protection* of Schnorr signatures has grown in the recent past. One common method of mitigating single points of failure in the storage and use of such keys is via *threshold* signatures [Des88]. Informally, a threshold signature allows its key to be secret shared amongst multiple devices, so that a qualified quorum of these devices must collaborate in order to perform signing operations with the key.

The linear nature of Schnorr's signature scheme is known to be conducive to simple distributed signing protocols, with several classic [SS01, NKDM03, GJKR07], and recent [MPSW19, KG20, NRS21, AB21] constructions. At a high level, a Schnorr signature under public key $\mathsf{pk} = \mathsf{sk} \cdot G$ is of the form $(R = r \cdot G, s = \mathsf{sk} \cdot e + r)$, where $e = H(R, pk, m)$, and so a multiparty signing protocol consists of sampling the nonce R with standard distributed key generation techniques [Ped91] and computing s by taking a linear combination the shares of sk and r.

Deterministic Nonce Derivation. Schnorr's original proposal required the random sampling of r, and a line of works on secret key recovery from nonce bias [HS01, ANT+20, MH20] has shown that even a slight noticeable deviation from the uniform distribution can induce catastrophic failure. Taken in combination with empirical observations on the ubiquity of randomness failures in practice [Hen22], this sensitivity to nonce bias makes the nonce sampling step a significant potential attack surface for Schnorr signatures. In order to mitigate this threat, early proposals [Bar97, Wig97] called for *deterministic* nonce generation.

Statelessness. One may consider deriving nonces by maintaining a stateful pseudorandom number generator, for example. However such an approach then becomes vulnerable to state reuse, which implies nonce reuse in this context. Reliably maintaining state can be surprisingly non-trivial in a variety of scenarios, and is studied under the umbrella of *state continuity* [PLD+11, SP16, MAK+17] in the systems literature. One such scenario that is particularly relevant to modern cloud deployment conditions is that of incorrectly instantiating Virtual Machines [EZJ+14, KASN15]—stale state is not typically detectable within the context provided to cryptographic APIs.

This problem could in principle be addressed by general solutions for state continuity based on trusted hardware [PLD+11, SP16]. However, such solutions come with extra hardware costs and qualitative disadvantages due to heuristic hardware-based trust assumptions, in addition to suffering a high latency for simple operations (over 60ms to increment an SGX Trusted Monotonic Counter [BCLK17]). We refer the reader to Garillot et al. [GKMN21] for a more detailed discussion.

Modern instantiations of Schnorr's scheme such as EdDSA are therefore both *deterministic*, and *stateless*—nonces are derived by applying a hash function (or PRF) on the message being signed, along with a long-term secret. In particular, given a PRF $\mathsf{F} : \{0,1\}^* \mapsto \mathbb{Z}_q$ the signer samples a

PRF seed $\mathsf{sd} \leftarrow \{0,1\}^\kappa$ during key generation, and computes $r \leftarrow \mathsf{F}_{\mathsf{sd}}(m)$ to use in signing the message. Our goal in this work is to translate this template for stateless derandomization to the distributed setting, which is known to be a challenging problem [MPSW19]. We focus on the two-party setting in this work, as it is the base case for the challenging dishonest majority setting, and is already sufficient for a number of useful applications including cryptocurrency wallets, two-factor authentication, etc. As one indicator of real-world interest in solutions to this problem, the recent draft NIST Internal Report on Threshold EdDSA/Schnorr signatures [BD22] and subsequent call for multiparty threshold schemes considers stateless deterministic signing as a potentially desirable mode of operation.

1.1 Distributed Schnorr Signing with Stateless Determinism

We first explain the obstacle to translating the benefits of the simple stateless deterministic nonce derivation technique described above to the distributed signing setting, and then we discuss existing approaches to the problem.

The Obstacle. Consider a setting in which parties P_0, P_1 have signing key shares $\mathsf{sk}_0, \mathsf{sk}_1$, and wish to sign a message m under their joint signing key $\mathsf{sk} = \mathsf{sk}_0 + \mathsf{sk}_1$. The protocol of Nicolosi et al. [NKDM03] roughly proceeds as follows: each P_i samples $r_i \leftarrow \mathbb{Z}_q$ and computes $R_i = r_i \cdot G$, and P_0 first sends a commitment $C = \mathsf{Com}(R_0)$ to P_1. Upon receiving C, party P_1 sends R_1 to P_0, who then establishes the signing nonce $R = R_0 + R_1$, which in turn determines $e = H(R, \mathsf{pk}, m)$. Finally, P_0 computes $s_0 = \mathsf{sk}_0 \cdot e + r_0$ locally and sends this value (along with the opening of C to R_0) to P_1, who is able to compute $s_1 = \mathsf{sk}_1 \cdot e + r_1$ and complete the signature $(R = R_0 + R_1, s = s_0 + s_1)$.

A naive method to statelessly derandomize such a protocol would be to instruct each P_i to sample a long-term secret seed sd_i, and begin the protocol by computing $r_i = \mathsf{F}_{\mathsf{sd}_i}(m)$ using a pseudorandom function F rather than sampling one afresh. This naive approach already solves the problem in the semi-honest setting. However, it runs into the following issue for malicious security: a corrupt P_1 could initiate two signature sessions to sign the same message, behaving honestly in the first instance, and in the second instance using $r_1^* \neq \mathsf{F}_{\mathsf{sd}_1}(m)$ in place of the correct r_1 (equivalently $R_1^* \neq R_1$). As P_0's choice of r_0 depends only on (sd_0, m), its nonce share R_0 stays the same in both instances. This leads to different nonces $R = R_0 + R_1$ and $R^* = R_0 + R_1^*$ with corresponding $e = H(R, \mathsf{pk}, m)$ and $e^* = H(R^*, \mathsf{pk}, m)$ in the two instances, which induces P_0 to reveal $s_0 = \mathsf{sk}_0 e + r_0$ and $s_0^* = \mathsf{sk}_0 e^* + r_0$ to P_1. As s_0, s_0^* constitute two independent linear combinations of the same two values (one of which is sk_0), P_1 can simply solve for sk_0 and recover the whole signing key. This flavour of issue was first documented by Maxwell et al. [MPSW19], and has been notoriously difficult to mitigate.

Existing Approaches. To our knowledge, there are only two papers in the literature that present techniques to mitigate the above issue: those of Nick et al. [NRSW20], and Garillot et al. [GKMN21]. Both works take a GMW-type approach [GMW87] of having each party prove in zero-knowledge that it executed the naive semi-honest stateless deterministic protocol correctly. Conceptually, the approach is simple: each P_i provides a one-time commitment to its long-term seed sd_i—perhaps during distributed key generation—and subsequently when signing a message, each P_i proves in zero-knowledge that its claimed nonce R_i is indeed of the form $R_i = \mathsf{F}_{\mathsf{sd}_i}(m) \cdot G$, where sd_i is contained in the commitment.

This simple approach is non-trivial to implement with concrete efficiency, as the statement being proven consists of an algebraic component (by virtue of the elliptic curve operations), and a complex non-algebraic one (due to the PRF). Nick et al. [NRSW20] reconcile this difference by

designing a custom arithmetization-friendly PRF to use along with a succinct proof system, i.e. Bulletproofs [BBB+18]. On the other hand, Garillot et al. [GKMN21] take the opposite approach; they start with a standard block-cipher based PRF—such as AES or SHA—and optimize garbled circuit based ZK proofs [JKO13] for this setting.

Both approaches (and indeed any approach based on the GMW paradigm) are bottlenecked by having to prove cryptographic statements involving F in zero-knowledge; the efficiency of such approaches inherently depends on the circuit complexity of pseudorandom functions. It is therefore natural to ask,

Can we design a distributed, stateless, deterministic, Schnorr signing scheme that makes only blackbox use of cryptographic primitives?

Our Results. In this work, we develop a new methodology for distributing Schnorr signing while retaining stateless determinism, that makes blackbox use of an increasingly general cryptographic primitive—*pseudorandom correlation functions*, or PCFs [BCG+20]. Just as the single party algorithm is easily derandomized with PRFs, we invoke PCFs to natively translate the derandomization technique to the distributed setting. Roughly, PCFs are the multiparty analogue of PRFs—they compress exponentially large correlated random tapes into short keys. This way, the random tape for all signing nonces is effectively committed during distributed key generation; when signing a message the parties access the relevant portion to obtain their nonce shares, and use the correlation to validate each other's shares.

The complexity of the correlation determines the efficiency of the PCF, and so we show that the relatively simple Vector Oblivious Linear Evaluation (VOLE) correlation suffices for our task. We present two instantiations of our methodology:

- Our first construction is based on the SoftSpoken VOLE PCF by Roy [Roy22], and makes blackbox use of any PRF. While the original construction worked only for small fields, we generalize it to arbitrary ones to be able to use it in our context. The computation overhead of this approach is barely noticeable relative to the naive semi-honest protocol (only additions in $\mathbb{Z}_q$), and the bandwidth overhead is a single group element. However, it only achieves *covert* security [AL07].
- Our second construction achieves full malicious security, and correspondingly requires a VOLE PCF for a large field. Our starting point is the Paillier encryption based PCF of Orlandi et al. [OSY21]. Since their PCF generates correlation modulo a biprime, we need to carefully devise a technique to obtain VOLE correlations modulo a prime while preserving active security which can be used in our context. We construct a secure translation and consistency checking mechanism for this purpose, whose security we reduce to the Decisional Composite Residuosity (DCR) and Strong RSA assumptions. Besides instantiating a fundamentally new approach, our construction achieves the lowest bandwidth consumption of any known stateless deterministic Schnorr signing scheme (only a few hundred bytes), at reasonable computation overhead (a few exponentiations). We report on a proof-of-concept implementation in Section 5.2.

As an interesting additional benefit, our constructions achieve two-round signing, which has been notoriously difficult to accomplish even in the randomized setting [DEF+19]. Existing two-round signing protocols either rely on the Algebraic Group Model or interactive assumptions [KG20, NRS21, AB21], or make use of non-black-box zero-knowledge proofs of cryptographic statements [NRSW20]. We give a comparison of our approaches to those from the literature in Table 1.1.

Key Generation/Setup. Our protocols make use of an ideal setup oracle $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$ to sample and distribute the PCF keys. In principle, this oracle can be realized by MPC to obtain a fully distributed

	Stateless & Deterministic	Rounds	Bandwidth (KB)	Assumptions	Blackbox	Security
[Lin22]	✗	3	0.9	RO	✓	Malicious
[NRS21]	✗	2	0.1	RO+OMDL	✓	Malicious
[NRSW20]	✓	2	1.1	RO+DDH[a]	✗	Malicious
[GKMN21]	✓	3	307	RO+PRF	✗	Malicious
This work	✓	2	0.1	OT+PRF	✓	Covert
This work	✓	2	0.5	RO+DCR+Strong RSA	✓	Malicious

[a] DDH in a custom elliptic curve, not the same one as the signature.

Table 1.1: Comparison of different techniques to distribute Schnorr signing. Computation complexity is not represented in this table as it is best measured empirically, and not all works are implemented. Note that our protocols invoke an ideal oracle $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$ during key generation, which is not the focus in this work. Bandwidth cost represents data transmitted per party when signing with a 256-bit elliptic curve, and is either derived analytically here (see Appendix C for [Lin22, NRS21] and Sections 4.1 and 5.1 for our work) or taken from previous work. "Blackbox" refers to the use of cryptographic tools as a black box, in order to avoid dependence on their circuit complexity. For instance, Schnorr's proof of knowledge of discrete logarithm is blackbox as it uses the group as an oracle, whereas proving statements that involve the circuit representation of group operations is not.

protocol. The focus of this work is on the signing phase, and so we do not focus on the concrete efficiency or optimizations in realizing $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$ beyond remarking that a single all-but-one OT suffices for our covert construction, and a note for our maliciously secure construction in Section 5.1. Moreover, as discussed by Abram et al. [ANO+22], in many practical applications of threshold cryptography there is a natural "trusted dealer" that can execute $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$, namely the owner of the cryptocurrency wallet distributing its secret key.

Future Work. As shown in Table 1.1, our results achieve *either* malicious security, or security under generic assumptions (OT and PRF). Achieving both at the same time remains an interesting open question.

1.2 Our Techniques

In this work, we retain the approach of having each P_i prove that their claimed nonce was derived correctly, but we do not look to the literature on zero-knowledge proofs to instantiate such an object. Instead, we view this problem through the lens of Pseudorandom Correlation Functions (PCFs) [BCG+20]. Informally, a two-party PCF produces two keys k_0, k_1 so that for any public $x \in \{0,1\}^\kappa$, it holds that $a_0 = \mathsf{PCF.Eval}(k_0, x)$ and $a_1 = \mathsf{PCF.Eval}(k_1, x)$ are correlated per some useful function. As an example, a PCF for the (Random) Oblivious Transfer correlation would enforce that $a_0 = (m_0, m_1) \in \{0,1\}^{2\times\kappa}$, and $a_1 = (b, m_b) \in \{0,1\} \times \{0,1\}^\kappa$. The 'pseudorandomness' property intuitively guarantees that (a_0, a_1) is distributed pseudorandomly in the appropriate domain.

Feasibility of the Ideal PCF. We begin by considering what the ideal PCF might look like for the task of nonce derivation. Consider a "nonce correlation", which enforces that (a_0, a_1) are of the form $a_0 \in \mathbb{Z}_q$ and $a_1 = a_0 \cdot G \in \mathbb{G}$. It is immediate how such a PCF would be useful in the design of two-party Schnorr signing—to sign a message m, each P_i derives its nonce as $r_i = \mathsf{PCF.Eval}(k_{i,0}, m)$ while P_{1-i} derives the corresponding $R_i = \mathsf{PCF.Eval}(k_{i,1}, m)$. As each party P_i can derive the other's nonce share R_{1-i} locally via the PCF, the common nonce $R = R_0 + R_1$ is established non-interactively just by fixing the message, and it only remains for the parties to locally compute and exchange their s_i values. This gives a conceptually simple non-interactive stateless deterministic two-party Schnorr signing protocol. While it is feasible to construct such a PCF via generic techniques [DHRW16, BCG+20], it is unclear how to instantiate it with reasonable concrete efficiency, or even blackbox in the group $\mathbb{G}$.

The VOLE PCF is Nearly Ideal. The Vector Oblivious Linear Evaluation (VOLE) correlation enforces that (a_0, a_1) be of the form $a_0 = (u, v)$ and $a_1 = (w, \Delta)$ such that $w - v = u\Delta$, where Δ is fixed across all evaluations with the same key pair. If the correlation holds mod q, the value u can be used as r_i directly, and v, w, Δ can serve to *authenticate* $R_i = u \cdot G$. In particular, if P_0 and P_1 derive (a_0, a_1) as above, and P_0 sends $R_0 = u \cdot G$ and $V = v \cdot G$ to P_1, party P_1 can then check the correlation *in the exponent*, i.e. validate $w \cdot G - V \stackrel{?}{=} \Delta \cdot R_i$. The probability that P_0 tricks P_1 into accepting a false $R_0^* \neq R_0$ is essentially equivalent to the probability that P_0 is able to guess Δ.

