





# Verifiable Encryption from MPC-in-the-Head

Akira Takahashi<sup>1</sup>   and Greg Zaverucha<sup>2</sup>

<sup>1</sup> J.P.Morgan AI Research & AlgoCRYPT CoE, New York, USA

<sup>2</sup> 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.

## 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  $\text{pk}$  (of which the corresponding decryption key  $\text{sk}$  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  $\text{sk}$  to obtain a plaintext  $w$  satisfying  $R(x, w) = 1$ .
- **Zero knowledge:** As  $\mathcal{V}$  does not have the decryption key  $\text{sk}$ , 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

E-mail: [takahashi.akira.58s@gmail.com](mailto:takahashi.akira.58s@gmail.com) (Akira Takahashi), [gregz@microsoft.com](mailto:gregz@microsoft.com) (Greg Zaverucha)



(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. We briefly review some closely related work here, and defer a more complete review to the full version [TZ21]. 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<sup>+</sup>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  $R' = \{(\text{ct}, x), (\text{pt}, r) : R(x, \text{pt}) = 1 \wedge \text{ct} = \text{Enc}(\text{pt}; r)\}$ . In general, the main technical challenge in constructing VE is to minimize the cost of proof-of-plaintext-knowledge (PoPK) “ $\text{ct} = \text{Enc}(\text{pt}; r)$ ” 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  $\Sigma$ -protocol exists, or otherwise (2) proving correct evaluation of the circuit  $\text{Enc}$ , which would be costly depending on the encryption scheme (e.g., Kyber-KEM standardized by NIST [SAB<sup>+</sup>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<sup>+</sup>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 revokable 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<sup>+</sup>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<sup>+</sup>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  $\tau$  is large. The second approach, called the *equality proof method*, is optimal as it achieves constant size ciphertexts,  $O(|w|)$ .

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 [TZ21] 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  $\Sigma$ -protocols. To verifiably encrypt AES keys, we apply our transform to the Banquet [BDK<sup>+</sup>21] and Helium [KZ22] ZK proofs, where  $\mathcal{P}$  proves knowledge of an AES key used to generate a public plaintext/ciphertext pair. 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.<sup>1</sup> 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<sup>+</sup>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<sup>+</sup>22, Table 1].

<sup>1</sup>Our implementations are available at <https://github.com/akiratk0355/verenc-mpcith>

**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 how their construction can be securely instantiated with statistically correct PKE schemes. Due to space constraints this material is in [TZ21].

**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<sup>+</sup>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<sup>+</sup>21], Limbo [DOT21], Liger [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<sup>+</sup>20] or FrodoKEM [NAB<sup>+</sup>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 [TZ21] 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 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:  $C_i \leftarrow \text{Enc}(\text{pk}, w_i)$  for  $i = 1, 2, 3$  (and the remaining  $\text{view}_i$  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  $\text{sk}$  can decrypt the unopened ciphertext  $C_{\bar{i}}$  (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 [TZ21].

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.

## 2 Preliminaries

First we introduce some notation and conventions used throughout the paper. The security parameter is denoted  $\lambda$ , and for an integer  $x$ ,  $[x]$  is short for the set  $\{1, \dots, x\}$ . 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 [TZ21], 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 [TZ21]. 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  $\mathbf{VE}_R$  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  $(\mathbf{pk}, \mathbf{sk})$ .
- $(\mathcal{P}, \mathcal{V})$ : A two-party protocol, where both  $\mathcal{P}$  and  $\mathcal{V}$  take  $(x, \mathbf{pk})$  and  $\mathcal{P}$  additionally takes a plaintext  $w$  as inputs. We let  $(b, \mathbf{tr}) \leftarrow \langle \mathcal{P}(w), \mathcal{V} \rangle(\mathbf{pk}, x)$  denote the output pair of  $\mathcal{V}$  on common input  $(\mathbf{pk}, x)$  when interacting with  $\mathcal{P}(w)$ , where  $b \in \{0, 1\}$  indicates whether  $\mathcal{V}$  accepts or rejects, and  $\mathbf{tr}$  denotes a transcript exchanged between  $\mathcal{P}$  and  $\mathcal{V}$ .
- $\mathcal{C}(x, \mathbf{tr})$ : A compression algorithm that outputs a compressed ciphertext  $C$ .

