Verifiable Encryption from MPC-in-the-Head

Akira Takahashi[1] ⓘ ↗ and Greg Zaverucha[2]

[1] J.P.Morgan AI Research & AlgoCRYPT CoE, New York, USA
[2] Microsoft Research, Redmond, USA

Abstract. Verifiable encryption (VE) is a protocol where one can provide assurance that an encrypted plaintext satisfies certain properties, or relations. It is an important building block in cryptography with many useful applications, such as key escrow, group signatures, optimistic fair exchange, and others. However, the majority of previous VE schemes are restricted to instantiation with specific public-key encryption schemes or relations.

In this work, we propose a novel framework that realizes VE protocols using zero-knowledge proof systems based on the MPC-in-the-head paradigm (Ishai et al. STOC 2007). Our generic compiler can turn a large class of zero-knowledge proofs into secure VE protocols for any secure public-key encryption scheme with the *undeniability* property, a notion that essentially guarantees binding of encryption when used as a commitment scheme.

Our framework is versatile: because the circuit proven by the MPC-in-the-head prover is decoupled from a complex encryption function, the work of the prover is focused on proving the encrypted data satisfies the relation, not the proof of plaintext knowledge. Hence, our approach allows for instantiation with various combinations of properties about the encrypted data and encryption functions. We then consider concrete applications, to demonstrate the efficiency of our framework, by first giving a new approach and implementation to verifiably encrypt discrete logarithms in any prime order group more efficiently than was previously known. Then we give the first practical verifiable encryption scheme for AES keys with post-quantum security, along with an implementation and benchmarks.

Contents

1 Introduction

A verifiable encryption (VE) scheme is a public-key encryption scheme where one party (called a *prover*, $\mathcal{P}$) can encrypt data w with a public key pk (of which the corresponding decryption key sk is held by the *receiver*, $\mathcal{R}$), and convince a third party (called the *verifier*, $\mathcal{V}$) that the data satisfies a relation R, i.e., $R(x, w) = 1$ with respect to a public statement x. At a very high-level, an (interactive) VE scheme should satisfy the following security properties [CD00]:

- **Completeness**: If $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{R}$ are honest then $\mathcal{V}$ accepts after interacting with $\mathcal{P}$, and $\mathcal{R}$ uses sk to obtain a plaintext w satisfying $R(x, w) = 1$.

- **Zero knowledge**: As $\mathcal{V}$ does not have the decryption key sk, she learns nothing about the plaintext from interacting with $\mathcal{P}$.

- **Validity**: If $\mathcal{V}$ accepts after interacting with a prover $\mathcal{P}^*$, $\mathcal{R}$ is guaranteed to obtain a plaintext w such that $R(x, w) = 1$, even if $\mathcal{P}^*$ is malicious.

Our Motivating Example for verifiable encryption is the *verifiable backup problem*, where a cryptographic device (such as a hardware security module (HSM)) or cloud service (such as [AWS22a, AWS22b, AKV22, GK22]) that is entrusted to store key material must securely export it for backup in case of hardware failure. These backups must be encrypted (or "wrapped") with the public key of another device, so that the plaintext keys are never exposed outside of the secure hardware [YC22, PK15]. The administrator of the device, responsible for creating backups, does not get assurance that the backup is well-formed, and will import successfully on the new device. She could try the import operation, but this may be expensive (e.g., if the backup device is in a separate facility), or risky (as it spreads the key around more than necessary). This latter risk is well illustrated in the case of cloud-based HSMs, where testing a backup by importing a key into a secondary cloud provider greatly expands the trust boundary.

Even if the import operation succeeds, the admin should still test that the imported private key corresponds to the expected public key, which typically requires using it to create a test signature or decryption. This is undesirable for two reasons: it adds extra use(s) of the key which must be logged for auditing, and it may also involve using the key for a different purpose than it was created for. Ideally, the exporting device could prove to the administrator that the ciphertext is a well-formed encryption under the receiving device's public key, and further, that the plaintext is a private key corresponding to a particular public key, e.g., the device claims "I encrypted the ECDSA signing key x for a public verification key y" and the administrator should be convinced that $y = g^x$ without access to the plaintext x. If the exported key is a symmetric key, then the device should prove that the plaintext is a key consistent with a commitment to the key, or a ciphertext or MAC created with the key. Verifiable encryption is a natural solution to this problem.

Verifiable Encryption Despite being introduced decades ago by Stadler [Sta96] and becoming a well-defined primitive with a relatively general solution in the work of Camenisch and Damgård [CD00], constructions suitable for the verifiable backup problem are limited. There are multiple challenges. We need generality, to allow multiple types of relation to be supported, not only a single one (as in [CS03, NRSW20, LN17]). Our use case requires verifiable encryption of many types of keys (potentially all the types here [PK22]), and at least ECC, RSA, and AES (the common types supported by cloud providers [AWS22a, AKV22, GK22]). We also want to minimize the additional assumptions required, ideally not requiring any new assumptions; for example if an AES key is to be exported, encrypted under an RSA key, we should not need to make assumptions in elliptic curve groups (perhaps with a pairing), as might be the case if certain SNARK proof systems were used for verifiability [Gro16, MBKM19, BBB$^+$18, LCKO19]. We also want flexibility in the receiver's public-key encryption (PKE) scheme, again to minimize new assumptions, but also to

support security goals like threshold decryption or post-quantum security, rather than have a VE scheme that dictates the PKE the receiver must use (as in [CS03, NRSW20, LN17]).

While any PKE can be made verifiable using a sufficiently general NIZK proof system (e.g., [BFM88, Mic00, GOS06]), the cost will be high. First, feasibility of VE for an arbitrary relation R is trivial once general-purpose NIZK for NP is given: attach a NIZK proof for modified relation $R' = \{(\mathsf{ct}, x), (\mathsf{pt}, r)) : R(x, \mathsf{pt}) = 1 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pt}; r)\}$. In general, the main technical challenge in constructing VE is to minimize the cost of proof-of-plaintext-knowledge (PoPK) "$\mathsf{ct} = \mathsf{Enc}(\mathsf{pt}; r)$" while supporting a large class of relation R; the naive approach would either require (1) re-designing a special encryption scheme for which an efficient PoPK Σ-protocol exists, or otherwise (2) proving correct evaluation of the circuit Enc, which would be costly depending on the encryption scheme (e.g., Kyber-KEM standardized by NIST [SAB+20] involves both algebraic operations over a cyclotomic ring and hashing). In summary, we desire a construction that is as general as possible, introduces no new assumptions, is versatile enough to support as many encryption schemes as possible, and is performant enough to be practical.

There are multiple applications of verifiable encryption in the literature. Some early examples include publicly verifiable secret sharing [Sta96], group/ring signatures [CD00, BSZ05, BKM09] and verifiable encryption of signatures for optimistic fair exchange [ASW98, Ate99], and more recent applications include blockchains [DHMW23, CDK+22]. Key escrow [YY98, PS00], where parties encrypt their private key to a trusted escrow authority, can be achieved with verifiable encryption, since it becomes possible for other parties on the network to ensure that the correct key has been escrowed. A common theme is *identity escrow* (or revocable anonymity) in privacy systems and group signatures, where an anonymous party encrypts their identity for an authority, who can de-anonymize them under certain circumstances. In cryptographic voting systems, voters often encrypt their votes and prove that their selection is in a set of valid choices (e.g., in $\{0, 1\}$ to encode a "yes" or "no" vote). The earliest paper with this idea predates the literature on verifiable encryption [CF85] and VE is still used in cryptographic voting systems today, see for example [EG21, CCFG16].

ZK from MPC The MPC-in-the-head (MPCitH) paradigm [IKOS07] is a way to create a zero-knowledge (ZK) proof for a relation R, given a secure multiparty computation protocol (MPC) to compute R. Some of the advantages of this approach make it well suited to our verifiable encryption problem. First, MPC protocols are *very flexible*, so that we can instantiate ZK proofs for many choices of R, typically expressed as binary or arithmetic circuits. The paradigm extends beyond circuits as well: we give an MPCitH protocol to prove knowledge of a discrete logarithm, and use our results to verifiably encrypt discrete logs.

Second, if the MPC protocol is information theoretically secure, converting it to a ZK proof only requires a secure commitment scheme, which can be instantiated with a cryptographic hash function, so that the proof system requires minimal assumptions, and is post-quantum secure. Finally, the performance of MPCitH proof systems in terms of prover and verification costs and proof sizes are practical, and have been steadily improving as has been demonstrated in the area of post-quantum signatures. To use the AES-128 circuit as an example, proof sizes went from 209 KB [GCZ16] to 32 KB [DDOS19] to 13 KB [BDK+21] to 9.9 KB [KZ22] in the past six years, and the running time of the prover and verifier is below 20ms (see the implementation benchmarks in [KZ22]). Taken together, these properties will allow us to construct verifiable encryption schemes that are very general, make minimal assumptions, achieve post-quantum (PQ) security and are efficient enough for practical use.

1.1 Our Contributions and Techniques

We outline our four main contributions and then discuss some of the techniques used in the paper.

Generic compiler for MPC-in-the-head-based VE Our results apply to a broad class of MPCitH proofs: those that can be viewed as an interactive oracle proof (IOP). The original class from [IKOS07] is captured by the IOP framework as well as many more recent MPCitH proofs aimed at concrete efficiency, such as [GMO16, KKW18, BN20, BDK+21, BD20, Beu20, DOT21]. In Section 3 we give a compiler that takes a proof protocol from the MPCitH-IOP class and converts it into a verifiable encryption scheme, denoted MPCitH-VE. Analogous to a series of work on black-box commit-and-prove [GLOV12, KOS18, Kiy20], our compiler treats PKE in a black-box manner, avoids dedicated proof-of-plaintext-knowledge, and is thus compatible with the majority of existing schemes. Unlike these feasibility results, our result focuses on concretely efficient instantiations and compatibility with typical relations and (possibly imperfectly correct) encryption schemes relevant to the verifiable key backup problem. Compared to the naïve approach that generates proof-of-plaintext-knowledge by representing the encryption function as an arithmetic circuit, our approach generally offers lower prover complexity while increasing the ciphertext sizes due to the number of MPC parties and parallel repetitions. With our approach the prover complexity is dominated by the cost of generating MPCitH-based ZK proof for relation R satisfied by the VE plaintext. With recent progress in the MPCitH literature, a VE prover can perform this task with low complexity for many types of relations such as the AES circuit.

Methods for compressing ciphertext In our compiler, the ciphertext size is independent of the relation R, but does depend on the witness and the number of parallel repetitions required for soundness. To narrow this gap, in Section 4 we give two methods that $\mathcal{V}$ can use to compress the VE ciphertexts. The first, called the *random subset method*, is very simple, incurs no computational overhead, and can reduce ciphertext size by a factor of three when the number of parallel repetitions τ is large. If τ is already small, it is also possible to reduce ciphertext size by increasing τ, which might be desirable depending on the application.
The second approach, called the *equality proof method*, is optimal as it achieves constant size ciphertexts, $O(|w|)$ (provided PKE has constant ciphertext expansion). However, the other costs of the resulting VE scheme (such as basic encryption and decryption operations) increase significantly, and this approach limits the class of compatible PKE to the homomorphic ones for which an efficient equality proof exists. See Appendix G for further details. We highlight improving compression as an interesting direction for future work.

Concrete Instantiation and Implementation In Section 5 we apply our transform to encrypt different types of keys, and quantify performance. We first give a new proof of knowledge and a VE scheme for discrete logarithms (DL) based on a new non-interactive ZK proof called *distributed key generation in the head (DKG-in-the-head)*. The prover emulates a protocol where parties run a DKG protocol to compute $y = g^x$. Since the DKG protocol only needs to have passive security and a broadcast channel is available for free in the MPC-in-the-head setting, our proposed protocol is extremely simple, requiring only a single round of interaction between parties. We also give a variant of this scheme based on a *robust DKG protocol*, that has faster verification and does not require parallel repetition for soundness, and show that our DKG-based protocols may also be used to verifiably encrypt RSA keys and plaintexts.

As discussed earlier, one of our goals is to design versatile VE suitable for wrapping a variety of keys with arbitrary PKE schemes, while benefiting from modern MPCitH ZK proofs which have more complex structures than Σ-protocols. To verifiably encrypt AES

keys, we apply our transform to the Banquet [BDK$^+$21] and Helium [KZ22] ZK proofs, where $\mathcal{P}$ proves knowledge of an AES key used to generate a public plaintext/ciphertext pair, i.e., it is specialized for the relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$. Prior to this work, there has been no practical VE scheme for AES keys, which may find interesting applications in the post-quantum setting when instantiated with quantum-resilient PKE.

We implement all three of these schemes, along with the scheme from [CD00] for comparison, and give benchmark results.[1] Overall we demonstrate that our new schemes are efficient and practical. In the DL setting, we find that the DKG-based schemes each offer a different tradeoff, e.g., allowing one to choose a scheme with short ciphertexts (1 KB), or fast verification (2ms on a 3.6GHz CPU) at the expense of lower performance in one of the other metrics. We conclude that none of the schemes in our comparison is strictly better than the others across all performance metrics. However, we believe the range of tradeoffs our parameters can offer may be useful depending on the application scenario. For our AES implementation we pair Helium with the post-quantum encryption algorithm Kyber [SAB$^+$20], and we show that an AES key can be verifiably encrypted under a Kyber public key with 22 KB proofs, 13 KB ciphertexts, prover and verifier times of 68 ms and decryption times of 2ms. For perspective, proofs of plaintext knowledge for lattices generally require proofs that are tens to hundreds of KB in size [GHL$^+$22, Table 1].

Revisiting the Camenisch-Damgård VE Construction We show that the existing verifiable encryption transform of Camenisch and Damgård [CD00] fails to retain the validity property when instantiated with IND-CPA PKE schemes that are only *statistically correct*, as opposed to perfectly correct. We describe concrete attacks in which a malicious prover can convince the verifier to accept a ciphertext that decrypts to random data unrelated to R. Finally, we show that by additionally assuming the undeniability property their construction can also be securely instantiated with statistically correct PKE schemes.

Our Techniques The output of our compiler is MPCitH-VE, a public-coin three-round interactive protocol, which can be made non-interactive using the standard Fiat-Shamir transform [FS87]. The first input to our compiler is a protocol MPCitH-IOP, this abstraction captures several three-round protocols, including [IKOS07], ZKBoo [GMO16], ZKB++ [CDG$^+$17], and our new DKG-in-the-head protocol. We also discuss using the same ideas to compile IOP versions of KKW [KKW18], Banquet [BDK$^+$21], Limbo [DOT21], Ligero [AHIV17], and Shamir secret sharing-based MPCitH [FR23].

The other input to the compiler is a public key encryption (PKE) scheme, such as Elgamal, RSA-OAEP or PQ-secure options like Kyber [SAB$^+$20] or FrodoKEM [NAB$^+$19]. We define and prove the requirements the PKE must have to ensure MPCitH-VE is secure. In short, ciphertexts created by the PKE must be a secure commitment (both hiding and binding) to the plaintext. Hiding is provided by CPA security (security against chosen-plaintext attacks), and for binding, we define a new property called *undeniability*, which is trivial for PKE schemes with perfect correctness, but may be absent otherwise. Notably, lattice-based PQ schemes are usually not perfectly correct. In Appendix B we prove that the Fujisaki-Okamoto transform [FO99, FO13, HHK17] (and simpler variants of it) can be used to upgrade any statistically correct PKE scheme to obtain undeniability, making our construction compatible with many existing schemes. An implication is that encryption schemes using the FO transform are secure commitment schemes, which might be of independent interest.

Our approach can easily instantiate VE with various combinations of R and PKE, since the circuit for R is decoupled from the encryption function of PKE. The prover's work is focused on proving the statement R about the encrypted data, not on the proof of plaintext knowledge. Proof of plaintext knowledge is achieved with existing mechanisms in the MPCitH proof. To illustrate the core idea of our transform, we sketch an example VE scheme based on the ZKBoo proof system [GMO16]. The original ZKBoo protocol

---

[1] Our implementations are available at https://github.com/akiratk0355/verenc-mpcith

for relation $R(x, w) := (f(w) =_? x)$ (where f is typically a one-way function) proceeds as follows: the prover $\mathcal{P}$ first distributes to three parties additive shares (w_1, w_2, w_3) of the secret witness w. Then $\mathcal{P}$ runs an MPC protocol computing R "in the head", to produce the *view* of each party, i.e., a string consisting of the input share, output, communication, and random tape. $\mathcal{P}$ sends commitments to the views as its first message, and the verifier $\mathcal{V}$ returns a challenge $\bar{i} \in \{1, 2, 3\}$, indicating party $\bar{i}$'s view is supposed to remain secret. $\mathcal{P}$ then responds with the views of party $i \neq \bar{i}$ and commitment randomness. $\mathcal{V}$ accepts if the commitments are correctly opened and the views agree with a correct run of the MPC.

Notice that one can immediately recover the witness once the commitment to party $\bar{i}$ is revealed. Our main observation is that a technique similar to *straight-line extractable* ZK proofs gives rise to a secure VE scheme: by replacing commitments in the original proof system with public-key encryptions, the prover $\mathcal{P}$ now sends three ciphertexts containing witness shares: $\mathsf{C}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, w_i)$ for $i = 1, 2, 3$ (and the remaining view_i can be committed with a cheaper hash-based commitment). The verifier still learns nothing about the encrypted data w since one of its additive shares is kept encrypted. By contrast, the receiver $\mathcal{R}$ with knowledge of the decryption key sk can decrypt the unopened ciphertext $\mathsf{C}_{\bar{i}}$ (or commitment) to obtain the remaining share $w_{\bar{i}}$, from which the plaintext w can be recovered using the shares $(w_i)_{i \neq \bar{i}}$ revealed in the public transcript. As usual, by applying the Fiat–Shamir transform [FS87], the above interactive protocol can be turned into a non-interactive VE scheme in the random oracle model, as we formally discuss in Appendix C.

While the idea of the construction is relatively simple (given the machinery of MPCitH), and its analysis may be straightforward for limited types of MPCitH proofs, the challenge is in defining and analyzing the compiler so that it is practically useful. Recent highly optimized, concretely efficient MPCitH proofs deviate significantly from the simpler IKOS/ZKBoo [GMO16, IKOS07] example given above for performance (e.g., they have more than three rounds, use broadcast channels, preprocessing, open more than two parties, etc.). With our comprehensive approach, most of the literature describing MPCitH proof systems can now be used for VE in an efficient way – arguably efficient enough for practice, as we demonstrate with AES and DL.

1.2 Related Work

Camenisch–Damgård transform Although our generic transform is similar in spirit to that of [CD00], there are some differences. Our starting point is any MPC-in-the-head IOP with the straight-line extractable property, while [CD00] is focused on 2-special sound Σ-protocols with 1-bit challenge space (though it seems possible to generalize their transform to k-special sound protocols for any k as well). Although one can naïvely apply [CD00] to some MPC-in-the-head protocols with k-special soundness, such as ZKBoo and IKOS, our method directly modifies the committing function and thus leads to better communication complexity. Moreover, [CD00] does not apply to more efficient MPC-in-the-head constructions, including KKW and Banquet: because the challenge spaces of these protocols are not limited to party indices the notion of special soundness is not well-defined. In contrast, our transform relies on straight-line extractability, and therefore applies to KKW and Banquet as well.

Camenisch–Shoup scheme Camenisch and Shoup [CS03] propose protocols for efficient verifiable encryption and decryption of discrete logarithms. However, it only works for discrete logarithms in a group where Paillier's decision composite residuosity (DCR) assumption holds, and the PKE is fixed to (a variant of) Paillier's scheme as well. The scheme is therefore not suitable for encrypting ECC, RSA or AES keys, one of our motivating examples. In theory it is possible to prove that the plaintext of a Camenisch-Shoup ciphertext corresponds to a discrete log from a prime order group. However, this

would require range proofs across the two groups, and security still relies on DCR (and possibly more, depending on how the range proofs are done). Finally, in Section 5.1 we estimate that the prover and verifier costs, as well as the ciphertext size of the [CS03] scheme are higher than encrypting discrete logarithms in prime order groups directly using our new DKGitH protocol.

SNARK-based constructions Lee et al. [LCKO19] gives a construction of a verifiable encryption scheme that is tailored to use in voting schemes as it is additively homomorphic and supports rerandomization. The construction is pairing-based, Elgamal-like and thus integrates well with SNARK proof systems. Just like our framework, theirs also decouples the encryption function from the circuit describing the relation, using the *commit-and-prove* SNARK of [CFQ19]. It requires a trusted setup assumption due to the use of CRS-based SNARK, while ours is naturally transparent thanks to the underlying MPC-in-the-head paradigm.

Nick et al. [NRSW20] gives a construction which can encrypt a discrete logarithm in an elliptic curve group, using a special PRF called Purify. The scheme does allow, e.g., encryption of an ECDSA private key without any trusted setup assumption thanks to the use of Bulletproofs [BBB+18], but requires that encryption be done with an Elgamal-like PKE. As we compare in Table 1, their ciphertext and proof are more compact than those of our DKG-in-the-head VE scheme, while ours requires less prover time. A complication related to implementation of the Purify PRF is that one must choose an additional pair of elliptic curves, related to the group order of the curve where the discrete logarithm is defined, such that the DDH assumption holds. In contrast, our framework does not introduce any additional assumption other than IND-CPA and undeniability of PKE (already satisfied by perfectly correct schemes and many statistical ones as we analyze).

Lattice-based constructions   Lyubashevsky and Neven [LN17] give a verifiable encryption scheme for lattices, based on the hardness of the ring learning with errors (RLWE) problem. They give a proof of plaintext knowledge, secure in the ROM that does not use parallel repetition to boost soundness. Their scheme can be further extended to support CCA security. The analysis of our VE construction does not consider CCA security and it is not "one-shot" as MPC-in-the-head proofs usually rely on parallel repetition or cut-and-choose unlike [LN17]. The construction comes with multiple caveats.

- A malicious prover may create a ciphertext that takes variable time to decrypt. In particular decryption requires $O(q)$ time to decrypt, where q is the number of hash queries made by the prover. This makes decryption potentially very expensive.

- Decryption is not guaranteed to recover the original plaintext, but vectors with small coefficients. It's argued that this is sufficient for some of the applications considered in [LN17], but may not be sufficient in general.

- The size of the proof and ciphertext are relatively large, for example proof sizes are 38–54 KB and ciphertext sizes are 48–71 KB. Proofs and ciphertexts may be as short 9 KB however, this is for verifiable encryption when there is no plaintext.

Isogeny-based construction Beullens et al. [BDK+22] proposed a VE scheme based on isogenies (or more generally, any cryptographically-hard group action). Their main motivation is to construct a building block of a ring signature: their VE prover essentially proves that (1) it encrypted a verification key vk, (2) vk belongs to a ring, and (3) it knows the secret sk corresponding to vk. Although fairly efficient, their approach inherently relies on Elgamal-like PKEs and it is highly specialized for the above limited class of relation. On the other hand, our focus is to build a general framework to support a large class of PKEs and relations as required for concrete solutions to the verifiable key export problem described earlier.

MPCitH and IOP proof systems We briefly survey some of the many existing MPCitH-based proof systems, optimized for different relations, as these immediately give verifiable

encryption schemes by applying our transform. [Beu20] gives an MPCitH-based proof of a solution of an SIS (short integer solution) instance. We can apply our transform to construct a verifiable encryption of SIS witnesses (here the witness is exact, not relaxed as in [LN17]). The proof protocols in [Beu20] for other relations, such as the PKP and MQ problems, are also compatible with our transform. [BN20] also gives multiple MPCitH-based proofs for lattice problems (SIS), which are also amenable to our transform, but are outperformed by the proofs of [Beu20]. The Limbo proof system [DOT21] is efficient for general R described as circuits, making it a good choice for hash functions, or as an alternative to Banquet. Gjøsteen et al. [GHM⁺22] present verifiable *decryption* protocols from MPCitH proofs, by designing suitable distributed decryption protocols for Elgamal and BGV lattice-based encryption schemes.

Aurora [BCR⁺19] and Ligero [AHIV17] are non-interactive proof systems for R1CS that are constructed by defining an IOP, then making it non-interactive using the transform in [BCS16]. Both have short proofs for relations involving lattices, and Aurora has the shortest proofs for SIS, about 10x shorter than [Beu20, BN20]. As we mention in Section 7, for this reason a more general transform for building VE schemes from IOPs in the [BCS16] framework is interesting future work.

Commit-and-prove zero knowledge Verifiable encryption somewhat resembles *commit-and-prove* zero knowledge (CPZK) proofs (e.g., [GLOV12, KOS18, Kiy20, BHH⁺19] based on MPC-in-the-Head and [LCKO19, CFF⁺21] based on SNARKs), where the prover is tasked with proving some statement about (a part of) secret witness that has been committed to in advance. In fact, one could also see many existing VE schemes as CPZK with a committing function instantiated with $\mathsf{Enc}(\mathsf{pk}, \cdot)$. However, the crucial difference is that in the VE setting commitments must always be non-interactive and straight-line extractable, whereas in context of CPZK either commitments require no extractability or otherwise they are interactive and have extraction via rewinding (as in [GLOV12, KOS18, Kiy20]). We are also motivated to support PKEs with imperfect correctness for practical instantiation of post-quantum VE, which introduces additional technical challenges. Moreover, previous works do not explore newer, practical MPC-in-the-head proofs that deviate from the basic framework of IKOS, such as KKW and Banquet.

Connection between straight-line extraction and verifiable encryption *Straight-line extractability (SLE)* (or sometimes called *online extractability*) is a special type of extractability, specialized to proof systems in the ROM or in the CRS model. The prover commits to witness-dependent strings via extractable commitments instantiated with the RO or PKE, and the extractor is given the statement, the transcript, and the prover's query history (in the ROM) or a secret trapdoor (in the CRS model) to extract a witness. In particular the straight-line extractor does not get any access to the prover, or ability to rewind them. SLE is especially crucial for security in the QROM, since rewinding techniques are generally prohibitively expensive in that setting. Numerous works achieve SLE of commit-and-open-type proof systems (including MPC-in-the-head) [Pas03, KKW18, DFMS22, HLR21], lattice-based ZK proof systems [Kat21], and straight-line extractable alternatives to the Fiat-Shamir transform [Fis05, Unr15]. A receiver $\mathcal{R}$ of our MPCitH-VE essentially behaves like a straight-line extractor for the MPC-in-the-head proof systems whose commitments are replaced with PKE. In this work, we formally draw a connection between the validity property of VE and SLE of IOP, a setting where commitments are idealized and thus SLE holds very naturally.

Encryption as a Commitment Most natural public key encryption schemes are committing, and constructing a non-committing one (a deniable scheme) is challenging.

[GH03] defines committing public-key encryption, but defines the verification algorithm in a more generic way than what is used in our verifiable encryption scheme and the one of [CD00]. Rather than having the verifier recompute the ciphertext as we do, given the purported (message, randomness) pair, the verify algorithm can be any function that takes

as input the message, and a hint produced by the opening function.

[GLR17, DGRW18] looks at committing encryption for symmetric-key AEAD schemes, to support an analysis of a primitive called *message franking*, where participants in a messaging platform can report abusive messages to the service provider. The name *encryptment* is also used, a portmanteau of the terms encryption and commitment. The schemes support many additional features beyond what is required for verifiable encryption in our setting, and the definitions are consequently more complicated than those of [GH03].

[BDD22] proved that Pointcheval's IND-CCA PKE [Poi00] can be used as a secure commitment scheme as is, and it is thus plausible that their analysis can be adapted to show undeniability of the scheme as well. Our analysis of the Fujisaki-Okamoto transform also suggests that CCA conversions of this type are useful for obtaining undeniability (and thus binding). It is an interesting follow-up question whether CCA security in general is sufficient for PKE to be committing and/or undeniable.

The opposite of what we need is called *deniable encryption* [CDNO97]. Here the scheme is carefully constructed so that the encryption is *not* a binding commitment to the message and randomness, allowing a sender of a ciphertext to later claim they sent a different message (hence denying the original message). After sending a ciphertext $c = \mathsf{Enc}(m; r)$ then sender can later claim they sent (m', r'), and anyone can check that $c = \mathsf{Enc}(m'; r')$ as well. This is why we use the name *undeniable encryption* to describe a scheme where this is not possible. While a "sender-deniable encryption scheme" in the terminology of [CDNO97] is sufficient as a counterexample to the analysis of [CD00], the example encryption schemes we describe for this purpose in Section 6 only require a weaker type of deniability.

Non-committing encryption [CFGN96, DN00] is related to deniable encryption, but constructions are interactive and the goal is to improve certain types MPC protocols. Briefly, in the security analysis, the simulator can use the fact that the encryption is not a commitment to be able to create simulated ciphertexts, then later open them to plaintexts that are consistent with later information.

Finally we mention *witness encryption* [GGSW13], which superficially sounds related, since in VE we are encrypting a witness w that is associated to a statement x by a relation R. However, a witness encryption scheme for R is a PKE-like primitive that allows us to use x as a public key and w is the secret key. No witnesses are ever encrypted! Witness encryption can be viewed as an encryption analog to *signatures of knowledge* [CL06], where w is a signing key and x is a public key verification key.

2 Preliminaries

First we introduce some notation and conventions used throughout the paper. The security parameter is denoted λ, and for an integer x, $[x]$ is short for the set $\{1, \ldots, x\}$. Whenever we have a two-part adversary, written as a pair, e.g, $(\mathbf{A}^*, \mathbf{P}^*)$, we assume that $\mathbf{A}^*$ and $\mathbf{P}^*$ share state, and do not explicitly write it as an output of $\mathbf{A}^*$ and an input to $\mathbf{P}^*$. For a set S, we denote by $x \leftarrow_\$ S$ sampling an element x from S uniformly at random.

In Appendix A, we recall the standard notions of public-key encryption (PKE) and interactive oracle proofs (IOP). We also introduce (straightline) extractable commitments (ECOM) required by our transformation.

2.1 Verifiable Encryption

We define a secure verifiable encryption scheme by adapting the definition from [CD00]. Non-interactive VE is formally defined in Appendix C. The main difference with [CD00] is that we additionally consider a *compression* algorithm $\mathcal{C}$ that postprocesses a transcript exchanged between a prover and a verifier, and outputs a corresponding ciphertext. In

practice, $\mathcal{C}$ would be run by the verifier right after interacting with the prover and obtaining a valid transcript. We explicitly introduce this because our proposed construction will benefit from different optimization strategies that post process accepting transcripts to produce a highly compressed ciphertext. The usefulness of the compression algorithm is motivated by the verifiable key backup scenario we sketched in the introduction. While the communication from prover to verifier is unchanged, compression reduces both the communication costs between the verifier and the receiver, and cost of storing ciphertexts. In the key backup scenario, the administrator is a verifier, and thus after the backup from the source HSM (prover) is verified, the administrator only needs to store a compressed ciphertext to be sent to the destination HSM (receiver). Moreover, unlike [CD00] we only consider ZK against *honest* verifiers, since this is sufficient to prove ZK of non-interactive VE in the random oracle model using the Fiat-Shamir transform.