The VOLE PCF is therefore conducive to a two-round Schnorr signing protocol: each P_i derives $r_i, v_i, w_{1-i}, \Delta_{1-i}$ locally—correlated as $w_i - v_i = r_i\Delta$ for each $i \in \{0, 1\}$—and sends the corresponding R_i, V_i to P_{1-i}. Upon receiving R_{1-i}, V_{1-i}, each P_i validates the correlation with w_{1-i}, Δ_i in the exponent, and sets $R = R_0 + R_1$. The parties then locally compute and exchange their respective $s_i = \mathsf{sk}_i \cdot e + r_i$ values, and complete the signature as $(R, s = s_0 + s_1)$.

First Instantiation: SoftSpoken VOLE. As a building block for OT extension, Roy [Roy22] introduced a PCF for the VOLE correlation called SoftSpoken VOLE, which makes blackbox use of any PRF. However, this PCF was designed to create VOLE correlations in $\mathbb{F}_{2^{O(\log \kappa)}}$ as a generalization of the IKNP OT extension technique [IKNP03]. In order to use it in our context, we generalize the SoftSpoken VOLE technique so that it can produce such correlations in any field, even exponentially large ones. However, an important caveat of this adjustment is that while u is uniform in say $\mathbb{Z}_q$, in order for the PCF evaluation to be efficient (i.e. polynomial in κ), the Δ component of the correlation is restricted to a polynomially large subset of $\mathbb{Z}_q$. This means that a corrupt P_i^* could fool P_{1-i} into accepting an incorrect R_i^* with small but noticeable probability, and thus our first instantiation only achieves covert security.

Second Instantiation: Paillier PCF. Orlandi et al. [OSY21] presented a PCF for the VOLE correlation based on Paillier's encryption scheme. For concreteness, assume that P_0 owns k_0, and P_1 owns k_1 which allow them to non-interactively generate shares $w - v$ of $u\Delta$ in $\mathbb{Z}_N$. Although the PCF can produce exponentially large Δ values, the correlations it produces hold over the ring $\mathbb{Z}_N$ where N is the product of two large primes—this factorization of N is known only to the party that holds key k_0. We therefore have to design an additional protocol to carefully 'translate' this correlation from $\mathbb{Z}_N$ to a correlation in $\mathbb{Z}_q$.

This translation problem turns out to be surprisingly non-trivial, as demonstrated by our initial failed approaches. We first explored techniques such as rounded division ($u' = \lfloor \frac{uq}{N} \rceil$) and remainder ($u' = (u \bmod N) \bmod q$) to convert from $\mathbb{Z}_N$ to $\mathbb{Z}_q$. Any such conversion introduces errors into the shares, which is quite difficult to securely correct to a valid correlation over $\mathbb{Z}_q$. Consider an IKNP OT extension [IKNP03] type approach, where P_0 sends a "correction word" to account for

the error. The challenge then is to ensure that P_0 is unable to send a malformed correction word without being detected. One might try adding a VOLE consistency checking protocol to verify the correlation, but validity of the correlation alone does not suffice in our setting; P_0 must be unable to *change* its output $u' \in \mathbb{Z}_q$ in repeated invocations of the protocol. One approach that we explored at this point is having P_0 prove that the corrected u' is "small", i.e. fully reduced modulo q. However this led us to a general issue with (approximate) range proofs: any such proof system that we could develop achieved at most a soundness of $\frac{1}{2}$ per repetition. This stems from an issue where P_0 can add an error of $\frac{1}{2} \bmod N$ at some point in the protocol, which has a noticeable chance of escaping detection when multiplied by a (small) even number. We refer the reader to Couteau et al. [CKLR21, Section 2.1] for a detailed discussion of this flavour of issue.

In Section 5 we present our solution to this translation problem that finally worked. Roughly, our translation to $\mathbb{Z}_q$ mechanism works as follows: First, the parties translate their shares over $\mathbb{Z}_N$ into shares over $\mathbb{Z}$, removing the modulus by using a technique from the Paillier HSS constructions of [OSY21, RS21]. This requires sending a correction, to force u to be small compared to N. Then, the parties take their shares modulo q to get shares suitable for use in two-party Schnorr. To handle P_0's possible lies, we introduce some carefully crafted checks which only allow P_0 to lie by a multiple of some parameter M, which we require to be divisible by q (thus having no impact on the final result). The most important of these checks is an "integer consistency check" in the exponent, using a different group generated by $g \in (\mathbb{Z}_{N_\mathsf{V}})^\times$, where N_V is another semiprime, making g's order be unknown. Performing a check over $\mathbb{Z}$ instead of $\mathbb{Z}_N$ avoids the issue of wraparound, as multiplying two large numbers cannot output a small number. We defer a more detailed technical description to Section 5.

1.3 Related Work

As discussed earlier, to our knowledge the works of Nick et al. [NRSW20] and Garillot et al. [GKMN21] are the only ones to present distributed Schnorr signing protocols where signing is stateless and deterministic. Both use zero-knowledge proofs to prove the consistency of claimed nonces with respect to committed nonce derivation keys. Our full security PCF construction achieves better communication complexity than both, while staying within the realm of standard assumptions.

Smart and Alaoui [ST19], and Dalskov et al. [DOK+20] observed that SPDZ-style MACs can be checked in the exponent, and showed how to apply this principle in the context of distributing the computation of ECDSA signatures. Our techniques in this paper can be viewed as a PCF interpretation of a similar idea, i.e. that VOLE correlations can be useful for authentication in the exponent.

Abram et al. [ANO+22] used Pseudorandom Correlation *Generators* (PCGs) to generate useful correlations to distribute the computation of ECDSA. Like us, they follow a pseudorandom correlation paradigm in a threshold signature context, however the setting in their paper is entirely different; their work optimizes storage and 'online' bandwidth complexity of distributing ECDSA, and their techniques rely on maintaining state.

Bonte et al. [BST21] investigate the cost of using MPC to distribute the EdDSA signing algorithm, which derives nonces by hashing the message to be signed with a long-term secret. While the ideal functionality that they achieve is stateless and deterministic, the *protocol* itself is not—i.e. their protocol relies on keeping state and/or sampling fresh randomness online.

Several works on threshold ECDSA make use of Paillier encryption [GG18, LN18, CGG+20] for Oblivious Linear Evaluation (OLE). Their techniques are fundamentally different; they rely on a classic OLE protocol that leverages the additive homomorphism of Paillier encryption [Gil99], and make use of expensive range proofs to enforce the correctness of the OLE. In contrast, our full

security construction does not directly make use of the homomorphism of Paillier encryption, and rather than use range proofs to enforce honesty, we design a custom mechanism to guarantee that the VOLE correlation is correctly 'translated' from $\mathbb{Z}_N$ to $\mathbb{Z}_q$.

2 Definitions

Notation. We define the modulus operation to be symmetric, meaning that $a \bmod b \in [-\frac{b}{2}, \frac{b}{2}) \cap \mathbb{Z}$. This works together with rounding $\lfloor \frac{a}{b} \rceil$ as quotient and remainder: $a = b\lfloor \frac{a}{b} \rceil + a \bmod b$.

2.1 Semiprime-Related Assumptions

There are many cryptographic schemes built on using a semiprime's factorization as the trapdoor. Here, we will present the assumptions we need. First, we need to choose a distribution for the semiprimes.

$N, \varphi \leftarrow$ **RSA.Gen**(1^κ)**:** Sample primes $p, q \in (2^{\ell(\kappa)/2-1}, 2^{\ell(\kappa)/2})$ uniformly at random. Output $N = pq \in (2^{\ell(\kappa)-2}, 2^{\ell(\kappa)})$ and $\varphi = (p-1)(q-1)$.

$N, \varphi \leftarrow$ **RSA.GenSafe**(1^κ)**:** Sample *safe* primes $p, q \in (2^{\ell(\kappa)/2-1}, 2^{\ell(\kappa)/2})$ uniformly at random. I.e. sample primes p, q such that $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are also prime. Compute N, φ as in RSA.Gen.

The polynomial $\ell(\kappa)$ should be chosen to make the related hardness assumptions achieve κ-bit security.[1]

First, we the DCR assumption for the security of the Paillier cryptosystem.

Definition 2.1. The decisional composite residuosity (DCR) assumption states that the following distributions are indistinguishable. RSA.Gen

$(N, \phi) \leftarrow \mathsf{RSA.Gen}(1^\kappa)$ $r \leftarrow (\mathbb{Z}_{N^2})^\times$ return N, r	$(N, \phi) \leftarrow \mathsf{RSA.Gen}(1^\kappa)$ $r \leftarrow (\mathbb{Z}_{N^2})^\times$ return N, r^N

Second, we will use a consistency check based on a group of unknown order, for which we use multiplication modulo a semiprime (similarly to [DF02]). For security, we need two properties. First, it must be hard to find roots of unity other than ± 1, i.e., to find elements $x \in (\mathbb{Z}_N)^\times \setminus \{\pm 1\}$ of low multiplicative order. If N is sampled with GenSafe, the only small order possible is 2, and a square root of unity would allow N to be factored. Second, it must be hard to compute modular roots. We use a version of the RSA assumption.

Definition 2.2. The Strong RSA assumption states that all PPT adversaries $\mathcal{A}$ must have negligible chance of winning the following game.

$(N, \phi) \leftarrow \mathsf{RSA.GenSafe}(1^\kappa)$ $g \leftarrow (\mathbb{Z}_N)^\times$ $(z, e) \leftarrow \mathcal{A}(N)$ win if $\lvert e \rvert > 1$ and $z^e = g$

[1] Assuming that the adversary uses the general number field sieve, $\ell(\kappa) = \widetilde{\Theta}(\kappa^3)$.

2.2 Pseudorandom Correlation Functions

We will use the Pailler-based pseudorandom correlation function (PCF) of [OSY21]. There are a couple of issues, however, with applying their definition to Pailler PCF for VOLE correlations. First, the output ring of the correlation, $\mathbb{Z}_N$ for a semiprime N, is sampled randomly. However, there is no feature in their definition that allows the group to be sampled. Second, the master secret key msk for VOLE is Δ, and it is supposed to be both output as part of y_1. But their reverse sampling definition requires that the output distribution be the same for two different choices of msk.

Below we have modified their definitions to fix these issues.

Definition 2.3. Let $1 \leq \ell_0(\kappa), \ell_1(\kappa) \leq \mathsf{poly}(\kappa)$ be output-length functions, and let $\mathcal{M}_i$ be a set of allowed master keys for party i. Let $(\mathsf{Setup}, \mathcal{Y})$ be a tuple of probabilistic algorithms, such that

- $\mathsf{Setup}(1^\kappa, \mathsf{msk}_0 \in \mathcal{M}_0, \mathcal{M}_1)$ samples a distribution key pk and an (optional) trapdoor sk.
- $\mathcal{Y}(\mathsf{pk}, \mathsf{msk}_0, \mathsf{msk}_1)$, returns a pair of outputs $(y_0, y_1) \in \{0,1\}^{\ell_0(\kappa)} \times \{0,1\}^{\ell_1(\kappa)}$.

We say that the tuple $(\mathsf{Setup}, \mathcal{Y})$ defines a reverse sampleable correlation with setup if there exists a probabilistic polynomial time algorithm $\mathsf{RSample}$ such that

- $\mathsf{RSample}(\mathsf{pk}, \mathsf{msk}_0, \mathsf{msk}_1, \sigma \in \{0,1\}, y_\sigma \in \{0,1\}^{\ell_\sigma(\kappa)})$ returns $y_{1-\sigma} \in \{0,1\}^{\ell_{1-\sigma}(\kappa)}$ such that for all $\mathsf{msk}_0 \in \mathcal{M}_0, \mathsf{msk}_1 \in \mathcal{M}_1$ and all $\sigma \in \{0,1\}$, the distributions of $(\mathsf{pk}, \mathsf{sk}, y_\sigma, y_{1-\sigma})$ and $(\mathsf{pk}, \mathsf{sk}, y_\sigma, y^*)$ are statistically close, where:

$$\left\{ (\mathsf{pk}, \mathsf{sk}, y_0, y_1, y^*) \;\middle|\; \begin{array}{rl} (\mathsf{pk}, \mathsf{sk}) \leftarrow & \mathsf{Setup}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1) \\ (y_0, y_1) \leftarrow & \mathcal{Y}(\mathsf{pk}, \mathsf{msk}_0, \mathsf{msk}_1) \\ y^* \leftarrow & \mathsf{RSample}(\mathsf{pk}, \mathsf{msk}_0, \mathsf{msk}_1, \sigma, y_\sigma) \end{array} \right\}$$

To show how this reverse sampling definition works, we next give the distribution for VOLE correlations.

Definition 2.4. A reverse sampleable correlation $(\mathsf{Setup}, \mathcal{Y})$ is a *VOLE correlation* if $\mathcal{M}_0 = \{\perp\}$, $\mathcal{M}_1 \subseteq \mathbb{Z}$, pk outputs a public modulus N, and the distribution $\mathcal{Y}(N, \perp, \Delta)$ samples $u, v \leftarrow \mathbb{Z}_N$, computes $w := u\Delta + v$, and outputs $((u, v), w)$.

Definition 2.5. Let $(\mathsf{Setup}, \mathcal{Y})$ fix a reverse-sampleable correlation with setup which has output length functions $\ell_0(\kappa), \ell_1(\kappa)$ and sets $\mathcal{M}_0, \mathcal{M}_1$ of allowed master keys, and let $\kappa \leq n(\kappa) \leq \mathsf{poly}(\kappa)$ be an input length function. Let $(\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ be a pair of algorithms with the following syntax:

- $\mathsf{PCF.Gen}(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}_0, \mathsf{msk}_1)$ is a probabilistic polynomial time algorithm that outputs a pair of keys (k_0, k_1);
- $\mathsf{PCF.Eval}(\sigma, k_\sigma, x)$ is a deterministic polynomial-time algorithm that on input $\sigma \in \{0,1\}$, key k_σ and input value $x \in \{0,1\}^{n(\kappa)}$, outputs a value $y_\sigma \in \{0,1\}^{\ell_\sigma(\kappa)}$.