- $\mathcal{R}(\text{sk}, C)$ : A receiver (or recovery) algorithm that outputs a plaintext  $w$ .

VE is secure if it satisfies the following three properties.

**Completeness**  $\text{VE}_R$  is  $\epsilon_{\text{comp}}$ -complete if for all  $(x, w) \in R$ .

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

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

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

**Computational Honest Verifier Zero-knowledge**  $\text{VE}_R$  is  $\epsilon_{\text{zk}}$ -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,

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

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  $w' = \mathcal{R}(\text{sk}, C)$  with all possible  $\text{sk}$ , checking whether  $(x, w') \in R$ .

## 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 [TZ21] 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], Liger, 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 [TZ21] 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<sup>+</sup>20, CFF<sup>+</sup>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<sub>R</sub>,  $\mathbf{P}$  proves knowledge of a witness  $w$  such that  $R(x, w) = 1$ , where  $\Pi_f$  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

**Protocol 1:** MPCitH-IOP<sub>R</sub>

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

**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  $w_1, \dots, w_N$  such that  $w = \sum_{i=1}^N w_i$ .
2. Emulate “in her head” the execution of  $\Pi_f$  on input  $(x, w_1, \dots, w_N)$ .
3. Prepare, based on the execution, the share of the witness, and the randomness, the views  $V_1, \dots, V_N$  of the  $N$  parties;  $\mathbf{P}$  outputs the proof string  $\pi = (V_1, \dots, V_N)$ .

**Query phase**

1.  $\mathbf{V}$  chooses a random  $\mathbf{e} \in \text{Ch}$  and queries the oracle for  $\pi$  with  $\mathbf{e}$ .
2. The oracle returns  $(V_i)_{i \in \mathbf{e}}$ .

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

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

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 Ligerio [BFH<sup>+</sup>20] and Limbo [DOT21]. As we shall see below, as an IOP protocol it is straightforward to prove straight-line extractability of MPCitH-IOP<sub>R</sub>. 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  $\tau$  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  $\text{CheckView}$  in MPCitH-IOP<sub>R</sub> 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  $\Pi_f$ ,

and returns 0 otherwise. We further define a utility function  $\text{GetW}$ , 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 MPCitH-IOP<sub>R</sub> is called *canonical* if on input  $x$  and  $\pi = (V_1, \dots, V_N)$ , it works as follows:  $\mathbf{E}$  obtains witness shares via  $w_i = \text{GetW}(V_i)$  for  $i \in [N]$  and then outputs a candidate witness  $w := \sum_{i \in [N]} w_i$ . For  $\tau$  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  $\perp$ .

It is rather straightforward to check that the protocol MPCitH-IOP<sub>R</sub> is (1) straight-line extractable with respect to the canonical knowledge extractor  $\mathbf{E}^\tau$  with  $\epsilon_{\text{sle-iop}} \leq ((k-1)/|\text{Ch}|)^\tau$  assuming the notion called  $k$ -consistency, and (2) HVZK if the underlying MPC protocol is  $t$ -private. For completeness, [TZ21] 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 **MPCitH-VE** from an MPCitH-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  $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$  be a public key encryption scheme. We construct an extractable commitment scheme  $\text{ECOM} = (\text{CGen}, \text{Commit}, \text{CExt})$  (see [TZ21] 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.

- $\text{CGen}(1^\lambda)$  runs  $\text{PKE.Gen}(1^\lambda)$  and outputs  $pk$  as the commitment key.
- $\text{Commit}(pk, m; r)$  outputs  $c = \text{PKE.Enc}(pk, m; r)$ .
- The opening of the commitment  $c$  is  $(m, r)$ , and the verifier checks  $(m, r)$  against  $c$  by computing  $c' = \text{Enc}(pk, m, r)$ ; the opening is accepted iff  $c' = c$ ,  $m \in S_m$  and  $r \in S_r$ .
- $\text{CExt}(sk, c)$  outputs  $m = \text{PKE.Dec}(sk, c)$ .

It is rather straightforward to show the above construction is perfectly extractable, perfectly binding and computationally hiding, assuming  $\text{PKE}$  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  $c = \text{Enc}(pk, m; r)$ . 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  $\text{Dec}(sk, (\text{Enc}(pk, m^*; r^*))) \neq m^*$ . 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.