Definition 1 (Verifiable Encryption Scheme). Let R be a relation with language $L_R := \{x : \exists w : (x, w) \in R\}$. A secure verifiable encryption scheme VE_R for R consists of a tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{R})$:

- $\mathcal{G}(1^\kappa)$: A key generation algorithm that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $(\mathcal{P}, \mathcal{V})$: A two-party protocol, where both $\mathcal{P}$ and $\mathcal{V}$ take (x, pk) and $\mathcal{P}$ additionally takes a plaintext w as inputs. We let $(b, \mathsf{tr}) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x)$ denote the output pair of $\mathcal{V}$ on common input (pk, x) when interacting with $\mathcal{P}(w)$, where $b \in \{0, 1\}$ indicates whether $\mathcal{V}$ accepts or rejects, and tr denotes a transcript exchanged between $\mathcal{P}$ and $\mathcal{V}$.
- $\mathcal{C}(x, \mathsf{tr})$: A compression algorithm that outputs a compressed ciphertext C.
- $\mathcal{R}(\mathsf{sk}, C)$: A receiver (or recovery) algorithm that outputs a plaintext w.

VE is secure if it satisfies the following three properties.

Completeness VE_R is ϵ_{comp}-complete if for all $(x, w) \in R$.

$$\Pr \left[b \neq 1 \vee (x, w') \notin R : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{comp}}(\kappa)$$

Validity VE_R is ϵ_{val}-valid if for all pairs of PPT adversaries $(\mathcal{A}^*, \mathcal{P}^*)$,

$$\Pr \left[b = 1 \wedge (x, w') \notin R : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}); \\ (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}^*(\mathsf{sk}), \mathcal{V} \rangle (\mathsf{pk}, x); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{val}}(\kappa)$$

Computational Honest Verifier Zero-knowledge VE_R is ϵ_{zk}-HVZK if there exists a PPT simulator $\mathcal{S}$ such that for all pairs of PPT adversaries $(\mathcal{A}, \mathcal{D})$ such that $\mathcal{A}$ always outputs a valid statement-witness pair,

$$\left| \Pr \left[i = i' : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ (x, w) \leftarrow \mathcal{A}(\mathsf{pk}); \\ (b, \mathsf{tr}_0) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle (\mathsf{pk}, x); \\ \mathsf{tr}_1 \leftarrow \mathcal{S}(\mathsf{pk}, x); \\ i \leftarrow_\$ \{0, 1\}; i' \leftarrow \mathcal{D}(\mathsf{pk}, x, \mathsf{tr}_i); \end{array} \right] - \frac{1}{2} \right| \leq \epsilon_{\mathsf{zk}}(\kappa)$$

Note that computational HVZK (as opposed to perfect, or statistical) is the best possible in the context of verifiable encryption, as an unbounded adversary can always try $w' = \mathcal{R}(\mathsf{sk}, C)$ with all possible sk, checking whether $(x, w') \in R$.

Protocol 1: MPCitH-IOP$_R$

Parameters: The number of parties N; the number of parallel repetitions τ; the number of opened parties t; the challenge set $\mathsf{Ch} = \big\{ \mathbf{e} \subset [N] \: : \: |\mathbf{e}| = t \big\}$.

Inputs: prover $\mathbf{P}$ receives (x, w); verifier $\mathbf{V}$ receives x.

Committing phase The first-round message of $\mathbf{V}$ is empty. $\mathbf{P}$ proceeds as follows.

1. Choose random $w_1, \ldots, w_N$ such that $w = \sum_{i=1}^{N} w_i$.

2. Emulate "in her head" the execution of Π_f on input $(x, w_1, \ldots, w_N)$.

3. Prepare, based on the execution, the share of the witness, and the randomness, the views $V_1, \ldots, V_N$ of the N parties; $\mathbf{P}$ outputs the proof string $\pi = (V_1, \ldots, V_N)$.

Query phase

1. $\mathbf{V}$ chooses a random $\mathbf{e} \in \mathsf{Ch}$ and queries the oracle for π with $\mathbf{e}$.

2. The oracle returns $(V_i)_{i \in \mathbf{e}}$.

Decision phase: $\mathbf{V}$ accepts if and only if $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$. See Section 2.2 for the definition of $\mathsf{CheckView}$

$\mathbf{P}$ and $\mathbf{V}$ execute τ instances of the above procedures in parallel. If $\mathbf{V}$ accepts in all τ executions, it outputs $b = 1$; otherwise it outputs $b = 0$.

2.2 MPC-in-the-Head Proofs as IOPs

In Protocol 1 we describe the blueprint of a simple MPC-in-the-head protocol characterized as a single-round IOP. In Appendix E.1 we provide further details of the original IKOS framework. We mainly focus our description on MPC-in-the-head ZK proofs derived from semi-honest MPC, because in the parameter regime of the simple NP-relations considered in this paper, they typically perform better than the ones from robust MPC. Since newer protocols such as, [FR23], Ligero, KKW and Banquet deviate from MPCitH-IOP significantly (e.g., they have more than 3 rounds, perform cut-and-choose, use Sharmir secret sharing, etc.), we separately describe them as IOPs in Appendix E.2 and formulate our definitions and analysis accordingly.

The framework of IOPs allows for a modular design of ZK proof systems and is becoming increasingly common for constructing efficient SNARKs and MPC-in-the-head ZK proofs (e.g., [CHM+20, CFF+21, DOT21]). As in prior work, we first design an information-theoretically secure protocol in the form of an IOP, where commitments are idealized in that both hiding and binding hold unconditionally. This is why the security properties for IOPs are defined w.r.t. unbounded adversaries, and the computational assumptions will only come into play when we later compile the IOP into a verifiable encryption scheme via a cryptographic commitment scheme with straight-line extractability.

In MPCitH-IOP$_R$, $\mathbf{P}$ proves knowledge of a witness w such that $R(x, w) = 1$, where Π_f is an MPC protocol computing f that uses additive secret sharing over some finite field $\mathbb{F}$, and $R(x, w) := (f(w) =_? x)$. This protocol is similar to the one from [IKOS07] relying on the "idealized commitment functionality", but modified to cover MPC protocols with a broadcast functionality, so the prover may open $2 < t < N$ parties' views instead of two. We also employ the IOP framework following more recent MPC-in-the-head protocols such as Ligero [BFH+20] and Limbo [DOT21]. As we shall see below, as an IOP protocol it is straightforward to prove straight-line extractability of MPCitH-IOP$_R$. This will allow a smooth transition to SLE of the MPCitH proof systems we compile (with suitable commitment schemes), then to the validity of the resulting verifiable encryption schemes.

Our description also has parallel repetition: a simpler protocol is repeated τ times in parallel to increase soundness. These changes make presentation consistent with many practical MPCitH proof protocols (e.g., ZKB++, KKW and Banquet all use $(N{-}1)$-private

MPC protocols with broadcast channels).

The helper function CheckView in $\mathsf{MPCitH\text{-}IOP}_R$ takes the statement and a set of views as input and returns 1 if:

1. The outputs of the opened parties (determined by their views) are 1, and

2. The opened views are consistent with each other, with respect to x and Π_f,

and returns 0 otherwise. We further define a utility function GetW, which takes a party's view and extracts their share of the witness from it.

We further recall the *canonical* way of extracting a witness from any MPC-in-the-head proofs, which is often implicit in the literature.

Definition 2 (Canonical extractor). An extractor $\mathbf{E}$ for one repetition of $\mathsf{MPCitH\text{-}IOP}_R$ is called *canonical* if on input x and $\pi = (V_1, \ldots, V_N)$, it works as follows: $\mathbf{E}$ obtains witness shares via $w_i = \mathsf{GetW}(V_i)$ for $i \in [N]$ and then outputs a candidate witness $w := \sum_{i \in [N]} w_i$. For τ repetitions, the canonical extractor $\mathbf{E}^\tau$ runs $\mathbf{E}$ on each repetition $j \in [\tau]$ and outputs $w^{(j)}$ if $(x, w^{(j)}) \in R$ for some j, otherwise it outputs $\bot$.

It is rather straightforward to check that the protocol $\mathsf{MPCitH\text{-}IOP}_R$ is (1) straight-line extractable (Definition 7) with respect to the canonical knowledge extractor $\mathbf{E}^\tau$ with $\epsilon_{\mathsf{sle\text{-}iop}} \leq ((k-1)/|\mathsf{Ch}|)^\tau$ assuming the notion called k-consistency (Definition 10), and (2) HVZK (Definition 8) if the underlying MPC protocol is t-private (Definition 11). For completeness, Appendix E formally introduces these notions and proves SLE and HVZK.

3   Our Transform

In this section we present our transform, which generically constructs a verifiable encryption scheme $\mathsf{MPCitH\text{-}VE}$ from an $\mathsf{MPCitH\text{-}IOP}$ protocol in the class described in Protocol 1. We start with a simple construction of extractable commitments from public-key encryption, then come to our compiler in Section 3.2.

3.1   Extractable Commitments from Undeniable PKE

In the following we show that most commonly used public-key encryption schemes give rise to extractable commitments. A similar construction appears in [GH03], and its analysis in the perfectly correct case is somewhat folklore, below we describe the exact construction we will use, and analyze its security. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. We construct an extractable commitment scheme $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ (see Section A.2 for definition and syntax). For simplicity we assume throughout that the message space S_m and random space S_r of the commitment schemes are identical to those of the encryption schemes. We remark that our formulation of ECOM is specifically tailored to *non-interactive* and *straight-line extractable* schemes, as opposed to what's referred to as an "extractable commitment" in some previous works (e.g., [GLOV12]) where the committing phase is interactive and the extractor has to rewind the prover. Interactivity is not suitable for VE, since the receiver does not directly interact with the prover.

- $\mathsf{CGen}(1^\lambda)$ runs $\mathsf{PKE.Gen}(1^\lambda)$ and outputs pk as the commitment key.

- $\mathsf{Commit}(pk, m; r)$ outputs $c = \mathsf{PKE.Enc}(pk, m; r)$.

- The opening of the commitment c is (m, r), and the verifier checks (m, r) against c by computing $c' = \mathsf{Enc}(pk, m, r)$; the opening is accepted iff $c' = c$, $m \in S_m$ and $r \in S_r$.

- $\mathsf{CExt}(sk, c)$ outputs $m = \mathsf{PKE.Dec}(sk, c)$.

It is rather straightforward to show the above construction is perfectly extractable, perfectly binding and computationally hiding, assuming PKE is *perfectly* correct and IND-CPA secure. This is because perfect correctness implies, for every valid ciphertext c, there exists a unique message-randomness pair (m, r) such that $c = \mathsf{Enc}(\mathsf{pk}, m; r)$. The two most commonly used choices of PKE, RSA and Elgamal, both meet these requirements, and can be used as commitment schemes.

However, IND-CPA security of PKE is not sufficient for guaranteeing validity of the resulting verifiable encryption, if the correctness is imperfect. For encryption schemes that are not perfectly correct, there can exist (m^*, r^*) such that $\mathsf{Dec}(\mathsf{sk}, (\mathsf{Enc}(\mathsf{pk}, m^*; r^*))) \neq m^*$. Then a malicious prover may be able to craft a ciphertext c^* that can be correctly opened to plaintext m^* such that it passes validity checks performed by a verifier, while c^* decrypts to junk during the recovery phase.

We will show two examples of such schemes, one based on decisional composite residuosity (Section 6.2), and one based on the learning with errors (LWE) problem (Section 6.4). In general, the base encryption scheme of post-quantum lattice-based candidates like FrodoKEM [NAB+19] and Kyber [SAB+20] are CPA secure, but not perfectly correct, and even the complete CCA-secure schemes may still be incorrect with bounded probability.

To prevent this attack, we require an additional property called *undeniability*. Intuitively, undeniability forces an adversary $\mathcal{A}$ to open any ciphertext to the plaintext identical to the result of decryption even if $\mathcal{A}$ may bias the randomness r. In Section 1.2 we discuss some similar (but different) notions from the literature.

Definition 3 (Undeniability)**.** We say that a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is ϵ_{und}-*undeniable* if for any PPT adversary $\mathcal{A}$:

$$\Pr\left[m \neq m' \,\wedge\, c = \mathsf{Enc}(\mathsf{pk}, m; r) \,:\, \begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa); \\ (c, m, r) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}); \\ m' := \mathsf{Dec}(\mathsf{sk}, c) \end{array} \right] \leq \epsilon_{\mathsf{und}}(\lambda)$$

The following utility lemma guarantees that an undeniable IND-CPA encryption scheme can be used as a secure extractable commitment with the simple construction given above.

Lemma 1. *If* PKE *is* ϵ_{und}-*undeniable and* ϵ_{cpa}-*IND-CPA secure, then the commitment scheme* ECOM *constructed from* PKE *is* ϵ_{cext}-*extractable with* $\epsilon_{\mathsf{cext}} \leq \epsilon_{\mathsf{und}}$, ϵ_{bind}-*binding with* $\epsilon_{\mathsf{bind}} \leq \epsilon_{\mathsf{und}}$ *and* ϵ_{hide}-*hiding with* $\epsilon_{\mathsf{hide}} \leq \epsilon_{\mathsf{cpa}}$.

Proof. We prove the three properties separately.

Extractability follows from undeniability. That is, if the adversary can output a tuple (c, m, r) breaking the extractability of ECOM, it also holds that $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and $m \neq \mathsf{Dec}(\mathsf{sk}, c)$. Therefore, (c, m, r) is also an instance breaking undeniability.

Binding follows from undeniability. Suppose there exists an adversary that outputs a tuple (m, r, m', r', c) such that it breaks binding with non-negligible probability, i.e., $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$. Given such an efficient adversary $\mathcal{A}$ against the binding game, we construct another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break undeniability as follows.

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ as input, $\mathcal{B}$ forwards it to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs (c, m, r, m', r') such that $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$, the $\mathcal{B}$ first decrypts c: $\tilde{m} = \mathsf{Dec}(\mathsf{sk}, c)$ and proceeds as follows: (a) If $\tilde{m} \neq m$, then $\mathcal{B}$ outputs (c, m, r) in the undeniability game, and (b) If $\tilde{m} \neq m'$, then $\mathcal{B}$ outputs (c, m', r') in the undeniability game.

Note that at least one of 2(a) or 2(b) must occur since $m \neq m'$. In either case, $\mathcal{B}$ successfully wins the undeniability game as long as $\mathcal{A}$ breaks binding. Clearly $\mathcal{B}$ succeeds with the same probability as $\mathcal{A}$, and $\mathcal{B}$'s runtime is the same as $\mathcal{A}$'s plus the cost of one Dec operation.

Hiding follows from the IND-CPA security of PKE. Concretely, if there exists a PPT distinguisher for commitment $c = \mathsf{Commit}(\mathsf{pk}, m_b; r) = \mathsf{Enc}(\mathsf{pk}, m_b; r)$ one can clearly construct a distinguisher for the IND-CPA game. That is, to break the IND-CPA game, the reduction first receives two messages (m_0, m_1) from the hiding adversary. Then by forwarding (m_0, m_1) to the IND-CPA challenger the reduction obtains the challenge ciphertext c^*, which can be also seen as a challenge commitment in the hiding game. If the hiding adversary can distinguish whether c^* is a commitment to m_0 or m_1 with advantage ϵ_{hide}, then the reduction can also distinguish whether c^* encrypts m_0 or m_1 with advantage ϵ_{hide}, which is at most ϵ_{cpa} by definition. $\qquad\square$

How to construct undeniable PKE Validity of our generic compiler described in the next section heavily relies on extractable commitments. The straightforward construction of ECOM requires undeniability, which is not necessarily satisfied by public-key encryption schemes with *statistical correctness*. As we shall see in Section 6.2, this is not just a limitation in a security proof; a lack of undeniability actually allows cheating provers to break validity entirely. A natural question is whether one can generically add the undeniable property to *any* IND-CPA-secure encryption scheme with statistical correctness. We answer this question in the affirmative by proving that several variants of the Fujisaki–Okamoto transform [FO99, FO13, HHK17] can make a given PKE scheme undeniable in the random oracle model.

For example, suppose we are given an encryption function Enc that takes a public key, message, and random value as input, and a random oracle G that hashes into the randomness space of Enc. The simplest FO transform [FO99] defines Enc' such that

$$\mathsf{Enc}'(\mathsf{pk}, m; r) := \mathsf{Enc}(\mathsf{pk}, m||r; \mathsf{G}(m||r)). \tag{1}$$

A crucial observation is that cheating provers are now forced to derive encryption randomness uniformly by querying the random oracle G. This makes it difficult to craft a malicious ciphertext c from biased randomness, which decrypts to a plaintext inconsistent with what she originally encrypted. Using the same observation we can also prove that well-known FO-based CCA conversion methods employed by Kyber and $\mathsf{FrodoKEM}$ achieve undeniability. Details are deferred to Appendix B.

3.2 Compiling MPCitH-IOP Into Verifiable Encryption

Our construction MPCitH-VE is given in Protocol 2. The description already incorporates the random subset optimization that will be analyzed in the next section. Here, we focus on the case of $n = \tau$ for simplicity. As for the intuition for our construction, we observed in Section 2.2 that for any MPCitH IOP following the [IKOS07] paradigm, there exists a (canonical) straight-line extractor that recovers the witness from the committed values of all parties. Recall that:

- The MPC protocol evaluates R with inputs x and w.

- The input x is public and w is shared amongst the parties.

- The view of each party must include their share of the witness and random tapes in order to allow verification to check consistency, since some of the outgoing messages of the parties must depend on both of these values.

Therefore, given the opening of the commitments of all parties (all N views), the extractor can recover the witness based on the shares of all parties. For constructing ZK proofs or signatures allowing for straight-line witness extraction, one can compile

MPCitH-IOP by letting a prover commit to every per-party view with random oracle commitments as in [Pas03, KKW18, ZCD⁺20, DFMS22]: the extractor can reconstruct a witness by observing the RO query history. However, this does not suffice for instantiating verifiable encryption because the receiver (i.e., decryptor) in the real-world clearly has no access to the query history.

Our compiler takes an alternative approach similar to [Kat21, HLR21], which simultaneously realizes a straight-line extractable ZK proof system and valid verifiable encryption scheme. By replacing the commitment function with an *extractable* commitment ECOM (as defined in previous section) where the recipient has the decryption key sk, the recipient can decrypt the commitments to the unopened view(s) and recover all openings, then use the extractor algorithm to recover a witness. We remark that our transform naturally generalizes to other types of MPCitH protocols as well, since all such protocols (we are aware of) allow extraction of a witness given the openings of the per-party commitments (and indeed use this in their security reductions).

Because our presentation assumes the witness is shared with an additive secret sharing scheme, we make use of this to compress the ciphertext, by summing the t revealed shares into the single value $\tilde{w}$. If the secret sharing scheme of Π_f does not allow such partial reconstruction, then the ciphertext may simply include all shares. When generalizing to other types of secret sharing schemes the decryption operation must also be generalized to reconstruct w from the shares of all parties.

Theorem 1. *Let* MPCitH-IOP$_R$ *be an MPC-in-the-head-based IOP in the class described by Protocol 1 that is perfectly HVZK and SLE with knowledge error $\epsilon_{\text{sle-iop}}$. Let* ECOM *be an extractable commitment scheme that has ϵ_{cext}-extractability and is ϵ_{hide}-hiding. Then the compiled protocol,* MPCitH-VE$_R$ *described in Protocol 2 with $n = \tau$, is ϵ_{val}-valid with validity error $\epsilon_{\text{val}} = \epsilon_{\text{sle-iop}} + \epsilon_{\text{cext}}$, and ϵ_{zk}-HVZK with $\epsilon_{\text{zk}} = \tau(N - t)\epsilon_{\text{hide}}$.*

HVZK directly follows from hiding of ECOM (and thus from IND-CPA of the underlying PKE). Proof of validity essentially proceeds as follows: if an MPCitH-VE cheating prover $\mathcal{P}^*$ can convince the verifier $\mathcal{V}$ while the receiver fails to decrypt a correct witness, then it must be that either (1) $\mathcal{P}^*$ broke extractability of ECOM, or (2) one can construct a pair of adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ that break SLE of MPCitH-IOP$_R$. Adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ first extract views from the commitments sent by $\mathcal{P}^*$ and then forward them as a complete set of N views in the SLE-IOP game. Formal proof is deferred to Appendix H.1.

Optimizations While the prover in our generic compiler MPCitH-VE commits to a complete per-party view V_i using ECOM, several standard optimization techniques in the literature also are applicable in our setting for better computational and communication complexities. Notice that $\mathcal{R}$ would only need witness shares $(w_i)_{i \in [N]}$ to be able to recover the plaintext. Hence, it would be sufficient to have the prover $\mathcal{P}$ commit to w_i using ECOM, and to the rest of the strings in V_i using the random oracle commitments as the ZKBoo/ZKB++ prover does [GMO16, CDG⁺17]. Since ECOM is instantiated with PKE in practice while the RO is instantiated with cryptographic hash functions, this would significantly reduce the size of transcripts and could save both prover and verifier time for creating/opening commitments.

Moreover, following [KKW18, §2.3] and subsequent works [ZCD⁺20, BDK⁺21, DOT21, KZ22], in case the MPC protocol Π_f relies on a broadcast channel and thus $N - 1$ out of N views are revealed, we can decouple broadcast messages $(\text{msgs}_i)_{i \in [N]}$ from per-party views to reduce the communication complexity, where each msgs_i consists of messages broadcast by party i. That is, the prover $\mathcal{P}$ first generates a root seed sd^* to derive per-party seeds $(\text{sd}_i)_{i \in [N]}$ with a binary tree construction. $\mathcal{P}$ now only commits to each seed sd_i used for deriving a witness share and a random tape of party i using ECOM, and sends $h = \mathsf{H}((\text{msgs}_i)_{i \in [N]})$. On receiving challenge $\bar{i} \in [N]$ from $\mathcal{V}$, indicating the index of unopened party, $\mathcal{P}$ reveals $\text{msgs}_{\bar{i}}$ and $\lceil \log_2(N) \rceil$ nodes in the tree, which are sufficient to

compute $(\mathsf{sd}_i)_{i \in [N] \setminus \{\bar{i}\}}$. From such information, $\mathcal{V}$ can reconstruct the remaining broadcast messages, check h against broadcast messages sent by all N parties, and check that $N - 1$ parties on input $(\mathsf{sd}_i)_{i \in [N] \setminus \{\bar{i}\}}$ lead to a correct output with respect to x and $\mathsf{msgs}_{\bar{i}}$. Our DKG-in-the-head protocol in Section 5.1 benefits from these optimizations.

Protocol 2: MPCitH-VE$_R$

Converts the MPCitH-IOP prover **P** and verifier **V** to an MPCitH-VE prover $\mathcal{P}$ and verifier $\mathcal{V}$ using the the extractable commitment scheme $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as constructed in Section 3.1.

Parameters: The number of parties N; the number of parallel repetitions τ; the number of opened parties t; the challenge set $\mathsf{Ch} = \{\mathbf{e} \in [N] : |\mathbf{e}| = t\}$; the subset size for compression n.

Key Generation $\mathcal{G}(1^\kappa)$: It invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^\kappa)$ and outputs $(\mathsf{pk}, \mathsf{sk})$.

Two-party protocol $\langle \mathcal{P}(w), \mathcal{V} \rangle(\mathsf{pk}, x)$**:**

1. $\mathcal{P}$ runs **P** on input (x, w) to obtain the proof string $\pi = (V_1, \ldots, V_N)$.

2. $\mathcal{P}$ separately commits to each of these N views to generate per-party commitments $(\mathsf{C}_1, \ldots, \mathsf{C}_N)$ where $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ and r_i is commitment randomness uniformly sampled from S_r.

3. $\mathcal{V}$ invokes **V** on input x to obtain challenge $\mathbf{e} \in \mathsf{Ch}$, and sends it to $\mathcal{P}$.

4. $\mathcal{P}$ opens the commitments of the t parties, by revealing $(V_i, r_i)_{i \in \mathbf{e}}$.

5. $\mathcal{V}$ sends the views $(V_i)_{i \in \mathbf{e}}$ to **V** as a response to the oracle query. It accepts if and only if:

   (a) $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ and $r \in S_r$ for all $i \in \mathbf{e}$, i.e., $\mathcal{P}$ opened the views corresponding to $(\mathsf{C}_i)_{i \in \mathbf{e}}$ successfully, and

   (b) **V** outputs 1.

$\mathcal{P}$ and $\mathcal{V}$ execute τ instances of the above protocol in parallel. If $\mathcal{V}$ accepts in all τ executions, it outputs $b = 1$ and a transcript

$$\mathsf{tr} = ((\mathsf{C}_i^{(j)})_{i \in [N]}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]} \ .$$

Otherwise, $\mathcal{V}$ outputs $b = 0$ and $\mathsf{tr} = \bot$.

Compression $\mathcal{C}(x, \mathsf{tr})$**:**

1. It samples a subset $S \subseteq [\tau]$ of size $n \leq \tau$ uniformly at random.

2. For $j \in S$, extract the t witness shares $w_i^{(j)} = \mathsf{GetW}(V_i^{(j)})$ for $i \in \mathbf{e}^{(j)}$ and partially reconstruct the witness $\tilde{w}^{(j)} = \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)}$.

3. Output the compressed ciphertext $C = (\tilde{w}^{(j)}, (\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}})_{j \in S}$.

Receiver $\mathcal{R}(\mathsf{sk}, C)$: To decrypt the ciphertext C, the receiver proceeds as follows.

1. For $j \in S$ and $i \notin \mathbf{e}^{(j)}$, extract the unopened parties' views $\hat{V}_i^{(j)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i^{(j)})$ and computes the corresponding witness shares $\hat{w}_i^{(j)} = \mathsf{GetW}(\hat{V}_i^{(j)})$. Let $w^{(j)} = \tilde{w}^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}$ be the jth candidate witness.

2. If there exists some $j \in S$ such that $(x, w^{(j)}) \in R$, output $w^{(j)}$. Otherwise, output $\bot$.

3.3 Compiling Other MPC-in-the-Head Proofs

Although the IOPs corresponding to KKW and Banquet (given in Appendix E) are not exactly in the class described by MPCitH-IOP, we can compile them into verifiable encryption schemes using essentially the same idea.

To compile Banquet-IOP, it is sufficient to have the VE prover $\mathcal{P}$ commit to the per-party seeds $(\mathsf{sd}_i)_{i \in [N]}$ with an extractable commitment scheme during the first round. The second and third round operations are identical to the original Banquet-IOP protocol, and the VE verifier $\mathcal{V}$ proceeds by following the decision phase of Banquet-IOP and accepts iff **V** accepts and the $N-1$ per-party commitments are opened correctly. The compression and receiver algorithms $\mathcal{C}$ and $\mathcal{R}$ are defined analogously to those of MPCitH-VE, except that the witness offset Δw is added by $\mathcal{C}$ when creating a partially reconstructed witness $\tilde{w}$. Since the receiver tries to decrypt by using the SLE extractor algorithm defined in Lemma 8, the compiled protocol has ϵ_{val}-validity with $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{cext}} + \epsilon_{\mathsf{sle}}$, assuming ϵ_{cext}-extractability of ECOM and ϵ_{sle}-SLE of Banquet-IOP. Appendix F provides detailed analysis of Banquet-based VE.

Likewise, we can compile KKW-IOP by having the VE prover $\mathcal{P}$ commit to the offline per-party states $(\mathsf{st}_i^{(j)})_{i \in [N]}$ with ECOM. On the other hand, the other commitments in KKW-IOP can be instantiated with the usual random oracle commitments as in the original KKW protocol. As we only need τ revealed online executions to recover a witness, the compression algorithm $\mathcal{C}$ outputs as a ciphertext $\tilde{w}^{(j)} = \sum_{i \neq \bar{i}_j} \lambda_i^w \oplus \hat{w}^{(j)}$ and $\mathsf{C}_{\bar{i}_j}^{(j)}$ for $j \in T \subset [M]$, where each witness mask share λ_i^w is obtained from the revealed value $\mathsf{st}_i^{(j)}$. Then the receiver $\mathcal{R}$ extracts the unopened share of the witness mask from $\mathsf{C}_{\bar{i}_j}^{(j)}$ and XORs it with $\hat{w}^{(j)}$ to recover a candidate witness.

Compiling Limbo is straightforward since the protocol of [DOT21, Fig.5] is already presented using the language of IOPs. The VE prover uses ECOM to commit to each witness share as part of the first oracle, and the rest of the proof string is committed with the existing commitment scheme.

Compiling protocols utilizing robustness and/or (t, N)-threshold LSSS-based protocols such as [AHIV17, BFH+20, FR23] (§E.2.1-E.2.2) is also possible and eliminates the need for parallel repetitions. E.g., if instantiated with Shamir secret sharing, the prover encrypts $w_i = f_w(i)$ for $i = 1, \ldots, N$ where $f_w \in \mathbb{F}[X]$ is a degree t polynomial encoding w in its constant term and containing uniformly random coefficients otherwise. The verifier asks the prover to open views of parties in $I \subset [N]$ with $|I| = t$. To recover the witness, the receiver then descrypts remaining shares and invokes the reconstruction algorithm of Shamir or a suitable decoding algorithm if the scheme relies on Reed-Solomon code. The validity analysis is rather straightforward given the knowledge soundness of these schemes, because undeniable PKE can be seen as a straight-line extractable commitment that replaces RO-based commitment, and the receiver of VE essentially acts as a knowledge extractor. However, our concrete instantiations mainly focus on simple non-robust MPC-in-the-Head since they typically perform better in the context of the simple NP-relations considered in our intended applications (e.g. AES circuits).