We say $(\mathsf{PCF.Gen}, \mathsf{PCF.Eval})$ is a (weak) pseudorandom correlation function (PCF) for $\mathcal{Y}$, if the following conditions hold:

- Pseudorandom $\mathcal{Y}$-correlated outputs. For every $\mathsf{msk}_0 \in \mathcal{M}_0$, $\mathsf{msk}_1 \in \mathcal{M}_1$, and non-uniform adversary $\mathcal{A}$ of size $\mathsf{poly}(\kappa)$, and every $Q = \mathsf{poly}(\kappa)$, it holds that

$$|\Pr[\mathrm{Exp}_0^{pr}(\kappa) = 1] - \Pr[\mathrm{Exp}_1^{pr}(\kappa) = 1]| \leq negl(\kappa)$$

  for all sufficiently large κ, where $\mathrm{Exp}_b^{pr}(\kappa)$ for $b \in \{0,1\}$ is defined as follows. (In particular, where the adversary is given access to $Q(\kappa)$ samples.)

$\underline{\mathrm{Exp}_0^{pr}(\kappa)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1)$

for $i = 1$ to $Q(\kappa)$:
$x^{(i)} \leftarrow \{0,1\}^{n(\kappa)}$
$(y_0^{(i)}, y_1^{(i)}) \leftarrow Y(1^\kappa, \mathsf{msk})$
$b \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk}, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i\in[Q(\kappa)]})$
return b

$\underline{\mathrm{Exp}_1^{pr}(\kappa)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1)$
$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}_0, \mathsf{msk}_1)$
for $i = 1$ to $Q(\kappa)$:
$x^{(i)} \leftarrow \{0,1\}^{n(\kappa)}$
for $\sigma \in \{0,1\}$: $y_\sigma^{(i)} \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk}, (x^{(i)}, y_0^{(i)}, y_1^{(i)})_{i\in[Q(\kappa)]})$
return b

– Security. For each $\sigma \in \{0,1\}$ there is a simulator $\mathcal{S}_\sigma$s such that for every $\mathsf{msk}_0 \in \mathcal{M}_0$, $\mathsf{msk}_1 \in \mathcal{M}_1$, any every non-uniform adversary $\mathcal{A}$ of size $B(\kappa)$, and every $Q = \mathsf{poly}(\kappa)$, it holds that

$$|\Pr[\mathrm{Exp}_0^{sec}(\kappa) = 1] - \Pr[\mathrm{Exp}_1^{sec}(\kappa) = 1]| \leq negl(\kappa)$$

for all sufficiently large κ, where $\mathrm{Exp}_b^{sec}(\kappa)$ for $b \in \{0,1\}$ is defined as follows (again, with $Q(\kappa)$ samples).

$\underline{\mathrm{Exp}_0^{sec}(\kappa)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1)$
$(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}_0, \mathsf{msk}_1)$
for $i = 1$ to $Q(\kappa)$:
$x^{(i)} \leftarrow \{0,1\}^{n(\kappa)}$
$y_{1-\sigma}^{(i)} \leftarrow \mathsf{PCF.Eval}(1-\sigma, k_{1-\sigma}, x^{(i)})$
$b \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[Q(\kappa)]})$
return b

$\underline{\mathrm{Exp}_1^{sec}(\kappa)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1)$
$k_\sigma \leftarrow \mathcal{S}_\sigma(\mathsf{pk}, \mathsf{sk}, \mathsf{msk}_\sigma)$
for $i = 1$ to $Q(\kappa)$:
$x^{(i)} \leftarrow \{0,1\}^{n(\kappa)}$
$y_\sigma^{(i)} \leftarrow \mathsf{PCF.Eval}(\sigma, k_\sigma, x^{(i)})$
$y_{1-\sigma}^{(i)} \leftarrow \mathsf{RSample}(1^\kappa, \mathsf{msk}_0, \mathsf{msk}_1, \sigma, y_\sigma^{(i)})$
$b \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk}, k_\sigma, (x^{(i)}, y_{1-\sigma}^{(i)})_{i\in[Q(\kappa)]})$
return b

2.3 Discrete Log Pseudorandom Correlation Functions

As we wish to enable a diversity of techniques and instantiations for our approach, rather than present a protocol that uses a specific type of PCF directly, we instead make use of an intermediate object that we define here, called an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$. Roughly, an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ produces two keys $(k_\mathsf{P}, k_\mathsf{V})$ through a setup algorithm. Given public input m (which can be for e.g. a message to be signed), k_P can be used to derive a $r_m \in \mathbb{Z}_q$ and an accompanying π_R, which serves as a proof that the nonce $R_m = r_m \cdot G$ was correctly derived. These nonces must be pseudorandom, and the probability that a verifier is fooled into accepting an incorrect $R_m^* \neq R_m$ is bounded by ε.

We formalize these intuitive properties below, accounting for subtleties that will be important for the simulation of our UC-secure signing protocol.

Definition 2.6. An $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ is characterized by five algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathcal{S}, \mathcal{V})$. The algorithms P and V are not provided with random tapes. The security parameter is taken to be equal to the size of the group, i.e. $\kappa = |\mathbb{G}|$. These algorithms must satisfy the following properties:

– **Completeness**: For any efficient adversary $\mathcal{A}$ interacting with the oracle $\mathcal{O}_{\mathsf{compl}}$, the chance of an abort is negligible.

$(k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(\kappa)$
$\underline{\mathcal{O}_{\mathsf{compl}}(m)}$:
$(r_m, \pi_R) \leftarrow \mathsf{P}(k_\mathsf{P}, m)$
$R_m = r_m \cdot G$
abort if $\mathsf{V}(k_\mathsf{V}, m, R_m, \pi_R) \neq 1$
return r_m, π_R

– ε-**Soundness**: For any efficient algorithm $\mathcal{A}$:

$$\Pr\left[R^* \neq R_m \ \wedge \ \mathsf{V}(k_\mathsf{V}, m, R^*, \pi_R^*) = 1 \ : \ \begin{array}{rcl}(k_\mathsf{P}, k_\mathsf{V}) & \leftarrow & \mathsf{Setup}(\kappa) \\ (m, R^*, \pi_R^*) & \leftarrow & \mathcal{A}(k_\mathsf{P}) \\ (r_m, \pi_R) & \leftarrow & \mathsf{P}(k_\mathsf{P}, m) \\ R_m & = & r_m \cdot G\end{array}\right] \leq \varepsilon(\kappa)$$

– **Pseudorandom Nonces With Simulatable Proofs**: Define two oracles, the first being $\mathcal{O}_\mathsf{P}(k_\mathsf{P})$, which on query $m \in \{0,1\}^\kappa$ computes $(r_m, \pi_R) \leftarrow \mathsf{P}(k_\mathsf{P}, m)$ and returns (R_m, π_R) where $R_m = r_m \cdot G$. The second oracle $\mathcal{O}_\mathcal{S}(k_\mathsf{V})$ is defined as follows:

1. Upon initialization, sample a random tape $\rho \in \{0,1\}^\kappa$ for $\mathcal{S}$
2. Upon receiving a query $m \in \{0,1\}^*$, if R_m is undefined, then sample $R_m \leftarrow \mathbb{G}$ and $\pi_R \leftarrow \mathcal{S}(k_\mathsf{V}, m, R_m, ; \rho)$
3. Return (R_m, π_R)

There is a negligible function negl such that for any efficient adversary $\mathcal{A}$,

$$\left| \begin{array}{l}\Pr[\mathcal{A}^{\mathcal{O}_\mathcal{S}(k_\mathsf{V})}(k_\mathsf{V}) = 1 \ : \ (k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(\kappa)] \\ -\Pr[\mathcal{A}^{\mathcal{O}_\mathsf{P}(k_\mathsf{P})}(k_\mathsf{V}) = 1 \ : \ (k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(\kappa)]\end{array} \right| \leq \mathsf{negl}(\kappa)$$

– **Simulatable Proof Validation**: Define the oracle $\mathcal{O}_\mathcal{V}(k_\mathsf{P}, \cdot)$ as follows:
Upon receiving (m, R_m^*, π_R^*) as input:

1. If $\texttt{flag}$ is already defined, and $\texttt{flag} \neq 1$, then ignore the query.
2. Otherwise if $\texttt{flag}$ is undefined, sample a random integer $\texttt{coin} \leftarrow [1, 1/\varepsilon(\kappa)]$.
3. Set $\texttt{flag} = \texttt{coin} \cdot \mathcal{V}(k_\mathsf{P}, m, R_m^*, \pi_R^*)$, and return $\texttt{flag} \stackrel{?}{=} 1$.

The oracles $\mathcal{O}_\mathcal{V}(k_\mathsf{P}, \cdot)$ and $\mathsf{V}(k_\mathsf{V}, \cdot)$ are indistinguishable to any efficient $\mathcal{A}(k_\mathsf{P})$ that queries only *incorrect* nonces (i.e. $R_m^* \neq r_m \cdot G$ where $(r_m, \cdot) \leftarrow \mathsf{P}(k_\mathsf{P}, m)$) over choice of $(k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(\kappa)$.

Remark 2.7. Simulatable Proof Validation follows directly from ε-Soundness when ε is negligible in κ. This is due to a canonical $\mathcal{V}$ that simply rejects all incorrect R_m^*—by soundness, it follows that *any* incorrect R_m^* will be accepted by a verifier only with negligible probability.

Remark 2.8. The P and V algorithms are deterministic, and do not allow state by syntax, which will result in a stateless deterministic distributed Schnorr protocol later on.

3 Deterministic Signing from Pseudorandom Discrete Logarithm Nonce Derivation Functions

Given an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_\mathsf{DL}$, we show how to distribute the computation of Schnorr signatures so that the signing protocol enjoys stateless determinism. We begin by describing the ideal UC functionality that we will realize.

Functionality 3.1. $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$**. Threshold Schnorr Signing With Error**

This two-party functionality is parameterized by the group $(\mathbb{G}, G, q)$, error ε such that $\eta = 1/\varepsilon$ is an integer, and the hash function H. All messages are adversarially delayed.

Key Generation: Run once.

1. Upon receiving $(sid, \texttt{init})$ from both parties, sample $\mathsf{sk} \leftarrow \mathbb{Z}_q$, and compute $\mathsf{pk} = \mathsf{sk} \cdot G$
2. In case an entry prefixed by sid does not already exist, send $(sid, \texttt{public-key}, \mathsf{pk})$ to both parties, and store $(sid, \texttt{keys}, \mathsf{pk}, \mathsf{sk})$ in memory.

Signing a message: Run arbitrarily many times.

1. Ignore any queries prefixed by sid if $(sid, \texttt{keys}, \mathsf{pk}, \mathsf{sk})$ does not exist in memory.
2. Upon receiving $(sid, \texttt{sign}, m)$ from both parties, if m has not previously been signed, sample $r_m \leftarrow \mathbb{Z}_q$ and store $(sid, \texttt{nonce}, m, r_m)$ in memory. Otherwise retrieve $(sid, \texttt{nonce}, m, r_m)$ from memory.
3. Compute $R_m = r_m \cdot G$ and in case P_i is corrupt, send it $(sid, \texttt{nonce}, R_m)$ and wait for $(sid, \texttt{proceed}, m)$.
4. Compute
$$s = \mathsf{sk} \cdot H(R_m, \mathsf{pk}, m) + r_m$$
and send $(sid, \texttt{sig}, R_m, s)$ to both parties.

Cheat: If P_i is corrupt, it may send $(sid, \texttt{cheat}, m)$ instead of $(sid, \texttt{proceed}, m)$ upon receiving $(sid, \texttt{nonce}, R_m)$. If initialized and `cheat` has not previously been sent, then:

1. Uniformly sample a random integer $\texttt{coin} \leftarrow [1, \eta]$
2. If $\texttt{coin} = 1$ then P_i is given control of this functionality (i.e. henceforth P_i receives all messages and responds on behalf of $\mathcal{F}^{\varepsilon}_{\mathsf{Schnorr}}$) without notifying P_{1-i}.
3. Otherwise, send $(sid, \texttt{cheat-detected})$ to both P_0 and P_1, and stop accepting further instructions.

Observe that if $\varepsilon(\kappa)$ is negligible in κ, then $\mathcal{F}^{\varepsilon}_{\mathsf{Schnorr}}$ is effectively equivalent to the standard Schnorr signing functionality (up to syntax); sending $(sid, \texttt{cheat}, m)$ to $\mathcal{F}^{\varepsilon}_{\mathsf{Schnorr}}$ in this case is equivalent to instructing the standard Schnorr signing functionality to abort.

Our protocol makes use of an $\mathcal{F}^{\mathsf{PCF}}_{\mathsf{Setup}}$ hybrid functionality, which upon receipt of the initialization commands for an sid (i.e. `prover-init` and `verifier-init`) simply executes the Setup algorithm of the $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ and returns the resulting k_{P} and k_{V} to the appropriate parties. We give the exact description in Appendix A. While such a functionality can always be instantiated generically by MPC, we discuss instantiations tailored to each $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ construction in their respective sections. The focus of this work is on the round complexity and stateless determinism of the *signing* protocol, and so we do not prioritize the optimization of the instantiation of the one-time setup phase. We now give the our distributed Schnorr signing protocol.

Protocol 3.2. π_{Sch}**. Stateless Deterministic Threshold Schnorr Signing**

This two-party protocol is parameterized by the group $(\mathbb{G}, G, q)$, where $|\mathbb{G}| = \kappa$, an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ characterized by algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathcal{S}, \mathcal{V})$, and optionally $\varepsilon_\kappa = \varepsilon(\kappa)$ such that $\eta = 1/\varepsilon_\kappa$ is an integer. This protocol additionally makes use of ideal functionalities $\mathcal{F}^{\mathsf{PCF}}_{\mathsf{Setup}}$ and $\mathcal{F}^{\mathsf{R}_{DL}}_{\mathsf{Com-ZK}}$, which are given in Appendix A.

All superscripts in the protocol descriptions are indices.