In [TZ21], we will show two examples of such schemes, one based on decisional composite residuosity, and one based on the learning with errors (LWE) problem. In general, the base encryption scheme of post-quantum lattice-based candidates like FrodoKEM [NAB<sup>+</sup>19] and Kyber [SAB<sup>+</sup>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  $r$ .

**Definition 3** (Undeniability). We say that a public-key encryption scheme  $\text{PKE} =$

$(\text{Gen}, \text{Enc}, \text{Dec})$  is  $\epsilon_{\text{und}}$ -undeniable if for any PPT adversary  $\mathcal{A}$ :

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

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

*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  $c = \text{Enc}(\text{pk}, m; r)$  and  $m \neq \text{Dec}(\text{sk}, c)$ . 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.,  $c = \text{Enc}(\text{pk}, m; r) = \text{Enc}(\text{pk}, m'; r')$  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  $(\text{pk}, \text{sk})$  as input,  $\mathcal{B}$  forwards it to  $\mathcal{A}$ .
2. When  $\mathcal{A}$  outputs  $(c, m, r, m', r')$  such that  $c = \text{Enc}(\text{pk}, m; r) = \text{Enc}(\text{pk}, m'; r')$  and  $m \neq m'$ , the  $\mathcal{B}$  first decrypts  $c$ :  $\tilde{m} = \text{Dec}(\text{sk}, c)$  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. That is, if there exists a PPT distinguisher for commitment  $c = \text{Commit}(\text{pk}, m_b; r) = \text{Enc}(\text{pk}, m_b; r)$  one can construct a reduction algorithm that maps a challenge ciphertext of the IND-CPA game to  $c$ .  $\square$

**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 [TZ21], 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  $\text{Enc}$  that takes a public key, message, and random value as input, and a random oracle  $\mathbf{G}$  that hashes into the randomness space of  $\text{Enc}$ . The simplest FO transform [FO99] defines  $\text{Enc}'$  such that

$$\text{Enc}'(\text{pk}, m; r) := \text{Enc}(\text{pk}, m \| r; \mathbf{G}(m \| r)). \quad (1)$$

A crucial observation is that cheating provers are now forced to derive encryption randomness uniformly by querying the random oracle  $\mathbf{G}$ . 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 FrodoKEM achieve undeniability. Details are deferred to [TZ21].

### 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<sup>+</sup>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  $\text{sk}$ , 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  $\Pi_f$  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  $\text{MPCitH-IOP}_R$  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  $\epsilon_{\text{cext}}$ -extractability and is  $\epsilon_{\text{hide}}$ -hiding. Then the compiled protocol, **MPCitH-VE** <sub>$R$</sub>  described in Protocol 2 with  $n = \tau$ , is  $\epsilon_{\text{val}}$ -valid with validity error  $\epsilon_{\text{val}} = \epsilon_{\text{sle-iop}} + \epsilon_{\text{cext}}$ , and  $\epsilon_{\text{zk}}$ -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  $\text{MPCitH-IOP}_R$ . 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 [TZ21].

**Protocol 2: MPCitH-VE<sub>R</sub>**

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

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

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

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

1.  $\mathcal{P}$  runs  $\mathbf{P}$  on input  $(x, w)$  to obtain the proof string  $\pi = (V_1, \dots, V_N)$ .
2.  $\mathcal{P}$  separately commits to each of these  $N$  views to generate per-party commitments  $(\text{C}_1, \dots, \text{C}_N)$  where  $\text{C}_i = \text{Commit}(\text{pk}, V_i; r_i)$  and  $r_i$  is commitment randomness uniformly sampled from  $S_r$ .
3.  $\mathcal{V}$  invokes  $\mathbf{V}$  on input  $x$  to obtain challenge  $\mathbf{e} \in \text{Ch}$ , 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  $\mathbf{V}$  as a response to the oracle query. It accepts if and only if:
  - (a)  $\text{C}_i = \text{Commit}(\text{pk}, V_i; r_i)$  and  $r \in S_r$  for all  $i \in \mathbf{e}$ , i.e.,  $\mathcal{P}$  opened the views corresponding to  $(\text{C}_i)_{i \in \mathbf{e}}$  successfully, and
  - (b)  $\mathbf{V}$  outputs 1.

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

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

Otherwise,  $\mathcal{V}$  outputs  $b = 0$  and  $\text{tr} = \perp$ .