3.4 Applying Fiat–Shamir

Following the standard Fiat–Shamir transform [FS87], we can make our verifiable encryption protocol MPCitH-VE non-interactive in the random oracle model, by hashing the first prover messages together with x and pk to obtain the challenge $\mathbf{e} \in \mathsf{Ch}$. Since the base interactive protocol has three rounds, the FS transform introduces a multiplicative factor of q security loss in validity, where q is the number of random oracle queries made by a non-interactive cheating prover. Note that this loss is well-known in (knowledge) soundness analysis for FS-NIZK proofs and EUF-KOA security of signatures constructed from canonical

identification schemes [KMP16]. Formal analysis is deferred to Appendix C. Banquet-based verifiable encryption however requires a separate concrete analysis dedicated to the non-interactive version, since it has 7 rounds of interaction. Because the EUF-KOA security analysis of Banquet as a signature scheme [BDK$^+$21, Theorem 2] already evaluates the probability that the witness (i.e., secret signing key) extraction fails, their analysis can be reused in large part to derive the concrete validity error of non-interactive Banquet-VE. Construction of Banquet-NIVE and validity analysis are deferred to Appendix F.

3.5   Achieving Strong Validity

To the best of our knowledge, prior definitions of validity for verifiable encryption in the literature assume that the key generation phase is always performed honestly. One can strengthen the validity property so that a cheating prover takes control of key generation. Formally, we say a VE scheme has ϵ_{sval}-*strong validity* if for all pairs of PPT adversaries $(\mathcal{A}^*, \mathcal{P}^*)$,

$$
\Pr \left[\begin{array}{cc} b = 1 \wedge & (x, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{A}^*(1^\kappa); \\ (x, w') \notin R \wedge : & (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}^*(\mathsf{sk}), \mathcal{V} \rangle(\mathsf{pk}, x); \\ (\mathsf{pk}, \mathsf{sk}) \in \mathcal{G}(1^\kappa) & C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{sval}}(\kappa).
$$

We remark that allowing $\mathcal{A}^*$ to choose $(\mathsf{pk}, \mathsf{sk})$ is very strong, and that in practice it's not possible to check whether $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{G}(1^\lambda)$. However, without this condition, note that $\mathcal{A}^*$ can trivially break strong validity by generating a keypair then setting sk to 0. In the context of our verifiable key backup scenario, the device could be encrypting the key to a future instance of itself, or to another device in the same security domain. Here the user must trust that the device importing the key has generated its keypair honestly. This seems to be the best possible validity assurance when the device is responsible to store sk.

If ECOM is instantiated with a perfectly correct PKE, we can achieve strong validity of MPCitH-VE. Observe that if PKE has perfect correctness, then for every key pair and for every ciphertext, the corresponding plaintext is uniquely determined. Therefore, as long as the key pair is in the right domain (which the receiver can easily check) undeniability can never be broken regardless of the distribution of keys.

4   Compressing Ciphertexts

Because MPCitH protocols use τ parallel repetitions to boost soundness, the ciphertexts output by our transform can be large. For example, for 128-bit security, τ could range from 20 to 219. Each repetition outputs one PKE ciphertext and a share of the witness, so the total size is $\tau(|\mathsf{PKE.Enc}| + |w|)$. Also, in the post-quantum PKE case, lattice-based constructions can have relatively large ciphertexts. An interesting question is whether these can be compressed, since these ciphertexts will usually be very redundant: note that for an honestly created proof all τ repetitions encrypt the same witness (in different ways), and the receiver will only need to decrypt one.

In this section we give two methods to compress the verifiable encryption ciphertexts output by schemes created with our transform. The first, called the *random subset method*, is very simple, incurs no computational overhead, and can reduce ciphertext size by a factor of three when τ is large.

The second approach, called the *equality proof method*, is optimal as it achieves constant size ciphertexts, $O(|w|)$ (provided PKE has constant ciphertext expansion). However, it requires special properties of PKE, increases proof size, prover and verifier computational costs significantly, so it is more of a possibility result rather than a practical construction. We defer detailed description of the equality proof method to Appendix G, and give only
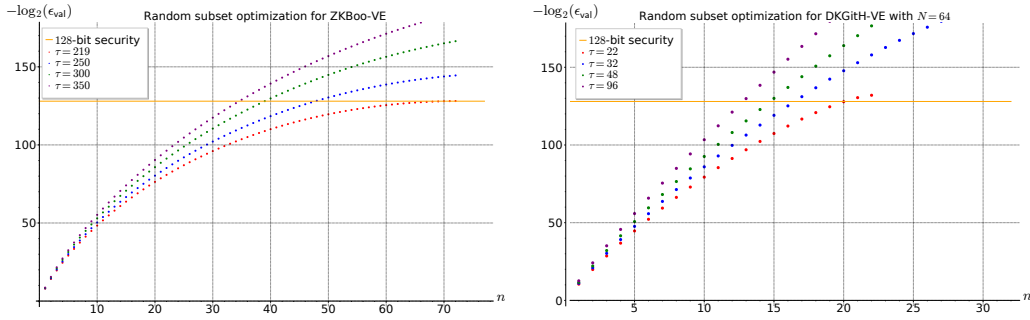
Figure 1: Approximate minimum cost of breaking validity of ZKBoo-based VE (left) and DKGitH-based VE (right) with a random subset of size n. The parameter τ denotes the number of parallel repetitions. The number of parties N is fixed to 3 for ZKBoo and 64 for DKGitH, respectively. Note that $\tau = 219$ corresponds to the `picnic-L1` parameters from the Picnic spec [ZCD+20].

the high-level idea here. In an honestly generated proof, all component ciphertexts are valid, and decryption will always succeed on the first attempt. If after the VE protocol, the prover were able to additionally prove that $\mathcal{R}$ would output the same witness from all of the component ciphertexts, then the verifier could keep only one of the component ciphertexts, making the VE ciphertext constant size. This is because either: all values are equal and correct, or all values are equal and incorrect, but the latter case is equivalent to creating an invalid proof, which is possible with only negligible probability by soundness of the proof protocol.

Note that the equality proof proves that $\mathcal{R}$ outputs the same value for all component ciphertexts – *and is not requiring that we prove the relation.* The crux of $\mathcal{R}$ for MPCitH protocols is recombining additive shares of the witness, a comparatively simple operation. However one of the shares is encrypted, meaning we are back to proving something about encrypted data. We describe one instantiation of the idea to show that this is possible without resorting to general methods, by using PKE in a non-black-box way.

4.1   The Random Subset Method

This compression method is rather simple, but the impact on ciphertext size can be significant, and the cost to the prover is nothing, and almost nothing to the verifier. That is, we set $n < \tau$ in Protocol 2 to optimize the compression and receiver algorithms. Upon receiving a verifiable encryption proof with our transform, the verifier has a set of τ ciphertext components, corresponding to the τ parallel repetitions used to produce the proof. The verifier chooses a subset of the ciphertexts to keep at random, and discards the others. The size of the subset is denoted n, and is a parameter of the method.

We stress that soundness of the proof is unchanged, since the entire is proof is communicated to the verifier and checked. Only the analysis of the validity error must be updated, since the receiver now has only n ciphertexts. Naïvely, one may be able to bound the validity error by $\epsilon_{\mathsf{sle-iop}}(n)$ which corresponds to the adversary's advantage in convincing the verifier while using n bad repetitions (out of τ) leading to an invalid witness if decrypted. However, this loose bound does not help us reduce the ciphertext size. In fact, thanks to the fact that VE verifier picks a random subset *after* the verification checks are completed, the actual validity error is significantly lower than $\epsilon_{\mathsf{sle-iop}}(n)$, since the adversary must make sure that all the bad repetitions fall in the uniformly selected subset of size n for the receiver to fail.

More formally, let s be the number of ciphertexts in the initial set of size τ that are

bad, meaning they do not decrypt to the witness. For the proof systems we consider, having $s > 0$ is quite easy, as it only requires guessing a small part of the challenge. Note that s must be at least n, otherwise the attack against compression never succeeds, since $\mathcal{V}$'s output always contains one or more valid ciphertexts.

Below we will choose parameters for the random subset method applied to different proof systems, in the interactive case. The adversary $\mathcal{P}^*$, is a cheating prover who knows the witness, and tries to create a verifiable ciphertext where decryption fails. Then the general form of $\mathcal{P}^*$'s success probability is

$$\Pr\left[\mathcal{C} \text{ selects } n \text{ of } s \text{ bad ctexts } \wedge \ \mathcal{V} \text{ accepts a proof with } s \text{ bad ctexts}\right]$$

$$= \frac{\#\text{subsets with } n \text{ bad ctexts}}{\# \text{ of subsets}} \cdot \Pr\left[\mathcal{V} \text{ accepts a proof with } s \text{ bad ctexts}\right] = \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\epsilon_{\mathsf{sle\text{-}iop}}(s) + \epsilon_{\mathsf{cext}}\right)$$

where $\epsilon_{\mathsf{sle\text{-}iop}}(s)$ is the probability that an IOP prover wins the SLE-IOP game with s parallel repetitions, and "ctexts" is short for ciphertexts. A more formal analysis is given in Appendix H.2, where we prove the following theorem.

Theorem 2. *Let* $\mathsf{MPCitH\text{-}IOP}_R$ *be an MPC-in-the-head-based IOP in the class described by Protocol 1 with SLE knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}}$. *Let* ECOM *be an extractable commitment scheme with* ϵ_{cext}*-extractability. Then* $\mathsf{MPCitH\text{-}VE}_R$ *is* ϵ_{val}*-valid with validity error*

$$\epsilon_{\mathsf{val}}(\tau, n) = \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\epsilon_{\mathsf{sle\text{-}iop}}(s) + \epsilon_{\mathsf{cext}}\right).$$

Generally, the amount of compression possible is larger when τ is larger, as demonstrated by the ZKB++ example (where $\tau = 219$ for 128-bit security). The DKGitH example requires much smaller τ (in the range 16–32), and compression is limited, or none at all. However, we can increase τ to larger values than strictly necessary, in order to compress the ciphertext further, see Fig. 1 for a range of options with fixed N and the first row of Table 1 for a concrete example. This reduces ciphertext size at the expense of proof size, which can be beneficial in applications that check the proof then discard it, but store the ciphertext.

Application to IKOS/ZKBoo/ZKB++ We consider interactive IKOS-style protocols, such as ZKBoo and ZKB++. For each repetition of the protocol, they have $\binom{N}{2}$-consistency, where N is the number of parties. As ZKBoo and ZKB++ have $N = 3$ and $\mathsf{Ch} = \{1, 2, 3\}$ they have 3-consistency and thus are SLE with knowledge error $\epsilon_{\mathsf{sle\text{-}iop}}(s) \leq 2/3$ from Lemma 4. In Fig. 1 we show the costs of breaking validity $-\log_2(\epsilon_{\mathsf{val}}(\tau, n))$ for different combinations of τ and n assuming ϵ_{cext} is negligible. We see that $n = 70$ provides 128-bit security with $\tau = 219$ repetitions, meaning we can compress ciphertexts by a factor 3 at no cost. If we increase τ slightly to 250 (meaning proof size and prover/verifier time increase by roughly 1.14x) then we can set $n = 50$ and compress ciphertexts by a factor 4.4.

Application to DKGitH This is similar to IKOS, except that the default soundness error is different. Because the corresponding MPC protocol uses a broadcast functionality, the prover reveals $N-1$ parties' views and thereby the knowledge error is at most $1/N$, instead of $1 - 1/\binom{N}{2}$. In Fig. 1 we show the costs of breaking validity for different combinations of τ and n assuming ϵ_{cext} is negligible. As τ is smaller, the amount of compression we get for free is limited to only 2 ciphertexts (i.e., we can set $n = 20$ when $\tau = 22$). The option of increasing τ is again possible, but provides less compression and at a higher cost. In addition to the choices of (n, τ) given in Fig. 1, Table 1 gives some concrete examples showing proof and ciphertext size along with estimates of the prover and verifier times.

5   Concrete Instantiations

In this section we give some instantiations of our transform. We implement verifiable encryption of AES keys, and three schemes for discrete logarithms (suitable for encrypting, e.g., ECDSA, ECDH and Ed25519 private keys) and provide performance benchmarks. We also describe how our scheme for discrete logs can be adapted to verifiably encrypt RSA private keys and plaintexts.

Interactivity All of the benchmarks are given for the non-interactive versions of proofs. However, we note that it is also possible in many applications (such as in verifiable key backup) where the verifier will only accept a small number of failed attempts by a prover, to use an interactive proof with 40–64 bits of interactive security (analogous to the case of interactive identification schemes [FS87, Section 2.3]). For the MPCitH protocols we consider that use parallel repetition, this reduces number repetitions significantly, in turn reducing the prover and verifier time, proof size and ciphertext size by a factor 2–3.

5.1   Verifiable Encryption of Discrete Logs in Prime Order Groups

Perhaps the most fundamental relation in cryptography is the discrete logarithm in a prime order group $\mathbb{G}$, i.e., (y, x) such that $y = g^x$ where $\langle g \rangle = \mathbb{G}$. As an application our transform, we give a new protocol to verifiably encrypt a discrete logarithm. We construct an MPC protocol to compute y from shares of x, which naturally gives an MPCitH protocol to prove knowledge of x. When compared to the most efficient proof of knowledge for discrete logarithms, the Schnorr proof, our new protocol is much less efficient, but it is amenable to our transform, and can therefore be used to verifiably encrypt discrete logs. We can then verifiably encrypt DH, ECDH, DSA and ECDSA keys directly as key pairs for these algorithms are discrete log instances, and in Section 5.1.3 we explain how this scheme can also be used to encrypt RSA keys.

As an aside, we remark that our new proof protocol has a tight reduction to the discrete logarithm problem in the random oracle model. This feature is of theoretical interest as it implies a signature scheme based on the discrete logarithm problem with a tight security reduction.

Baselines for Comparison We compare to two protocols from the literature. The first is the Camenisch-Damgård protocol [CD00] for a generic Σ protocol, combined with Schnorr's Σ-protocol [Sch91] for discrete logs with binary challenges. This is the only verifiable encryption scheme we are aware of that works for discrete logarithms in any cyclic group, and allows a flexible choice of PKE (as our protocol does). It also requires the random oracle assumption to make the proof non-interactive.

The second, more efficient, protocol in [CD00] has k parallel repetitions, and the verifier selects a subset to form the output, and audits the encryption step of the $k - u$ other repetitions (and the verifier checks all repetitions have a valid transcript for the Σ protocol with one challenge). No parameters are given for concrete, non-interactive security – we found that for λ-bit security, (k, u) must be chosen so that $\binom{k}{u} \geq 2^\lambda$. Then there are multiple possible choices for (k, u), which trade ciphertext size for computation: we can have a small decrease in ciphertext size, for a large increase in computation and proof size. Our comparison in Table 1 gives some of the options.

Another VE scheme we compare to is from [NRSW20], which can encrypt a discrete logarithm in an elliptic curve group, using a special PRF called Purify. The scheme does allow, e.g., encryption of an ECDSA private key, but requires that encryption be done with an Elgamal-like PKE. A complication related to implementation of the Purify PRF is that one must choose an additional pair of elliptic curves, related to the group order of the curve where the discrete logarithm is defined, such that the DDH assumption holds. In addition to making these additional parameter choices, we must also make an assumption beyond the DLP + PKE assumptions in $\mathbb{G}$ (as in [CD00] and our scheme).

We omit a detailed comparison to [CS03] since it only works for discrete logarithms in a group suitable for Paillier's encryption scheme, and the PKE is fixed to Paillier's scheme. The scheme is not suitable for encrypting an ECDSA private key, one of our motivating examples. That said, due to the high cost of arithmetic mod $\mathbb{Z}_{n^2}$ where n is 3072–4096 bits, we estimate that our DKGitH proof always outperforms [CS03] in terms of prover time, verifier time and ciphertext size. To support these conclusions, our software package provides some detailed estimates for [CS03], along with the software used to benchmark $\mathbb{Z}_{n^2}$ arithmetic.

We chose [NRSW20] and [CS03] as baselines for comparison for encrypting discrete logarithms rather than a zkSNARK since SNARKs generally require significantly stronger assupmptions when compared to our protocol. An exception in this regard is Spartan [Set20], instantiated in the group where the discrete log is defined. In this case the assumptions are comparable to our scheme and [CS03], however VE with Elgamal would require proving two scalar multiplications "non-natively", i.e., as an arithmetic circuit modulo the group order, which would require at least two million R1CS constraints. An R1CS instance of this size would require tens of seconds to prove and have proofs over 100 KB in size [Set20, Figures 7-10].

5.1.1   Encrypting Discrete Logs with DKG-in-the-head

We first describe the base non-interactive ZK proof system DKGitH for relation $R = \{(y, x) : y = g^x\}$.[2] The core idea of the protocol is based on the additive homomorphism of private keys, under multiplication of public keys, and may be folklore (an early reference describing it is [Ped92]). To compute $f(x) = g^x =_? y$ in a distributed manner, the prover $\mathcal{P}$ provides shares of x to the N parties such that $x = \sum_{i=1}^{N} x_i \pmod{p}$. Then $\mathcal{P}$ emulates a simple distributed key generation (DKG) protocol Π_f that proceeds as follows.

1. Each party i computes $y_i = g^{x_i}$, and broadcasts y_i.
2. Output the public key $y = \prod_{i=1}^{N} y_i$

$\mathcal{P}$ commits to the shares of the parties, and the y_i values (together these two values makeup party P_i's view), then the verifier $\mathcal{P}$ selects one party to remain unopened, having index $\bar{i}$. In the response, the prover sends the views of the other $N-1$ parties, along with $y_{\bar{i}}$, and a commitment to $x_{\bar{i}}$. Based on the revealed values, $\mathcal{V}$ checks that $y = y_{\bar{i}} \prod_{i \in [N], i \neq \bar{i}} g^{x_i}$ and that each y_i is computed correctly.

To realize VE, $\mathcal{P}$ encrypts x_i's in the committing phase, which allows a receiver $\mathcal{R}$ to reconstruct the DLog of y by decrypting an unopened share $x_{\bar{i}}$. Along with the core idea, the full protocol in Protocol 4 uses two ideas (originating in [KKW18]) that are now standard in protocols of this type. First, the shares of the parties are computed by reading random values from their tapes, and the first share is corrected with an auxiliary value that depends on the secret. Second, the tapes are derived from a seed with a binary tree construction, so that the $N-1$ revealed seeds can be communicated more efficiently by revealing $\lceil \log_2(N) \rceil$ seeds.

In Appendix D.1, we provide the full protocol, prove security, describe how to chose parameters for 128-bit concrete security, describe the hashed Elgamal PKE we use, and describe the optimizations we apply once these choices are fixed. We then explain how we obtained the size and speed benchmarks used in this section.

---

[2]We present the final VE scheme to illustrate our implementation details. The IOP implicit in DKGitH is sketched in the validity analysis in Appendix D.1.

5.1.2 Encrypting Discrete Logs with Robust DKG-in-the-head

One can consider a variant of the above protocol by having a prover $\mathcal{P}$ run Feldman's VSS protocol in-the-head [Fel87]:[3]

1. Party 0 (dealer), upon receiving the witness (DLOG) x as input, samples uniformly random $a_i \in \mathbb{F}$ for $i = 1, \ldots, t$ and lets $a_0 = x$. Define a degree-t polynomial $a(X) = a_0 + a_1 X + \ldots + a_t X^t$.

2. For $i = 1, \ldots, N$, send to party i a Shamir secret share $x_i = a(i)$ of x. Moreover, broadcast a commitment to the polynomial $A_0 = g^x, A_1 = g^{a_1}, \ldots, A_t = g^{a_t}$.

3. Each party i checks the validity of its share: $g^{x_i} =_? \prod_{j=0}^{t} A_j^{i^j}$.

The corresponding VE protocol instantiated with the hashed Elgamal PKE is detailed in Appendix D.2. At a high-level, $\mathcal{P}$ encrypts the N shares separately, and upon receiving a challenge $I \subset [N]$ with $|I| = t$ from $\mathcal{V}$, she opens x_i for $i \in I$. Then $\mathcal{V}$ checks the validity of revealed shares. At this point, the receiver $\mathcal{R}$ could take the t opened shares and the remaining ciphertexts as input, and run Lagrange interpolation to recover the witness. But by exploiting the additive homomorphism (over $\mathbb{Z}_p$) of hashed Elgamal, we observe that one can delegate this task to the compression algorithm. Our optimized instantiation entirely avoids interpolation within the recovery algorithm.

5.1.3 Verifiable Encryption of RSA Keys

There are two natural ways to generalize the DKG-in-the-head idea to the RSA setting. First, we can verifiably encrypt an RSA private exponent d, by using the above proof of a discrete logarithm to prove knowledge of d such that $(m^e)^d = m \pmod{n}$, where (e, n) is an RSA public key and m is an arbitrary value. Thus we can efficiently verifiably encrypt RSA encryption and signing keys.

Second, we can prove knowledge of a preimage of a one-way group homomorphism. For example, if the homomorphism is $\phi : m \mapsto m^e \bmod n$ with $n = p \cdot q$, one can design a simple MPCitH protocol for knowledge of an RSA preimage: the parties share m multiplicatively, $m = m_1 \cdots m_N \pmod{n}$ then broadcast $\phi(m_i) = m_i^e$, and then check that $c = \prod m_i^e \pmod{n}$. This can be used to prove knowledge of an RSA plaintext corresponding to a given ciphertext (a more direct type of verifiable encryption), or knowledge of a message corresponding to a given signature. The MPC protocol can be extended to prove additional properties of m as well.

5.2 Verifiable Encryption of AES Keys

With our transform applied to Banquet-IOP, one can verifiably encrypt an AES private key used for generating a given public ciphertext. Concretely, since Banquet-IOP is specialized for the relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$, one can verifiably encrypt K satisfying the relation R with any PKE. The compiled VE scheme Banquet-NIVE and validity proof are detailed in Appendix F. To the best of our knowledge, no prior work proposed a verifiable encryption scheme for AES private keys. As AES keys are commonly stored in hardware, this is also relevant for our verifiable backup scenario. Since AES is considered PQ-secure, and encrypted data may have a long lifetime, in some systems it is important that AES keys be exported with a matching level of security. If PKE is instantiated with a quantum-resilient scheme, such as a lattice-based one, our verifiable encryption has PQ security, in the sense that both the *encryption scheme* and *relation* to be proven about the plaintext may withstand quantum attacks.

---

[3]We thank Yashvanth Kondi for letting us know the existence of a similar protocol in his unpublished manuscript.

Verifiably encrypting an AES key with the hashed Elgamal scheme described in Appendix D.1 has proof sizes that are only slightly larger than Banquet proofs for AES, since the ciphertexts are only 16 bytes larger than hash-based commitments. For example, proofs are 20.4 KB ($N = 16, \tau = 41$), an overhead of less than 1KB, and ciphertexts are 2 KB. Prover and verifier times are dominated by the cost of computing encryptions, but we estimate the total time to be below 100ms (based on the time estimates used above and those from [BDK$^+$21]).

As we analyze in Appendix B, variants of the FO transform can be used for achieving undeniability and thus many efficient post-quantum PKE schemes, including Kyber [SAB$^+$20] and FrodoKEM [NAB$^+$19], are compatible with our framework.

Our implementation, AES-VE, uses Kyber as a PKE and is based on the Helium-AES proof system [KZ22], an IOP with the same structure as Banquet but further optimized for the AES relation. Table 1 gives benchmarks for our implementation, for three choices of parameters, showing that one can trade larger proof and ciphertext sizes for speed of the prover and verifier. In the parameters yielding the shortest proofs and ciphertexts (approx. 22 KB and 13 KB, respectively) the prover and verifier run in about 67 ms, compression is about 2 ms and decryption (not shown) is below 1 ms.

5.3   Benchmarks and Comparison

In Table 1 we present benchmarks from the four verifiable encryption schemes we implemented, and one that we provide estimates from the literature.[4] We implement both the robust and normal variants of the DKG-in-the-head protocol for proving knowledge of discrete logarithms in prime order groups. Our Rust implementation uses the `secp256r1` elliptic curve group, via generic APIs of the `arkworks` library [ac22] allowing our code to change to one of the many other curves `arkworks` supports. Then for comparison, we implement the CD (Camenisch-Damgard [CD00]) scheme, as described above. We provide sizes for the NRSW scheme from [NRSW20] scheme, and estimate the runtimes of their proof generation and verification by scaling their reported runtimes to the frequency of our processor. Note however that their implementation is optimized to use properties of the `secp256k1` elliptic curve and may not perform as well on other curves. Finally, our AES-VE implementation is based on the C++ implementation of Helium-AES [Kal22] for AES-128 and uses the AVX2 optimized Kyber implementation from PQClean [KSSW22]. All of the parameters were chosen to meet the 128-bit security level; for Kyber we use the L1 parameter set which is expected to match the security of AES-128. Our benchmark machine has an Intel Xeon W-2133 CPU @ 3.60GHz. The table gives the parameters we use for each scheme, the size in bytes of the transcript tr, the VE ciphertext $|C|$, (also with random subset (RS) compression, column $|C|_{RS}$), as well as the computational costs of the prover $\mathcal{P}$, verifier $\mathcal{V}$ and the compression algorithm $\mathcal{C}$. The final decryption cost by the receiver $\mathcal{R}$ was always well below 1ms, so we omit it from the table.

All of the schemes in Table 1 (except [NRSW20]) allow one to trade-off size of proof and ciphertext for speed. We provide benchmarks for different combinations of parameters to showcase the range of options available to applications. For instance, in a group signature or identity escrow scheme the ciphertext size may be the most important metric to optimize. When exporting an encrypted key from a constrained device (such as a YubiKey) prover time may be most important. The best parameter set depends on the importance assigned to each of the metrics.

Despite the caveats mentioned above, it seems reasonable to conclude that the NRSW scheme has the shortest ciphertext sizes and the slowest prover of the schemes compared. For all schemes but NRSW, by selecting different parameters (at the same security level) we can achieve various tradeoffs, and overall we find that no scheme is strictly better than

---

[4]Our implementations are available at `https://github.com/akiratk0355/verenc-mpcith`.

Table 1: Parameters and benchmarks for verifiable encryption of discrete logarithm and AES keys. Our new schemes are DKGitH (§5.1), RDKGitH (§5.1.2) and AES-VE (§5.2). CD is our implementation of the generic scheme from [CD00], followed by (estimates for) the NRSW [NRSW20] construction. Sizes are given in bytes and run times in milliseconds.

Scheme	Parameters	$|\mathsf{tr}|$	$|C|$	$|C|_{\mathrm{RS}}$	$\mathcal{P}$ (ms)	$\mathcal{V}$ (ms)	$\mathcal{C}$ (ms)
DKGitH	(64, 48, 15)	9 360	3 120	975	182.45	181.31	2.57
(N, τ, n)	(85, 20, 20)	4 276	1 300	1 300	101.36	100.93	4.76
	(16, 32, 30)	5 248	2 080	1 950	30.55	29.15	1.19
	(4, 64, 48)	8 352	4 160	3 120	15.33	12.28	0.38
RDKGitH	(132, 64, 67)	11 781	6 596	6 499	6.65	4.36	45.76
(N, t, n)	(192, 36, 145)	15 265	15 132	14 065	8.46	2.81	42.90
	(160, 80, 55)	14 337	7 760	5 335	9.03	5.34	54.99
	(256, 226, 30)	26 017	2 910	2 910	16.93	16.60	242.92
CD [CD00]	(712, 20)	52 968	1 300		42.29	37.88	
(k, u)	(250, 30)	20 524	1 950		15.22	12.61	
	(132, 64)	14 816	4 160		8.30	5.13	
NRSW [NRSW20]		1100	64		759.64†	40.28†	
AES-VE	(16, 31)	40 396	24 894		12.41	12.13	0.31
(N, τ)	(57, 22)	29 400	17 676		23.63	22.81	0.77
	(256, 16)	21 920	12 864		67.24	66.43	2.16

all others, across all metrics. The DKGitH scheme has the 2nd shortest proofs following NRSW and modest timings. The RDKGitH scheme can achieve the fastest verification of all schemes, since it scales only with the t parameter. However, this comes at the cost of larger proofs and ciphertexts, and significant cost for compression (or decryption, depending on where one does the interpolation step). Still, this tradeoff of shifting work from $\mathcal{P}$ and $\mathcal{V}$ to decryption may be appealing in scenarios where decryption is done infrequently, and higher latency is tolerable. The CD scheme provides good run times when compared to DKGitH, but with significantly larger proof sizes.

Finally, we note that our AES-VE scheme proves knowledge of an AES key with respect to a single block, and since this relation is not a binding commitment to the key, in practice we would prove knowledge of a key relating the encryption of two plaintext blocks to two ciphertext blocks. Since many parts of the proof are re-used in this larger circuit (e.g., the seed tree and witness shares of each party), the resulting proof is easily seen to be at most 2x larger/slower than the benchmarks in Table 1 and we estimate 1.5x is possible with a direct implementation. Ciphertext sizes, compression and decryption time would remain as reported.

6 Camenisch–Damgård Verifiable Encryption with Imperfect Correctness

6.1 The Camenisch–Damgård framework [CD00]

Σ-protocol A Σ-protocol for relation R is an interactive proof system consisting of three rounds. In a Σ-protocol, the prover sends a message a, the verifier replies with a random bit string e, and the prover responds with z. The verifier decides to accept or reject based on the transcript (a, e, z). A Σ-protocol can be efficiently compiled into a non-interactive zero-knowledge proof of knowledge (in the random oracle model) through the Fiat-Shamir transform [FS87]. The usual requirements for a Σ-protocol are *special soundness* and *honest verifier zero knowledge*. In particular, special soundness implies existence of an

$\mathcal{P}(w, x, \mathsf{pk})$ $\mathcal{V}(x, \mathsf{pk})$

Run Σ-protocol prover on w, x to get a.

Run Σ-protocol prover on input

challenges 0 and 1, respectively.

Obtain responses z_0, z_1.

$c_e := \mathsf{Enc}(\mathsf{pk}, z_e; r_e)$ for $e = 0, 1$. $\xrightarrow{\quad a, c_0, c_1 \quad}$

$\xleftarrow{\quad e \in \{0, 1\} \quad}$

$\xrightarrow{\quad z_e, r_e \quad}$

Run Σ-protocol verifier to check (a, e, z_e)

Check $c_e =_? \mathsf{Enc}(\mathsf{pk}, z_e; r_e)$

If check passes output $C := (c_{\bar{e}}, a, e, z_e)$

....