Key Generation: For $i \in \{0, 1\}$, each P_i does the following:

1. Send $(sid_i, \texttt{prover-init})$ and $(sid_{1-i}, \texttt{verifier-init})$ to $\mathcal{F}^{\mathsf{PCF}}_{\mathsf{Setup}}$ and wait for responses $(sid, \texttt{prover-key}, k^i_{\mathsf{P}})$ and $(sid, \texttt{verifier-key}, k^{1-i}_{\mathsf{V}})$ respectively.
2. Sample $\mathsf{sk}_i \leftarrow \mathbb{Z}_q$, compute $\mathsf{pk}_i = \mathsf{sk}_i \cdot G$ and send $(\texttt{commit}, sid_i, \mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{F}^{\mathsf{R}_{DL}}_{\mathsf{Com-ZK}}$
3. Wait for $(\texttt{committed}, sid_{1-i})$ from $\mathcal{F}^{\mathsf{R}_{DL}}_{\mathsf{Com-ZK}}$ and send $(\texttt{open}, sid_i)$
4. Wait for $(\texttt{opened}, sid_{1-i}, \mathsf{pk}_{1-i})$ and set $\mathsf{pk} = \mathsf{pk}_0 + \mathsf{pk}_1$

Signing a message m**:** For $i \in \{0,1\}$, each P_i does the following:

1. Compute $(r_m^i, \pi_R^i) \leftarrow \mathsf{P}(k_\mathsf{P}^i)$ and $R_m^i = r_m^i \cdot G$
2. Send (R_m^i, π_R^i) to P_{1-i} and wait for (R_m^{1-i}, π_R^{1-i}) in response
3. If $\mathsf{V}(k_\mathsf{V}^i, m, R_m^{1-i}, \pi_R^{1-i}) = 0$, then abort. Otherwise, set $R_m = R_m^0 + R_m^1$
4. Send $s_i = \mathsf{sk}_i \cdot H(R_m, \mathsf{pk}, m) + r_m^i$ to P_{1-i}, and wait for s_{1-i} in response
5. Compute $s = s_0 + s_1$, and output (s, R_m) after verifying that it is a Schnorr signature on m.

Theorem 3.3. π_{Sch} UC-realizes $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$ in the $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$,$\mathcal{F}_{\mathsf{Com-ZK}}^{\mathsf{R}_{DL}}$-hybrid model in the presence of up to one static active corruption, assuming that $(\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathcal{S}, \mathcal{V})$ is an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$.

Proof. (Sketch) We briefly describe how to simulate the view of a corrupt P_i.
Key Generation. For each $i \in \{0,1\}$, the simulator runs $(k_\mathsf{P}^i, k_\mathsf{V}^i) \leftarrow \mathsf{Setup}(1^\kappa)$. The keys $(k_\mathsf{P}^i, k_\mathsf{V}^{1-i})$ are given to P_i on behalf of $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$. Additionally, $(\mathsf{pk}_i, \mathsf{sk}_i)$ is received from P_i on behalf of $\mathcal{F}_{\mathsf{Com-ZK}}^{\mathsf{R}_{DL}}$. Upon receiving pk from $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$, compute $\mathsf{pk}_{1-i} = \mathsf{pk} - \mathsf{pk}_i$ and send pk_{1-i} to P_i on behalf of $\mathcal{F}_{\mathsf{Com-ZK}}^{\mathsf{R}_{DL}}$.
Signing a message m**.**

1. Upon receiving the nonce R_m from $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$, compute $(r_m^i, \pi_R^i) \leftarrow \mathsf{P}(k_\mathsf{P}^i)$ and $R_m^i = r_m^i \cdot G$.
2. Compute $R_m^{1-i} = R_m - R_m^i$ and $\pi_R^{1-i} \leftarrow \mathcal{S}(k_\mathsf{P}, R_m^{1-i})$, and send (R_m^{1-i}, π_R^{1-i}) to P_i on behalf of P_{1-i}.
3. Upon receiving (R_m^{i*}, π_R^{i*}) from P_i,
   (a) If $R_m^{i*} = R_m^i$ and $\mathsf{V}(k_\mathsf{V}, m, R_m^i, \pi_R^{i*}) = 1$ then send `proceed` to $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$, and receive the signature (R, s) in response. Compute $s_{1-i} = s - (\mathsf{sk}_i \cdot H(R_m, \mathsf{pk}, m) + r_m^i)$ and send s_{1-i} to P_i.
   (b) Otherwise when $R_m^{i*} \neq R_m^i$, if no cheat has been attempted before, then send `cheat` to $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$. If the cheat is successful (or has been successful in the past), and $\mathcal{V}(k_\mathsf{P}, m, R_m^{i*}, \pi_R^{i*}) = 1$, obtain sk and r_m from the $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$, set $R'_m = R_m^{1-i} + R_m^{i*}$, and $\mathsf{sk}_{1-i} = \mathsf{sk} - \mathsf{sk}_i$ and $r_m^{1-i} = r_m - r_m^i$. Compute $s_{1-i} = \mathsf{sk}_{1-i} \cdot H(R'_m, \mathsf{pk}, m) + r_m^i$ and send s_{1-i} to P_i. Additionally, upon receiving s_i from P_i, instruct $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$ to send $(R'_m, s_i + s_{1-i})$ to P_{1-i}.
   (c) Abort in any case not explicitly handled above.

Indistinguishability of simulation. The simulation of the key generation phase is merely syntactically different from the real protocol. In the simulation of the signing phase, (R_m^{1-i}, π_R^{1-i}) is distributed indistinguishably from the real protocol by virtue of the Pseudorandom Nonces With Simulatable Proofs property. As for (R_m^{i*}, π_R^{i*}) being 'accepted', there are two main cases:

– $R_m^{i*} = R_m^i$, in which case the real protocol and simulation behave identically (i.e. they proceed or abort conditional on $\mathsf{V}(k_\mathsf{V}, m, R_m^i, \pi_R^{i*})$).
– $R_m^{i*} \neq R_m^i$, in which case the real protocol proceeds conditional on $\mathsf{V}(k_\mathsf{V}, m, R_m^{i*}, \pi_R^{i*})$, whereas the simulation proceeds conditional on $\mathtt{coin} = 1$ where `coin` is a random integer sampled from $[1/\eta]$ by $\mathcal{F}_{\mathsf{Schnorr}}^{\varepsilon}$ *and* $\mathcal{V}(k_\mathsf{P}, m, R_m^{i*}, \pi_R^{i*}) = 1$. These two methods of validating (R_m^{i*}, π_R^{i*}) are indistinguishable, by the Simulatable Proof Validation property. Note that since this property only accounts for *incorrect* nonces, in the reduction to this property, one must be able to simulate the output of $\mathsf{V}(k_\mathsf{V})$ when it is sent correct nonces. We claim that this oracle can be simulated in the reduction by simply always accepting correct nonces (even if the accompanying π_R^* is incorrect), at no loss of advantage. This is because a correct R_m accompanied by an π_R^* that

was not computed honestly induces one of two outcomes: (1) the honest V accepts anyway, in which case the reduction made the correct choice, or (2) the honest V aborts, in which case the adversary (and hence the reduction) would have failed anyway.

Finally, conditional on (R_m^{i*}, π_R^{i*}) having been 'accepted', both the simulation and and the real protocol compute s_{1-i} in the exact same way (up to syntax). □

Corollary 3.4. π_{Sch} UC-realizes $\mathcal{F}_{\mathsf{Schnorr}}$ in the presence of up to one static active corruption, assuming that $(\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathcal{S})$ is an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ where ε is negligible in κ.

4 Covert Security from SoftSpoken VOLE

With our $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ abstraction and the protocol π_{Sch} to use it in place, we turn our focus to how to instantiate an $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$. As a first construction, we adapt the SoftSpoken VOLE PCF of Roy [Roy22] to our context. Roughly, k_{P} consists of $\eta = 1/\varepsilon(\kappa)$ independently sampled PRF keys, and k_{V} consists of all but one of them—the missing index is labelled Δ, and k_{P} contains no information about Δ. In order to evaluate $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ at a public point m, P simply adds up the result of evaluating the PRF with each of the keys to derive r_m, and computes π_R by taking a linear combination of the PRF evaluations. Given Δ and all but the Δ^{th} PRF keys, V uses the structure of the SoftSpoken VOLE PCF to derive w correlated with π_R, R_m, Δ and verifies the correlation in the exponent. The omission of the Δ^{th} PRF key in k_{V} keeps R_m pseudorandom, and soundness follows from the fact that forging an π_R for an incorrect R_m is exactly equivalent to guessing Δ. We give our entire construction below.

Algorithm 4.1. $\mathsf{ssPCF}_{\varepsilon,\mathbb{G},\mathsf{F}}$. $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ **from SoftSpoken VOLE**

This algorithm is parameterized by the group $(\mathbb{G}, G, q)$, where $|\mathbb{G}| = \kappa$, and $\varepsilon_\kappa = \varepsilon(\kappa)$ such that $\eta = 1/\varepsilon_\kappa$ is an integer. Additionally, this algorithm makes blackbox use of a pseudorandom function $\mathsf{F} : \{0,1\}^\kappa \to \mathbb{Z}_q$

$\mathsf{Setup}(\kappa)$**:** .

1. Sample a random integer index $\Delta \leftarrow [\eta]$
2. Sample η keys, $\{k_i\}_{i\in[\eta]} \leftarrow \{0,1\}^{\kappa\times\eta}$
3. Assemble and output $k_{\mathsf{P}} = \{k_i\}_{i\in[\eta]}$, and $k_{\mathsf{V}} = \left(\Delta, \{k_i\}_{i\in[\eta]\setminus\Delta}\right)$

$\mathsf{P}(k_{\mathsf{P}}, m)$**:** Output $r_m = -\sum_{i\in[\eta]} \mathsf{F}_{k_i}(m)$, and $\pi_R = \left(\sum_{i\in[\eta]} i \cdot \mathsf{F}_{k_i}(m)\right) \cdot G$

$\mathsf{V}(k_{\mathsf{V}}, m, R, \pi_R)$**:** .

1. Compute $w = \sum_{i\in[\eta]\setminus\Delta} (i - \Delta) \cdot \mathsf{F}_{k_i}(m)$
2. Output 1 iff $w \cdot G \stackrel{?}{=} \pi_R - \Delta \cdot R_m$, and 0 otherwise

Theorem 4.2. Algorithm $\mathsf{ssPCF}_{\varepsilon,\mathbb{G},\mathsf{F}}$ is a $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ for any $\mathbb{G}$ and $\varepsilon(\kappa) \in \mathsf{poly}(\kappa)$, assuming that F is a pseudorandom function.

Proof. **Completeness.** Observe that since $(i - \Delta) = 0$ when $i = \Delta$, for any $(R_m, \pi_R) \leftarrow \mathsf{P}(k_{\mathsf{P}} =$

$\{k_i\}_{i\in[\eta]}$) it holds that

$$\begin{aligned} w \cdot G &= \left(\sum_{i\in[\eta]\setminus\Delta} (i-\Delta)\cdot \mathsf{F}_{k_i}(m)\right)\cdot G = \left(\sum_{i\in[\eta]} (i-\Delta)\cdot \mathsf{F}_{k_i}(m)\right)\cdot G \\ &= \left(\sum_{i\in[\eta]} i\cdot \mathsf{F}_{k_i}(m)\right)\cdot G - \Delta\cdot\left(\sum_{i\in[\eta]} \mathsf{F}_{k_i}(m)\right)\cdot G \\ &= \pi_R - \Delta\cdot R_m \end{aligned}$$

which is exactly the condition that induces V to output 1, proving completeness.

Soundness. First, note that any given prover's key k_P allows for η different corresponding k_V values, each of which is equally likely. For a given k_P and message m, we will call a nonce $R \in \mathbb{G}$ 'honest' if $(r_m, \pi_R) = \mathsf{P}(k_\mathsf{P}, m)$ and $r_m \cdot G = R$, and 'malformed' otherwise. A given k_P, m induces a unique honest nonce and proof which we will denote (R_m, π_R^m). Now, consider the following claims:

Claim 4.3. For a given prover's key $k_\mathsf{P} = \{k_i\}_{i\in[\eta]}$, message m, and honest nonce R_m, there is a unique value $\pi_R^m \in \mathbb{G}$ such that $\mathsf{V}(k_\mathsf{V}, m, R^*, \pi_R^m) = 1$.

Claim 4.4. For a given prover's key $k_\mathsf{P} = \{k_i\}_{i\in[\eta]}$, message m, and malformed nonce R^*, each possible choice of verification key $k_\mathsf{V} = (\Delta, \{k_i\}_{i\in[\eta]\setminus\Delta})$ implies a unique value $\pi^*_{R,\Delta} \in \mathbb{G}$ such that $\mathsf{V}(k_\mathsf{V}, m, R^*, \pi^*_{R,\Delta}) = 1$.

Both of the above claims follow from the fact that for any $R^* = R_m + D$, the proof $\pi^*_{R,\Delta} \in \mathbb{G}$ that induces the verifier to accept is given by:

$$\begin{aligned} \pi^*_{R,\Delta} &= w\cdot G + \Delta\cdot R^* \\ &= \pi_R^m - \Delta\cdot R_m + \Delta\cdot R^* \\ &= \pi_R^m + \Delta\cdot D \end{aligned}$$

therefore if $D = 0$ the only accepting proof is π_R^m independent of Δ, otherwise $\pi^*_{R,\Delta}$ is unique for each choice of $\Delta \in [\eta]$.

Corollary 4.5. Denote by $\mathsf{fooled}_\mathcal{B}$ the event that $(m, R^*, \pi_R^*) \leftarrow \mathcal{B}(k_\mathsf{P})$ induces $\mathsf{V}(k_\mathsf{V}, m, R^*, \pi_R^*) = 1$, where $(k_\mathsf{P}, k_\mathsf{V}) \leftarrow \mathsf{Setup}(\kappa)$. Then, $\Pr[\mathsf{fooled}_\mathcal{B}] \le 1/\eta$ for *any* algorithm $\mathcal{B}$.

The corollary follows immediately from Claim 4.4 and the fact that there are η different, equally likely choices of k_V for each k_P.

Corollary 4.5 therefore gives us that for any algorithm $\mathcal{A}$:

$$\begin{aligned} &\Pr\left[R^* \neq R_m \ \wedge\ \mathsf{V}(k_\mathsf{V}, m, R^*, \pi_R^*) = 1 \ :\ \begin{array}{rcl} (k_\mathsf{P}, k_\mathsf{V}) & \leftarrow & \mathsf{Setup}(\kappa) \\ (m, R^*, \pi_R^*) & \leftarrow & \mathcal{A}(k_\mathsf{P}) \\ (r_m, \pi_R) & \leftarrow & \mathsf{P}(k_\mathsf{P}, m) \\ R_m & = & r_m \cdot G \end{array}\right] \\ &= \Pr[\mathsf{fooled}_{\mathcal{A}^*}] \le 1/\eta = \varepsilon(\kappa) \end{aligned}$$

and this satisfies soundness.

Extending the above soundness proof to Simulatable Proof Validation is straightforward—given a dishonest (R_m^*, π_R^*) the algorithm $\mathcal{V}$ simply enumerates all possible $\pi^*_{R,\Delta}$ values (as per Claim 4.4),

and outputs 1 if π_R^* is among them, and 0 otherwise.