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

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  $w_i^{(j)} = \text{GetW}(V_i^{(j)})$  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  $C = (\tilde{w}^{(j)}, (\text{C}_i^{(j)})_{i \notin \mathbf{e}^{(j)}})_{j \in S}$ .

**Receiver**  $\mathcal{R}(\text{sk}, C)$ : 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  $\hat{V}_i^{(j)} = \text{CExt}(\text{sk}, \text{C}_i^{(j)})$  and computes the corresponding witness shares  $\hat{w}_i^{(j)} = \text{GetW}(\hat{V}_i^{(j)})$ . Let  $w^{(j)} = \tilde{w}^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}$  be the  $j$ th candidate witness.
2. If there exists some  $j \in S$  such that  $(x, w^{(j)}) \in R$ , output  $w^{(j)}$ . Otherwise, output  $\perp$ .

**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<sup>+</sup>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<sup>+</sup>20, BDK<sup>+</sup>21, DOT21, KZ22], in case the MPC protocol  $\Pi_f$  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  $\text{msgs}_i$  consists of messages broadcast by party  $i$ . That is, the prover  $\mathcal{P}$  first generates a root seed  $\text{sd}^*$  to derive per-party seeds  $(\text{sd}_i)_{i \in [N]}$  with a binary tree construction.  $\mathcal{P}$  now only commits to each seed  $\text{sd}_i$  used for deriving a witness share and a random tape of party  $i$  using ECOM, and sends  $h = H((\text{msgs}_i)_{i \in [N]})$ . 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  $(\text{sd}_i)_{i \in [N] \setminus \{\bar{i}\}}$ . 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  $(\text{sd}_i)_{i \in [N] \setminus \{\bar{i}\}}$  lead to a correct output with respect to  $x$  and  $\text{msgs}_{\bar{i}}$ . Our DKG-in-the-head protocol in section 5.1 benefits from these optimizations.

### 3.3 Compiling Other MPC-in-the-Head Proofs

Although the IOPs corresponding to KKW and Banquet (given in [TZ21]) 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  $(\text{sd}_i)_{i \in [N]}$  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  $\mathbf{V}$  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  $\Delta w$  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 [TZ21], the compiled protocol has  $\epsilon_{\text{val}}$ -validity with  $\epsilon_{\text{val}} = \epsilon_{\text{cext}} + \epsilon_{\text{sle}}$ , assuming  $\epsilon_{\text{cext}}$ -extractability of ECOM and  $\epsilon_{\text{sle}}$ -SLE of Banquet-IOP. [TZ21] 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  $(\text{st}_i^{(j)})_{i \in [N]}$  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  $\tau$  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  $\mathcal{C}_{\bar{i}_j}^{(j)}$  for  $j \in T \subset [M]$ , where each witness mask share  $\lambda_i^w$  is obtained from the revealed value  $\text{st}_i^{(j)}$ . Then the receiver  $\mathcal{R}$  extracts the unopened share of the witness mask from  $\mathcal{C}_{\bar{i}_j}^{(j)}$  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<sup>+</sup>20, FR23] 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  $i = 1, \dots, N$  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  $\mathbf{pk}$  to obtain the challenge  $\mathbf{e} \in \text{Ch}$ . 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 [TZ21]. 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<sup>+</sup>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 [TZ21].

### 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  $\epsilon_{\text{val}}\text{-strong validity}$  if for all pairs of PPT adversaries  $(\mathcal{A}^*, \mathcal{P}^*)$ ,

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

We remark that allowing  $\mathcal{A}^*$  to choose  $(\mathbf{pk}, \mathbf{sk})$  is very strong, and that in practice it's not possible to check whether  $(\mathbf{pk}, \mathbf{sk}) \in \mathcal{G}(1^\lambda)$ . However, without this condition, note that  $\mathcal{A}^*$  can trivially break strong validity by generating a keypair then setting  $\mathbf{sk}$  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  $\mathbf{sk}$ .

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.