$\mathcal{R}(c_{\bar{e}}, a, e, z_e, \mathsf{sk})$

Get $z_{\bar{e}} := \mathsf{Dec}(\mathsf{sk}, c_{\bar{e}})$

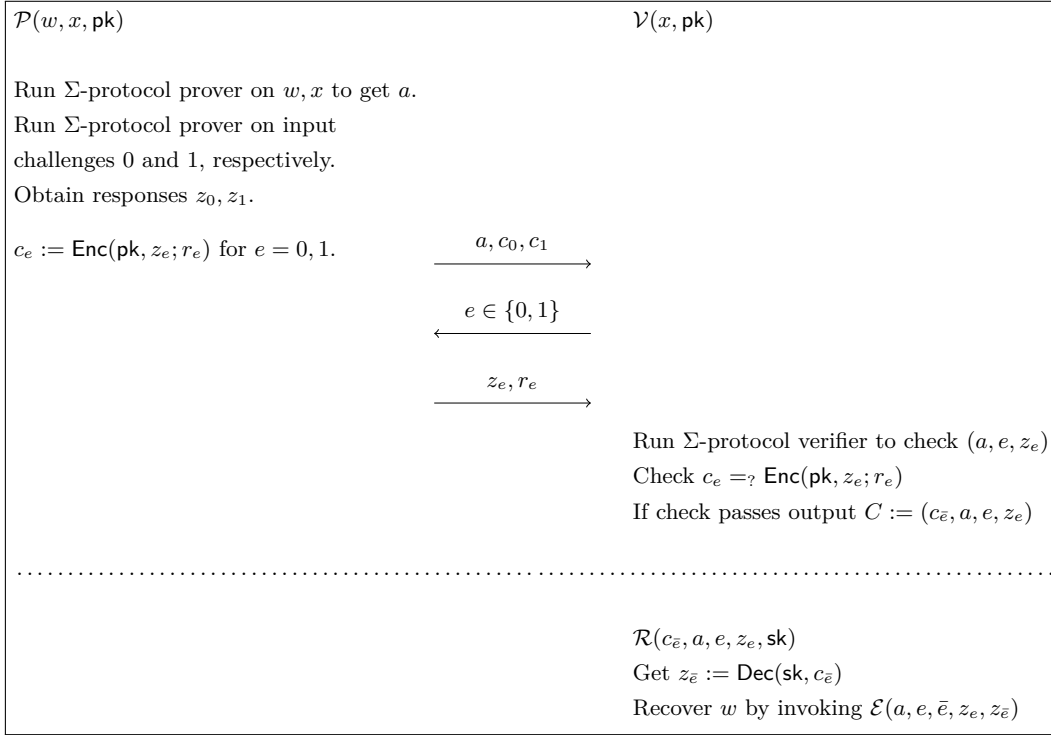Recover w by invoking $\mathcal{E}(a, e, \bar{e}, z_e, z_{\bar{e}})$

Figure 2: Camenisch–Damgård verifiable encryption.

efficient extractor $\mathcal{E}$ that outputs a valid witness, given two accepting transcripts (a, e, z) and (a, e', z') such that $e \neq e'$.

The transform See Fig. 2. We assume that there exists a Σ-protocol with one-bit challenge for the relation R. At a high-level, the receiver $\mathcal{R}$ can obtain a witness by first decrypting the unopened response and then by invoking the extractor $\mathcal{E}$ of the underlying Σ-protocol.

6.2 Undeniable Encryption is Required for [CD00]

In the analysis of the VE scheme in Fig. 2, Camenisch and Damgård assume only that Enc is semantically secure, which means that ciphertexts are indistinguishable against chosen plaintext attacks (also called IND-CPA security, or CPA security for short). We first note that CPA security does not imply that an encryption scheme is committing *if its correctness is not perfect*, so the analysis of [CD00] suggests that this property is not required for the security of their VE scheme.

We provide two counterexamples to the security analysis of [CD00]. An example with LWE-based encryption is deferred to Section 6.4. We describe encryption schemes that are semantically secure, and non-committing, in a way that allows a malicious prover, who knows the witness, to compute a proof and ciphertext guaranteed to decrypt to junk, rather than the witness, breaking the validity property of Definition 1.

Here we present the scheme that is essentially an instance of the basic construction of [CDNO97], and is much simpler than a fully deniable encryption scheme, as we only need the prover to be able to open ciphertexts by flipping bits in one direction. In our attack the unopened/junk encryptions are encryptions of the all-ones string, and when we open one to a specific value, we must flip some of the ones to zero, but we never need to change a zero to a one bit. In a more general deniable encryption scheme we would

need both one-to-zero and zero-to-one, in order to open one arbitrary message to another arbitrary message.

6.2.1   A Secure Encryption Scheme that is Not Undeniable

We describe the encryption scheme now. Let λ be a security parameter and for an RSA modulus n, let $\mathbb{QR}_n$ be the set of quadratic residues in $\mathbb{Z}_n^*$.

Key generation: Generate an RSA modulus $n = pq$, along with a generator g of a cyclic subgroup of $\mathbb{QR}_n$, of size ρ. Output the secret key $sk = (p, q)$ and public key $pk = (n, g)$

We leave the details to [Gro05], where this is called an *RSA subgroup pair*. We note only that when generating (n, g) there is flexibility for ρ, in particular we can have ρ be cryptographically large (e.g., $\rho \approx 2^{2\lambda}$), but still only be a negligible part of $\mathbb{QR}_n$.

Encryption: To encrypt the n-bit string $(m_1, \ldots, m_n)$, under the public key (n, g), encryption outputs the vector $(c_1, \ldots, c_n) \in \mathbb{QR}_n$, where

$$c_i = \begin{cases} g^{r_i} & \text{if } m_i = 1, \text{ or} \\ r_i & \text{if } m_i = 0 \end{cases}$$

and r_i is a uniformly chosen random integer in $\mathbb{QR}_n$.

Decryption: To decrypt the ciphertext $(c_1, \ldots, c_n)$ using secret key (p, q), output $(m_1, \ldots, m_n)$ where

$$m_i = \begin{cases} 1 & \text{if } c_i \in \langle g \rangle \\ 0 & \text{if } c_i \notin \langle g \rangle. \end{cases}$$

Using the secret key, we can efficiently test if $c_i \in \langle g \rangle$, by checking the order of c_i.

Fake opening: We first explain faking for a single-bit ciphertext. The scheme allows an encryption of one to be opened as if it were an encryption of zero. Let the ciphertext be $c = g^r$, the encryptor outputs $m = 0$ and $r = c$ (claiming they they generated c at random, as specified by the 0 case of encryption). Note that re-encryption with $m = 0$, $r = c$ outputs c, as required. For longer ciphertexts, the same step can be repeated for each bit, and we can open an encryption of the all-ones string to any string.

Corrrectness: When encrypting a one bit, the ciphertext is always in $\langle g \rangle$ and will be decrypted correctly. When the plaintext is a zero, decryption can fail if by chance r_i is in $\langle g \rangle$. Since key generation ensures that $\rho/|\mathbb{QR}_n|$ is negligible, this happens with negligible probability for polynomially bounded message length.

IND-CPA security Security relies on the hardness of the decisional RSA subgroup assumption, described by Groth in [Gro05], which states that distinguishing elements in $\langle g \rangle$ from elements in $\mathbb{QR}_n$ is hard. This is closely related to the *prime residuosity assumption* introduced by Benaloh and Fisher [CF85] and later used in other constructions, e.g. [BCP03, NS98]. More generally, our scheme can be constructed with any subgroup indistinguishability assumption, as defined by Brakerski and Goldwasser [BG10], provided the size of the subgroup is much smaller than the group (as required for correctness). Other options for instantiating the subgroup indistinguishability assumption are given in [BG10], in particular the instantiation based on the Damgård-Jurik [DJ01], generalization of the decisional composite residuosity assumption [Pai99] would be suitable for our construction (because it allows the subgroup to be smaller; see [BG10, Footnote 8]).

Under the decisional RSA subgroup assumption, IND-CPA security of the the single-bit case follows directly, by noting that the set of ciphertexts corresponding to an encryption of one is $\langle g \rangle$ and the set of ciphertexts corresponding to an encryption of zero is $\mathbb{QR}_n$. Distinguishing $\langle g \rangle$ from $\mathbb{QR}_n$ immediately breaks CPA security, and we can trivially construct an attacker B for the the decisional RSA subgroup problem given a CPA attacker (with the same success probability). With n-bit messages, B's advantage degrades by a factor $1/n$.

6.2.2   An Attack on [CD00] Verifiable Encryption

We now give a simple attack on the Camenisch–Damgård verifiable encryption scheme, given in Fig. 2, when the recipient's public key encryption algorithm is our semantically secure encryption scheme given above.

In the first message, the two encryptions c_e are computed as $\mathsf{Enc}(\mathsf{pk}, 1^\ell; r_e)$, i.e., the malicious prover $\mathcal{P}^*$ replaces the plaintext z_e with an encryption of the all-ones string having the same length.[5]

After seeing the challenge e, $\mathcal{P}^*$ must open the ciphertext c_e, and provide the plaintext and randomness so that $\mathcal{V}$ can check it. $\mathcal{P}^*$ uses the faking algorithm of Enc in order to claim that c_e was in fact an encryption of z_e, as required. This ensures that $\mathcal{V}$'s check $c_e = \mathsf{Enc}(\mathsf{pk}, z_e; r_e)$ succeeds. Since z_e is computed honestly (using the witness), $\mathcal{V}$'s check of the transcript (a, e, z_e) will also pass.

However, the VE ciphertext output consists of an encryption of 1^ℓ, and so the decryption will not produce a witness with probability 1, breaking the validity property, which was claimed to hold with probability $1/2$ by [CD00, Theorem 2][6].

6.3   Fixing the [CD00] Security Analysis

By additionally assuming *undeniability* of PKE (Definition 3) we can prove validity of the scheme described in Fig. 2. Recall that a cheating prover $\mathcal{P}^*$ wins the validity game if the receiver $\mathcal{R}$ failed to decrypt a witness while the verifier $\mathcal{V}$ accepts. Similar to the validity analysis of Theorem 1, we consider two cases: (1) $z_e \neq \mathsf{Dec}(\mathsf{sk}, C_e)$ while $c_e = \mathsf{Enc}(\mathsf{pk}, z_e; r_e)$ and (2) $z_e = \mathsf{Dec}(\mathsf{sk}, C_e)$ while (a, e, z_e) is an accepting transcript, where the challenge bit e is chosen uniformly. In the former case, one can break undeniability of PKE using a cheating prover $\mathcal{P}^*$. In the latter case, due to special soundness, if the extractor $\mathcal{E}$ (internally invoked by $\mathcal{R}$) fails to obtain a valid witness, it must be that the unopened response $z_{\bar{e}} = \mathsf{Dec}(\mathsf{sk}, c_{\bar{e}})$ is non-accepting w.r.t. $(a, \bar{e})$. Since z_e and $z_{\bar{e}}$ are determined *before* a cheating prover $\mathcal{P}^*$ gets to see the challenge e, the probability that $\mathcal{P}^*$ can correctly guess e is at most $1/2$. Overall, the validity error is $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{cext}} + 1/2$.

6.4   Attacking [CD00] Instantiated with LWE-based Encryption Schemes

In this section we discuss why a plain IND-CPA-secure LWE-based encryption doesn't satisfy undeniability and how it affects concrete security of [CD00].

6.4.1   LWE encryption scheme

Below we first recall a construction presented in [Lyu20, §2], a simplified version of the Regev encryption [Reg05]. The scheme is proven IND-CPA secure under the (decisional) LWE assumption, but we show that it fails to satisfy the undeniability property.

Key generation. Let q be a prime. Following [Lyu20] let us denote $[\beta] := \{-\beta, -\beta + 1, \dots, \beta\}$ with $\beta \ll q$. The key generation algorithms samples $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$, $\mathbf{s} \in [\beta]^m$, and $\mathbf{e}_1 \in [\beta]^m$ uniformly at random. It outputs public key $(\mathbf{A}, \mathbf{t})$ and secret decryption key $\mathbf{s}$, where

$$\mathbf{t} = \mathbf{As} + \mathbf{e}_1 \mod q.$$

Encryption. An encryptor is given $(\mathbf{A}, \mathbf{t})$ and message $\mu \in \{0, 1\}$ as inputs. It first samples $\mathbf{r} \in [\beta]^m$, $\mathbf{e}_2 \in [\beta]^m$ and $e_3 \in [\beta]^m$ uniformly at random and outputs ciphertext

---

[5]For simplicity we assume here that all z_e values are encoded to have the same length.
[6]Theorem 2 in the full version of [CD00] the paper, the BRICS technical report https://brics.dk/RS/98/32/BRICS-RS-98-32.pdf

$(\mathbf{u}, v)$, where

$$\mathbf{u}^T = \mathbf{r}^T \mathbf{A} + \mathbf{e}_2^T \quad \mathrm{mod}\ q$$

$$v = \mathbf{r}^T \mathbf{t} + e_3 + \frac{q}{2} \cdot \mu \quad \mathrm{mod}\ q$$

Decryption. A decryptor is given $\mathbf{s}$ and $(\mathbf{u}, v)$. It computes $v - \mathbf{u}^T \mathbf{s}$ and outputs 1 if the result is closer to $q/2$ than to 0, and outputs 0 otherwise.

Statistical correctness. The above encryption scheme is statistically correct. Note that

$$v - \mathbf{u}^T \mathbf{s} = \mathbf{r}^T (\mathbf{As} + \mathbf{e}_1) + e_3 + \frac{q}{2} \cdot \mu - \mathbf{r}^T \mathbf{As} - \mathbf{e}_2^T \mathbf{s}$$

$$= \mathbf{r}^T \mathbf{e}_1 + e_3 - \mathbf{e}_2^T \mathbf{s} + \frac{q}{2} \cdot \mu$$

For decryption to be correct the noise term $\mathbf{r}^T \mathbf{e}_1 + e_3 - \mathbf{e}_2^T \mathbf{s}$ must have a norm smaller than $q/4$. Hence, the parameters (q, m, β) should be chosen such that the following probability is negligible.

$$\delta := \Pr \left[|\mathbf{r}^T \mathbf{e}_1 + e_3 - \mathbf{e}_2^T \mathbf{s}| \geq \frac{q}{4} \ : \ \mathbf{s}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{r} \leftarrow^\$ [\beta]^m; \mathbf{e}_3 \leftarrow^\$ [\beta]; \right].$$

As $|\mathbf{r}^t \mathbf{e}_1| \leq m\beta^2$ and $|\mathbf{e}_2^t \mathbf{s}| \leq m\beta^2$, by setting $2m\beta^2 + \beta < q/4$ one can actually achieve perfect correctness. For the sake of efficiency, however, such a parameter choice is unusual. For example, if $(q, m, \beta) = (3329, 4096, 1)$ one can achieve the decryption error probability $\delta \approx 2^{-142}$. In practice, the lower bits of ciphertext are often truncated, which trades ciphertext size for decryption error, but we ignore this optimization for simplicity.

6.4.2   Attacks breaking undeniability

We present three scenarios, depending on the adversarial power.

- **Case 1. Adversary generates a key pair (i.e. breaking strong undeniability)** If the adversary has control over key generation then the attack is straightforward: by setting $\mathbf{s} = (-\beta, \ldots, -\beta)$ and $\mathbf{e}_1 = (\beta, \ldots, \beta)$, they encrypt $\mu = 1$ with randomness $\mathbf{r} = \mathbf{e}_2 = (\beta, \ldots, \beta)$ and $e_3 = \beta$. Clearly, the resulting ciphertext $(\mathbf{u}, v)$ decrypts to 0, since the noise term $\mathbf{r}^T \mathbf{e}_1 + e_3 - \mathbf{e}_2^T \mathbf{s}$ exceeds $q/4$ for the example parameter $(q, m, \beta) = (3329, 4096, 1)$.

- **Case 2. Key pair is generated honestly, but the adversary knows the decryption key (i.e. breaking undeniability as defined in Definition 3)** Even if a key pair is generated honestly, one may observe that the attack succeeds if the adversary sees the secret decryption key $(\mathbf{s}, \mathbf{e}_1)$ as in Definition 3. To maximize the norm of noise term, the adversary looks at each element of $\mathbf{e}_1$ and adaptively choose the corresponding position of $\mathbf{r}^T$. That is, $r_i = \beta$ if $\mathbf{e}_{1,i}$ is positive, and $r_i = -\beta$ otherwise. The $\mathbf{e}_2$ is chosen in the same fashion and e_3 doesn't matter as it has little impact on the resulting norm. For the example parameter $(q, m, \beta) = (3329, 4096, 1)$ this strategy succeeds with overwhelming probability (where the probability is taken over random coins used in key generation).

- **Case 3. Key pair is generated honestly, and the adversary only receives the public key (i.e. breaking a variant of undeniability weaker than Definition 3)** Even if the adversary does not get to see the decryption key, which may be the case in some practical scenarios, one can still significantly increase the chance of decryption failure. If the adversary deterministically uses the largest possible randomness $\mathbf{r}^T = \mathbf{e}_2^T = (\beta, \ldots, \beta)$ and $e_3 = \beta$, it amounts to evaluating the following probability.

$$\delta' := \Pr\left[|\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}| \geq \frac{q}{4} \ : \ \begin{array}{c} \mathbf{s}, \mathbf{e}_1 \leftarrow_\$ [\beta]^m; \\ \mathbf{r}^T = \mathbf{e}_2^T = (\beta, \ldots, \beta); e_3 = \beta \end{array}\right].$$

For the example parameter $(q, m, \beta) = (3329, 4096, 1)$, the decryption error is now $\delta' \approx 2^{-96}$, which is significantly larger than the correctness error in the honest encryption case.

6.4.3   Attacking validity of [CD00]

The VE scheme described in Fig. 2 can be instantiated with the above LWE-based encryption by having a prover encrypt the responses bit-by-bit, or with its improvements such as [GPV08, PVW08, LPR10, LPR13] allowing for packing many bits in the plaintext per one encrypting operation. A cheating prover with knowledge of a valid witness follows the protocol honestly, except that the encryption randomness is always chosen to be large as above. In this case, the verifier always gets convinced while the receiver fails to decrypt the unopened response correctly with some probability, depending on the scenarios. The original analysis of [CD00] only claims to achieve a weaker variant of validity in which a prover does not receive sk as inputs (corresponding to Case 3 above) and therefore the probability that decryption fails is still small with the above simple attack strategy. Although this scenario may not lead to a practical attack against validity of VE, the example illustrates how it fails to achieve 128-bit security in the validity game, even though the underlying encryption scheme has 128-bit security in terms of decryption correctness.

Remark 1. We remark that, unlike the counterexample presented in Section 6.2, the randomness submitted by the adversary when opening the plaintext looks somewhat suspicious: because the norm of revealed randomness is often large, the verifier may be able to detect that encryption did not sample fromthe correct (uniform) distribution over the randomness space. However, it still serves as another counterexample to the security analysis of [CD00], because it does not specify how the verifier should check randomness when the plaintext is revealed. Applying the FO transform is one way to circumvent the issue, as we observe in Appendix B. An interesting follow-up question is whether the verifier could impose the norm bound on revealed randomness to ensure that the scheme retains both correctness and binding when used as a commitment.

7   Conclusion and Future Work

As our construction gives a practical way to verifiably encrypt ECC, DSA, DH, RSA and AES keys, we have a complete and flexible solution to the verifiable backup problem for the most common key types stored in hardware and cloud services. A notable exception are keys for the HMAC algorithm. They can be handled with our transform and ZKB++ or KKW, but with larger proof sizes due to the larger circuit size of the SHA2 or SHA3 hash function. Using Limbo [DOT21] with our transform (see Section 3.3) would be the best option as Limbo can create proofs for SHA-256 that are 100-200 KB in size.

For even larger circuits, an ideal approach would be a generalization of our compiler to construct VE schemes from more general IOPs, in order to make use of proof systems where communication is sub-linear in the circuit size (such as Aurora [BCR+19]), which currently outperform MPCitH proofs for large circuits. However, a direct modification of our compiler is not obvious. With, e.g., polynomial IOPs, there are no concise secret shares of the witness to be encrypted as in MPCitH. Then the prover would likely need to perform proof of plaintext knowledge to guarantee consistency between encrypted data and the statement to be proven.   Another minor gap in our solution to the the verifiable backup problem is that strong validity (Section 3.5) is only guaranteed for perfectly correct PKE, which means we require a stronger assumption when PKE is lattice-based.

Related to encryption, can the ciphertexts produced by our construction be made CCA-secure? Currently they are if the entire proof transcript is sent to the receiver, however, once compression outputs a ciphertext, note that the ciphertext can be modified by dropping one of the individual PKE ciphertexts from one parallel repetition (even if PKE is CCA secure). Having PKE support labels (as discussed in [CS03]) might allow the set of PKE ciphertexts to be bound together. Also on the subject of CCA security, does CCA security imply undeniability?

The DKG-in-the-head design strategy proved useful here, and may be worth exploring further, since there is a large literature on distributed (or threshold) key generation upon which to draw inspiration. It is also an interesting open question whether our approach to VE leads to interesting instantiations of group and ring signatures, especially those targeting post-quantum security as was done in [BDK+22], or those based only on symmetric-key primitives such as [KKW18, DRS18].

Changelog

- **October 16th 2023** Major updates including implementation results and the analysis of non-interactive verifiable encryption schemes and RDKGitH. We also fixed the sizes of DKGitH ciphertexts in Table 1 which were not estimated correctly in the initial version.

- **April 20th 2024** Updated benchmark results for DKGiH in Table 1 and accommodated the reviewer feedback of IACR CiC. The current implementation is consistent with Protocol 4 and uses a randomized first component of Elgamal. Although the de-randomization optimization still retains a weak variant of HVZK where the statement is sampled uniformly (as in the verifiable key backup scenario), we dropped it to minimize the required assumption since the proof of security seems to require the generic group model (see Appendix D.1 for details).

Acknowledgments

This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

[ac22]   arkworks contributors. `arkworks` zksnark ecosystem, 2022. URL: https://arkworks.rs.

[AHIV17]   Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017. doi:10.1145/3133956.3134104.

[AHIV22]   Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. Cryptology ePrint Archive, Report 2022/1608, 2022. https://eprint.iacr.org/2022/1608.

[AKV22]   Microsoft Azure Key Vault Documentation: Key types, algorithms, and operations, February 2022. https://docs.microsoft.com/en-us/azure/key-vault/keys/about-keys-details.

[ASW98]   N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures (extended abstract). In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 591–606. Springer, Heidelberg, May / June 1998. doi:10.1007/BFb0054156.

[Ate99]   Giuseppe Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In Juzar Motiwalla and Gene Tsudik, editors, *ACM CCS 99*, pages 138–146. ACM Press, November 1999. doi:10.1145/319709.319728.

[AWS22a]   Amazon Web Services CloudHSM Documentation: Using the command line to manage keys, 2022. https://docs.aws.amazon.com/cloudhsm/latest/userguide/using-kmu.html.

[AWS22b]   Amazon Web Services Key Management Service Documentation: AWS KMS Keys, 2022. https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#kms_keys.

[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*, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020.

[BCP03]   Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54. Springer, Heidelberg, November / December 2003. doi:10.1007/978-3-540-40061-5_3.

[BCR+19]   Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_4.

[BCS16]   Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_2.

[BD20]   Ward Beullens and Cyprien Delpech de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 130–150. Springer, Heidelberg, 2020. `doi:10.1007/978-3-030-44223-1_8`.

[BDD22]   Carsten Baum, Bernardo David, and Rafael Dowsley. (public) verifiability for composable protocols without adaptivity or zero-knowledge. In Chunpeng Ge and Fuchun Guo, editors, *Provable and Practical Security - 16th International Conference, ProvSec 2022, Nanjing, China, November 11-12, 2022, Proceedings*, volume 13600 of *Lecture Notes in Computer Science*, pages 249–272. Springer, 2022. `doi:10.1007/978-3-031-20917-8\_17`.

[BDK+21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021. `doi:10.1007/978-3-030-75245-3_11`.

[BDK+22] Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 95–126. Springer, Heidelberg, May / June 2022. `doi:10.1007/978-3-031-07085-3_4`.

[Beu20]   Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45727-3_7`.

[BFH+20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020. `doi:10.1145/3372297.3417893`.

[BFM88]   Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. `doi:10.1145/62212.62222`.

[BG10]   Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2010. `doi:10.1007/978-3-642-14623-7_1`.

[BHH+19] Michael Backes, Lucjan Hanzlik, Amir Herzberg, Aniket Kate, and Ivan Pryvalov. Efficient non-interactive zero-knowledge proofs in cross-domains without trusted setup. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 286–313. Springer, Heidelberg, April 2019. `doi:10.1007/978-3-030-17253-4_10`.

[BKM09]   Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, January 2009. `doi:10.1007/s00145-007-9011-9`.

[BN20]   Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments
            for arithmetic circuits and their application to lattice-based cryptography.
            In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas,
            editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer,
            Heidelberg, May 2020. `doi:10.1007/978-3-030-45374-9_17`.

[BS20]   Dan Boneh and Victor Shoup. A graduate course in applied cryptography,
            2020. Available online `https://crypto.stanford.edu/~dabo/cryptobook/`.

[BSZ05]   Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures:
            The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*,
            volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.
            `doi:10.1007/978-3-540-30574-3_11`.

[CCFG16]   Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Be-
            leniosRF: A non-interactive receipt-free electronic voting scheme. In Edgar R.
            Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and
            Shai Halevi, editors, *ACM CCS 2016*, pages 1614–1625. ACM Press, October
            2016. `doi:10.1145/2976749.2978337`.

[CD00]   Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption,
            and their applications to separable group signatures and signature sharing
            schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of
            LNCS, pages 331–345. Springer, Heidelberg, December 2000. `doi:10.1007/
            3-540-44448-3_25`.

[CDG+17]   Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian
            Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-
            quantum zero-knowledge and signatures from symmetric-key primitives. In
            Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu,
            editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November
            2017. `doi:10.1145/3133956.3133997`.

[CDK+22]   Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Kon-
            ring, and Jesper Buus Nielsen. Encryption to the future - A paradigm
            for sending secret messages to future (anonymous) committees. In Shweta
            Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume
            13793 of *LNCS*, pages 151–180. Springer, Heidelberg, December 2022. `doi:
            10.1007/978-3-031-22969-5_6`.

[CDNO97]   Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable
            encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294
            of *LNCS*, pages 90–104. Springer, Heidelberg, August 1997. `doi:10.1007/
            BFb0052229`.

[CF85]   Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographi-
            cally secure election scheme (extended abstract). In *26th FOCS*, pages 372–382.
            IEEE Computer Society Press, October 1985. `doi:10.1109/SFCS.1985.2`.

[CFF+21]   Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián
            Rodríguez. Lunar: A toolbox for more efficient universal and updatable
            zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and
            Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of
            LNCS, pages 3–33. Springer, Heidelberg, December 2021. `doi:10.1007/
            978-3-030-92078-4_1`.

[CFGN96]   Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. `doi:10.1145/237814.238015`.

[CFQ19]   Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019. `doi:10.1145/3319535.3339820`.

[CHM⁺20]   Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EURO-CRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45721-1_26`.

[CL06]   Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006. `doi:10.1007/11818175_5`.

[COS20]   Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45721-1_27`.

[CS03]   Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003. `doi:10.1007/978-3-540-45146-4_8`.

[DDOS19]   Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in picnic signatures. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019. `doi:10.1007/978-3-030-38471-5_27`.

[DFMS22]   Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 677–706. Springer, Heidelberg, May / June 2022. `doi:10.1007/978-3-031-07082-2_24`.

[DGRW18]   Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96884-1_6`.

[DHMW23]   Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: Verifiable encryption to the future made practical. In Foteini Baldimtsi and Christian Cachin, editors, *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part I*, volume 13950 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2023. `doi:10.1007/978-3-031-47754-6\_15`.

[DJ01]   Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Heidelberg, February 2001. `doi:10.1007/3-540-44586-2_9`.

[DN00]   Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, Heidelberg, August 2000. `doi:10.1007/3-540-44598-6_27`.

[DOT21]   Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021. `doi:10.1145/3460120.3484595`.

[DRS18]   David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 419–440. Springer, Heidelberg, 2018. `doi:10.1007/978-3-319-79063-3_20`.

[EG21]   Electionguard specification, 2021. `https://www.electionguard.vote/`.

[Fel87]   Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987. `doi:10.1109/SFCS.1987.4`.

[Fis05]   Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005. `doi:10.1007/11535218_10`.

[FO99]   Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 53–68. Springer, Heidelberg, March 1999. `doi:10.1007/3-540-49162-7_5`.

[FO13]   Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013. `doi:10.1007/s00145-011-9114-1`.

[FR23]   Thibauld Feneuil and Matthieu Rivain. Threshold linear secret sharing to the rescue of MPC-in-the-head. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 441–473. Springer, Heidelberg, December 2023. `doi:10.1007/978-981-99-8721-4_14`.

[FS87]   Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. `doi:10.1007/3-540-47721-7_12`.

[GCZ16]   Steven Goldfeder, Melissa Chase, and Greg Zaverucha. Efficient post-quantum zero-knowledge and signatures. Cryptology ePrint Archive, Report 2016/1110, 2016. `https://eprint.iacr.org/2016/1110`.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. `doi:10.1145/2488608.2488667`.

[GH03]   Yitchak Gertner and Amir Herzberg. Committing encryption and publicly-verifiable signcryption. Cryptology ePrint Archive, Report 2003/254, 2003. `https://eprint.iacr.org/2003/254`.

[GHL+22] Tim Güneysu, Philip Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. Proof-of-possession for kem certificates using verifiable generation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1337–1351, New York, NY, USA, 2022. ACM. URL: `https://ia.cr/2022/703`, `doi:10.1145/3548606.3560560`.

[GHM+22] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter B. Rønne, and Tjerand Silde. Verifiable decryption in the head. In Khoa Nguyen, Guomin Yang, Fuchun Guo, and Willy Susilo, editors, *ACISP 22*, volume 13494 of *LNCS*, pages 355–374. Springer, Heidelberg, November 2022. `doi:10.1007/978-3-031-22301-3_18`.

[GK22]   Google Cloud Key Management Service Documentation: Key purposes and algorithms, February 2022. `https://cloud.google.com/kms/docs/algorithms`.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. `doi:10.1109/FOCS.2012.47`.

[GLR17]   Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, August 2017. `doi:10.1007/978-3-319-63697-9_3`.

[GMO16]   Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

[GMP21]   Paul Grubbs, Varun Maram, and Kenneth G. Patterson. Anonymous, robust post-quantum public key encryption. NIST Third PQC Standardization Conference, 2021.

[GOS06]   Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006. `doi:10.1007/11761679_21`.

[GPV08]   Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. `doi:10.1145/1374376.1374407`.

[Gro05]   Jens Groth. Cryptography in subgroups of zn. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 50–65. Springer, Heidelberg, February 2005. `doi:10.1007/978-3-540-30576-7_4`.

[Gro16]   Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. `doi:10.1007/978-3-662-49896-5_11`.

[GT21]   Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg. `doi:10.1007/978-3-030-84252-9_3`.