Pseudorandom Nonces With Simulatable Proofs. We first define the simulator $\mathcal{S}$ as follows. $\mathcal{S}(k_{\mathsf{V}}, m, R_m; \rho)$:

1. Parse $(\Delta, \{k_i\}_{i\in[\eta]\setminus\Delta})$ from k_{V}, and compute $w = \sum_{i\in[\eta]\setminus\Delta} (i-\Delta)\cdot \mathsf{F}_{k_i}(m)$
2. Output $\pi_R = w \cdot G + \Delta \cdot R_m$

We now show that with the above simulator, $\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})$ and $\mathcal{O}_{\mathcal{S}}(k_{\mathsf{V}})$ are computationally indistinguishable assuming that F is a pseudorandom function. Consider a hybrid oracle $\mathcal{O}_{\mathcal{H}1}(k_{\mathsf{P}}, k_{\mathsf{V}})$ defined as follows:

1. Compute $r_{m,\Delta} = \mathsf{F}_{k_\Delta}(m)$
2. Compute $r_m = -r_{m,\Delta} - \sum_{i\in[\eta]\setminus\Delta} \mathsf{F}_{k_i}(m)$ (equivalent to $\mathsf{P}(k_{\mathsf{P}})$)
3. Set $R_m = r_m \cdot G$, and compute $\pi_R = \mathcal{O}_{\mathcal{S}}(k_{\mathsf{V}})$
4. Output (R_m, π_R)

Claim 4.6. The oracles $\mathcal{O}_{\mathcal{H}1}$ and $\mathcal{O}_{\mathsf{P}}$ are distributed identically, i.e. for any adversary $\mathcal{A}$,

$$\left|\begin{array}{l}\Pr[\mathcal{A}^{\mathcal{O}_{\mathcal{H}1}(k_{\mathsf{P}},k_{\mathsf{V}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)] \\ -\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)]\end{array}\right| = 0$$

The above claim follows from the fact that r_m is computed in exactly the same way by both $\mathcal{O}_{\mathsf{P}}$ and $\mathcal{O}_{\mathcal{H}1}$, and that the computation of π_R by $\mathcal{S}$ (denote it $\pi_R^{\mathcal{S}}$) is equivalent to the way that P computes it (denote it π_R^{P}), in particular:

$$\begin{aligned}\pi_R^{\mathcal{S}} &= w \cdot G + \Delta \cdot R_m \\ &= \sum_{i\in[\eta]\setminus\Delta} (i-\Delta)\cdot \mathsf{F}_{k_i}(m) + \Delta \cdot \sum_{i\in[\eta]} \mathsf{F}_{k_i}(m) \\ &= \sum_{i\in[\eta]} ((i-\Delta)\cdot \mathsf{F}_{k_i}(m) + \Delta \cdot \mathsf{F}_{k_i}(m)) \\ &= \sum_{i\in[\eta]} i \cdot \mathsf{F}_{k_i}(m) \\ &= \pi_R^{\mathsf{P}}\end{aligned}$$

Now we define the next hybrid oracle $\mathcal{O}_{\mathcal{H}2}(k_{\mathsf{V}})$ as follows:

1. Sample $r_{m,\Delta} \leftarrow \mathbb{Z}_q$
2. Compute $r_m = -r_{m,\Delta} - \sum_{i\in[\eta]\setminus\Delta} \mathsf{F}_{k_i}(m)$
3. Set $R_m = r_m \cdot G$, and compute $\pi_R = \mathcal{O}_{\mathcal{S}}(k_{\mathsf{V}})$
4. Output (R_m, π_R)

Claim 4.7. Assuming that F is a pseudorandom function, the oracles $\mathcal{O}_{\mathcal{H}1}$ and $\mathcal{O}_{\mathcal{H}2}$ are computationally indistinguishable. i.e. There is a negligible function negl such that for any efficient adversary $\mathcal{A}$:

$$\left|\begin{array}{l}\Pr[\mathcal{A}^{\mathcal{O}_{\mathcal{H}2}(k_{\mathsf{V}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)] \\ -\Pr[\mathcal{A}^{\mathcal{O}_{\mathcal{H}1}(k_{\mathsf{P}},k_{\mathsf{V}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)]\end{array}\right| \leq \mathsf{negl}(\kappa)$$

The above claim follows from a straightforward perfect reduction to the PRF game. Notice that the difference between $\mathcal{O}_{\mathcal{H}2}$ and $\mathcal{O}_{\mathcal{S}}$ is merely syntactic. From Claims 4.6 and 4.7, we have that

$$\left| \begin{array}{l} \Pr[\mathcal{A}^{\mathcal{O}_{\mathcal{S}}(k_{\mathsf{V}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)] \\ -\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})}(k_{\mathsf{V}}) = 1 \ : \ (k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(\kappa)] \end{array} \right| \leq \mathsf{negl}(\kappa)$$

for some negligible function negl. This completes the proof of the Pseudorandom Nonces With Simulatable Proofs property, and hence proves the theorem. □

4.1 Efficiency

A single P evaluation costs η F invocations, one (fixed base) exponentiation in $\mathbb{G}$, and 2η additions in $\mathbb{Z}_q$—the partial sums produced when computing r_m can be saved and reused to compute π_R. A verification by V costs $\eta - 1$ F invocations, η *small* $\mathbb{Z}_q$ multipications, as many $\mathbb{Z}_q$ additions, and two $\mathbb{G}$ exponentiations (one fixed base and one variable base). The dominant computation cost for say $\varepsilon = 10\%$ (i.e. a 90% chance of a cheater being caught) in most situations will likely be the three $\mathbb{G}$ exponentiations—i.e. roughly the same computation cost profile as generating and then verifying a Schnorr signature.

In terms of bandwidth, π_R consists of a single $\mathbb{G}$ element, i.e. half the size of a Schnorr signature in most common representations.

5 Full Security from Pseudorandom Correlation Functions

Algorithm 5.1 provides a formal description of our protocol. We provide some intuitions here: in a nutshell, the PCF setup provides P and V with keys k_0, k_1 which allow them to non-interactively generate shares $w - v$ of $u\Delta$ in $\mathbb{Z}_{N_{\mathsf{P}}}$. The main challenge we need to solve is to translate these shares into $\mathbb{Z}_q$, where q is the prime used in Schnorr signatures. Roughly, our translation mechanism works as follows: First, the parties translate their correlated shares into shares over $\mathbb{Z}$, removing the modulus by using a technique similar to what is used in the Paillier HSS constructions of [OSY21, RS21]. Essentially, note that shares $w - v$ of $u\Delta$ (modulo N_{P}) are very likely to be shares of $u\Delta$ without any modulus, as long as $|u\Delta| \ll N_{\mathsf{P}}$. However, u is uniformly random in $\mathbb{Z}_{N_{\mathsf{P}}}$, so party P must partially derandomize u to get a correlation for a smaller value u_{lo}. That is, P expresses u in terms of its quotient and residue w.r.t. a second modulus $M \ll \frac{N_{\mathsf{P}}}{\Delta}$, i.e. (u_{hi}, u_{lo}) such that $u_{hi} \cdot M + u_{lo} = u$. Then, P reveals u_{hi} so both parties can locally compute shares $w_{lo} - v$ of $u_{lo}\Delta$. Taking these shares modulo q then gives the desired VOLE correlation, with $r = u_{lo} \bmod q$.

But what if P is malicious and tries to cheat? Note that P could send an incorrect u_{hi} to get shares of $u_{lo}\Delta$ modulo N_{P}, where $u_{lo} = u^*_{lo} + M(u^*_{hi} - u_{hi})$, and (u^*_{hi}, u^*_{lo}) are what these would be if P were honest. We combine three protections to stop P from cheating. First, we have $q \mid M$, so that if the parties get shares of $u_{lo}\Delta$ over $\mathbb{Z}$ (i.e., if u_{lo} is small) then the cheating makes no difference, as $u_{lo} \equiv u^*_{lo} \mod q$ anyway. Second, we add an "integer consistency check" in the exponent, using a different group generated by $g \in (\mathbb{Z}_{N_{\mathsf{V}}})^\times$, where N_{V} is a semiprime so that g's order is unknown. We let P send $g^{u_{lo}}$ and g^v, and V checks that $g^{w_{lo}-v} = g^{u_{lo}\Delta}$. To pass this check, a corrupted P must guess a linear function of Δ that equals the number of times $u_{lo}\Delta \bmod N_{\mathsf{P}}$ wraps around, which is only possible when u_{lo} is very close to a multiple of N_{P}. Finally, V enforces an upper bound on $|u_{hi}|$ so that the only values of u_{lo} near a multiple of N_{P} are near 0 (i.e., the case we already solved by requiring $q \mid M$).

Dealing with a corrupted V is much easier, and only requires setting M to be large enough so that $g^{u_{lo}}$ and $u_{lo} \bmod q$ are statistically independent.

Algorithm 5.1. $\mathsf{modPCF}_{\mathbb{G},\mathbf{PCF}}$. $(\varepsilon,\mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ **from Paillier PCF**

Parameters and constants:

1. A random oracle $H\colon \{0,1\}^* \to \{0,1\}^{n(\kappa)}$.
2. A group $\mathbb{G}$ of odd order q, written additively. We require that $q < 2^{\ell(\kappa)/2-2}$.
3. The number $\eta \geq 2\ln(2)\ell' 2^\kappa$ of possible verifier secrets Δ. We require that $\eta < 2^{\ell(\kappa)/2-2}$ and that q has no factors below η.
4. The Paillier key size, $\ell' \geq \log_2(q\eta) + \kappa + \ell(\kappa) + 2$.

$\mathsf{Setup}(\kappa)$**:**

1. Sample $(N_\mathsf{P}, \varphi_\mathsf{P}) \leftarrow \mathsf{RSA.Gen}(1^{\kappa'})$, where $\kappa' > \kappa$ is chosen so that $\ell(\kappa') = \ell'$.
2. Sample $(N_\mathsf{V}, \varphi_\mathsf{V}) \leftarrow \mathsf{RSA.GenSafe}(1^\kappa)$ and $g \leftarrow (\mathbb{Z}_{N_\mathsf{V}})^\times$.
3. Setup the smaller modulus $M := qN_\mathsf{V}$.
4. Sample the verifier secret index $\Delta \leftarrow [-\frac{\eta}{2}, \frac{\eta}{2}) \cap \mathbb{Z}$.
5. Set up the PCF: $(k_0, k_1) \leftarrow \mathsf{PCF.Gen}(1^\kappa, N_\mathsf{P}, \varphi_\mathsf{P}, \bot, \Delta)$.
6. Output $k_\mathsf{P} = (k_0, N_\mathsf{P}, N_\mathsf{V}, M, g)$, and $k_\mathsf{V} = (k_1, N_\mathsf{P}, N_\mathsf{V}, M, g, \Delta)$.

$\mathsf{P}((k_0, N_\mathsf{P}, N_\mathsf{V}, M, g), m)$**:**

1. Compute $(u, v) := \mathsf{PCF.Eval}(0, k_0, H(m))$.
2. Find $u_{hi} := \lfloor \frac{u}{M} \rceil$, $u_{lo} := u - Mu_{hi} = u \bmod M$.
3. Compute the proof $s := g^{u_{lo}}$ and $t := g^v$.
4. Output $r_m := u \bmod q$, and $\pi_R := (u_{hi}, s, t, v \cdot G)$.

$\mathsf{V}((k_1, N_\mathsf{P}, N_\mathsf{V}, M, g, \Delta), m, R_m, \pi_R)$**:**

1. Compute $w := \mathsf{PCF.Eval}(1, k_1, H(m))$.
2. Let $(u_{hi}, s, t, V) := \pi_R$.
3. Correct w as $w_{lo} := (w - Mu_{hi}\Delta) \bmod N_\mathsf{P}$.
4. Output 1 if all of the following checks pass. Otherwise output 0.
   (a) $|u_{hi}| \leq \frac{N_\mathsf{P}}{2M}$
   (b) $g^{w_{lo}} \stackrel{?}{=} \pm t\, s^\Delta$
   (c) $w_{lo} \cdot G \stackrel{?}{=} V + \Delta \cdot R_m$

5.1 Efficiency

Notice that t and V are only used to check an equality. This allows an optimization where P sends $h_{tV} = H'(|t|, V)$ instead of t and V, and V checks that $h_{tV} \stackrel{?}{=} H'(|g^{w_{lo}} s^{-\Delta}|, w_{lo} \cdot G - \Delta \cdot R_m)$. This is still sound, because in the soundness proof we can extract $|t| = \pm t$ and V from h_{tV}.

We now give a summary of the complexity of the protocol with the hash optimization, for both communication and computation. The proof π_R that P sends contains u_{hi} for $\log_2(N_\mathsf{P}/M)$ bits, s for $\log_2(N_\mathsf{V})$ bits, and h_{tV} for 2κ bits. The total is $C = \ell' - \log_2(q) + 2\kappa$ bits, because $M = qN_\mathsf{V}$. Set $\eta = 2\ln(2)\ell' 2^\kappa$. [2] We get that $\ell' - \log_2(\ell') \geq \log_2(q) + 2\kappa + \ell(\kappa) + 3 + \log_2(\ln(2))$. Since $A - \log_2(A) \geq B$ can be solved by $A = B + \log_2(2B)$ when $B \geq 2$, and since $\log_2(\ln(2)) < 0$, we

[2] η really should be rounded up to an integer, but this makes almost no difference.

can set

$$\ell' \geq \log_2(q) + 2\kappa + \ell(\kappa) + 4 + \log_2(\log_2(q) + 2\kappa + \ell(\kappa) + 3).$$

Since $q \ll \ell(\kappa)$ for our application, we have that $\ell(\kappa)$ is bigger than all the other terms put together, which let's us simplify by choosing a slightly bigger ℓ'. We can compute the total communication cost using ℓ'.

$$\ell' = \log_2(q) + 2\kappa + \ell(\kappa) + \log_2(\ell(\kappa)) + 5$$
$$C = \ell(\kappa) + \log_2(\ell(\kappa)) + 5 + 4\kappa$$

For computation, P performs two exponentiations in $(\mathbb{Z}_{N_\mathsf{P}^2})^\times$ (for the PCF) and two in $(\mathbb{Z}_{N_\mathsf{V}})^\times$ (for the check), while V computes one exponentiation in $(\mathbb{Z}_{N_\mathsf{P}^2})^\times$ and two in $(\mathbb{Z}_{N_\mathsf{V}})^\times$. The CRT optimization can be applied to P's exponentiations in $(\mathbb{Z}_{N_\mathsf{P}^2})^\times$ and to V's exponentiations in $(\mathbb{Z}_{N_\mathsf{V}})^\times$, since they know the factorizations of their respective moduli.

We illustrate our scheme's efficiency with concrete parameters for the $\kappa = 128$-bit security level. To evaluate $\ell(\kappa)$, we follow NIST's recommendations for RSA key sizes, which is 3072 bits for 128-bit security [Bar20, Table 2]. Let $q \approx 2^{252}$ be the prime order subgroup used for the Ed25519 signature scheme [BDL+11]. We then get $\ell' = 3597$ bits (rounding up), with a total communication cost of $C = 3601$ bits, or 451 bytes. When used with Protocol 3.2, each party must send an additional curve point and element of $\mathbb{Z}_q$, so the per-party communication cost is 514 bytes.