[HHK17]   Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. `doi:10.1007/978-3-319-70500-2_12`.

[HLR21]   Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 750–760. ACM, 2021. `doi:10.1145/3406325.3451116`.

[IKOS07]   Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007. `doi:10.1145/1250790.1250794`.

[Kal22]   Daniel Kales. Implementation of bn++ and helium signatures, 2022. `https://github.com/IAIK/bnpp_helium_signatures`.

[Kat21]   Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 580–610, Virtual Event, August 2021. Springer, Heidelberg. `doi:10.1007/978-3-030-84245-1_20`.

[Kiy20]   Susumu Kiyoshima. Round-optimal black-box commit-and-prove with succinct communication. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 533–561. Springer, Heidelberg, August 2020. `doi:10.1007/978-3-030-56880-1_19`.

[KKW18]   Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018. `doi:10.1145/3243734.3243805`.

[KMP16]   Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016. `doi:10.1007/978-3-662-53008-5_2`.

[KOS18]   Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 286–313. Springer, Heidelberg, November 2018. `doi:10.1007/978-3-030-03807-6_11`.

[KSSW22]   Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom
            Wiggers. Improving software quality in cryptography standardization
            projects. In *IEEE European Symposium on Security and Privacy, Eu-
            roS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*, pages 19–30.
            IEEE, 2022. URL: https://doi.org/10.1109/EuroSPW55150.2022.00010,
            doi:10.1109/EUROSPW55150.2022.00010.

[KZ22]   Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge
            proofs and post-quantum signatures. Cryptology ePrint Archive, Report
            2022/588, 2022. https://eprint.iacr.org/2022/588. URL: https://
            eprint.iacr.org/2022/588.

[LCKO19]   Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. SAVER: Snark-
            friendly, additively-homomorphic, and verifiable encryption and decryption
            with rerandomization. Cryptology ePrint Archive, Report 2019/1270, 2019.
            https://eprint.iacr.org/2019/1270.

[LN17]   Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption
            from lattices. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors,
            EUROCRYPT 2017, Part I, volume 10210 of *LNCS*, pages 293–323. Springer,
            Heidelberg, April / May 2017. doi:10.1007/978-3-319-56620-7_11.

[LPR10]   Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and
            learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*,
            volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.
            doi:10.1007/978-3-642-13190-5_1.

[LPR13]   Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-
            LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors,
            EUROCRYPT 2013, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg,
            May 2013. doi:10.1007/978-3-642-38348-9_3.

[Lyu20]   Vadim Lyubashevsky. Basic lattice cryptography: Encryption and Fiat-
            Shamir signatures. Manuscript, 2020. https://drive.google.com/file/d/
            1JTdW5ryznp-dUBBjN12QbvWz9R41NDGU/view?usp=sharing.

[MBKM19]   Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic:
            Zero-knowledge SNARKs from linear-size universal and updatable structured
            reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang,
            and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press,
            November 2019. doi:10.1145/3319535.3339817.

[Mic00]   Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*,
            30(4):1253–1298, 2000.

[NAB+19]   Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook,
            Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christo-
            pher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Tech-
            nical report, National Institute of Standards and Technology, 2019. avail-
            able at https://csrc.nist.gov/projects/post-quantum-cryptography/
            post-quantum-cryptography-standardization/round-2-submissions.

[NRSW20]   Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN:
            Schnorr multi-signatures with verifiably deterministic nonces. In Jay Lig-
            atti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS
            2020*, pages 1717–1731. ACM Press, November 2020. doi:10.1145/3372297.
            3417236.

[NS98]   David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Li Gong and Michael K. Reiter, editors, *ACM CCS 98*, pages 59–66. ACM Press, November 1998. `doi:10.1145/288090.288106`.

[Pai99]   Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999. `doi:10.1007/3-540-48910-X_16`.

[Pas03]   Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003. `doi:10.1007/978-3-540-45146-4_19`.

[Ped92]   Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992. `doi:10.1007/3-540-46766-1_9`.

[PK15]   OASIS Standard: PKCS #11 Cryptographic Token Interface Base Specification Version 2.40, April 2015. `http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf`.

[PK22]   OASIS Standard: PKCS #11 Cryptographic Token Interface Current Mechanisms Specification Version 3.0, 2022. `https://docs.oasis-open.org/pkcs11/pkcs11-curr/v3.0/csprd01/pkcs11-curr-v3.0-csprd01.pdf`.

[Poi00]   David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 129–146. Springer, Heidelberg, January 2000. `doi:10.1007/978-3-540-46588-1_10`.

[PS00]   Guillaume Poupard and Jacques Stern. Fair encryption of RSA keys. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 172–189. Springer, Heidelberg, May 2000. `doi:10.1007/3-540-45539-6_13`.

[PVW08]   Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. `doi:10.1007/978-3-540-85174-5_31`.

[Reg05]   Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. `doi:10.1145/1060590.1060603`.

[SAB⁺20]   Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

[Sch91]   Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. `doi:10.1007/BF00196725`.

[Set20]   Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Heidelberg, August 2020. `doi:10.1007/978-3-030-56877-1_25`.

[Sta96]   Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer, Heidelberg, May 1996. `doi:10.1007/3-540-68339-9_17`.

[Unr15]   Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015. `doi:10.1007/978-3-662-46803-6_25`.

[YC22]   Yubico YubiHSM2 Guide: Backing Up Key Material, 2022. `https://developers.yubico.com/YubiHSM2/Usage_Guides/YubiHSM2_for_ADCS_Guide/Backing_Up_Key_Material.html`.

[YY98]   Adam Young and Moti Yung. Auto-recoverable auto-certifiable cryptosystems. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 17–31. Springer, Heidelberg, May / June 1998. `doi:10.1007/BFb0054114`.

[ZCD⁺20]   Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`.

A   Additional Preliminaries

In this section, we provide additional preliminaries on public-key encryption, extractable commitment schemes and interactive oracle proofs.

A.1   Public Key Encryption

A public key encryption scheme PKE is a tuple of three algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Let S_m be a message space and S_r be a set from which randomness is sampled.

- $\mathsf{Gen}(1^\kappa)$ outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Enc}(\mathsf{pk}, m; r)$ outputs a ciphertext c on public key pk, message $m \in S_m$ and randomness $r \in S_r$ as inputs.
- $\mathsf{Dec}(\mathsf{sk}, c)$ outputs a plaintext m or $\perp$ on decryption key sk and ciphertext c as inputs.

Definition 4. PKE is ϵ_{cpa}-IND-CPA secure if for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$

$$\left| \Pr \left[b = b' : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa); \\ (m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{pk}); \\ b \leftarrow_\$ \{0,1\}; r \leftarrow_\$ S_r; \\ c := \mathsf{Enc}(\mathsf{pk}, m_b; r); b' \leftarrow \mathcal{A}_2(c); \end{array} \right] - \frac{1}{2} \right| \le \epsilon_{\mathsf{cpa}}(\kappa)$$

Throughout we assume PKE satisfies IND-CPA security.

Following [HHK17], we define statistical correctness relative to a random oracle $\mathsf{G} : \{0,1\}^* \to S_r$.

Definition 5. [HHK17] Let q_{G} be the number of queries to G made by an adversary. We say that PKE is $\delta(q_{\mathsf{G}})$-correct if for all (possibly unbounded) adversaries $\mathcal{A}$

$$\Pr\left[m \neq m' : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\kappa}); m \leftarrow \mathcal{A}^{\mathsf{G}(\cdot)}(\mathsf{pk}, \mathsf{sk}); \\ r \leftarrow_{\$} S_r; c := \mathsf{Enc}(\mathsf{pk}, m; r); m' := \mathsf{Dec}(\mathsf{sk}, c); \end{array}\right] \leq \delta(q_{\mathsf{G}})$$

Note that statistical correctness in the standard model can be defined as a special case of the above definition, where $q_{\mathsf{G}} = 0$ and therefore δ doesn't rely on q_{G}.

A.2 Extractable Commitment Schemes

An extractable commitment scheme ECOM is a tuple of algorithms $(\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$.

- $\mathsf{CGen}(1^{\kappa})$ outputs a commitment key pk and an extraction key sk.
- $\mathsf{Commit}(\mathsf{pk}, m; r)$ outputs a commitment c on commitment key pk, message $m \in S_m$ and randomness $r \in S_r$ as inputs.
- $\mathsf{CExt}(\mathsf{sk}, c)$ outputs a message m on an extraction key sk and a commitment c as inputs.

We require ECOM to satisfy hiding, binding and extractability.

Hiding ECOM is statistically (resp. computationally) ϵ_{hide}-hiding, if for any adversary (resp. any PPT adversary) $(\mathcal{A}_1, \mathcal{A}_2)$

$$\left|\Pr\left[b = b' : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^{\kappa}); \\ (m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{pk}); \\ b \leftarrow_{\$} \{0, 1\}; r \leftarrow_{\$} S_r; \\ c := \mathsf{Commit}(\mathsf{pk}, m_b; r); b' \leftarrow \mathcal{A}_2(c); \end{array}\right] - \frac{1}{2}\right| \leq \epsilon_{\mathsf{hide}}(\kappa)$$

Binding ECOM is statistically (resp. computationally) ϵ_{bind}-binding if for any adversary (resp. any PPT adversary) $\mathcal{A}$

$$\Pr\left[\begin{array}{l} m \neq m' \\ \wedge\ c = \mathsf{Commit}(\mathsf{pk}, m; r) : \\ \wedge\ c = \mathsf{Commit}(\mathsf{pk}, m'; r') \end{array} \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^{\kappa}) \\ (c, m, r, m', r') \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}) \end{array}\right] \leq \epsilon_{\mathsf{bind}}(\kappa)$$

In particular, statistically binding implies that the following probability is also negligible in κ, since otherwise a computationally unbounded adversary could simply check all possible values of (c, m, r, m', r') to find a tuple that breaks binding.

$$\Pr\left[\begin{array}{l} \exists (c, m, r, m', r') : m \neq m' \\ \wedge\ c = \mathsf{Commit}(\mathsf{pk}, m; r) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^{\kappa}) \\ \wedge\ c = \mathsf{Commit}(\mathsf{pk}, m'; r') \end{array}\right]$$

Extractability ECOM is statistically (resp. computationally) ϵ_{cext}-extractable if for any adversary (resp. any PPT adversary) $\mathcal{A}$

$$\Pr\left[\begin{array}{l} m \neq m' \\ \wedge\ c = \mathsf{Commit}(\mathsf{pk}, m; r) \end{array} : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^{\kappa}) \\ (c, m, r) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}) \\ m' := \mathsf{CExt}(\mathsf{sk}, c) \end{array}\right] \leq \epsilon_{\mathsf{cext}}(\kappa)$$

Note that without CExt and the extractable property, it is a usual commitment scheme COM.

A.3   Interactive Oracle Proofs

We recall interactive oracle proofs (IOP) originally introduced by [BCS16]. As the MPC-in-the-head proofs relevant to our work have public-coin verifiers that make non-adaptive queries (i.e., queries made by the verifier are solely determined by the verifier's randomness and inputs), we consider a slightly restricted class of IOPs satisfying those properties. This allows us to divide the protocol into three phases similar to those of the AHP framework [CHM$^+$20] (although we do not require a preprocessing phase).

Definition 6 (IOP). Let R be a relation and $L_R := \{x : \exists w : (x, w) \in R\}$. A (public-coin) r-round interactive oracle proof for a relation R consists of a tuple $(\mathbf{P}, \mathbf{V})$. The protocol proceeds as follows.

- **Committing phase** For $i \in [1, r]$, the verifier $\mathbf{V}$ sends a random message ρ_i and the prover $\mathbf{P}$ outputs a proof string π_i, to which the verifier has oracle access.
- **Query phase** For $i \in [1, r]$, $\mathbf{V}$ can query oracle i to access π_i with a query string q_i. The oracle returns the corresponding response string s_i.
- **Decision phase** Based on the responses from oracles, $\mathbf{V}$ accepts or rejects.

While not present in the general definition, in order to ensure that an IOP is ZK, concrete protocols define limits on the queries the verifier can make, to ensure that information about w is not leaked. For many IOPs (as suggested by the next definition), given all proof strings it becomes possible to recover w.

Definition 7 (Straight-line extractability (SLE)). An IOP $(\mathbf{P}, \mathbf{V})$ is *straight-line extractable* with knowledge error $\epsilon_{\mathsf{sle\text{-}iop}}$ if there exists an efficient extractor $\mathbf{E}$ such that for all pairs of unbounded adversaries $(\mathbf{A}^*, \mathbf{P}^*)$

$$
\Pr\left[b = 1 \wedge (x, w') \notin R : \begin{array}{l} x \leftarrow \mathbf{A}^*(1^\kappa); \\ b \leftarrow \langle \mathbf{P}^*, \mathbf{V}\rangle(x); \\ w' \leftarrow \mathbf{E}(x, \pi_1, \ldots, \pi_r); \end{array}\right] \leq \epsilon_{\mathsf{sle\text{-}iop}}(\kappa)
$$

Definition 8 (Honest-verifier zero knowledge (HVZK)). An IOP $(\mathbf{P}, \mathbf{V})$ is $\epsilon_{\mathsf{zk\text{-}iop}}$-*statistical honest-verifier zero knowledge* if there exists a PPT simulator $\mathbf{S}$ such that for every $(x, w) \in R$, the statistical distance between $\mathbf{S}(x)$ and $\mathbf{V}$'s view of the honest interaction with $\mathbf{P}$ on input x and w is at most $\epsilon_{\mathsf{zk\text{-}iop}}$.

Remark 2 (Security proofs for IOPs). In our security proofs, we assume that an IOP prover $\mathbf{P}$ gets to see the query strings q_i. This naturally models all concrete protocols we consider where the verifier queries are fixed or sent to the prover, and does not affect security because the query phase happens after the prover has sent all proof strings.

Also, since most protocols realize the oracles by committing to the proofs strings π_i, all π_i are available to the extractor by using extractable commitments (Appendix A.2) or reading the query history in the ROM. For examples of IOPs that follow this paradigm see Marlin [CHM$^+$20] and Lunar [CFF$^+$21].

Remark 3 (Uniqueness of extracted witness). While we are guaranteed that the witness w' output by extractor algorithm $\mathbf{E}$ of Definition 7 satisfies $(x, w') \in R$, w' might not be the same witness used by $\mathbf{P}$ when creating the proof, if there are multiple valid witness per statement. For example, when proving knowledge of a symmetric key that relates a given plaintext-ciphertext pair (as Banquet does for AES) it may be easy to find keys k_1, k_2 such that $E_{k_1}(p) = E_{k_2}(p)$ where p is a fixed plaintext block. In the context of our verifiable encryption construction, where decryption invokes the IOP extractor, it will be important that R is such that x is a binding commitment to w.

A.4   Bitwise Commitment and Encryption

Our equality proof method (Appendix G) to compress ciphertexts requires an extractable commitment scheme where the messages are bitstrings, and the commitments must be (somewhat) homomorphic with respect to the XOR operation. For a committed value $c = \mathsf{Commit}(a)$ and a public value b, where a and b are both bitstrings, we must be able to compute $c' = \mathsf{Commit}(a \oplus b)$. In Appendix G our construction uses Paillier encryption, but what we describe here also applies to Elgamal encryption and can be adapted to Pedersen commitments. The setup is a group $\mathbb{G}$ with generator g and public key h. Encryption is defined as follows.

$\mathsf{Enc}_h(a_1, \ldots, a_\lambda; r_1, \ldots r_\lambda)$ The inputs $a_1, \ldots, a_\lambda$ are bits, and the inputs $r_1, \ldots, r_n$ are random values modulo the order of $\mathbb{G}$. The output is $c = ((e_1, d_1), \ldots, (e_\lambda, d_\lambda)) \in \mathbb{G}^{2n}$ where $(e_i, d_i) = (g^{r_i}, g^{a_i} h^{r_i})$.

XOR of committed and public values Given $c = \mathsf{Enc}_h(a_1, \ldots, a_\lambda; r_1, \ldots, r_\lambda)$, and $b = (b_1, \ldots, b_\lambda)$, to compute $c' = \mathsf{Enc}_h(a \oplus b; r'_1, \ldots, r'_\lambda)$, first note that, when a_i and b_i are bits, $a_i \oplus b_i = a_i + b_i - 2a_i b_i$, and the latter computation can be done correctly modulo the group order of $\mathbb{G}$ since a_i and b_i are bits. Therefore, we compute (e'_i, d'_i) as $((e_i \cdot g^{b_i})/(e_i^{2b_i}), (d_i \cdot g^{b_i})/d_i^{2b_i})$, and the prover can compute $r'_i = r_i + b_i - 2r_i b_i$ (over $\mathbb{Z}$ when the group is of unknown order).

Converting from single-bit to bitstring encryptions Once all homomorphic operations have been performed, we can convert a bitwise encryption to $a = (a_1, \ldots, a_\lambda)_2$ whenever $\lambda < \lfloor \log_2(|\mathbb{G}|) \rfloor$, by simply computing an encryption of the integer $a = \sum_{i=1}^{\lambda} a_i 2^i$, as $(\prod e_i^{2^i}, \prod d_i^{2^i})$, with opening $r' = \sum_{i=1}^{\lambda} r_i 2^i$.

Proving equality of committed values It is straightforward to prove two bitwise encrypted values are the same by proving the individual bits are the same. Once the ciphertext has been converted an integer, proving equality is standard in the literature. Using a generalization of Schnorr's proof (called the "general linear protocol" in [BS20, §19.5.3]), we can prove knowledge of (a, r_1, r_2) such that $e_1 = g^a h^{r_1} \wedge e_2 = g^a h^{r_2}$ (and this generalizes to multiple commitments in a straightforward way).

B   Undeniability and Binding of the Fujisaki–Okamoto Transform

As we observed in the previous section it turns out that an IND-CPA-secure PKE does not necessarily satisfy the undeniability property. The issue is especially critical in the post-quantum scenario, because typical lattice-based public key encryption schemes allow a small probability of decryption failure, which can be exploited by a malicious adversary to break undeniability.

Motivated by this we consider simple generic constructions of undeniable encryption from any CPA-secure scheme with statistical correctness. We analyze two variants of the Fujisaki–Okamoto transform [FO99, FO13, HHK17] and prove that both provide computational undeniability in the random oracle model. Note that from Lemma 1, binding of PKE when used as a commitment scheme is implied by undeniability. Hence, the result in this section also implies these variants of the FO transform can be used to construct a secure commitment scheme, which might be of independent interest. Throughout this section, we denote the number of queries to a random oracle G by q_{G}.

B.1   PKE$_1$ [HHK17]

We first present a simple transform that forms the basis of both conversions. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure scheme with statistical correctness, message space

$S_m = \{0,1\}^k$, and randomness space $S_r = \{0,1\}^l$. Let $\mathsf{G} : \{0,1\}^* \to S_r$ be a random oracle. Then we define a deterministic encryption scheme $\mathsf{PKE_1} = (\mathsf{Gen_1}, \mathsf{Enc_1}, \mathsf{Dec_1})$ as follows.

- $\mathsf{Gen_1}$ is identical to Gen
- $\mathsf{Enc_1}(\mathsf{pk}, M)$ takes $M \in S_m$ and outputs $c := \mathsf{Enc}(\mathsf{pk}, M, \mathsf{G}(M))$.
- $\mathsf{Dec_1}(\mathsf{sk}, c)$ first obtains $M := \mathsf{Dec}(\mathsf{sk}, c)$, and
    - if $M' = \bot$ or $c \neq \mathsf{Enc}(\mathsf{pk}, M'; \mathsf{G}(M'))$, outputs $\bot$;
    - otherwise outputs M'.

From [HHK17, Theorem 3.1], if PKE is δ-correct, then $\mathsf{PKE_1}$ is δ_1-correct in the random oracle model with $\delta_1(q_\mathsf{G}) = q_\mathsf{G} \cdot \delta$. Note that this also implies that $\mathsf{PKE_1}$ is undeniable as well albeit not CPA secure, since it is a deterministic encryption scheme.

B.2   PKE₂ [FO99]

As a randomized variant of the previous conversion, we define $\mathsf{PKE_2} = (\mathsf{Gen_2}, \mathsf{Enc_2}, \mathsf{Dec_2})$ parameterized by bitlength k_0 of randomness as follows.

- $\mathsf{Gen_2}$ is identical to Gen
- $\mathsf{Enc_2}(\mathsf{pk}, m; \rho)$ takes $m \in \{0,1\}^{k-k_0}$ and $\rho \in \{0,1\}^{k_0}$ such that $m || \rho \in S_m$ and outputs $c := \mathsf{Enc_1}(\mathsf{pk}, m || \rho) = \mathsf{Enc}(\mathsf{pk}, m || \rho, \mathsf{G}(m || \rho))$.
- $\mathsf{Dec_2}(\mathsf{sk}, c)$ first obtains $M' := \mathsf{Dec_1}(\mathsf{sk}, c)$, and
    - if $M' = \bot$, outputs $\bot$;
    - otherwise parses M' as $m' || \rho'$ and outputs m'.

In [FO99], Fujisaki and Okamoto proved that the above conversion preserves IND-CPA security. Hence, a commitment scheme constructed from $\mathsf{PKE_2}$ is computationally hiding. Moreover, by additionally assuming γ-uniformity of PKE, they showed that $\mathsf{PKE_2}$ is IND-CCA secure, but we omit details in this paper as we only require IND-CPA security.

Below we prove our new result about $\mathsf{PKE_2}$.

Lemma 2. *If PKE is δ-correct, then in the random oracle model, $\mathsf{PKE_2}$ is ϵ_und-undeniable with $\epsilon_\mathsf{und}(q_\mathsf{G}) = q_\mathsf{G} \cdot \delta$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks undeniability. We construct a reduction $\mathcal{B}$ that breaks δ_1-correctness of $\mathsf{PKE_1}$ and thus δ-correctness of PKE.

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ in the correctness game, $\mathcal{B}$ forwards $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}$.
2. Whenever $\mathcal{A}$ makes a query to G, the $\mathcal{B}$ forwards the same query to G in the correctness game.
3. When $\mathcal{A}$ outputs (c, m, ρ) such that $c = \mathsf{Enc_2}(\mathsf{pk}, m; \rho)$ and $m \neq \mathsf{Dec_2}(\mathsf{sk}, c)$, the $\mathcal{B}$ outputs $M := m || \rho$ in the correctness game w.r.t. $\mathsf{PKE_1}$.

Note that the output M of $\mathcal{B}$ satisfies $c = \mathsf{Enc_1}(\mathsf{pk}, M)$, because of how $\mathsf{Enc_2}$ works internally.

We argue that, as long as $m \neq \mathsf{Dec_2}(\mathsf{sk}, c)$, we have $M \neq \mathsf{Dec_1}(\mathsf{sk}, c)$ in the correctness game w.r.t. $\mathsf{PKE_1}$. There are two cases where $\mathcal{A}$ wins the undeniability game w.r.t. $\mathsf{PKE_2}$: (1) $\bot = \mathsf{Dec_1}(\mathsf{sk}, c)$, and (2) $m' \neq m$, where $m' || \rho' = \mathsf{Dec_1}(\mathsf{sk}, c)$. In case (1), $\mathcal{B}$ clearly breaks correctness of $\mathsf{PKE_1}$; in case (2), $\mathsf{Dec_1}(\mathsf{sk}, c)$ outputs $M' := m' || \rho' \neq M$. Hence, whenever $\mathcal{A}$ successfully breaks undeniability, $\mathcal{B}$ successfully breaks correctness of $\mathsf{PKE_1}$. Overall, we have $\epsilon_\mathsf{und} = \delta_1 = q_\mathsf{G} \cdot \delta$.

$\square$

B.3 PKE$_3$ [HHK17]

We now define $\mathsf{PKE}_3 = (\mathsf{Gen}_3, \mathsf{Enc}_3, \mathsf{Dec}_3)$, a hybrid encryption scheme obtained by applying a variant of the FO transform to a deterministic public-key encryption scheme $\mathsf{PKE}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ as defined in Appendix B.1 and a symmetric-key encryption scheme $\mathsf{SKE} = (\mathsf{SEnc}, \mathsf{SDec})$ with message space $\mathcal{M}_{\mathrm{sym}}$ and key space $\mathcal{K}$, respectively.

The transform we analyze here corresponds to $\mathsf{FO}^{\not\perp}$ ("FO with implicit rejection") of [HHK17] and it is used in the Kyber [SAB$^+$20][7] and FrodoKEM [NAB$^+$19] lattice-based encryption schemes. With slight modification our undeniability analysis below also applies to other similar variants such as $\mathsf{FO}^{\not\perp}_m$ of [HHK17] or the original FO transform proposed in [FO13]. We let S_m be the message space of PKE_1. The scheme below relies on two random oracles $\mathsf{G} : \{0,1\}^* \to S_r$ and $\mathsf{H} : \{0,1\}^* \to \mathcal{K}$.

- Gen_3 first obtains $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_1(1^\kappa)$ and then samples a secret random seed $s \leftarrow\!\!\$\ S_m$. It outputs $(\mathsf{pk}', \mathsf{sk}') := (\mathsf{pk}, (\mathsf{sk}, s))$.

- $\mathsf{Enc}_3(\mathsf{pk}, m; \rho)$ takes $m \in \mathcal{M}_{\mathrm{sym}}$ and $\rho \in S_m$. It computes $c_1 := \mathsf{Enc}_1(\mathsf{pk}, \rho)$, $K := \mathsf{H}(\rho, c_1)$, and $c_2 := \mathsf{SEnc}(K, m)$. It outputs (c_1, c_2).

- $\mathsf{Dec}_3((\mathsf{sk}, s), (c_1, c_2))$ first obtains $\rho' := \mathsf{Dec}_1(\mathsf{sk}, c_1)$, and
  - if $\rho' = \bot$, let $K' := \mathsf{H}(s, c_1)$;
  - otherwise let $K' := \mathsf{H}(\rho', c_1)$.

  It finally outputs $m' := \mathsf{SDec}(K', c_2)$.

Hofheinz, Hövelmanns, and Kiltz proved IND-CCA security of the underlying KEM implicit in the above construction. From [HHK17, Theorem 3.4], if PKE_1 is δ_1-correct and SKE is perfectly correct, then PKE_3 is δ_1-correct in the random oracle model.

Below we prove our new result about PKE_3.

Lemma 3. *If PKE_1 is δ_1-correct and SKE is perfectly correct, then in the random oracle model, PKE_3 is ϵ_{und}-undeniable with $\epsilon_{\mathsf{und}} = \delta_1$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks undeniability. We construct a reduction $\mathcal{B}$ that breaks δ_1-correctness of PKE_1.

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ in the correctness game, $\mathcal{B}$ samples a random seed $s \leftarrow\!\!\$\ S_m$ and forwards $(\mathsf{pk}, (\mathsf{sk}, s))$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs $((c_1, c_2), m, \rho)$ such that $(c_1, c_2) = \mathsf{Enc}_3(\mathsf{pk}, m; \rho)$ and $m \neq \mathsf{Dec}_3((\mathsf{sk}, s), (c_1, c_2))$, the $\mathcal{B}$ outputs ρ in the correctness game w.r.t. PKE_1.

Note that if $\mathcal{B}$ outputs ρ, we have that $c_1 = \mathsf{Enc}_1(\mathsf{pk}, \rho)$, $c_2 = \mathsf{SEnc}(K, m)$, and $m \neq \mathsf{SDec}(K', c_2)$, where $K = \mathsf{H}(\rho, c_1)$ and K' is as defined in Dec_3. Hence it must be that $K' \neq K$, since SKE is perfectly correct. We consider two cases depending on how K' is derived in Dec_3: (1) $\bot = \mathsf{Dec}_1(\mathsf{sk}, c_1)$, leading to $K' = \mathsf{H}(s, c_1)$, (2) $\rho' = \mathsf{Dec}_1(\mathsf{sk}, c_1)$ and $\rho' \neq \bot$, leading to $K' = \mathsf{H}(\rho', c_1)$. In case (1), $\mathcal{B}$ clearly breaks correctness of PKE_1; in case (2), it must be that $\rho' \neq \rho$ for $K' \neq K$ to happen, and thus correctness of PKE_1 is broken. Hence, whenever $\mathcal{A}$ successfully breaks undeniability, $\mathcal{B}$ successfully breaks δ_1-correctness of PKE_1.

$\square$

---

[7]We remark that the specification of Kyber slightly deviates from this transformation. Although Grubbs, Maram and Patterson recently pointed out that this subtle difference makes CCA proof in the QROM challenging [GMP21], it is not an issue in our setting because we only need CPA security.

C   Constructing Non-interactive Verifiable Encryption in the Random Oracle Model

C.1   Non-interactive verifiable encryption

Below we define a variant of Definition 1 tailored to the non-interactive setting. We assume that if the scheme is instantiated in the (programmable) random oracle model, all parties have access to the RO and a ZK simulator has ability to program the RO.

Definition 9 (Non-interactive Verifiable Encryption Scheme)**.** Let R be a relation and $L_R := \left\{ x \, : \, \exists w : (x, w) \in R \right\}$. A secure non-interactive verifiable encryption scheme NIVE_R for a relation R consists of a tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{R})$:

- $\mathcal{G}(1^\kappa)$: A key generation algorithm that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathcal{P}(\mathsf{pk}, x, w)$: A prover algorithm that outputs a transcript tr.
- $\mathcal{V}(\mathsf{pk}, x, \mathsf{tr})$: A verifier algorithm that that outputs a bit $b \in \{0, 1\}$ indicating whether $\mathcal{V}$ accepts or rejects.
- $\mathcal{C}(x, \mathsf{tr})$: A compression algorithm that outputs a compressed ciphertext C.
- $\mathcal{R}(\mathsf{sk}, C)$: A receiver (or recovery) algorithm that outputs a plaintext w.

We require NIVE to satisfy completeness, validity and zero knowledge.

Completeness NIVE_R is ϵ_{comp}-complete if for all $(x, w) \in R$.

$$\Pr \left[b \neq 1 \vee (x, w') \notin R \, : \, \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ \mathsf{tr} \leftarrow \mathcal{P}(\mathsf{pk}, x, w); \\ b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr}); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{comp}}(\kappa)$$

Validity NIVE_R is ϵ_{val}-valid if for all pairs of PPT adversary $(\mathcal{A}^*, \mathcal{P}^*)$,

$$\Pr \left[b = 1 \wedge (x, w') \notin R \, : \, \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}); \\ \mathsf{tr} \leftarrow \mathcal{P}^*(\mathsf{pk}, \mathsf{sk}, x); \\ b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr}); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right] \leq \epsilon_{\mathsf{val}}(\kappa)$$

Computational zero-knowledge NIVE_R is ϵ_{zk}-ZK if there exists a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$, all $(x, w) \in R$,

$$\left| \Pr \left[i = i' \, : \, \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); \\ \mathsf{tr}_0 \leftarrow \mathcal{P}(\mathsf{pk}, x, w); \\ \mathsf{tr}_1 \leftarrow \mathcal{S}(\mathsf{pk}, x); \\ i \leftarrow_\$ \{0, 1\}; i' \leftarrow \mathcal{D}(\mathsf{pk}, x, \mathsf{tr}_i); \end{array} \right] - \frac{1}{2} \right| \leq \epsilon_{\mathsf{zk}}(\kappa)$$

C.2   Applying Fiat–Shamir to MPCitH-VE

See Protocol 3 for a non-interactive VE scheme constructed from $\mathsf{MPCitH\text{-}VE}_R$.

Protocol 3: MPCitH-NIVE$_R$

Converts the MPCitH-VE prover $\mathcal{P}$ and verifier $\mathcal{V}$ to an MPCitH-NIVE prover $\mathcal{P}'$ and verifier $\mathcal{V}'$ using the random oracle $\mathsf{H} : \{0,1\}^* \to \mathsf{Ch}^\tau$. $\mathcal{G}$, $\mathcal{C}$, and $\mathcal{R}$ are identical to those of MPCitH-VE.[a]

Prover $\mathcal{P}'(\mathsf{pk}, x, w)$**:**

1. $\mathcal{P}'$ runs the $\mathcal{P}$ on input (pk, x, w) to obtain the first-round message $\sigma := (\mathsf{C}_i^{(j)})_{i \in [N], j \in [\tau]}$ as well as $\mathcal{P}$'s internal state.

2. $\mathcal{P}'$ derives challenge $h := (\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\tau)}) = \mathsf{H}(\sigma, x, \mathsf{pk})$

3. $\mathcal{P}'$ invokes $\mathcal{P}$ on challenge $(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\tau)})$ as well as its internal state to obtain $((V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$.

4. $\mathcal{P}'$ outputs $\mathsf{tr}' = ((\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$

Verifier $\mathcal{V}'(\mathsf{pk}, x, \mathsf{tr}')$**:**

1. $\mathcal{V}'$ parses tr' as $((\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$ and let $h := (\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\tau)})$

2. $\mathcal{V}'$ computes $\mathsf{C}_i^{(j)} := \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)})$ for $i \in \mathbf{e}^{(j)}$ and for $j \in [\tau]$. Let $\mathsf{tr} = ((\mathsf{C}_i^{(j)})_{i \in [N]}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$ and $\sigma := (\mathsf{C}_i^{(j)})_{i \in [N], j \in [\tau]}$

3. $\mathcal{V}'$ accepts iff $\mathcal{V}$ on input $(\mathsf{pk}, x, \mathsf{tr})$ accepts and $h =_? \mathsf{H}(\sigma, x, \mathsf{pk})$.

---

[a]Although a transcript tr' in the non-interactive scheme lacks commitments to the opened views, omitting those as inputs to $\mathcal{C}$ and $\mathcal{R}$ doesn't affect correctness/security since these algorithms do not use them.

C.3   Security

Theorem 3. *Let MPCitH-VE$_R$ be an MPC-in-the-head-based VE scheme in the class described in Protocol 2 that is ϵ_{val}-valid and ϵ_{zk}-HVZK. Suppose the Commit function of ECOM used for instantiating MPCitH-VE$_R$ has α-bit min-entropy. Then the corresponding non-interactive VE scheme, MPCitH-NIVE$_R$ described in Protocol 3 is ϵ'_{val}-valid with validity error $\epsilon'_{\mathsf{val}} \leq (Q+1) \cdot \epsilon_{\mathsf{val}}$ and ϵ'_{zk}-ZK with $\epsilon'_{\mathsf{zk}} \leq \epsilon_{\mathsf{zk}} + Q/2^{\tau N \alpha}$ against adversaries making at most q queries to the random oracle H.*

Proof. We prove both properties separately.

ZK follows from HVZK of the base protocol MPCitH-VE$_R$. Let us denote a simulator for MPCitH-VE$_R$ by $\mathcal{S}$. The corresponding ZK simulator $\mathcal{S}'$ proceeds as follows: (1) invoke $\mathcal{S}$ on input (pk, x) to obtain

$$\mathsf{tr} = ((\mathsf{C}_i^{(j)})_{i \in [N]}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]} ,$$

(2) program the random oracle such that then $\mathsf{H}((\mathsf{C}_i^{(j)})_{i \in [N], j \in [\tau]}, x, \mathsf{pk}) := (\mathbf{e}^{(j)})_{j \in [\tau]}$ if the corresponding entry is not defined (otherwise $\mathcal{S}'$ aborts), and (3) output $\mathsf{tr}' = ((\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$. Since each commitment $\mathsf{C}_i^{(j)}$ has α bits of min-entropy, the min-entropy of in total $\tau \cdot N$ commitments is lower-bounded by $\tau N \alpha$. Hence the probability that programming the RO fails is at most $Q/2^{\tau N \alpha}$. Unless programming the RO fails, a ZK distinguisher against MPCitH-NIVE$_R$ can be used to break HVZK of MPCitH-VE$_R$. Hence, we obtain the bound $\epsilon_{\mathsf{zk}} + Q/2^{\tau N \alpha}$. We remark that will always have $\alpha \geq \lambda$, since even if the commitments are not randomized, the random share of the witness has $|w|$ bits of min-entropy, and $|w|$ must be λ bits long otherwise w may be recovered from x in less than $O(2^\lambda)$ work.

Validity follows from validity of the base protocol $\mathsf{MPCitH\text{-}VE}_R$. Given a pair of validity adversaries $(\mathcal{A}', \mathcal{P}')$ against $\mathsf{MPCitH\text{-}NIVE}_R$, we construct a reduction $(\mathcal{A}, \mathcal{P})$ breaking $\mathsf{MPCitH\text{-}VE}_R$ as follows.

1. On receiving a key pair $(\mathsf{pk}, \mathsf{sk})$ from Gen, $\mathcal{A}$ forwards it to $\mathcal{A}'$

2. On receiving a statement x from $\mathcal{A}'$, $\mathcal{A}$ outputs x. $\mathcal{A}$ also forwards $(\mathsf{pk}, \mathsf{sk}, x)$ to $\mathcal{P}'$.

3. $\mathcal{P}'$ initially chooses $k \in [1, Q+1]$ uniformly at random. On receiving the kth query to the RO of the form (σ, x, pk) from $\mathcal{P}'$, $\mathcal{P}$ forwards σ to $\mathcal{V}$ to obtain the corresponding challenge h. It then programs the RO such that $\mathsf{H}(\sigma, x, \mathsf{pk}) := h$. For all the other queries, it lazily samples challenge from Ch^τ and returns it as a response.

4. On receiving a transcript $\mathsf{tr}' = ((\mathsf{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}}, \mathbf{e}^{(j)}, (V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$ from $\mathcal{P}'$, $\mathcal{P}$ reconstructs the opened commitments and makes a query to the RO with input $((\mathsf{C}_i^{(j)})_{i \in [N], j \in [\tau]}, x, \mathsf{pk})$. If $\mathcal{V}'$ accepts tr' and $(\mathsf{C}_i^{(j)})_{i \in [N], j \in [\tau]} = \sigma$, i.e., the commitments match the kth query input, it forwards $((V_i^{(j)}, r_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$ to $\mathcal{V}$. Otherwise, it aborts.

Because the kth RO query σ is forwarded to the validity game for $\mathsf{MPCitH\text{-}VE}_R$, the corresponding RO output matches the challenge sent by the interactive verifier $\mathcal{V}$. In that case, since $\mathcal{C}$ and $\mathcal{R}$ of the non-interactive scheme are identical to those of the interactive scheme, $\mathcal{P}$ wins the validity game for $\mathsf{MPCitH\text{-}VE}_R$[8]. The probability that a random guess of the query index k succeeds is $1/(Q+1)$. Hence, we have that $\epsilon'_{\mathsf{val}}/(Q+1) \leq \epsilon_{\mathsf{val}}$.

$\square$

D   Details of Our DKG-in-the-Head Protocols

In Protocols 4 and 5 we give a detailed description of our DKG-in-the-head protocols to prove knowledge of a discrete logarithm. The protocols are discussed in Section 5.1. In this section we also describe how we instantiate the protocols (choices of the PKE and parameters) and what optimizations we make. Then we evaluate the costs in terms of proof size and prover and verifier time.

D.1   Verifiable Encryption from DKGitH

Following Protocol 2, we can directly convert Protocol 4 to a verifiable encryption scheme for discrete logarithms, using any PKE meeting the requirements of Section 3. In this section we make a specific choice of PKE that allows some optimizations and comparison to previous work. We use an instance of the hashed Elgamal scheme in the same group $\mathbb{G}$ as our discrete logarithm relation, where plaintexts are elements of $\mathbb{Z}_p$. The scheme uses a hash function $H_p : \{0,1\}^* \to \mathbb{Z}_p$ modeled as a random oracle. Key pairs are of the form $(sk, pk) = (z, g^z)$ and $\mathsf{Enc}(pk, m)$ outputs (c_1, c_2) where $c_1 = g^r$ for $r \leftarrow_\$ \mathbb{Z}_p$ and $c_2 = H_p(pk^r) + m \pmod{p}$.

Creating a VE ciphertext from a transcript is similar to Protocol 2, but we can compress ciphertexts even further by using the homomorphic property of Elgamal to combine the partially reconstructed witness with the encrypted share. The compression function $\mathcal{C}(\mathsf{tr})$ proceeds as follows:

1. For each execution $j \in [\tau]$

   (a) Recompute $x_i^{(j)}$ as in Verify, then compute $\tilde{x}^{(j)} = \sum_{i \neq \bar{i}_j} x_i^{(j)}$.

---

[8]We remark that $\mathcal{C}$ may be randomized and it is thus only guaranteed that the *probability* that $\mathcal{P}'$ wins is identical to that of $\mathcal{P}$ conditioned on the event that guessing k succeeded.

      (b) Parse $C_{\bar{i}_j}$ as the Elgamal ciphertext (c_1, c_2). Set $C^{(j)} = (c_1, c_2 + \tilde{x}^{(j)})$

  2. Output $C = (C^{(j)})_{j \in [\tau]}$ as the ciphertext.

Decryption is somewhat simpler than in Protocol 2 as $C^{(j)}$ in C now encrypt the witness directly, rather than encrypting shares. The function $\mathcal{R}(sk, C)$ proceeds as follows:

  1. For each repetition $j \in [\tau]$,

      (a) Decrypt $C^{(j)}$ using sk to get $x' \in \mathbb{Z}_p$

      (b) If $y = g^{x'}$ output x'.

As the soundness error of the proof protocol is assumed to be negligible, decryption will succeed in Item 1b for one of the repetitions with overwhelming probability.

We now point out another optimization that reduces the proof size. For each ciphertext the prover computes, we use $y_i^{(j)} = g^{x_i^{(j)}}$ as the c_1 component of the Elgamal ciphertext. This reduces the ciphertext size in the proof to a single element of $\mathbb{Z}_p$ and saves one exponentiation, reducing the prover's total number of exponentiations by τN. The size of the output ciphertext C stays the same though, since the value $g^{x_{\bar{i}}^{(j)}}$ must be present for decryption. However, this optimization makes the encryption scheme deterministic and therefore HVZK of the VE scheme cannot rely on the IND-CPA security anymore. Although one could provide a dedicated analysis in the GGM and the ROM under a relaxed version of HVZK (where an instance y is uniformly sampled),[9] our main goal is to avoid extra assumptions so we opt to omit this optimization in our implementations.

HVZK We prove perfect $(N-1)$-privacy of the underlying MPC protocol Π_f, to conclude HVZK via Theorem 1.

Suppose we are given the index of a party $\bar{i}$, we show that we can simulate the views of the other $N-1$ parties, such that simulated and real transcripts are perfectly indistinguishable. First the simulator chooses x_i at random, for $i \neq \bar{i}$ and computes $y_i = g^{x_i}$, as in the real protocol. Then for party $\bar{i}$, the simulator sets $y_{\bar{i}} = y/(\prod_{i \neq \bar{i}} y_i)$. Note that

$$y_{\bar{i}} = g^{x - \sum_{i \neq \bar{i}} x_i}, \text{ and } x_{\bar{i}} = x - \sum_{i \neq \bar{i}} x_i$$

are distributed exactly as in the real protocol.

Validity Although Protocol 4 already incorporates the optimizations described in Section 3.2, the underlying MPCitH-IOP protocol instantiated with Π_f does satisfy the requirements from Section 2.2 so that our general compiler theorem applies: the challenge space is $\mathsf{Ch} = \{\mathbf{e} \subset [N] : |\mathbf{e}| = N - 1\}$; party i's view V_i consists of $(x_i, (y_{i'})_{i' \neq i})$; the function $\mathsf{GetW}(V_i)$ outputs x_i; the function $\mathsf{CheckView}(y, (V_i)_{i \in \mathbf{e}})$ parses V_i as $(x_i, (y_{i',i})_{i' \neq i})$ and checks $y =_? g^{x_i} \prod_{i' \neq i} y_{i',i}$ for all $i \in \mathbf{e}$ and $y_{i',i} =_? g^{x_{i'}}$ for $i \in \mathbf{e}$ and $i' \in [N] \setminus \{i, \bar{i}\}$, where $\bar{i} \notin \mathbf{e}$ is the index of the unopened party. We can prove the protocol has (i) 2-consistency (Definition 10) and thus is (ii) SLE with $\epsilon_{\mathsf{sle\text{-}iop}} = 1/N$ (Lemma 4). Showing the condition 1. from 2. in Definition 10 is trivial. To show the converse, let $\mathbf{e}, \mathbf{e}'$ be two distinct challenges. If $\mathsf{CheckView}$ outputs 1 w.r.t. both challenges, then for some i such that $i \in \mathbf{e} \cap \mathbf{e}'$, it must be that $y = g^{x_i} \prod_{i' \neq i} g^{x_{i'}} = g^{x_1 + \cdots + x_N}$. Hence, $(V_1, \ldots, V_N)$ form an honest execution of Π_f on y and witness shares $(x_i)_{i \in [N]}$ as inputs.

Parameter choices As an interactive protocol, it's easy to see that the soundness error is $1/N$. When executing τ repetitions in parallel, we therefore need to set τ so that

---

[9] On a high-level, the proof of relaxed HVZK amounts to showing an unopened ciphertext $(g^{x_{\bar{i}}^{(j)}}, H_p(g^{\mathsf{sk} \cdot x_{\bar{i}}^{(j)}}) + x_{\bar{i}}^{(j)})$ is indistinguishable with $(g^{x_{\bar{i}}^{(j)}}, \rho)$ for uniformly random $\rho \in \mathbb{Z}_p$. If $g^{x_{\bar{i}}^{(j)}}$ and g^{sk} are uniformly random, an adversary in the GGM never queries the random oracle $H_p()$ with input $g^{\mathsf{sk} \cdot x_{\bar{i}}^{(j)}}$ except with negligible probability. Thus, we can regard the second component as a uniformly random and independent $\mathbb{Z}_p$-element.

$(1/N)^\tau < 2^{-\lambda}$, or equivalently $\tau \log_2(N) \geq \lambda$. This offers a range of choices for (N, τ) for each choice of λ, with a different balance of proof size and running time.

Efficiency Starting with the proof, without our transform, when Commit is implemented with a hash function with 2λ-bit digests, the size is

$$4\lambda + \tau(\lambda \lceil \log_2 N \rceil + 2\lambda + \ell_p) .$$

bits where ℓ_p is the bitlength of an integer modulo the group order. When our transform is applied, the term 2λ for $\mathsf{C}_{\bar{i}}^{(j)}$ is replaced with ℓ_C, the size of a PKE ciphertext. With our transform, the size of a VE ciphertext is $\tau(\ell_p + \ell_C)$ for any PKE, and $\tau \ell_p$ for our choice of hashed Elgamal. With Elgamal, the computational costs of the protocol can be roughly estimated by counting the exponentiations in $\mathbb{G}$. With the encryption scheme and optimizations described above, the prover must compute $2\tau N$ exponentiations and the verifier must compute $2\tau(N-1)$ exponentiations.

D.2   Verifiable Encryption from RDKGitH

Analogously, we can convert Protocol 5 into a verifiable encryption scheme by replacing a committing function with the hashed Elgamal encryption.

To precompute the Lagrange interpolation operations, the compression function $\mathcal{C}(\mathsf{tr})$ proceeds as follows:

1. For each unopened party $\bar{i} \in [N] \setminus I$

   (a) Let $S_{\bar{i}} = I \cup \{\bar{i}\}$. Define the Lagrange coefficients determined by $S_{\bar{i}}$ as follows: $\delta_{S_{\bar{i}}, i} := \prod_{j \neq i} \frac{j}{j-i}$ for $i \in S_{\bar{i}}$.

   (b) Parse $\mathsf{C}_{\bar{i}}$ as the hashed Elgamal ciphertext (c_1, c_2). Set $\mathsf{C}'_{\bar{i}} = (c_1, \delta_{S_{\bar{i}}, \bar{i}} \cdot c_2 + \sum_{i \in I} \delta_{S_{\bar{i}}, i} \cdot x_i)$.

   (c) Set $\mathsf{aux}_{\bar{i}} = \delta_{S_{\bar{i}}, \bar{i}}$

2. Output $C = (\mathsf{C}'_{\bar{i}}, \mathsf{aux}_{\bar{i}})_{\bar{i} \notin I}$ as the ciphertext.

The receiver function $\mathcal{R}(sk, C)$ proceeds as follows:

1. For each $\bar{i} \in [N] \setminus I$:

   (a) Parse $\mathsf{C}'_{\bar{i}}$ as (c_1, c'_2).

   (b) Compute $x' = c_2 - \mathsf{aux}_{\bar{i}} \cdot H_p(c_1^{\mathsf{sk}})$

   (c) If $y = g^{x'}$ output x'.

As the soundness error of the proof protocol is assumed to be negligible, decryption will succeed in Item 1c for one of the indices with overwhelming probability.

Validity We show that no cheating prover can cause extraction failure unless she guesses the exact I. This implies that the validity error is at most $1/\binom{N}{t}$.

- When a malicious prover $\mathcal{P}^*$ commits to $((A_i)_{i=1}^t, (\mathsf{C}_i)_{i=1}^N)$, the ciphertexts uniquely determine the corresponding shares x_i for $i \in [N]$. Moreover, note that $(A_0, \ldots, A_t)$ serves as a perfectly binding commitment to a degree-t polynomial a.

- If there exists some subset $I' \subset [N]$ of size at least $t+1$ such that for $i \in I'$, it holds that $g^{x_i} = \sum_j A_j^{i^j}$, then it must be that the polynomial a' interpolated from $(x_i)_{i \in I'}$ coincides with $a(x) = a_0 + \ldots + a_t X^t$ implicit in $(A_0, \ldots, A_t)$. Therefore, in this case the receiver always succeeds in extracting the correct DLog a_0 from a.

- Suppose at most t of all x_i's satisfy the verification equation. In this case, it is not guaranteed that the receiver succeeds in extracting the right DLog since it fails to recover the committed polynomial a. However, the prover $\mathcal{P}^*$ gets caught unless she guesses the exact I so that only $(x_i)_{i \in I}$ satisfy the verification equation. The probability that she can do so is at most $1/\binom{N}{t}$.

HVZK For ease of exposition, we use the additive notation for proof. The HVZK simulator, upon receiving a statement $y = A_0$ as input, first picks $I \subset [N]$ with $|I| = t$ uniformly at random. Then it simulates a transcript as follows.

1. For $i \in I = \{i_1, \ldots, i_t\}$, pick uniform $x_i \in \mathbb{Z}_p^*$ and let $B_i = x_i \cdot G - A_0$.

2. Consider the Vandermonde matrix

$$V = \begin{bmatrix} i_1, i_1^2, \ldots i_1^t \\ \vdots \\ i_t, i_t^2, \ldots i_t^t \end{bmatrix}$$

   Then one can find $A_1, \ldots, A_t$ such that

$$[B_{i_1}, \ldots, B_{i_t}]^\tau = V \cdot [A_1, \ldots, A_t]^\tau$$

   since V is invertible.

3. Finally, the simulator encrypts x_i for $i \in I$ as an actual prover would, and produces C_i for $i \notin I$ by encrypting 0 string.

Random Subset Optimization We can apply the random subset method to compress the ciphertext. Once the verifier verifies the proof, there are $N - t$ unopened ciphertexts in the transcript. The compression algorithm selects n of them uniformly at random. The validity error analysis is analogous to Section 4.1. Let $n \leq s \leq N - t$ be the number of "bad" ciphertexts, i.e., $\mathsf{Dec}(\mathsf{C}_i)$ does not satisfy the verification condition. Then the probability that a malicious prover breaks validity is:

$$\Pr[\mathcal{C} \text{ selects } n \text{ of } s \text{ bad ctexts} \wedge \mathcal{V} \text{ accepts a proof with } s \text{ bad ctexts}]$$

$$= \frac{\binom{s}{n}}{\binom{N-t}{n}} \cdot \left(\frac{\binom{N-s}{t}}{\binom{N}{t}} + \epsilon_{\text{cext}} \right)$$

We can then take the maximum over s to find the updated validity error.

E IOP Versions of MPC-in-the-Head Proofs

E.1 Further Details of MPCitH-IOP

We first introduce the notion of k-consistency, which essentially guarantees N views form an honest run of Π_f, as long as for any k distinct subsets of party indices, the corresponding parties' views are consistent with each other. This generalizes the notion of pairwise consistency introduced previously in [IKOS07, Def. 2.2].

Definition 10 (k-consistency)**.** A single repetition of the protocol MPCitH-IOP$_R$ has k-consistency if for any x, for any set of views $(V_1, \ldots, V_N)$ and for any subset of the the challenge space $S \subseteq \mathsf{Ch}$ such that $|S| \geq k$, the following two conditions are equivalent:

1. for every $\mathbf{e} \in S$, $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$;

2. $(V_1, \ldots, V_N)$ form an honest execution of Π_f on a public input x and the corresponding per-party private inputs, $w_i = \mathsf{GetW}(V_i)$, are such that $x = f(\sum_{i \in [N]} w_i)$.

Remark 4. The above notion captures several different instantiations of MPC-in-the-head protocols. For example, the original protocol from [IKOS07, §3] opens 2-out-of-N parties (i.e., $t = 2$ in MPCitH-IOP$_R$) and satisfies $\binom{N}{2}$-consistency because their Lemma 2.3 only guarantees the validity of N views as long as every possible pair of the views is consistent. ZKBoo and ZKB++ are essentially a special case of that protocol with $N = 3$ and therefore

they have 3-consistency. Looking ahead, our DKGitH protocol in Section 5.1 works with N parties and the challenge set Ch is all subsets of $[N]$ of size $t = N - 1$.[10] We will show it satisfies 2-consistency thanks to the use of a broadcast functionality.

Now we prove knowledge error bounds for generic IOPs with k-consistency.

Lemma 4. *If a single repetition of* MPCitH-IOP$_R$ *has k-consistency, then it is SLE with respect to the canonical extractor* $\mathbf{E}$ *with knowledge error* $\epsilon_{\mathsf{sle-iop}} \leq \frac{k-1}{|\mathsf{Ch}|}$.

Proof. Let V_i be the views output by a cheating prover $\mathbf{P}^*$ in the committing phase and $\mathbf{e} \in \mathsf{Ch}$ is the challenge sampled uniformly by the verifier $\mathbf{V}$ in the query phase. Further, let $w' = \sum_{i \in [N]} \mathsf{GetW}(V_i)$. Our goal is to bound the probability

$$\Pr\left[\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1 \wedge (x, w') \notin R\right]. \tag{2}$$

Define $\mathsf{GoodCh} := \left\{\mathbf{e} \in \mathsf{Ch} : \mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1\right\}$, i.e., a set of challenges that are accepting with respect to views $(V_i)_{i \in [N]}$ committed to by $\mathbf{P}^*$. If $|\mathsf{GoodCh}| \geq k$, then it must be that $(x, w') \in R$ due to k-consistency, so the canonical extractor always succeeds. If $|\mathsf{GoodCh}| < k$, then since $\mathbf{e}$ is sampled from Ch independently of V_i and thus of GoodCh as well, the probability that $\mathbf{e}$ falls in GoodCh is at most $\frac{k-1}{|\mathsf{Ch}|}$. $\qquad\square$

One can easily generalize the above argument to deal with parallel repetitions.

Lemma 5. *If one repetition of* MPCitH-IOP$_R$ *has k-consistency, then τ parallel repetitions are SLE with respect to the canonical knowledge extractor* $\mathbf{E}^\tau$ *with knowledge error* $\epsilon_{\mathsf{sle-iop}} \leq \left(\frac{k-1}{|\mathsf{Ch}|}\right)^\tau$.

Proof. This is a straightforward generalization of Lemma 4. For a set of views output by $\mathbf{P}^*$ in each parallel repetition $j \in [\tau]$, one can define

$$\mathsf{GoodCh}^{(j)} := \left\{\mathbf{e} \in \mathsf{Ch} : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}}) = 1\right\}.$$

If there exists $j \in [\tau]$ such that $|\mathsf{GoodCh}^{(j)}| \geq k$, then the extractor $\mathbf{E}^\tau$ succeeds in extracting a valid witness $w^{(j)}$ due to k-consistency. If for all $j \in [\tau]$ $|\mathsf{GoodCh}^{(j)}| < k$, then the probability that the τ independently sampled challenges simultaneously fall into $\mathsf{GoodCh}^{(j)}$ for $j \in [\tau]$ is at most $\left(\frac{k-1}{|\mathsf{Ch}|}\right)^\tau$. $\qquad\square$

Remark 5. We note that the above knowledge error is equivalent to the soundness error. For example, for ZKBoo and ZKB++ we have that $k = 3$ and $\mathsf{Ch} = \{\{1,2\}, \{2,3\}, \{3,1\}\}$ and therefore both the SLE knowledge error and soundness error are $(2/3)^\tau$.

Finally, we recall the notion of t-privacy for an MPC protocol from [IKOS07]. We show t-privacy implies HVZK of the MPC-in-the-head IOP. Although we only consider the case of *perfect* t-privacy and HVZK, one can obtain a similar claim for statistical security of the lemma following the result of [IKOS07].

Definition 11 (t-privacy). The protocol Π_f is said to be t-private if there exists a PPT simulator Sim such that for every $\mathbf{e} \in [N]$ of size at most t and for every input $(x, w_1, \ldots, w_N)$, the joint view of parties in $\mathbf{e}$ is distributed identically to $\mathsf{Sim}(\mathbf{e}, x, (w_i)_{i \in \mathbf{e}}, b)$ where $b = 1$ if $(x, \sum_{i \in [N]} w_i) \in R$ and $b = 0$ otherwise.

Lemma 6. *If the MPC protocol Π_f is t-private, then* MPCitH-IOP$_R$ *is perfectly HVZK.*

---

[10]In practice, it suffices to send a single party index $\bar{i}$ whose view is *not* to be revealed.

Proof. An IOP simulator $\mathbf{S}$ takes x as input and proceeds as follows: (1) sample $\mathbf{e} \subset [N]$ of size t uniformly at random, (2) choose uniformly random witness shares w_i for $i \in \mathbf{e}$, (3) invoke $\mathsf{Sim}(\mathbf{e}, x, (w_i)_{i \in \mathbf{e}}, 1)$ to obtain joint views V_i for $i \in \mathbf{e}$, and (4) output $(\mathbf{e}, (V_i)_{i \in \mathbf{e}})$. This perfectly simulates the view of $\mathbf{V}(x)$ in the honest interaction with $\mathbf{P}$, since an honest $\mathbf{V}$ always queries the oracle with a set of party indices of size t and thus the t-privacy property guarantees perfect simulation of revealed views in the MPC execution. $\square$

E.2   Protocols without k-consistency

While the notion of k-consistency has some generality and gives a convenient way to prove SLE of some three-round protocols, many MPC-in-the-head proof systems such as KKW and Banquet have challenge spaces not limited to party indices and therefore do not have k-consistency. However, we show that they are easily checked to be straight-line extractable since $\mathbf{P}$ outputs per-party views that include the shares of the witness in the first round of the committing phase. The existing soundness analysis thus implies SLE of the corresponding IOP protocols.

E.2.1   Protocols with Robustness

It is well-known that an MPC protocol with the *robustness* property allows the corresponding MPCitH proof system to avoid parallel repetitions for negligible (knowledge) soundness error [IKOS07]. To the best of our knowledge, Ligero [AHIV17] and Ligero++ [BFH+20] are the only concrete instantiation utilizing robustness within the MPCitH framework. In the full version of Ligero [AHIV22], the authors provide their version of MPCitH-based IOP, denoted by Π_{ZKIPCP}.[11] By their soundness analysis [AHIV22, Theorem 3.5], Π_{ZKIPCP} immediately satisfies SLE with knowledge error

$$\epsilon_{\mathsf{sle\text{-}iop}}(N, t_p, t_r) \leq \frac{\binom{N-t_r}{t_p}}{\binom{N}{t_p}} + \delta$$

where t_p is a parameter for privacy, t_r is a parameter for robustness, δ is the robustness error of the underlying MPC protocol, and the knowledge extractor $\mathbf{E}$ simply takes N views and reconstruct a witness from N witness shares supplied to computing servers.

E.2.2   Protocols using Threshold Linear Secret Sharing

The very recent work due to Feneuil and Rivain [FR23] showed that even a semi-honest secure MPC protocol allows for comparably low soundness error if combined with *threshold linear secret sharing scheme (LSSS)*. A (t, N)-LSSS scheme over $\mathbb{F}$ consists of a tuple (Share, Recon), where Share takes a secret $s \in \mathbb{F}$ as input and returns $(s_1, \ldots, s_N)$ as output, and Recon takes $t + 1$ shares $(s_{i_1}, \ldots, s_{i_{t+1}})$ as input and returns a reconstructed secret s as output, respectively. Their LSSS-based MPCitH protocol [FR23, Protocol 6] can be "de-compiled" into an IOP by having a prover send i-th party's view as an oracle instead of committing to it. Then by their soundness analysis [FR23, Theorem 2], the corresponding IOP satisfies SLE with knowledge error

$$\epsilon_{\mathsf{sle\text{-}iop}}(N, t, p) \leq \frac{1}{\binom{N}{t}} + p \cdot \frac{t(N - t)}{t + 1}$$

where p is the false-positive rate of the underlying MPC protocol, and the knowledge extractor $\mathbf{E}$ simply takes N views and reconstruct a witness from t opened witness shares and one of the unopened shares.