A Note on Setup Efficiency. As written, the setup functionality samples N_P and N_V itself, which appears to require distributed sampling of an RSA modulus. However, observe that the factorization of the moduli need not be hidden from both parties simultaneously—N_P's factorization is known by the prover, and it is fine to give N_V's to the verifier. This opens the door for protocols where the appropriate party simply samples the RSA modulus itself, and proves that it is well-formed (as is common in widely deployed Paillier-based Threshold ECDSA protocols [CGG+20]). As the focus is on the signing protocols in this work, rather than the instantiation of the setup functionality, we leave optimizing the setup phase for future work.

5.2 Implementation

We made a prototype implementation using the GMP library. When running on a single thread of a laptop (with a Ryzen 7 5800H processor), the prover takes 56ms while the verifier takes 130ms, leading to a per-party computation time of 188ms per ed25519 signature. These estimates indicate that our construction is considerably more computationally efficient than prior work based on Bulletproofs: from [NRSW20, Table 1], their prover runs in 943ms and the verifier between 10-50ms depending on the batch size, making our protocol approximately 5 times faster in total. Of course the benchmarks are not directly comparable as they are measured in different environments, but they provide an overall picture of the efficiency comparison. We do not estimate our construction to be computationally lighter than prior work based on garbled circuits [GKMN21], which is however much heavier on bandwidth consumption than this work.

Theorem 5.2. Algorithm $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$ is a $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$, assuming that PCF is a pseudorandom correlation function for a VOLE reverse sampleable correlation.

The proof is given in the following sections, split between the three properties.

5.3 Completeness

Theorem 5.3. $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$ satisfies completeness, as defined in $(\varepsilon,\mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$. More specifically, assume that at most Q unique queries to H are made (either by the adversary or P). Then the chance of failure is at most $Q(\frac{1}{2}+\eta^{-1})2^{-\kappa} + \mathsf{Adv}_{\mathsf{PCF.Pseudorandom}}$.

Proof. We will ignore the output (r_m, π_R) from $\mathcal{O}_{\mathsf{compl}}$, instead proving that $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$ will remain complete even when $\mathcal{A}$ is given k_{P}. In the worst case, every message m input to H will be given to $\mathcal{O}_{\mathsf{compl}}$, and because the time of abort does not matter, we can assume m is given to $\mathcal{O}_{\mathsf{compl}}$ at the same time as $H(m)$ is sampled. Therefore, we compute an upper bound A on the probability of abort for $\mathcal{O}_{\mathsf{compl}}(m)$, assuming that that $H(m)$ is freshly random. The overall game then aborts with probability at most QA, by the union bound.

First, let's check the bounds on u_{lo} and u_{hi}. We have $|u_{lo}| = |u_i \bmod M| \le \frac{M}{2}$, because we are using a symmetric modulus operation. Also,

$$|u_{hi}| = \left|\left\lfloor \frac{u}{M} \right\rceil\right| \le \left\lfloor \frac{N_{\mathsf{P}}}{2M} \right\rceil \le \left\lfloor \frac{N_{\mathsf{P}}}{2M} \right\rfloor + 1,$$

so check (a) will pass except for when $|u_{hi}|$ takes the maximal value $\left\lfloor \frac{N_{\mathsf{P}}}{2M} \right\rfloor + 1$. If it fails,

$$|u| \ge \left\lfloor \frac{N_{\mathsf{P}}}{2M} \right\rfloor M + \frac{M}{2} > \frac{N_{\mathsf{P}}}{2M} M - M + \frac{M}{2} = \frac{N_{\mathsf{P}} - M}{2}.$$

Therefore, u would be in a set of size at most $2\left(\frac{N_{\mathsf{P}}}{2} - \frac{N_{\mathsf{P}}-M}{2}\right) = M$, which has probability at most

$$\frac{M}{N_{\mathsf{P}}} < q2^{\ell(\kappa)-\ell'+2} < q2^{-\log_2(q\eta)-\kappa} = \eta^{-1}2^{-\kappa}, \tag{1}$$

by the pseudorandomness of PCF.

Next, for checks (b) and (c) we need the following lemma.

Lemma 5.4. ([RS21, Lemma 19])**.** For any $N_{\mathsf{P}} \in \mathbb{Z}^+$, $x \in \mathbb{Z}$, and uniformly random $r \in \mathbb{Z}_{N_{\mathsf{P}}}$, we have

$$\Pr[x = (r+x) \bmod N_{\mathsf{P}} - r \bmod N_{\mathsf{P}}] = \max\left(1 - \frac{|x|}{N_{\mathsf{P}}}, 0\right).$$

We will use this lemma on the correlation between w_{lo} and v. Notice that

$$w_{lo} - v \equiv w - v - u_{hi}M\Delta \equiv u\Delta - u_{hi}M\Delta \equiv u_{lo}\Delta \mod N_{\mathsf{P}},$$

and $|u_{lo}\Delta| \le M\frac{\eta}{2}$ is small compared to N_{P}. By the pseudorandomness of PCF, v will be uniform. As both v and w_{lo} are reduced modulo N_{P}, by Lemma 5.4 we can remove the modulus to get $w_{lo} - v = u_{lo}\Delta$, except with probability $\frac{M\eta}{2N_{\mathsf{P}}} \le 2^{-\kappa-1}$. Checks (b) and (c) then must be satisfied, as they are $g^{w_{lo}} \stackrel{?}{=} g^{v+\bar{u}\Delta}$ and $w_{lo} \cdot G \stackrel{?}{=} (v+\Delta u)\cdot G$ respectively. For check (c), notice that $u = u_{lo} + Mu_{hi} \equiv u \mod q$ because $q \mid M$.

Therefore, we have the bound $A = (\frac{1}{2}+\eta^{-1})2^{-\kappa}$, which after multiplying by Q matches the theorem statement. □

5.4 Soundness

Theorem 5.5. $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$ satisfies ε**-Soundness**, for

$$\varepsilon = Q(2^{-\kappa} + 5\eta^{-1}) + 2\eta^{-1} + 2^{-\ell(\kappa)/2+3} < Q(2^{-\kappa} + 6\eta^{-1}),$$

plus the advantages against the underlying hardness assumptions: $\Theta(\mathsf{Adv}_{\mathsf{StrongRSA}}) + \mathsf{Adv}_{\mathsf{PCF.Security}} + \mathsf{Adv}_{\mathsf{PCF.Pseudorandom}}$. Here, we assume that at most Q unique queries to H are made.

Proof. Our proof overall is structured as a hybrid argument. The first hybrid, $\mathcal{H}_1$, is a straightforward change from PCF evaluation to reverse sampling, based on the security of PCF. First, change k_0 to be sampled with $\mathsf{PCF}.\mathcal{S}_0(N_\mathsf{P}, \phi, \perp)$, and remove the call to $\mathsf{PCF.Gen}$. Let $(u, v) = \mathsf{PCF.Eval}(0, k_0, H(m))$ be the correlation that would be computed by an honest prover. Then change V to equivalently compute w as $v + u\Delta$, instead of evaluating the PCF with k_1 (which is no longer defined). Define $u_{lo} = u - Mu_{hi}$, where u_{hi} is the value sent by the adversary. Then

$$w_{lo} = (w - Mu_{hi}\Delta) \bmod N_\mathsf{P} = (v + u_{lo}\Delta) \bmod N_\mathsf{P}.$$

Let

$$\mathcal{D} = \{\Delta \in [-\frac{\eta}{2}, \frac{\eta}{2}) \mid g^{(v+u_{lo}\Delta) \bmod N_\mathsf{P}} = \pm t\, s^\Delta\},$$

and replace check (b) with the equivalent check $\Delta \in \mathcal{D}$.

Claims. With the notation defined in $\mathcal{H}_1$, we can now analyze the consistency check in $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$. The bulk of our proof will be about finding the size and structure of $\mathcal{D}$, as this is what determines whether the attacker can succeed in lying about R_m. We first give a series of claims that should hold when the consistency check passes, except with negligible probability. More precisely, a claim stating X means that $\Pr[\text{consistency check passes} \wedge \neg X]$ is negligible. The final claim will be that the scheme is sound, i.e., R_m must take its honest value. We will justify these claims with hybrids, changing the protocol until all of the claims are true unconditionally.

Claim 5.6. In addition to the actually checked value Δ, $\mathcal{D}$ contains a second element $\Delta' \neq \Delta$.

Claim 5.7. From Δ, Δ', we can efficiently find nonzero integers a, b such that $g^a = \pm s^b$, where $|a| \leq N_\mathsf{P}$ and $|b| \leq \eta$.

Claim 5.8. From Δ, Δ', we can efficiently extract integers $\bar{u}_{lo} = a/b$ and $\bar{v}$ such that $s = \pm g^{\bar{u}_{lo}}$ and $t = \pm g^{\bar{v}}$.

Claim 5.9. g^2 generates the subgroup of perfect squares in $(\mathbb{Z}_{N_\mathsf{V}})^\times$.

Claim 5.10. $\bar{u}_{lo} \equiv u_{lo} \mod N_\mathsf{P}$ and $\bar{v} \equiv v \mod N_\mathsf{P}$.

Claim 5.11. $\mathcal{D} = \left\{\Delta \in [-\frac{\eta}{2}, \frac{\eta}{2}) \mid |\bar{v} + \bar{u}_{lo}\Delta| \leq \frac{N_\mathsf{P}}{2}\right\}$,

Claim 5.12. We can assume without loss of generality that $\Delta' = \Delta \pm 1$.

Claim 5.13. $\bar{u}_{lo} = u_{lo}$.

Claim 5.14. $R_m = r_m \cdot G$.

Now we present the remaining hybrids to show that these claims hold.

$\mathcal{H}_2$. Change the verifier to abort if $|\mathcal{D}| < 2$, and otherwise sample $\mathcal{D}'$ as a uniformly random element of $\mathcal{D} \setminus \{\Delta\}$. This makes Claim 5.6 hold trivially. This change is only distinguishable when $|\mathcal{D}| \leq 1$, in which case the adversary has to guess that Δ is the unique value in $\mathcal{D}$, when Δ is uniformly random over η possibilities. Therefore, passing check (b) has probability at most η^{-1} in $\mathcal{H}_1$, and zero probability in $\mathcal{H}_2$. This bounds the advantage at η^{-1}.

Claim 5.7 also holds in this hybrid. Because $\Delta, \Delta' \in \mathcal{D}$, we have $g^{(v+u_{lo}\Delta) \bmod N_{\mathsf{P}}} = \pm t\, s^{\Delta}$, and the same for Δ'. Taking the ratio between the two equations gives $g^a = \pm s^b$, where

$$a = (v + u_{lo}\Delta') \bmod N_{\mathsf{P}} - (v + u_{lo}\Delta) \bmod N_{\mathsf{P}}, \qquad b = \Delta' - \Delta.$$

The bounds are $|a| \leq N_{\mathsf{P}}$ and $|b| \leq \eta$, because a is the difference of two values below N_{P} and b is the difference of two values below η.

Note that this hybrid is not efficiently computable. We will fix this in a later hybrid, but until then we will use only statistical security (like here), or give a reduction that avoids this issue.

$\mathcal{H}_3$. Add checks requiring that $b \mid a$ and $s = \pm g^{\bar{u}_{lo}}$. Here, we let

$$\bar{u}_{lo} = \frac{a}{b} \quad \text{and} \quad \bar{v} = (v + u_{lo}\Delta) \bmod N_{\mathsf{P}} - \bar{u}_{lo}\Delta. \tag{2}$$

That is, $\bar{u}_{lo}$ and $\bar{v}$ are defined to be the slope and intercept of the line through $(\Delta, (v + u_{lo}\Delta) \bmod N_{\mathsf{P}})$ and $(\Delta', (v+u_{lo}\Delta') \bmod N_{\mathsf{P}})$. Then $s = \pm g^{\bar{u}_{lo}}$ implies that $t = \pm g^{\bar{v}}$. Therefore, Claim 5.8 holds as of this hybrid.

The indistinguishability of this hybrid follows from a reduction to Strong RSA. This reduction is deferred to Appx. B.

$\mathcal{H}_4$. Require that g^2 generates the subgroup of perfect squares in $(\mathbb{Z}_{N_{\mathsf{V}}})^\times$, which can be checked efficiently using that we know a prime factorization $p' \cdot q'$ of N_{V}, and that p' and q' are safe primes. This makes Claim 5.9 hold trivially. The order of the group of perfect squares is $(\frac{p'-1}{2}) \cdot (\frac{q'-1}{2})$, which is a semiprime. Therefore, g^2 generates it with probability $\left(1 - \frac{2}{p'-1}\right)\left(1 - \frac{2}{q'-1}\right) \geq 1 - 2^{-\ell(\kappa)/2+3}$, so this change is indistinguishable.

Proof of Claim 5.10. Using Claim 5.8, we now have $\Delta^* \in \mathcal{D}$ if and only if

$$g^{(v+u_{lo}\Delta^*) \bmod N_{\mathsf{P}}} = \pm g^{\bar{v}+\bar{u}_{lo}\Delta^*}$$

Since g^2 generates the perfect squares of $(\mathbb{Z}_{N_{\mathsf{V}}})^\times$, which has odd order $\varphi_{\mathsf{V}}/4$, squaring both sides gives

$$(v + u_{lo}\Delta^*) \bmod N_{\mathsf{P}} \equiv \bar{v} + \bar{u}_{lo}\Delta^* \mod \varphi_{\mathsf{V}}/4. \tag{3}$$

Additionally, this last equation holds without the modulus for $\Delta^* = \Delta$ and $\Delta^* = \Delta'$, because $\bar{u}_{lo}$ and $\bar{v}$ come from interpolating a line through these two points over $\mathbb{Z}$ (see Equation (2)). Taking both sides modulo N_{P} gives

$$\begin{aligned} v + u_{lo}\Delta &\equiv \bar{v} + \bar{u}_{lo}\Delta && \mod N_{\mathsf{P}} \\ v + u_{lo}\Delta' &\equiv \bar{v} + \bar{u}_{lo}\Delta' && \mod N_{\mathsf{P}} \\ (u_{lo} - \bar{u}_{lo})(\Delta' - \Delta) &\equiv 0 && \mod N_{\mathsf{P}}. \end{aligned}$$

Since $\Delta' - \Delta \neq 0$ is smaller than $\eta < 2^{\ell(\kappa)/2-2} < 2^{\ell'/2-1} < \min(p, q)$, $\Delta' - \Delta$ is coprime to N_{P}. Therefore, $u_{lo} \equiv \bar{u}_{lo} \mod N_{\mathsf{P}}$, and so $v \equiv \bar{v} \mod N_{\mathsf{P}}$ as well.