---

[11]IPCP is a special case of IOP

E.2.3 KKW as an IOP

KKW [KKW18] is an MPC-in-the-head proof system that produces much more compact proofs than [IKOS07] thanks to the *preprocessing phase*. In this paradigm, the prover first runs an offline protocol (also "in the head") to compute correlated randomness. Then it proceeds by executing the corresponding online phase taking the secret witness as input. For completeness, we present in Protocol 6 the underlying interactive protocol of KKW characterized as a single-round IOP.

Notice that in the first message $\mathbf{P}$ includes per-party states $(\mathsf{st}_i)_{i \in [N]}$ and the masked witness $\hat{w}$ for each MPC execution. If the function GetW is defined such that it outputs the ith share of witness mask λ_i^w on input st_i, a witness is recovered by computing $w = \sum_{i \in [N]} \lambda_i^w \oplus \hat{w}$. Hence, an extractor $\mathbf{E}$ for a single repetition similar to the canonical extractor can be defined assuming it takes per-party states and a masked witness as input. The following claim is implicit in [KKW18].

Lemma 7. KKW-IOP *is SLE with knowledge error*

$$\epsilon_{\mathsf{sle\text{-}iop}}(N, M, \tau) \leq \max_{M-\tau \leq k \leq M} \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau}} \cdot \left(\frac{1}{N}\right)^{k-(M-\tau)}$$

where the parameters (N, M, τ) are as defined in Protocol 6 and an extractor $\mathbf{E}^M$ proceeds as follows: on receiving x and $\pi = (\mathsf{sd}^{(j)}, (\mathsf{st}_i^{(j)}, \mathsf{msgs}_i^{(j)})_{i \in [N]}, \hat{w}^{(j)})_{j \in [M]}$, it outputs $w^{(j)} = \mathbf{E}(x, (\mathsf{st}_i^{(j)})_{i \in [N]}, \hat{w}^{(j)})$ if $(x, w^{(j)}) \in R$ for some $j \in [M]$. Otherwise, it outputs $\perp$.

This is a direct consequence of "Beating parallel repetition" of [KKW18]. Suppose a witness w extracted from the proof string form "bad inputs" for MPC, i.e., $w = \sum_{i \in [N]} \lambda_i^w \oplus \hat{w}$ while $f(w) \neq x$. In that case, since the input does not lead to the correct output of the circuit, to pass verification checks it must be that for every MPC execution either (1) the offline computation is not carried out correctly, or (2) input and/or circuit has been modified during the online phase. Such cheating behaviors are caught with the above knowledge error as already analyzed in Theorem 2.2 of [KKW18].

E.2.4 Banquet as an IOP

Banquet [BDK+21] is a recent MPC-in-the-head proof system that allows a prover to prove knowledge of an AES key with a relatively short proof. The base MPC protocol for computing the AES circuit uses secret-sharings over the field $\mathbb{F}_{2^s}$. For completeness, we present in Protocol 7 the underlying interactive protocol of Banquet characterized as a 3-round IOP. The protocol is highly optimized for the relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$, where $(\mathsf{ct}, \mathsf{pt})$ is a public ciphertext-plaintext pair and $K = w$ is a witness. Notice that in the first message $\mathbf{P}$ includes commitments to per-party random seeds $(\mathsf{sd}_i)_{i \in [N]}$ and the offset Δw. If the function GetW is defined such that it outputs the ith witness share w_i on input sd_i, a witness can be easily recovered by computing $w = \sum_{i \in [N]} w_i + \Delta w$. Hence, an extractor $\mathbf{E}$ for a single repetition similar to the canonical extractor can be defined assuming it takes per-party seeds and the offset as input. The following claim is implicit in [BDK+21].

Lemma 8. Banquet-IOP *is SLE with knowledge error*

$$\epsilon_{\mathsf{sle\text{-}iop}}(\lambda, m_2, N, \tau) \leq (p_1 + (1-p_1)p_2 + (1-p_1)(1-p_2)p_3)^\tau$$

where $p_1 = 2^{-8\lambda}$, $p_2 = 2m_2/(2^{8\lambda} - m_2)$, and $p_3 = 1/N$ and an extractor $\mathbf{E}^\tau$ proceeds as follows: on receiving x and $\pi_1^{(e)} = ((\mathsf{sd}_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \in [N]}, \Delta w^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]})_{e \in [\tau]}$, it outputs $w^{(e)} = \mathbf{E}(x, (\mathsf{sd}_i^{(e)})_{i \in [N]}, \Delta w^{(e)})$ if $(x, w^{(e)}) \in R$ for some $e \in [\tau]$. Otherwise it outputs $\perp$.

Likewise, the above probability corresponds to the soundness error of interactive Banquet.[12] In case of "bad inputs", it must be that the cheating prover guessed at least one of three challenges for each parallel repetition to pass the verification checks. The p_1, p_2 and p_3 above correspond to the probabilities that $\mathbf{P}^*$ correctly guesses first, second, and third challenge, respectively.

Protocol 7: Banquet-IOP for relation $R = \big\{((\mathsf{ct}, \mathsf{pt}), K) \ : \ \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\big\}$

Parameters: The number of parties N; the parameter for field extension λ; the number of S-boxes m; the number of checking polynomials m_1 and the degree m_2 such that $m = m_1 \cdot m_2$ and $m_2 < 8\lambda$.

Inputs: $\mathbf{P}$ receives (x, w); $\mathbf{V}$ receives x, where $x = (\mathsf{ct}, \mathsf{pt})$ and $w = K$.

Committing phase 1:   The first-round message of $\mathbf{V}$ is empty. The prover proceeds as follows.

  1. The prover picks at random seeds $\mathsf{sd}_1, \ldots, \mathsf{sd}_N$.

  2. For each party $i \in [N]$:

     (a) Expand sd_i into tape_i

     (b) Sample witness share w_i from tape_i

  3. It computes witness offset $\Delta w = w - \sum_i w_i$ and adjust first share $w_1 := w_i + \Delta w$.

  4. For each S-box ℓ:

     (a) For each party $i \in [N]$, compute the local linear operations to obtain the share $s_{i,\ell}$ of the S-box of input s_ℓ.

     (b) Compute the S-box output $t_\ell = (\sum_i s_{i,\ell})^{-1}$.

     (c) For each party $i \in [N]$, sample the share of the output $t_{i,\ell}$ from tape_i

     (d) Compute output offset $\Delta t_\ell = t_\ell - \sum_i t_{i,\ell}$.

     (e) Adjust first share $t_{1,\ell} = t_{1,\ell} + \Delta t_\ell$

  5. Broadcast each party's share ct_i of the output.

  6. Send an oracle $\pi_1 = ((\mathsf{sd}_i, \mathsf{ct}_i)_{i \in [N]}, \Delta w, (\Delta t_\ell)_{\ell \in [m]})$

Committing phase 2:   The second-round message of $\mathbf{V}$ is $(r_j)_{j \in [m_1]}$ where $r_j \in \mathbb{F}_{2^{8\lambda}}$. The prover proceeds as follows.

  1. For each party $i \in [N]$ and S-box $\ell \in [m]$, lift $s_{i,\ell}$ and $t_{i,\ell}$ to $\mathbb{F}_{2^{8\lambda}}$.

  2. For $i \in [N]$ and $j \in [m_1]$:

     (a) For $k \in [0, m_2 - 1]$, set $s'_{i,j,k} = r_j \cdot s_{i,j+km_1}$ and $t'_{i,j,k} = t_{i,j+km_1}$

     (b) Sample masking points $\bar{s}_{i,j}$ and $\bar{t}_{i,j}$ from tape_i.

     (c) Interpolate degree m_2 polynomials $S, T \in \mathbb{F}_{2^{8\lambda}}[X]$ such that

     $$S_{i,j}(k) = s'_{i,j,k} \text{ for } k \in [0, m_2 - 1], \qquad S_{i,j}(m_2) = \bar{s}_{i,j}$$
     $$T_{i,j}(k) = t'_{i,j,k} \text{ for } k \in [0, m_2 - 1], \qquad T_{i,j}(m_2) = \bar{t}_{i,j}$$

---

[12] Technically, *soundness error* is the probability that the cheating prover convinces the verifier given an invalid statement $x \notin L$, while for *knowledge error* it might be that $x \in L$. However, the analysis given in §3.2 of [BDK$^+$21] already covers the latter case because it derives the probability that a cheating behavior can go undetected assuming the input does not satisfy the circuit.

3. Compute product polynomial

$$P := \sum_{j \in [m_1]} \left(\sum_i S_{i,j} \right) \cdot \left(\sum_i T_{i,j} \right)$$

4. For $k \in [m_2, 2m_2]$, compute offset $\Delta P(k) = P(k) - \sum_i \mathsf{Sample}(\mathsf{tape}_i)$

5. For $i \in [N]$ interpolate ith share of degree $2m_2$ polynomial $P_i \in \mathbb{F}_{2^{8\lambda}}[X]$ such that

$$\text{For } k \in [0, m_2 - 1]: \qquad P_1(k) = \sum_j r_j,$$
$$P_i(k) = 0 \text{ for } i \neq 1$$
$$\text{For } k \in [m_2, 2m_2]: \qquad P_1(k) = \mathsf{Sample}(\mathsf{tape}_1) + \Delta P(k),$$
$$P_i(k) = \mathsf{Sample}(\mathsf{tape}_i) \text{ for } i \neq 1$$

Send an oracle $\pi_2 = (\Delta P(k))_{k \in [m_2, 2m_2]}$

Committing phase 3:   The third-round message of **V** is $R \in \mathbb{F}_{2^{8\lambda}} \setminus [0, m_2 - 1]$. The prover proceeds as follows.

1. For each party $i \in [N]$, compute $a_{i,j} = S_{i,j}(R)$ and $b_{i,j} = T_{i,j}(R)$ for $j \in [m_1]$, and $c_i = P_i(R)$.

2. Send an oracle $\pi_3 = ((a_{i,j}, b_{i,j})_{j \in [m_1]}, c_i)_{i \in [N]}$

Query phase

1. The verifier picks at random $\bar{i} \in [N]$ and queries π_1 with $\bar{i}$.

2. The oracle π returns $((\mathsf{sd}_i)_{i \neq \bar{i}}, \mathsf{ct}_{\bar{i}}, \Delta w, (\Delta t_\ell)_{\ell \in [m]})$.

3. The verifier queries π_2 and π_3 with empty strings

4. The oracle π_2 and π_3 return $(\Delta P(k))_{k \in [m_2, 2m_2]}$ and $((a_{i,j}, b_{i,j})_{j \in [m_1]}, c_i)_{i \in [N]}$, respectively.

Decision phase

1. For $i \neq \bar{i}$:

   (a) Expand sd_i into tape_i

   (b) Sample witness share w_i from tape_i

   (c) If $i = 1$, adjust $w_1 = w_1 + \Delta w$

   (d) For each S-box ℓ:

      i. Compute local linear operations to obtain the share $s_{i,\ell}$

      ii. Sample output share $t_{i,\ell}$ from tape_i

      iii. If $i = 1$ adjust, $t_{1,\ell} = t_{1,\ell} + \Delta t_\ell$

   (e) Recompute output broadcast $\tilde{\mathsf{ct}}_i$ and missing $\tilde{\mathsf{ct}}_{\bar{i}} = \mathsf{ct} - \sum_{i \neq \bar{i}} \tilde{\mathsf{ct}}_i$

   (f) For $j \in [m_1]$, interpolate polynomials $S_{i,j}$ and $T_{i,j}$ as the prover would.

   (g) Interpolate product polynomial P_i using $(\Delta P(k))_{k \in [m_2, 2m_2]}$ as the prover would.

   (h) For $j \in [m_1]$, compute $\tilde{a}_{i,j} = S_{i,j}(R)$ and $\tilde{b}_{i,j} = T_{i,j}(R)$. Compute $\tilde{c}_i = P_i(R)$.

2. Accept iff

- For $i \neq \bar{i}$, $c_i = \tilde{c}_i$
- For $i \neq \bar{i}$ and $j \in [m_1]$, $a_{i,j} = \tilde{a}_{i,j}$ and $b_{i,j} = \tilde{b}_{i,j}$
- $\mathsf{ct}_{\bar{i}} = \tilde{\mathsf{ct}}_{\bar{i}}$
- $\sum_i c_i = \sum_j (\sum_i a_{i,j}) \cdot (\sum_i b_{i,j})$

The prover and verifier execute τ instances of the above procedures in parallel. If the verifier accepts all τ executions, it outputs $b = 1$; otherwise it outputs $b = 0$.

F Compiling Banquet

In Protocol 8, we present our Banquet-based non-interactive VE scheme for AES, more precisely, the relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$. Proof of validity requires extra care, because the underlying interactive protocol has more than three rounds and thereby we cannot directly apply the analysis for MPCitH-NIVE presented in Appendix C.

F.1 Security against State-restoration Adversaries

When considering concrete (knowledge) soundness of $2r+1$-round interactive proof systems made non-interactive with the Fiat-Shamir transform, it is often useful to prove a stronger notion called *state-restoration* soundness, which is known to be tightly related to the soundness of the non-interactive protocol in the random oracle model [BCS16, COS20] [GT21]. In this setting, a cheating prover is allowed to rewind the verifier Q times to obtain a fresh challenge at any point. This naturally models the ability of a malicious prover to change the inputs to the hash function used to derive the challenge in the FS transform. They win the game if there exists an accepting transcript in the execution tree while the extractor fails to get a witness from that accepting transcript. In the context of IOPs, knowledge extractors are typically given access to the entire proof strings and one should thus extend the notion of straight-line extractability (Definition 7) to protect against state-restoration attacks, as we recall below.

To formally capture state-restoration attacks, we denote by $(b, \pi_{\mathrm{sr}}) \leftarrow \langle \mathbf{P}^*, \mathbf{V} \rangle_{\mathrm{sr}}(x)$ the following procedures: it lets the adversary $\mathbf{P}^*$ interact with the verifier $\mathbf{V}$ while allowing them to reset $\mathbf{V}$'s state to any previous round $i \in [1, r]$ by re-sending a proof string π_i. This implies that from the ith round onwards, $\mathbf{V}$'s messages are freshly sampled (including query strings). After Q such interactions, a sequence of proof strings, verifier messages, and oracle responses form an *execution tree* of depth r, where each node is labeled by a verifier message/query string and each edge is labeled by a proof/response string, respectively.[13] Finally, $\mathbf{P}^*$ specifies an *execution path* in the tree. If $\mathbf{V}$ accepts in that path, the procedure outputs $b = 1$ and π_{sr} consisting of proof strings in the path; otherwise it outputs $(0, \bot)$.

Definition 12 (State-restoration straight-line extractability (SR-SLE))**.** An IOP $(\mathbf{P}, \mathbf{V})$ is *state-restoration straight-line extractable* with knowledge error $\epsilon_{\mathsf{sr\text{-}sle}}$ if there exists an efficient extractor $\mathbf{E}$ such that for all pairs of unbounded state-restoration adversaries $(\mathbf{A}^*, \mathbf{P}^*)$

$$\Pr\left[b = 1 \wedge (x, w') \notin R : \begin{array}{l} x \leftarrow \mathbf{A}^*(1^\kappa); \\ (b, \pi_{\mathrm{sr}}) \leftarrow \langle \mathbf{P}^*, \mathbf{V} \rangle_{\mathrm{sr}}(x); \\ w' \leftarrow \mathbf{E}(x, \pi_{\mathrm{sr}}); \end{array} \right] \leq \epsilon_{\mathsf{sr\text{-}sle}}(\kappa).$$

---

[13]Here, we are implicitly considering a limited class of IOPs where the verifier only makes a single query to the first proof string π_1 in the query phase, and all the subsequent proof strings are given to the verifier in the clear. This is indeed sufficient to capture Banquet-IOP.

F.2   State-restoration SLE of Banquet-IOP

We first mention SR-SLE of Banquet-IOP, which is implicit in the EUF-KOA security analysis of the Banquet signature scheme.

Lemma 9. Banquet-IOP *is SR-SLE with knowledge error*

$$\epsilon_{\mathsf{sr\text{-}sle}}(\lambda, m_2, N, \tau) \leq \Pr\left[X + Y + Z = \tau\right] \tag{3}$$

where

- *the extractor is the same as* $\mathbf{E}^\tau$ *of Lemma 8;*
- *the random variables* X, Y, Z *are:*

$$X := \max_{\varpi_1 \in \mathcal{Q}_1} X_{\varpi_1} \qquad\qquad X_{\varpi_1} \sim \mathscr{B}(\tau, p_1)$$

$$Y := \max_{\varpi_2 \in \mathcal{Q}_2} Y_{\varpi_2} \qquad\qquad Y_{\varpi_2} \sim \mathscr{B}(\tau - X, p_2)$$

$$Z := \max_{\varpi_3 \in \mathcal{Q}_3} Z_{\varpi_3} \qquad\qquad Z_{\varpi_3} \sim \mathscr{B}(\tau - X - Y, p_3)$$

  where for $i = 1, 2, 3$, $\mathcal{Q}_i$ *consists of vectors of* τ *proof strings* $\varpi_i = (\pi_i^{(e)})_{e \in [\tau]}$ *sent by* $\mathbf{P}^*$*, where* $p_1 = 2^{-8\lambda}$*,* $p_2 = 2m_2/(2^{8\lambda} - m_2)$*, and* $p_3 = 1/N$*, where* $\mathscr{B}(n, p)$ *denotes the binomial distribution with parameters* $n \in \mathbb{N}$ *and* $p \in [0, 1]$*, i.e., if* $X \sim \mathscr{B}(n, p)$ *we have* $\Pr[X = i] = \binom{n}{i}p^i(1 - p)^{n-i}$*;*

- *other parameters are as defined in E.2.4.*

As the analysis is essentially a simplified version of proof for Lemma 2 of [BDK+21], we only sketch how it should be adapted in our setting. For any $\varpi_1 = (\pi_1^{(e)})_{e \in [\tau]} \in \mathcal{Q}_1$, since each proof string contains all per-party seeds, as soon as ϖ_1 is committed by $\mathbf{P}^*$, it is already determined whether the extractor $\mathbf{E}^\tau$ will succeed in extracting a correct witness from the execution path started with ϖ_1. Fix the path started with ϖ_1 that gets accepted, and let us denote $\mathbf{V}$'s messages/queries in the path by $h_1^{(e)} := (r_j^{(e)})_{j \in [m_1]}$, $h_2^{(e)} := R^{(e)}$, $h_3^{(e)} := \bar{i}^{(e)}$ for $e \in [\tau]$, respectively. From the analysis of [BDK+21, Lemma 2], if ϖ_1 contains no set of seeds leading to a valid witness, it must be that the adversary has correctly guessed one of $h_1^{(e)}, h_2^{(e)}, h_3^{(e)}$ for all parallel repetitions $e \in [\tau]$. Indeed, this probability is given as the upper bound for "$\Pr\left[\mathcal{A} \text{ wins} \mid \perp\right]$" in their proof, which is as stated in our lemma.

F.3   Validity of Banquet-NIVE

Theorem 4. *Suppose* ECOM *is* ϵ_{cext}*-extractable. Then* Banquet-NIVE *is* ϵ_{val}*-valid with validity error*

$$\epsilon_{\mathsf{val}} \leq \Pr[X + Y + Z = \tau] + \epsilon_{\mathsf{cext}} + \frac{2(Q_1 + Q_2 + Q_3)^2}{2^{2\kappa}}. \tag{4}$$

where Q_i *is the number of queries to* H_i *made by the adversary* $\mathcal{P}^*$*.*

Proof. Following proof for Theorem 1, given a cheating Banquet-NIVE prover $\mathcal{P}^*$ having access to the random oracles, we construct an adversary that either breaks extractability of ECOM, or breaks SR-SLE of Banquet-IOP. Concretely, we consider the following game hops.

- G_0 is identical to the non-interactive validity game.

- G_1 is identical to G_0, except that it aborts under the following condition: if an RO output freshly sampled inside H_1, H_2, H_3 collides with any of previously generated outputs *or* any of previously queried h_1, h_2 (indicated by the Bad set in [BDK+21, Lemma 2]). Similarity to Eq.(3) of their proof, we get

$$\Pr[G_1 \text{ aborts}] = (\# \text{ times an random oracle output is sampled}) \cdot \Pr[G_1 \text{ aborts at that sample}]$$

$$\leq (Q_1 + Q_2 + Q_3) \cdot \frac{\max |\mathsf{Bad}|}{2^{2\kappa}}$$

$$\leq (Q_1 + Q_2 + Q_3) \cdot \frac{Q_1 + 2Q_2 + 2Q_3}{2^{2\kappa}}$$

$$\leq \frac{2(Q_1 + Q_2 + Q_3)^2}{2^{2\kappa}}$$

where the maximum size of Bad is as above because for each query to H_1, its output h_1 is added to Bad; for each query to H_2, both input h_1 and output h_2 are added to Bad; for each query to H_3, both input h_2 and output h_3 are added to Bad.

- G_2 is identical to G_1, except that it aborts under the following condition: when the adversary $\mathcal{P}^*$ outputs an accepting tr, G_2 recomputes $\mathsf{C}_i^{(e)}$ for $i \neq \bar{i}^{(e)}$ as $\mathcal{V}$ would, and aborts if there exists some $e \in [\tau], i \neq \bar{i}^{(e)}$ such that $\mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i^{(e)}) \neq \mathsf{sd}_i^{(e)}$. Since this means that extractability of ECOM is broken, we have

$$\Pr[G_2 \text{ aborts}] \leq \epsilon_{\mathsf{cext}}$$

- Given a cheating prover $(\mathcal{A}^*, \mathcal{P}^*)$ that wins G_2, we construct a state-restoration adversary pair $(\mathbf{A}^*, \mathbf{P}^*)$ against SR-SLE of Banquet-IOP:

  1. $\mathbf{A}^*(1^\kappa)$ invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa)$ and forwards $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}^*$.

  2. On receiving x from $\mathcal{A}^*$, $\mathbf{A}^*$ forwards x to the SR-SLE game.

  3. $\mathbf{P}^*(x)$ runs $\mathcal{P}^*$ on input x, and simulates the RO responses as follows.

     - On receiving a query to H_1 with input $(x, \mathsf{pk}, \sigma_1)$ from $\mathcal{P}^*$, $\mathbf{P}^*$ parses $\sigma_1 := (\mathsf{salt}, ((\mathsf{C}_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \in [N]}, \Delta w^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]})_{e \in [\tau]})$ and decrypts per-party seeds by invoking $\hat{\mathsf{sd}}_i^{(e)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i^{(e)})$ for $i \in [N], e \in [\tau]$. Then $\mathbf{P}^*$ constructs a vector of proof strings $\varpi_1 := (\pi_1^{(e)})_{e \in [\tau]}$ where $\pi_1^{(e)} = ((\hat{\mathsf{sd}}_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \in [N]}, \Delta w^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]})$. It then outputs ϖ_1 in the SR-SLE game, receives verifier messages $h_1 := (r_j^{(e)})_{j \in [m_1], e \in [\tau]}$, programs the RO such that $H_1(x, \mathsf{pk}, \sigma) := h_1$, and returns h_1 to $\mathcal{P}^*$.

     - On receiving a query to H_2 with input $(x, \mathsf{pk}, h_1, \sigma_2)$, $\mathbf{P}^*$ sends a vector of proof strings $\varpi_2 := \sigma_2$ in the SR-SLE game, in order to resume the execution path containing a node labeled by h_1 (if there is no such a node then $\mathbf{P}^*$ just lazily samples h_2 and returns it to $\mathcal{P}^*$). It then receives verifier messages $h_2 := (R^{(e)})_{e \in [\tau]}$, programs the RO such that $H_2(x, \mathsf{pk}, h_1, \sigma_2) := h_2$, and returns h_2 to $\mathcal{P}^*$.

     - Likewise, on receiving a query to H_3 with input $(x, \mathsf{pk}, h_1, h_2, \sigma_3)$, $\mathbf{P}^*$ sends a vector of proof strings $\varpi_3 := \sigma_3$ in the SR-SLE game, in order to resume the execution path containing a node labeled by h_2. It then receives verifier queries $h_3 := (\bar{i}^{(e)})_{e \in [\tau]}$, programs the RO such that $H_3(x, \mathsf{pk}, h_1, h_2, \sigma_3) := h_3$, and returns h_3 to $\mathcal{P}^*$.

  4. When $\mathcal{P}^*$ outputs an accepting tr, $\mathbf{P}^*$ specifies the corresponding execution path constructed in the SR-SLE game.

Note that, $\mathbf{P}^*$ can uniquely determine which path to choose, because a winning adversary G_2 is guaranteed to query an random oracle H_i with input h_{i-1} *after* h_{i-1} was derived through H_{i-1} and there should be no collision between RO outputs (obtained from verifier messages/queries). Moreover, similar to validity proof for Theorem 1, since per-party seeds decrypted during the RO simulation are always consistent with ones opened in the accepting tr, it must be that the receiver $\mathcal{R}$ fails to recover a witness from tr iff the SR-SLE extractor $\mathbf{E}^\tau$ fails in extraction from the corresponding execution path. Therefore, we have

$$\Pr[(\mathcal{A}^*, \mathcal{P}^*) \text{ wins } G_2] \leq \epsilon_{\mathsf{sr\text{-}sle}}(\lambda, m_2, N, \tau) \leq \Pr[X + Y + Z = \tau]$$

where the last inequality is due to Lemma 9.

Putting the bounds together, we obtain the theorem statement.

$\square$

G   Compressing Ciphertexts: The Equality Proof Method

Recall that in a VE scheme created with our compiler, decryption iterates over the component ciphertexts (from each parallel repetition) until the reconstruction function recovers a witness. It is guaranteed that at least one of the component ciphertexts will cause decryption to succeed.

In an honestly generated proof, all component ciphertexts are valid, and decryption will always succeed on the first attempt. If after the VE protocol, the prover were able to additionally prove that $\mathcal{R}$ would output the same witness from all of the component ciphertexts, then the verifier could keep only one of the component ciphertexts, making the VE ciphertext constant size. This is because either: all values are equal and correct, or all values are equal and incorrect, but the latter case is equivalent to creating an invalid proof, which is possible with only negligible probability by soundness of the proof protocol.

Note that the equality proof proves that $\mathcal{R}$ outputs the same value for all component ciphertexts – *and is not requiring that we prove the relation*. The crux of $\mathcal{R}$ for MPCitH protocols is recombining additive shares of the witness, a comparatively simple operation. However one of the shares is encrypted, meaning we are back to proving something about encrypted data. We describe one instantiation of the idea to show that this is possible without resorting to general methods, by using PKE in a non-black-box way.

Theorem 5. *Let Π be an MPCitH-based IOP in the class given by Protocol 1 with $t = N-1$, for a relation R where $|w| = \lambda$. Then there exists a VE scheme Π' with a compression algorithm that produces $O(\lambda)$ ciphertexts for Π', assuming Paillier's encryption scheme is IND-CPA secure.*

Sketch. We describe the construction of the verifiable encryption scheme Π'. First we compile Π to a VE scheme using a slight variant of Protocol 2. Namely, we split the per-party commitments into two so that the share of the witness and other information in the view are committed separately. Thus we have an additional commitment public key pk' for a second extractable commitment scheme Commit' (which may be the same as Commit, or a more efficient hash-based scheme, extractable in the ROM, since extraction won't be required for decryption). Then $C_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ is instead computed as $C_i = (\mathsf{Commit}(\mathsf{pk}, w_i; r_i), \mathsf{Commit}'(\mathsf{pk}', v_i; s_i))$, where (wlog) each view is assumed to be $V_i = w_i || v_i$. Additionally, we assume that w is shared with XOR, so the shares are λ-bit strings.

Next, Π' is instantiated with extractable commitments constructed from the Paillier encryption scheme. Paillier is IND-CPA secure under the decisional composite residuosity assumption [Pai99], and encryption is perfectly correct, so it is a secure commitment

scheme by Lemma 1. Further, each bit of the witness share is encrypted separately which will allow bitwise operations using the homomorphic properties of Paillier encryption.

The new compression algorithm $\mathcal{C}'$ requires input from $\mathcal{P}$ (the equality proof) and $\mathcal{V}$ runs it to check the proof and keep the final ciphertext. The steps for $\mathcal{P}$ are:

1. Run the compression algorithm $\mathcal{C}$ from Protocol 2, to get the set $C = (\tilde{w}^{(j)}, \widehat{\mathsf{C}}^{(j)})_{j \in [\tau]}$, where $\widehat{\mathsf{C}}^{(j)}$ is the unopened commitment for the jth parallel execution. Since $t = N - 1$ there is only one commitment per parallel repetition.

2. Recall that $\widehat{\mathsf{C}}^{(j)} = \mathsf{Enc}(\mathsf{pk}, \hat{w}^{(j)})$ and $w = \hat{w}^{(j)} \oplus \tilde{w}^{(j)}$. Using the additive homomomorphic property of encryption, compute $C' = (\mathsf{Enc}(\hat{w}^{(1)} \oplus \tilde{w}^{(1)}), \dots, \mathsf{Enc}(\hat{w}^{(\tau)} \oplus \tilde{w}^{(\tau)}))$ as described in Appendix A.4. This is possible because $\tilde{w}^{(j)}$ are public constants, and there is only one unopened party, so we only need to compute the XOR of one public and one encrypted value.

3. Convert the set C' of bitwise encryptions of w, to the set C'' of integer encryptions of w as described in Appendix A.4. This is again possible using the homomorphic property, by computing $w = \sum_{i=0}^{\lambda} 2^i w_i$ (converting from binary to integer), and choosing paramters such that λ-bit strings fit in the plaintext space of Enc.

4. Prove all ciphertexts in C'' have the same plaintext. This step can be realized, e.g., with a standard generalization of Schnorr's proof of knowledge of a discrete logarithm (details in Appendix A.4). A non-interactive equality proof π is output by $\mathcal{P}$ and sent to $\mathcal{V}$.

The verifier $\mathcal{V}$ repeats Steps 1-3 to compute C'', then checks that π is valid. If so, $\mathcal{V}$ outputs the first ciphertext in C'' as the encryption of w.

Since the output compression is one ciphertext, the resulting VE ciphertext clearly has size $O(\lambda)$.

In terms of security, the protocol until Step 2 of $\mathcal{C}'$ is secure by Theorem 1, since the modifications to the commitment scheme maintain the required extractability and hiding properties. For the next part of $\mathcal{C}'$, we argue that the plaintext transforms in Steps 2 and 3 to compute C'' are 1:1 and thus maintain validity. Because Theorem 1 guarantees that the scheme is valid, decryption of C succeeds with overwhelming probability, which means that at least one component ciphertext is an encryption of w that is valid, in particular the plaintexts are guaranteed to be single bits. When a ciphertext in C is an encryption of individual bits, then steps 2 and 3 are reversible, implying that if ciphertext j in C is valid, then ciphertext j in C'' is also valid. Finally, as argued above since C'' contains at least one valid encryption of w, all ciphertexts must encrypt w assuming the equality proof in Step 4 is sound with overwhelming probability. The assumptions required for the proof in Step 4 can vary, but with an interactive version of Schnorr's proof we need only assume that discrete logarithms are hard in the Paillier group, which is implied by the DCR assumption required for security of Paillier encryption. □   □

The construction has drawbacks that keep it from being practical, and it would be interesting to address them. Because we require some (relatively weak) homomorphic properties, we lose the flexibility in the choice of PKE, and a suitable PQ-secure instantiation requires investigation. The cost of creating and communicating of the proof soars because we require bitwise encryptions of witness shares, meaning the prover must compute $O(\tau N \lambda)$ individual Paillier encryptions. In practice this cost could be significantly reduced by using bitwise exponential Elgamal, however then the final ciphertext would have to remain in the bitwise representation (to allow efficient decryption) meaning the ciphertext would have size $O(\lambda^2)$, rather than $O(\lambda)$.

H   Deferred Proofs

H.1   Proof of Theorem 1

Proof. We prove both properties separately.

HVZK follows from hiding of the commitment scheme and perfect HVZK of the base protocol $\mathsf{MPCitH\text{-}IOP}_R$. The $\mathsf{MPCitH\text{-}VE}_R$ simulator $\mathcal{S}$ on input (pk, x) proceeds as follows, for each parallel repetition: (1) invoke the $\mathsf{MPCitH\text{-}IOP}_R$ simulator $\mathbf{S}$ on input x to obtain $(\mathbf{e}, (V_i)_{i \in \mathbf{e}})$, (2) for each $i \in \mathbf{e}$, define $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, V_i; r_i)$ and for each $i \notin \mathbf{e}$, define $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, 0^{|V|}; r_i)$, where $0^{|V|}$ is the zero-string with the length of a view and the commitment randomness r_i's are sampled uniformly, (3) output $((\mathsf{C}_i)_{i \in [N]}, \mathbf{e}, (V_i)_{i \in \mathbf{e}})$.

The computational indistinguishability of the simulation follows from a standard hybrid argument. Since $\mathsf{MPCitH\text{-}IOP}$ is perfect HVZK, the $\tau \cdot t$ views output by $\mathbf{S}$ are distributed identically to views revealed in the real executions. As the $\mathsf{MPCitH\text{-}VE}$ simulator $\mathcal{S}$ has to generate $\tau(N - t)$ unopened commitments in total, we require $\tau(N - t)$ hybrids to replace these commitments to the real views with simulated ones. For each hop, the distinguisher is able to distinguish two consecutive hybrids with probability at most ϵ_{hide}. We thus obtain the bound $\tau(N - t)\epsilon_{\mathsf{hide}}$.

Validity At a high-level, the proof proceeds as follows: if an $\mathsf{MPCitH\text{-}VE}$ cheating prover $\mathcal{P}^*$ can convince the verifier $\mathcal{V}$ while the receiver fails to decrypt a correct witness, then it must be that either (1) $\mathcal{P}^*$ broke extractability of ECOM, or (2) one can construct a pair of adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ that break SLE of $\mathsf{MPCitH\text{-}IOP}_R$. Adversaries $(\mathbf{A}^*, \mathbf{P}^*)$ first extract views from the commitments sent by $\mathcal{P}^*$ and then forward them as a complete set of N views in the SLE-IOP game.

Now let us turn to the formal proof. We give proof for the single repetition case, but the argument below naturally extends to τ parallel repetitions.[14] We bound the probability that the receiver fails to decrypt:

$$\mathsf{fail} := \Pr\left[b = 1 \wedge (x, w') \notin R : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}); \\ (b, \mathsf{tr}) \leftarrow \langle \mathcal{P}^*(\mathsf{sk}), \mathcal{V} \rangle(\mathsf{pk}, x); \\ C \leftarrow \mathcal{C}(x, \mathsf{tr}); w' \leftarrow \mathcal{R}(\mathsf{sk}, C) \end{array} \right]$$

For brevity we omit the variable definitions after the colon in the following. From the description of $\mathsf{MPCitH\text{-}VE}_R$, the probability fail can be re-stated as follows.

$$\mathsf{fail} = \Pr\left[\begin{array}{c} \mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1 \\ \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{C}_i \\ \wedge (x, \sum_{i \in \mathbf{e}} w_i + \sum_{i \notin \mathbf{e}} \hat{w}_i) \notin R \end{array} \right] \tag{5}$$

$$= \Pr\left[\begin{array}{c} \mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1 \\ \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{C}_i \\ \wedge (x, \sum_{i \in \mathbf{e}} w_i + \sum_{i \notin \mathbf{e}} \hat{w}_i) \notin R \\ \wedge \exists i \in \mathbf{e} : V_i \neq \hat{V}_i \end{array} \right] \tag{6}$$

$$+ \Pr\left[\begin{array}{c} \mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1 \\ \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{C}_i \\ \wedge (x, \sum_{i \in \mathbf{e}} w_i + \sum_{i \notin \mathbf{e}} \hat{w}_i) \notin R \\ \wedge \forall i \in \mathbf{e} : V_i = \hat{V}_i \end{array} \right] \tag{7}$$

---

[14]In fact, it is also implied by Theorem 2 in the next section by setting $n = \tau$.

where for $i \in [N]$, $\hat{V}_i = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i)$ are the views obtained by decrypting C_i; for $i \in \mathbf{e}$, $w_i = \mathsf{GetW}(V_i)$; for $i \notin \mathbf{e}$, $\hat{w}_i = \mathsf{GetW}(\hat{V}_i)$, according to the operations of $\mathcal{C}$ and $\mathcal{R}$.

If event (6) happens, then the extractability of ECOM is broken. That is, one can construct an adversary $\mathcal{A}$ that receives $(\mathsf{pk}, \mathsf{sk})$ in the extractability game, runs the validity adversary on that key pair, and if $\mathcal{P}^*$ outputs some C_i, V_i, r_i such that $\mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{C}_i \wedge \hat{V}_i \neq V_i$, then $\mathcal{A}$ wins with output (C_i, V_i, r_i). Hence (6) is bounded by ϵ_{cext}.

We now give an upper bound for (7). First, it can be bounded as follows

$$(7) = \Pr \begin{bmatrix} \mathsf{CheckView}(x, (\hat{V}_i)_{i \in \mathbf{e}}) = 1 \\ \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, \hat{V}_i; r_i) = \mathsf{C}_i \\ \wedge (x, \sum_{i \in \mathbf{e}} \hat{w}_i + \sum_{i \notin \mathbf{e}} \hat{w}_i) \notin R \\ \wedge \forall i \in \mathbf{e} : V_i = \hat{V}_i \end{bmatrix} \tag{8}$$

$$\leq \Pr \left[\mathsf{CheckView}(x, (\hat{V}_i)_{i \in \mathbf{e}}) = 1 \wedge (x, \sum_{i \in [N]} \hat{w}_i) \notin R \right] \tag{9}$$

because we are guaranteed that $\hat{V}_i = V_i$ and thus $\hat{w}_i = w_i$ for $i \in \mathbf{e}$. Given a cheating prover $(\mathcal{A}^*, \mathcal{P}^*)$ that causes event (9) to occur, we construct an adversary pair $(\mathbf{A}^*, \mathbf{P}^*)$ against SLE of $\mathsf{MPCitH\text{-}IOP}_R$.

1. $\mathbf{A}^*(1^\kappa)$ invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\kappa)$ and forwards $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}^*$.

2. On receiving x from $\mathcal{A}^*$, $\mathbf{A}^*$ forwards x to the IOP-SLE game.

3. $\mathbf{P}^*(x)$ runs $\mathcal{P}^*$ on input x and plays an $\mathsf{MPCitH\text{-}VE}$ verifier $\mathcal{V}$.

4. On receiving $(\mathsf{C}_i)_{i \in [N]}$ from $\mathcal{P}^*$, $\mathbf{P}^*$ extracts the views by invoking $\hat{V}_i = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i)$ for $i \in [N]$. Then $\mathbf{P}^*$ constructs the proof string $\pi = (\hat{V}_i)_{i \in [N]}$ and outputs in the SLE-IOP game.

5. On receiving an oracle query $\mathbf{e}$ from $\mathbf{V}$ in the SLE-IOP game, $\mathbf{P}^*$ forwards $\mathbf{e}$ to $\mathcal{P}^*$.

Since the canonical extractor $\mathbf{E}$ for $\mathsf{MPCitH\text{-}IOP}_R$ (see Definition 2) also constructs a witness candidate via $w = \sum_{i \in [N]} \mathsf{GetW}(\hat{V}_i) = \sum_{i \in [N]} \hat{w}_i$, it must be that $\mathbf{E}$ fails to extract a valid witness if and only if the receiver $\mathcal{R}$ fails to decrypt correctly, i.e.,

$$(9) = \Pr \left[\mathsf{CheckView}(x, (\hat{V}_i)_{i \in \mathbf{e}}) = 1 \wedge (x, \mathbf{E}(x, (\hat{V}_i)_{i \in [N]})) \notin R \right] \leq \epsilon_{\mathsf{sle\text{-}iop}} \tag{10}$$

Overall, we have that $\mathsf{fail} \leq \epsilon_{\mathsf{cext}} + \epsilon_{\mathsf{sle\text{-}iop}}$.

$\square$

H.2 Proof of Theorem 2

Proof. We extend the proof for Theorem 1. On a high-level the analysis amounts to evaluating the probability that the SLE-IOP extractor succeeds in extracting a valid witness from random subset of in total τ transcripts, while all τ repetitions are accepting. From the description of $\mathsf{MPCitH\text{-}VE}_R$, the probability fail can be parsed as follows.

$$\text{fail} \tag{11}$$

$$= \Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge \forall j \in S : (x, \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}) \notin R \end{array} \right]$$

$$= \Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge S \subset \mathsf{BadRun} \end{array} \right]$$

$$= \sum_{s=n}^{\tau} \Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \wedge S \subset \mathsf{BadRun} \end{array} \right]$$

$$= \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \end{array} \right] \tag{12}$$

where $\mathsf{BadRun} := \left\{ j \in [\tau] : (x, \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}) \notin R \right\}$, i.e., a set of "bad" parallel repetitions from which one fails to decrypt a valid witness. Note that the last equality holds since a random subset S is sampled independently of BadRun and thus the probability that a subset S is chosen from BadRun of size s is $\frac{\binom{s}{n}}{\binom{\tau}{n}}$.

Now we split (12) into the two cases, depending on whether extractability of ECOM is broken or not.

$$(12)$$

$$= \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \\ \wedge \exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)} \end{array} \right] \right.$$

$$\left. + \Pr \left[\begin{array}{c} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \\ \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : V_i^{(j)} = \hat{V}_i^{(j)} \end{array} \right] \right)$$

where for $j \in \mathsf{BadRun}$ and $i \in [N]$, $\hat{V}_i^{(j)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{C}_i^{(j)})$ are the views obtained by decrypting $\mathsf{C}_i^{(j)}$.

Following the proof for Theorem 1, the former case can be bounded as follows.

$$\Pr\begin{bmatrix} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \\ \wedge \exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)} \end{bmatrix}$$

$$\leq \Pr\left[\exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)} \;\middle|\; |\mathsf{BadRun}| = s\right] \tag{13}$$

$$\cdot \Pr\left[|\mathsf{BadRun}| = s\right]$$

$$\leq \epsilon_{\mathsf{cext}} \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{14}$$

Likewise, the latter case can be bounded as follows.

$$\Pr\begin{bmatrix} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{C}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \\ \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : V_i^{(j)} = \hat{V}_i^{(j)} \end{bmatrix}$$

$$\leq \Pr\begin{bmatrix} \forall j \in \mathsf{BadRun} : \mathsf{CheckView}(x, (\hat{V}_i^{(j)})_{i \in \mathbf{e}}) = 1 \\ \wedge (x, \sum_{i \in [N]} \hat{w}_i^{(j)}) \notin R \end{bmatrix} \;\middle|\; |\mathsf{BadRun}| = s \end{bmatrix} \tag{15}$$

$$\cdot \Pr\left[|\mathsf{BadRun}| = s\right]$$

$$\leq \epsilon_{\mathsf{sle\text{-}iop}}(s) \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{16}$$

because given a cheating prover $\mathcal{P}^*$ committing to views $\hat{V}_i^{(j)}$ that do not decode to a valid witness, one can construct an adversary pair $(\mathbf{A}^*, \mathbf{P}^*)$ against SLE of $\mathsf{MPCitH\text{-}IOP}_R$ as in the proof for Theorem 1. However, notice that $\mathbf{P}^*$ here only forwards as an oracle $(\hat{V}_i^{(j)})_{i \in [N]}$ for $j \in \mathsf{BadRun}$ to the SLE-IOP game, instead of the extracted views for all τ executions. Therefore, the probability upper bound is parameterized by the advantage of SLE-IOP prover that only runs s parallel repetitions.

Putting (12), (14) and (16) together, we obtain the desired bound.

$\square$

Protocol 4: DKGitH for relation $R = \{(y, x) \ : \ y = g^x\}$

Parameters Let $\mathbb{G}$ be a group of prime order p generated by g, written multiplicatively. Let τ be a parameter for the number of parallel repetitions. N denotes the number of parties. Let $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ be an extractable commitment scheme.

Key Generation $\mathcal{G}(1^\kappa)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^\kappa)$

Prover $\mathcal{P}(\mathsf{pk}, x, y)$:

 Commit

  1: Sample a random salt $\mathsf{salt} \leftarrow\!\!\$ \ \{0,1\}^{2\lambda}$.
  2: **for** each parallel repetition j **do**
  3:   Sample a root seed: $\mathsf{sd}^{(j)} \leftarrow\!\!\$ \ \{0,1\}^{\lambda}$.
  4:   Compute parties' seeds $\mathsf{sd}_1^{(j)}, \ldots, \mathsf{sd}_N^{(j)}$ as leaves of a binary tree with root $\mathsf{sd}^{(j)}$.
  5:   **for** each party i **do**
  6:    Expand seed to witness share tape: $x_i^{(j)} \leftarrow \mathsf{Expand}(\mathsf{salt}, j, i, \mathsf{sd}_i^{(j)})$
  7:    Commit to share: $\mathsf{C}_i^{(j)} \leftarrow \mathsf{Commit}(\mathsf{pk}, \mathsf{salt}, j, i, x_i^{(j)})$.
  8:   **end for**
  9:   Compute witness offset: $\Delta x^{(j)} \leftarrow x - \sum_i x_i^{(j)}$.
  10:   Correct first share: $x_1^{(j)} \leftarrow x_1^{(j)} + \Delta x^{(j)}$.
  11:   **for** each party i **do**
  12:    Compute and broadcast $y_i^{(j)} = g^{x_i^{(j)}}$
  13:   **end for**
  14: **end for**
  15: Set $\pi_1 \leftarrow (\mathsf{salt}, ((\mathsf{C}_i^{(j)}, y_i^{(j)})_{i \in [N]}, \Delta x^{(j)})_{j \in [\tau]})$.

 Challenge

  1: Let $h = (\bar{i}_1, \ldots, \bar{i}_\tau) = H(\pi_1, y, \mathsf{pk})$, where $\bar{i}_j \in [N]$

 Response and output

  1: **for** each parallel repetition j **do**
  2:   $\mathsf{sds}^{(j)} \leftarrow \{\log_2(N) \text{ nodes needed to compute } \mathsf{sd}_i^{(j)} \text{ for } i \in [N] \setminus \{\bar{i}_j\}\}$.
  3: **end for**
  4: Output $\mathsf{tr} \leftarrow (\mathsf{salt}, h, (\mathsf{sds}^{(j)}, \mathsf{C}_{\bar{i}_j}^{(j)}, \Delta x^{(j)})_{j \in [\tau]})$.

Verifier $\mathcal{V}(\mathsf{pk}, y, \mathsf{tr})$:

 1: Parse tr as $(\mathsf{salt}, h, (\mathsf{sds}^{(j)}, \mathsf{C}_{\bar{i}_j}^{(j)}, \Delta x^{(j)})_{j \in [\tau]})$ and h as $(\bar{i}_1, \ldots, \bar{i}_\tau)$
 2: **for** each parallel repetition j **do**
 3:   Use $\mathsf{sds}^{(j)}$ to compute $\mathsf{sd}_i^{(j)}$ for $i \in [N] \setminus \{\bar{i}_j\}$.
 4:   **for** each party $i \in [N] \setminus \{\bar{i}_j\}$ **do**
 5:    Recompute $x_i^{(j)} \leftarrow \mathsf{Expand}(\mathsf{salt}, j, i, \mathsf{sd}_i^{(j)})$, $\mathsf{C}_i^{(j)} \leftarrow \mathsf{Commit}(\mathsf{pk}, \mathsf{salt}, j, i, x_i^{(j)})$ and $y_i^{(j)} = g^{x_i^{(j)}}$.
 6:    **if** $i \overset{?}{=} 1$ **then**
 7:     Correct first share: $x_i^{(j)} \leftarrow x_i^{(j)} + \Delta x^{(j)}$.
 8:    **end if**
 9:   **end for**
 10:   Compute $y_{\bar{i}_j}^{(j)} = y/(\prod_{i \neq \bar{i}_j} y_i^{(j)})$
 11: **end for**
 12: Set $\pi_1 \leftarrow (\mathsf{salt}, ((\mathsf{C}_i^{(j)}, y_i^{(j)})_{i \in [N]}, \Delta x^{(j)})_{j \in [\tau]})$.
 13: Accept if $h \overset{?}{=} H(\pi_1, y, \mathsf{pk})$, otherwise reject.

Protocol 5: RDKGitH for relation $R = \left\{(y, x) \ : \ y = g^x\right\}$

Parameters Let t be the degree of polynomial. $\mathbb{G}, p, g, N, \mathsf{ECOM}$ are the same as ones in DKGitH.

Key Generation $\mathcal{G}(1^\kappa)$ outputs a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^\kappa)$

Prover $\mathcal{P}(\mathsf{pk}, x, y)$:

    Commit
        1: **for** each $j = 1, \ldots, t$ **do**
        2:     $a_j \leftarrow\!\!\$\ \mathbb{Z}_p^*; A_j \leftarrow g^{a_j}$
        3: **end for**
        4: $a_0 \leftarrow x; A_0 \leftarrow y$
        5: $a(X) \leftarrow \sum_{j=0}^{t} a_j X^j$
        6: **for** each party $i \in [N]$ **do**
        7:     Compute a witness share: $x_i \leftarrow a(i)$
        8:     Commit to share: $\mathsf{C}_i \leftarrow \mathsf{Commit}(\mathsf{pk}, x_i; r_i)$.
        9: **end for**
      10: Set $\pi_1 \leftarrow ((\mathsf{C}_i)_{i \in [N]}, A_1, \ldots, A_t)$.

    Challenge
        1: Let $I = (i_1, \ldots, i_t) = H(\pi_1, y, \mathsf{pk})$, where $i_j \in [N]$

    Response and output
        1: Output $\mathsf{tr} \leftarrow (I, (x_i, r_i)_{i \in I}, (\mathsf{C}_i)_{i \notin I}, A_1, \ldots, A_t)$.

Verifier $\mathcal{V}(\mathsf{pk}, y, \mathsf{tr})$:

    1: Parse tr as $(I, (x_i, r_i)_{i \in I}, (\mathsf{C}_i)_{i \notin I}, A_1, \ldots, A_t)$ and I as $(i_1, \ldots, i_\tau)$
    2: $A_0 \leftarrow y$
    3: **for** each party $i \in I$ **do**
    4:     Recompute $\mathsf{C}_i = \mathsf{Commit}(\mathsf{pk}, x_i; r_i)$.
    5:     Check $g^{x_i} =_? \prod_{j=0}^{t} A_j^{i^j}$, otherwise reject
    6: **end for**
    7: Set $\pi_1 \leftarrow ((\mathsf{C}_i)_{i \in [N]}, A_1, \ldots, A_t)$.
    8: Accept if $I \stackrel{?}{=} H(\pi_1, y, \mathsf{pk})$, otherwise reject.

Protocol 6: KKW-IOP$_R$

Parameters: The number of parties N; the number of MPC executions M; the number of revealed online phases τ.

Inputs: prover **P** receives (x, w); verifier **V** receives x.

Committing phase The first-round message of **V** is empty. For each $j \in [M]$, **P** proceeds as follows.

1. Choose uniform $\mathsf{sd}^{(j)}$ and use it to generate per-party seeds $(\mathsf{sd}_i^{(j)})_{i \in [N]}$. **P** computes $\mathsf{aux}^{(j)}$ by running the offline phase of MPC on input $(\mathsf{sd}_i^{(j)})_{i \in [N]}$. For each $i \in [1, N-1]$, let $\mathsf{st}_i^{(j)} = \mathsf{sd}_i^{(j)}$ and let $\mathsf{st}_N^{(j)} = \mathsf{sd}_N^{(j)} \| \mathsf{aux}^{(j)}$.

2. Compute the masked witness $\hat{w} = \lambda_1^w \oplus \ldots \oplus \lambda_N^w \oplus w$, where λ_i^w is party i's random share to mask the witness in jth MPC execution, and is read out from $\mathsf{st}_i^{(j)}$.

3. Emulate "in her head" the online phase of the N-party protocol for $f(w) =_? x$ by running the online phase of MPC on input x, $\hat{w}^{(j)}$ and $(\mathsf{st}_i^{(j)})_{i \in [N]}$. As a result the prover obtains per-party broadcast messages $(\mathsf{msgs}_i^{(j)})_{i \in [N]}$.

Finally, **P** outputs the proof string $\pi = (\mathsf{sd}^{(j)}, (\mathsf{st}_i^{(j)}, \mathsf{msgs}_i^{(j)})_{i \in [N]}, \hat{w}^{(j)})_{j \in [M]}$.

Query phase

1. **V** chooses a uniformly random subset $T \subset [M]$ of size τ and party indices $(\bar{i}_j)_{j \in T}$ where each $\bar{i}_j \in [N]$ is uniform. It queries the oracle for π with T and $(\bar{i}_j)_{j \in T}$.

2. The oracle returns $\mathsf{sd}^{(j)}$ and $(\mathsf{st}_i^{(j)})_{i \in [N]}$ for $j \notin T$ and $(\mathsf{st}_i^{(j)})_{i \neq \bar{i}_j}$, $(\mathsf{msgs}_i^{(j)})_{i \in [N]}$ and $\hat{w}^{(j)}$ for $j \in T$.

Decision phase:

1. For each $j \notin T$, **V** emulates the offline phase using $\mathsf{sd}^{(j)}$ to compute $(\tilde{\mathsf{st}}_i^{(j)})_{i \in [N]}$ as an honest prover would.

2. For each $j \in T$, **V** emulates the online phase using $(\mathsf{st}_i^{(j)})_{i \neq \bar{i}_j}$, masked witness $\hat{w}^{(j)}$ and $\mathsf{msgs}_{\bar{i}_j}^{(j)}$ to compute $(\tilde{\mathsf{msgs}}_i^{(j)})_{i \neq \bar{i}_j}$ and output bit $b^{(j)}$.

3. Accept iff
   - For $j \notin T$, the offline phases are computed correctly, i.e., $(\tilde{\mathsf{st}}_i^{(j)})_{i \in [N]} = (\mathsf{st}_i^{(j)})_{i \in [N]}$.
   - For $j \in T$, the online phases are computed correctly, i.e., $(\tilde{\mathsf{msgs}}_i^{(j)})_{i \neq \bar{i}_j} = (\mathsf{msgs}_i^{(j)})_{i \neq \bar{i}_j}$ and $b^{(j)} = 1$.

Protocol 8: Banquet-NIVE

Converts the Banquet-IOP prover $\mathbf{P}$ and verifier $\mathbf{V}$ to non-interactive VE prover $\mathcal{P}$ and verifier $\mathcal{V}$ using the the extractable commitment scheme $\mathsf{ECOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as constructed in Section 3.1, and the random oracles $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$. The protocol below closely follows the Banquet signature; differences are highlighted in orange.

Key Generation $\mathcal{G}(1^\kappa)$: It invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{CGen}(1^\kappa)$ and outputs $(\mathsf{pk}, \mathsf{sk})$.

Prover $\mathcal{P}(\mathsf{pk}, x, w)$ where $x = (\mathsf{ct}, \mathsf{pt})$ and $w = K$

    1: $\mathsf{salt} \leftarrow\!\!\$ \ \{0,1\}^\kappa$

    2: For $e \in [\tau]$: run $\mathbf{P}$ to get $\pi_1^{(e)} = ((\mathsf{sd}_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \in [N]}, \Delta w^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]})$

    3: For $e \in [\tau], i \in [N]$: $\rho_i^{(e)} \leftarrow\!\!\$ \ S_r$; $\mathsf{C}_i^{(e)} := \mathsf{Commit}(\mathsf{pk}, \mathsf{sd}_i^{(e)}; \rho_i^{(e)})$

    4: $\sigma_1 := (\mathsf{salt}, ((\mathsf{C}_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \in [N]}, \Delta w^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]})_{e \in [\tau]})$

    5: $(r_j^{(e)})_{j \in [m_1], e \in [\tau]} := h_1 = \mathsf{H}_1(\mathsf{pk}, x, \sigma_1)$

    6: For $e \in [\tau]$: run $\mathbf{P}$ to get $\pi_2^{(e)} = (\Delta P^{(e)}(k))_{k \in [m_2, 2m_2]}$; $\sigma_2 := (\pi_2^{(e)})_{e \in [\tau]}$.

    7: $(R^{(e)})_{e \in [\tau]} := h_2 = \mathsf{H}_2(\mathsf{pk}, x, h_1, \sigma_2)$

    8: For $e \in [\tau]$: run $\mathbf{P}$ to get $\pi_3^{(e)} = ((a_{i,j}^{(e)}, b_{i,j}^{(e)})_{j \in [m_1]}, c_i^{(e)})_{i \in [N]}$; $\sigma_3 := (\pi_3^{(e)})_{e \in [\tau]}$

    9: For $e \in [\tau], j \in [m_1]$: $a_j^{(e)} := \sum_i a_{i,j}^{(e)}$; $b_j^{(e)} := \sum_i b_{i,j}^{(e)}$

   10: For $e \in [\tau]$: $c^{(e)} := \sum_i c_i^{(e)}$

   11: $(\bar{i}^{(e)})_{e \in [\tau]} := h_3 = \mathsf{H}_3(\mathsf{pk}, x, h_1, h_2, \sigma_3)$

   12: For $e \in [\tau]$: $\mathsf{sds}^{(e)} := \{\log_2(N)$ nodes needed to compute $\mathsf{sd}_i^{(e)}$ for $i \in [N] \setminus \{\bar{i}^{(e)}\}\}$

   13: $\mathsf{tr} := (\mathsf{salt}, h_1, h_3, (\mathsf{sds}^{(e)}, (\rho_i^{(e)})_{i \neq \bar{i}^{(e)}}, \mathsf{C}_{\bar{i}^{(e)}}^{(e)}, \Delta x^{(e)}, (\Delta t_\ell^{(e)})_{\ell \in [m]}, \Delta P^{(e)}(k)_{k \in [m_2, 2m_2]},$
        $(a_j^{(e)}, b_j^{(e)})_{j \in [m_1]}, c^{(e)})_{e \in [\tau]})$.

Verifier $\mathcal{V}(\mathsf{pk}, x, \mathsf{tr})$

    1: For $e \in [\tau]; i \neq \bar{i}^{(e)}$: $\mathsf{C}_i^{(e)} := \mathsf{Commit}(\mathsf{pk}, \mathsf{sd}_i^{(e)}; \rho_i^{(e)})$

    2: For $e \in [\tau], i \neq \bar{i}^{(e)}$: recompute $((a_{i,j}^{(e)}, b_{i,j}^{(e)})_{j \in [m_1]}, c_i^{(e)}, \mathsf{ct}_i^{(e)})_{i \neq \bar{i}^{(e)}}$ and $\mathsf{ct}_i^{(e)}$ as $\mathbf{V}$ would.

    3: For $e \in [\tau]$ and $j \in [m_1]$: $a_{\bar{i}^{(e)}, j}^{(e)} := a_j^{(e)} - \sum_{i \neq \bar{i}^{(e)}} a_{i,j}^{(e)}$; $b_{\bar{i}^{(e)}, j}^{(e)} := b_j^{(e)} - \sum_{i \neq \bar{i}^{(e)}} b_{i,j}^{(e)}$

    4: For $e \in [\tau]$: $c_{\bar{i}^{(e)}}^{(e)} := c^{(e)} - \sum_{i \neq \bar{i}^{(e)}} c_i^{(e)}$

    5: Define σ_1 and σ_3 as $\mathcal{P}$ would

    6: $h_1' := \mathsf{H}_1(\mathsf{pk}, x, \sigma_1)$; $h_3' := \mathsf{H}_3(\mathsf{pk}, x, h_1', h_2, \sigma_3)$

    7: **return** 1 iff $h_1' =_? h_1$, $h_3' =_? h_3$, and for all $e \in [\tau]$ $\mathbf{V}$ accepts.

Compression $\mathcal{C}(x, \mathsf{tr})$:

    1: For $e \in [\tau], i \neq \bar{i}^{(e)}$: $w_i^{(e)} := \mathsf{GetW}(\mathsf{sd}_i^{(e)})$ and $\tilde{w}^{(e)} := \sum_{i \neq \bar{i}^{(e)}} w_i^{(e)} + \Delta w^{(e)}$.

    2: **return** $C := (\tilde{w}^{(e)}, (\mathsf{C}_{\bar{i}^{(e)}}^{(e)}))_{e \in [\tau]}$.

Receiver $\mathcal{R}(\mathsf{sk}, C)$:

    1: For $e \in [\tau]$: $w_{\bar{i}^{(e)}}^{(e)} := \mathsf{GetW}(\mathsf{CExt}(\mathsf{sk}, \mathsf{C}_{\bar{i}^{(e)}}^{(e)})))$ and $w^{(e)} := \tilde{w}^{(e)} + w_{\bar{i}^{(e)}}^{(e)}$.

    2: **return** $w^{(e)}$ if $\exists e : R(x, w^{(e)}) =_? 1$