Proof of Claim 5.11. We can now substitute $\bar{u}_{lo}, \bar{v}$ for u_{lo}, v in Equation (3), simplifying it to get

$$\begin{aligned}(\bar{v} + \bar{u}_{lo}\Delta^*) \bmod N_\mathsf{P} - \bar{v} - \bar{u}_{lo}\Delta^* &\equiv 0 \mod \varphi_\mathsf{V}/4 \\ -N_\mathsf{P}\left\lfloor \frac{\bar{v} + \bar{u}_{lo}\Delta^*}{N} \right\rceil_\mathsf{P} &\equiv 0 \mod \varphi_\mathsf{V}/4 \\ \left\lfloor \frac{\bar{v} + \bar{u}_{lo}\Delta^*}{N} \right\rceil_\mathsf{P} &\equiv 0 \mod \varphi_\mathsf{V}/4.\end{aligned}$$

We used that N_P is coprime to $\varphi_\mathsf{V}/4$ because they are both semiprimes and N_P's prime factors are much larger than $\varphi_\mathsf{V}/4$. Recall that $(v + u_{lo}\Delta) \bmod N_\mathsf{P} - \bar{v} - \bar{u}_{lo}\Delta$, so we can upper bound the value being rounded:

$$\left|\frac{\bar{v} + \bar{u}_{lo}\Delta^*}{N}\right\rceil_\mathsf{P}\Bigg| = \left|\frac{(v + u_{lo}\Delta^*) \bmod N_\mathsf{P} + \bar{u}_{lo}(\Delta^* - \Delta)}{N}\right\rceil_\mathsf{P}\Bigg| \leq \frac{1}{2} + \eta < \varphi_\mathsf{V}/4,$$

since $\bar{u}_{lo} \leq |a| \leq N_\mathsf{P}$. Therefore, we can remove the modulus, showing that $\Delta^* \in \mathcal{D}$ implies $\left\lfloor \frac{\bar{v}+\bar{u}_{lo}\Delta}{N_\mathsf{P}} \right\rceil = 0$, or equivalently $|\bar{v} + \bar{u}_{lo}\Delta| \leq \frac{N_\mathsf{P}}{2}$. Conversely, if $|\bar{v} + \bar{u}_{lo}\Delta| \leq \frac{N_\mathsf{P}}{2}$ then $g^{(v+u_{lo}\Delta^*) \bmod N_\mathsf{P}} = g^{(\bar{v}+\bar{u}_{lo}\Delta^*) \bmod N_\mathsf{P}} = g^{\bar{v}+\bar{u}_{lo}\Delta^*}$, so $\Delta^* \in \mathcal{D}$.

$\mathcal{H}_5$. Instead of sampling $\mathcal{D}'$ as a uniformly random element of $\mathcal{D}\backslash\{\Delta\}$, just check the two neighboring elements $\Delta \pm 1$. By Claim 5.11, $\mathcal{D}$ is an interval, and by Claim 5.6 it contains at least two elements. Therefore, any element $\Delta \in \mathcal{D}$ has a neighboring element $\Delta' \in \mathcal{D}$, so this change is indistinguishable. All the preceding claims merely require that Δ and Δ' be distinct elements of $\mathcal{D}$, so they will still hold with this choice of Δ'. Therefore, Claim 5.12 holds.

Note that this hybrid is now efficiently computable, by removing the sampling introduced in $\mathcal{H}_2$.

$\mathcal{H}_6$. In this hybrid, add a new check requiring that $\bar{u}_{lo} = u_{lo}$ so that Claim 5.13 is trivially true. To notice this change, the adversary must find $\bar{u}_{lo} \neq u_{lo}$ while passing the check $\Delta \in \mathcal{D}$. They have probability at most $\frac{|\mathcal{D}|}{\eta}$ of passing, so we want to upper bound $|\mathcal{D}|$. By Claim 5.11, Δ must be in a width $\frac{N_\mathsf{P}}{\bar{u}_{lo}}$ interval, which can contain at most $\frac{N_\mathsf{P}}{\bar{u}_{lo}} + 1$ integers.

The adversary is limited in how it can pick $\bar{u}_{lo}$. By Claim 5.10, $\bar{u}_{lo} \equiv u_{lo} \mod N_\mathsf{P}$, and we defined $u_{lo} = u - Mu_{hi}$. We also have $|u_{hi}| \leq \frac{N_\mathsf{P}}{2M}$ from check (a). Therefore, $|u_{lo}| \leq |u| + M|u_{hi}| \leq |u| + \frac{N_\mathsf{P}}{2}$. The adversary must shift u_{lo} by at least N_P in either direction to get $\bar{u}_{lo} \neq u_{lo}$, so $|\bar{u}_{lo}| \geq N_\mathsf{P} - |u_{lo}| \geq \frac{N_\mathsf{P}}{2} - |u|$.

The only control that the adversary exercises over u is through the message m, as u is the honest output from the PCF. For the ith random oracle query $H(m_i)$, let P_{m_i} be the probability that the adversary selects m_i and passes the check. Then $\eta P_{m_i} \leq \mathbb{E}\left[\frac{N_\mathsf{P}}{\frac{N_\mathsf{P}}{2} - |u_{m_i}|} + 1\right]$, where $(u_{m_i}, v_{m_i}) = \mathsf{PCF.Eval}(0, k_0, H(m_i))$. By the pseudorandomness of PCF, $|u_{m_i}|$ is indistinguish-

able from a uniformly random integer in $[0, \frac{N_\mathsf{P}-1}{2}]$. Therefore,

$$\begin{aligned}
\eta P_{m_i} &\le 1 + \frac{2}{N_\mathsf{P}+1} \sum_{j=0}^{\frac{N_\mathsf{P}-1}{2}} \frac{N_\mathsf{P}}{\frac{N_\mathsf{P}}{2} - j} \\
&= 1 + \frac{4N_\mathsf{P}}{N_\mathsf{P}+1} + \frac{2}{N_\mathsf{P}+1} \sum_{x=0}^{\frac{N_\mathsf{P}-3}{2}} \frac{N_\mathsf{P}}{\frac{N_\mathsf{P}}{2} - x} \\
&\le 5 + \int_0^{\frac{N_\mathsf{P}-1}{2}} \frac{2}{\frac{N_\mathsf{P}}{2} - x} dx \\
&= 5 + 2\ln(N_\mathsf{P}/2) - 2\ln(1/2) = 5 + 2\ln(N_\mathsf{P}).
\end{aligned}$$

The second inequality uses the summation as a left Riemann sum; this gives a lower bound on the integral because $\frac{N_\mathsf{P}}{\frac{N_\mathsf{P}}{2}-x}$ is an increasing function. Finally, apply the union bound over all queries i to upper bound the distinguisher's advantage at $Q\frac{5+2\ln(N_\mathsf{P})}{\eta} \le Q(5\eta^{-1} + 2^{-\kappa})$, where Q is the total number of queries to H.

$\mathcal{H}_7$. In this final hybrid, we require that Claim 5.14 be true. That is, we make V fail whenever R_m is not its honest value $r_m \cdot G = u \cdot G$. This makes the soundness game trivially impossible for the adversary to attack. We will bound the chance that the adversary passes all the previous checks, but fails this final check.

Let $A = R_m - u \cdot G$ and $B = V - \bar{v} \cdot G$. Then check (c) becomes

$$\begin{aligned}
B + \bar{v} \cdot G + \Delta \cdot A + \Delta u \cdot G &= w_{lo} \cdot G \\
B + \Delta \cdot A &= ((\bar{v} + \bar{u}_{lo}\Delta) \bmod N_\mathsf{P} - \bar{v} - \Delta\bar{u}_{lo}) \cdot G \\
B + \Delta \cdot A &= -N_\mathsf{P} \left\lfloor \frac{\bar{v} + \bar{u}_{lo}\Delta}{N} \right\rceil_\mathsf{P} \cdot G = 0,
\end{aligned}$$

because $\bar{u}_{lo} = u_{lo} = u - Mu_{hi} \equiv u \mod q$ since $q \mid M$. Next, since q has no prime factors below η, $\Delta \cdot A$ will uniquely identify Δ when $A \neq 0$. When $A = 0$, $R_m = r_m \cdot G$ and so the newly added check will pass. Therefore, any adversary distinguishing $\mathcal{H}_7$ from $\mathcal{H}_6$ has advantage at most η^{-1}. □

5.5 Pseudorandom Nonces

Theorem 5.15. $\mathsf{modPCF}_{\mathbb{G},\mathsf{PCF}}$ satisfies the pseudorandomness property defined in $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$. Assuming that at most Q unique queries to H are made, by either the adversary or the oracles, the distinguisher advantage is bounded by

$$Q((1+\eta^{-1})2^{-\kappa-1} + 2^{-\ell(\kappa)/2+2}) < Q2^{-\kappa}$$

plus the advantage against PCF, which is $\mathsf{Adv}_{\mathsf{PCF.Security}} + \mathsf{Adv}_{\mathsf{PCF.Pseudorandom}}$.

Proof. Let the simulator $\mathcal{S}(k_\mathsf{V}, m, R_m)$ work as follows.

1. Compute the verifier's share $w := \mathsf{PCF.Eval}(1, k_1, H(m))$.
2. Sample $u \leftarrow [-\frac{N_{\text{round}}}{2}, \frac{N_{\text{round}}}{2}) \cap \mathbb{Z}$, where $N_{\text{round}} = M\left\lfloor \frac{N_\mathsf{P}}{M} \right\rceil$ is the multiple of M nearest to N_P.
3. Compute u_{hi}, u_{lo}, and s as in the prover.

4. Compute $w_{lo} := (w - Mu_{hi}\Delta) \bmod N_{\mathsf{P}}$ as in the verifier, then set $t := g^{w_{lo}} s^{-\Delta}$ and $V := w_{lo} \cdot G - \Delta \cdot R_m$.
5. Output $\pi_R := (u_{hi}, s, t, V)$.

We need to show that the simulation $\mathcal{O}_{\mathcal{S}}(k_{\mathsf{V}})$ is indistinguishable from the real prover $\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})$. To do so, we give a hybrid proof, going from the real prover to the simulation. We assume that the queries to H are exactly the queries to $\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})$, as the adversary can always be made to query $\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})$ at the extra locations. During the proof, we focus on only a single execution of $\mathcal{O}_{\mathsf{P}}(k_{\mathsf{P}})$ and a single oracle query $H(m)$, so all statistical bounds must be multiplied by Q in the overall advantage.

$\mathcal{H}_1$. By the security of the PCF, we can replace the prover's PCF evaluations $(u, v) \leftarrow \mathsf{PCF.Eval}(0, k_0, H(m))$ with computing the verifier's share w and sampling $(u, v) \leftarrow \mathsf{RSample}(1^\kappa, \Delta, 1, w)$. More concretely, $\mathsf{RSample}$ chooses u uniformly in $[-\frac{N_{\mathsf{P}}}{2}, \frac{N_{\mathsf{P}}}{2}) \cap \mathbb{Z}$ and finds the corresponding $v := (w - u\Delta) \bmod N_{\mathsf{P}}$. We also have to change k_1 to be sampled with $\mathsf{PCF}.\mathcal{S}_1(N_{\mathsf{P}}, \phi, \Delta)$, and remove k_0 and the call to $\mathsf{PCF.Gen}$.

$\mathcal{H}_2$. Sample $u \leftarrow [-\frac{N_{\text{round}}}{2}, \frac{N_{\text{round}}}{2}) \cap \mathbb{Z}$ instead. This has advantage at most

$$\frac{|N_{\mathsf{P}} - N_{\text{round}}|}{N_{\mathsf{P}}} = \frac{|N_{\mathsf{P}} \bmod M|}{N_{\mathsf{P}}} \leq \frac{M}{2N} < \eta^{-1} 2^{-\kappa-1},$$

by Equation (1).

$\mathcal{H}_3$. Instead of computing $t := g^v$ and $V := v \cdot G$, calculate them based on the consistency checks as $t := g^{w_{lo}} s^{-\Delta}$ and $V := w_{lo} \cdot G - \Delta \cdot R_m$. These formulas are equivalent as long as $w_{lo} = v + u_{lo}\Delta$ (without any modulus). As argued in Theorem 5.3, this holds except with probability $2^{-\kappa-1} + \mathsf{Adv}_{\mathsf{PCF.Pseudorandom}}$.

$\mathcal{H}_4$. Currently $u_{lo} = u \bmod M$ is uniformly random in $\mathbb{Z}_M$. Add a bad event asserting that $u_{lo} \in [-\frac{M_{\varphi_{\mathsf{V}}}}{2}, \frac{M_{\varphi_{\mathsf{V}}}}{2})$, where $M_{\varphi_{\mathsf{V}}} = q\varphi_{\mathsf{V}}$. The bad event occurs with probability

$$\frac{M - M_{\varphi_{\mathsf{V}}}}{M} = \frac{N_{\mathsf{V}} - \varphi_{\mathsf{V}}}{N_{\mathsf{V}}} = \frac{1}{p'} + \frac{1}{q'} - \frac{1}{N_{\mathsf{V}}} < 2^{-\ell(\kappa)/2+2},$$

where $p' \cdot q'$ is the prime factorization of N_{V}.

Assuming that this bad event does not occur, we can divide u_{lo} into two components using the Chinese remainder theorem, which applies because q and φ_{V}) are coprime. That is, φ_{V} has prime factoriztaion $4 \cdot (\frac{p'-1}{2}) \cdot (\frac{q'-1}{2})$ because p' and q' are safe primes, and q is an odd number below $2^{\ell(\kappa)/2-1} < \min(\frac{p'-1}{2}, \frac{q'-1}{2})$. Therefore, the possible values of u_{lo} are in bijection with pairs $(u_q, u_{\varphi_{\mathsf{V}}})$, where $u_q = u_{lo} \bmod q$ and $u_{\varphi_{\mathsf{V}}} = u_{lo} \bmod \varphi_{\mathsf{V}}$. The only places u_{lo} is used are in $s := g^{u_{lo}}$ and $r_m := u \mod q$. Instead, compute these with $u_{\varphi_{\mathsf{V}}}$ and u_q, respectively.

$\mathcal{H}_5$. Instead of computing $u_q = u_{lo} \bmod q$, sample $u_q \leftarrow \mathbb{Z}_q$ (while leaving u_{lo} unchanged). This is indistinguishable because u_q is uniformly random in both hybrids, and because u and u_{lo} are not used directly.

$\mathcal{H}_6$. Factor out the computation of R_m into $\mathcal{O}_{\mathcal{S}}$. The simulator $\mathcal{S}$ is now exactly the modified prover, so we are now at the simulated distribution. □

+

Acknowledgments. The research described in this paper received funding from: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-0165-00107B (C3PO).

Bibliography

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In *CRYPTO 2021, Part I*, August 2021.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC 2007*, February 2007.

[ANO+22] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 2554–2572. IEEE, 2022.

[ANT+20] Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In *ACM CCS 2020*, November 2020.

[Bar97] George Barwood. Digital signatures using elliptic curves, message 32f519ad. 19609226@news.dial.pipex.com posted to sci.crypt., 1997.

[Bar20] Elaine Barker. Recommendation for key management: Part 1 – general. Technical Report NIST Special Publication (SP) 800-57, Part 1, Rev. 5, National Institute of Standards and Technology, Gaithersburg, MD, 2020.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, May 2018.

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, November 2020.

[BCLK17] Marcus Brandenburger, Christian Cachin, Matthias Lorenz, and Rüdiger Kapitza. Rollback and forking detection for trusted execution environments using lightweight collective memory. In *DSN 2017*, 2017.

[BD22] Luís T. A. N. Brandão and Michael Davidson. NISTIR 8214B, Notes on Threshold EdDSA/Schnorr Signatures. https://csrc.nist.gov/publications/detail/nistir/8214b/draft, 2022.

[BDL+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *CHES 2011*, September / October 2011.

[BST21] Charlotte Bonte, Nigel P. Smart, and Titouan Tanguy. Thresholdizing hasheddsa: MPC to the rescue. *Int. J. Inf. Sec.*, 20(6):879–894, 2021.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *ACM CCS 2020*, November 2020.

[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*, October 2021.

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, May 2019.

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO'87*, August 1988.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, December 2002.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO 2016, Part III*, August 2016.

[DOK+20] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS 2020, Part II*, September 2020.

[EZJ+14] Adam Everspaugh, Yan Zhai, Robert Jellinek, Thomas Ristenpart, and Michael M. Swift. Not-so-random numbers in virtualized linux and the whirlwind RNG. In *2014 IEEE Symposium on Security and Privacy*, May 2014.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, August 2005.

[GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *ACM CCS 2018*, October 2018.

[Gil99] Niv Gilboa. Two party RSA key generation. In *CRYPTO'99*, August 1999.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, (1), January 2007.

[GKMN21] François Garillot, Yashvanth Kondi, Payman Mohassel, and Valeria Nikolaenko. Threshold Schnorr with stateless deterministic signing from standard assumptions. In *CRYPTO 2021, Part I*, August 2021.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, August 1987.

[Hen22] Nadia Heninger. RSA, DH and DSA in the Wild. In Joppe Bos and Martijn Stam, editors, *Computational Cryptography*, chapter 6, pages 140–181. Cambridge University Press, 2022.

[HS01] Nick Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptogr.*, 23(3):283–290, 2001.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, August 2003.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 2013*, November 2013.

[KASN15] Rashmi Kumari, Mohsen Alimomeni, and Reihaneh Safavi-Naini. Performance Analysis of Linux RNG in Virtualized Environments. In *ACM Workshop on Cloud Computing Security Workshop - CCSW '15*, New York, New York, USA, 2015.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In *SAC 2020*, October 21-23, 2020.

[Lin22] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Report 2022/374, 2022. https://eprint.iacr.org/2022/374.

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *ACM CCS 2018*, October 2018.

[MAK+17] Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. ROTE: Rollback protection for trusted execution. In *USENIX Security 2017*, August 2017.

[MH20] Gabrielle De Micheli and Nadia Heninger. Recovering cryptographic keys from partial information, by example. Cryptology ePrint Archive, Report 2020/1506, 2020. https://eprint.iacr.org/2020/1506.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[NKDM03] Antonio Nicolosi, Maxwell N. Krohn, Yevgeniy Dodis, and David Mazières. Proactive two-party signatures for user authentication. In *NDSS 2003*, February 2003.

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In *CRYPTO 2021, Part I*, August 2021.

[NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In *ACM CCS 2020*, November 2020.

[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*, October 2021.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In *EUROCRYPT'91*, April 1991.

[PLD+11] Bryan Parno, Jacob R. Lorch, John R. Douceur, James W. Mickens, and Jonathan M. McCune. Memoir: Practical state continuity for protected modules. In *2011 IEEE Symposium on Security and Privacy*, May 2011.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, May 1996.

[Roy22] Lawrence Roy. SoftSpokenOT: Communication–Computation Tradeoffs in OT Extension. In *CRYPTO '22*, 2022.

[RS21] Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO 2021, Part III*, August 2021.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, (3), January 1991.

[SP16] Raoul Strackx and Frank Piessens. Ariadne: A minimal approach to state continuity. In *USENIX Security 2016*, August 2016.

[SS01] Douglas R. Stinson and Reto Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP 01*, July 2001.

[ST19] Nigel P. Smart and Younes Talibi Alaoui. Distributing any elliptic curve based protocol. In *17th IMA International Conference on Cryptography and Coding*, December 2019.

[Wig97] John Wigley. Removing need for rng in signatures, message 5gov5d$pad@wapping.ecs.soton.ac.uk posted to sci.crypt. http://groups.google.com/group/sci.crypt/msg/a6da45bcc8939a89, 1997.

[WNR] Pieter Wuille, Jonas Nick, and Tim Ruffing. BIP 340: Schnorr Signatures for secp256k1. https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

A Specification of $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$ and $\mathcal{F}_{\mathsf{Com-ZK}}^{\mathsf{R}_{DL}}$

Functionality A.1. $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{PCF}}$**. Threshold Schnorr Signing With Error**

This two-party functionality is parameterized by the $(\varepsilon, \mathbb{G})$-$\mathsf{PCF}_{\mathsf{DL}}$ characterized by algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathcal{S}, \mathcal{V})$. All messages are adversarially delayed.

Upon receiving $(sid, \texttt{prover-init})$ from party P_i and $(sid, \texttt{verifier-init})$ and $(sid, \texttt{verifier-init})$ from party P_{1-i} for some $i \in \{0, 1\}$ if no such message pair has previously been received for this sid, sample $(k_{\mathsf{P}}, k_{\mathsf{V}}) \leftarrow \mathsf{Setup}(1^{\kappa})$. Send $(sid, \texttt{prover-key}, k_{\mathsf{P}})$ and $(sid, \texttt{verifier-key}, k_{\mathsf{V}})$ to parties P_i and P_{1-i} respectively.

Functionality A.2. $\mathcal{F}_{\mathsf{Com-ZK}}^{\mathsf{R}_{DL}}$**. Commitment and ZKPoK of DLog**

This two-party functionality is parameterized by the group $(\mathbb{G}, G, q)$. All messages are adversarially delayed.

Commit Proof: On receiving $(\texttt{commit}, sid, X, x)$ from P_i, where $x \in \mathbb{Z}_q$ and $X \in \mathbb{G}$, store $(\texttt{received}, sid, X, x)$ and send $(\texttt{committed}, i)$ to P_{1-i}.

Decommit Proof: On receiving $(\texttt{open}, sid)$ from P_i, if $(\texttt{received}, sid, X, x)$ exists in memory:

1. If $X = x \cdot G$, send $(\texttt{opened}, sid, X)$ to P_{1-i} .
2. Otherwise send $(\texttt{failed}, sid)$ to P_{1-i}.

B Reduction to Strong RSA

Lemma B.1. In the soundness proof Theorem 5.5, hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable, assuming the hardness of Strong RSA.

Proof. **Reduction.** We prove that any adversary $\mathcal{A}$ that distinguishes $\mathcal{H}_3$ from the $\mathcal{H}_2$ with advantage P can be turned into an attack against Strong RSA (Definition 2.2). If $D \geq P$ is the chance of passing check (b), then the reduction will succeeds with probability $\Theta(D)$. It's expected computational complexity is 2 exponentiations ($2/D$ for worst case complexity), plus the computation in $\mathcal{A}$ and V. Note that if the adversary succeeds with non-negligible probability then $2/D \leq 2/P$ will be bounded by a polynomial infinitely often.

The reduction will be given a modulus N' and a uniformly random element $g^* \in (\mathbb{Z}_{N'})^{\times}$, and asked to find a pair (z, e) such that $g^* = z^e$. Sample a uniformly random sign $\sigma = \pm 1$, and set $g = \sigma g^*$. Now run the game representing the previous hybrid, except using these values of N' and g. When V runs, if the check $\Delta \in \mathcal{D}$ succeeds then try to find Δ' by repeatedly sampling $\Delta' \leftarrow [-\frac{\eta}{2}, \frac{\eta}{2})$ and checking whether $\Delta' \in \mathcal{D}$. Retry the sampling up to $2/D$ times, and abort if no $\Delta' \in \mathcal{D}$ is found.

Next, compute a and b as before, and use the extended Euclidean algorithm to find $x, y \in \mathbb{Z}$ such that $ax + by = \gcd(a, b)$. When $b \mid a$ we require the reduction to use $x = 1$ and $y = \frac{|b|-a}{b}$, which works because $\gcd(a, b) = |b|$. Claim 5.7 implies that $g^{\gcd(a,b)} = g^{ax+by} = \pm(s^x g^y)^b$, so if we let $r = (s^x g^y)^{b\,/\,\gcd(a,b)}$ then $(rg^{-1})^{\gcd(a,b)} = \pm 1$. That is, rg^{-1} is a $(2\gcd(a, b))$th root of unity. We claim that if $rg^{-1} \neq \pm 1$ then we can efficiently factor N'.

Let $p' \cdot q'$ be the prime factorization of N', and $\lambda(N') = \mathrm{lcm}(p' - 1, q' - 1)$ be the Carmichael function, so that the order of every element in $(\mathbb{Z}_{N'})^{\times}$ divides $\lambda(N')$. Then rg^{-1} must be a $\gcd(2a,$

$2b, \lambda(N'))$th root of unity. The prime factorization of $\lambda(N')$ is $2 \cdot (\frac{p'-1}{2}) \cdot (\frac{q'-1}{2})$, as p' and q' are safe primes. We can then infer that $\gcd(2a, 2b, \lambda(N')) = 2$, because

$$|b| \leq \eta < 2^{\ell(\kappa)/2-2} \leq \min\left(\frac{p'-1}{2}, \frac{q'-1}{2}\right).$$

Because rg^{-1} is a square-root of unity, if $rg^{-1} \neq \pm 1$ then we can factor N' by finding $\gcd(rg^{-1} - 1, N')$, and use the factorization to solve Strong RSA.

Otherwise, we have $rg^{-1} = \pm 1$, so $r\sigma g^{-1} = rg^{*-1} = \pm 1$ as well. Because σ is independent from (r, g), with probability 1/2 we have $rg^{*-1} = 1$. If so, $r = (s^x g^y)^{b/\gcd(a,b)} = g^*$, and the reduction outputs (z, e) for $z = s^x g^y$ and $e = b/\gcd(a, b)$. If $b \nmid a$ then $|e| > 1$ and the reduction succeeds.

Analysis. We need to show that the reduction will succeed with probability $\Theta(D)$. First, we lower bound the probability that Δ and Δ' will be found. It is

$$\begin{aligned}
\mathbb{E}\left[\frac{|\mathcal{D}|}{\eta}\left(1 - \left(1 - \frac{|\mathcal{D}|-1}{\eta}\right)^{2/D}\right)\right] &= D - \mathbb{E}\left[\frac{|\mathcal{D}|}{\eta}\left(1 + \eta^{-1} - \frac{|\mathcal{D}|}{\eta}\right)^{2/D}\right] \\
&\geq D - \frac{1+\eta^{-1}}{\frac{2}{D}+1}\left(1 + \eta^{-1} - \frac{1+\eta^{-1}}{\frac{2}{D}+1}\right)^{2/D} \\
&\geq D - \frac{1+\eta^{-1}}{\frac{2}{D}} = \frac{D}{2}(1 - \eta^{-1}) = \Theta(D).
\end{aligned}$$

In the first inequality we used that $\frac{|\mathcal{D}|}{\eta} = \frac{1+\eta^{-1}}{2/D+1}$ maximizes the expectation, and in the second we used $D \geq \frac{2}{\eta}$ because $\frac{2}{\eta}$ is negligible and D is not.

Assume that Δ' is found by the search. We now show that when the previous hybrid's checks pass, but this hybrid's new checks would abort, then the reduction will succeed with constant probability. That is, when $|\mathcal{D}| \geq 2$, $\Delta, \Delta' \in \mathcal{D}$, and either $b \nmid a$ or $s \neq \pm g^{\bar{u}_{lo}}$. If $b \nmid a$ then we argued above that either $rg^{-1} = \pm 1$, and we get a solution to Strong RSA with probability 1/2, or rg^{-1} is another square-root of unity, and we can factor N' to solve Strong RSA. If $b \mid a$ then if $rg^{-1} \neq \pm 1$ then we could factor N' like before. Alternatively, $\pm g = r = (s^x g^y)^{b/\gcd(a,b)}$. Assume without loss of generality that $b > 0$, so $\gcd(a, b) = b$, $x = 1$, and $y = 1 - \frac{a}{b}$. Then $\pm g = s g^{1-\frac{a}{b}}$ and $s = \pm g^{\frac{a}{b}} = g^{\bar{u}_{lo}}$, so both hybrids behave identically in this case.

The expected running time of the reduction is 2 exponentiations more than $\mathcal{A}$'s runtime. That is, we use an extra $2/D$ exponentiation, but only when $\Delta \in \mathcal{D}$, which has probability D. □

C Bandwidth Cost of [Lin22, NRS21]

We show here how we derived the bandwidth costs for Table 1.1.

C.1 Bandwidth Cost of [Lin22]

Each party transmits a UC commitment in the first round ($2\kappa \approx |\mathbb{G}|$ bits), and in the second round opens this to a group element ($1 \times |\mathbb{G}|$) and its proof of knowledge of discrete logarithm obtained via Fischlin's compiler [Fis05]. This PoK permits a tradeoff in computation and bandwidth by adjusting the number of repetitions in Fischlin's compiler; for our calculation we will assume 10 repetitions. Each repetition involves transmitting $|\mathbb{G}| + \mathbb{Z}_q + \kappa \approx 2.5|\mathbb{G}|$ bits. Finally, a single $\mathbb{Z}_q$ element is transmitted to assemble the signature. This brings the total to $\approx 28|\mathbb{G}| = 896$ Bytes.

C.2 Bandwidth Cost of [NRS21]

Each party transmits two group elements in the first round ($2|\mathbb{G}|$ bits), and a single $\mathbb{Z}_q$ element in the second round, bringing the total to $3|\mathbb{G}| = 96$ Bytes.