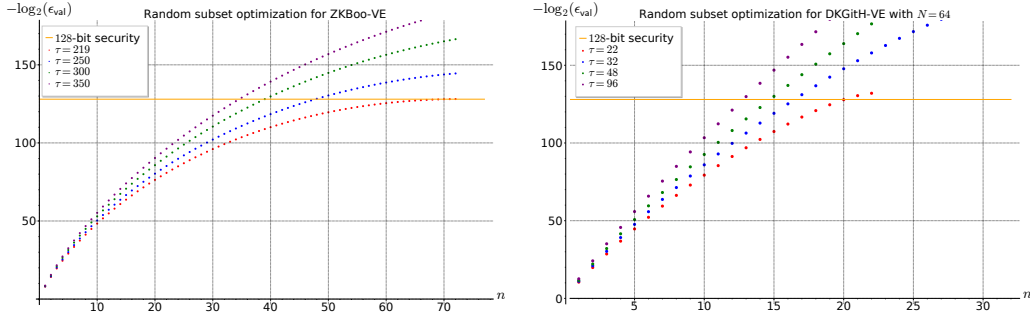


Figure 1: Approximate minimum cost of breaking validity of ZKBoo-based VE (left) and DKGith-based VE (right) with a random subset of size  $n$ . The parameter  $\tau$  denotes the number of parallel repetitions. The number of parties  $N$  is fixed to 3 for ZKBoo and 64 for DKGith, respectively. Note that  $\tau = 219$  corresponds to the `picnic-L1` parameters from the Picnic spec [ZCD<sup>+</sup>20].

## 4 Compressing Ciphertexts

Because MPCitH protocols use  $\tau$  parallel repetitions to boost soundness, the ciphertexts output by our transform can be large. For example, for 128-bit security,  $\tau$  could range from 20 to 219. Each repetition outputs one PKE ciphertext and a share of the witness, so the total size is  $\tau(|\text{PKE.Enc}| + |w|)$ . 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  $\tau$  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  $\tau$  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 [TZ21], and give only 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  $\tau$  ciphertext components, corresponding to the  $\tau$  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 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  $\epsilon_{\text{sle-iop}}(n)$  which corresponds to the adversary’s advantage in convincing the verifier while using  $n$  bad repetitions (out of  $\tau$ ) 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  $\epsilon_{\text{sle-iop}}(n)$ , 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  $\tau$  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  $V$ ’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

$$\begin{aligned} & \Pr [C \text{ selects } n \text{ of } s \text{ bad ctexts} \wedge V \text{ accepts a proof with } s \text{ bad ctexts}] \\ &= \frac{\# \text{subsets with } n \text{ bad ctexts}}{\# \text{ of subsets}} \cdot \Pr [V \text{ accepts a proof with } s \text{ bad ctexts}] = \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot (\epsilon_{\text{sle-iop}}(s) + \epsilon_{\text{cext}}) \end{aligned}$$

where  $\epsilon_{\text{sle-iop}}(s)$  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 [TZ21], where we prove the following theorem.

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

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

Generally, the amount of compression possible is larger when  $\tau$  is larger, as demonstrated by the ZKB++ example (where  $\tau = 219$  for 128-bit security). The DKGitH example requires much smaller  $\tau$  (in the range 16–32), and compression is limited, or none at all. However, we can increase  $\tau$  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  $\text{Ch} = \{1, 2, 3\}$  they have 3-consistency and thus are SLE with knowledge error  $\epsilon_{\text{sle-iop}}(s) \leq 2/3$ . In fig. 1 we show the costs of breaking validity  $-\log_2(\epsilon_{\text{val}}(\tau, n))$  for different combinations of  $\tau$  and  $n$  assuming  $\epsilon_{\text{cext}}$  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  $\tau$  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  $\tau$  and  $n$  assuming  $\epsilon_{\text{cext}}$  is negligible. As  $\tau$  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  $\tau$  is again possible, but provides less compression and at a higher cost. In addition to the choices of  $(n, \tau)$  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  $\Sigma$  protocol, combined with Schnorr's  $\Sigma$ -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  $\Sigma$  protocol with one challenge). No parameters are given for concrete, non-interactive security – we found that for  $\lambda$ -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 assumptions 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  $R = \{(y, x) : y = g^x\}$ .<sup>2</sup> 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  $\Pi_f$  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

<sup>2</sup>We present the final VE scheme to illustrate our implementation details. The IOP implicit in DKGitH is sketched in the validity analysis in [TZ21].

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 [TZ21] 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 [TZ21], 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.

### 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]:

1. Party 0 (dealer), upon receiving the witness (DLOG)  $x$  as input, samples uniformly random  $a_i \in \mathbb{F}$  for  $i = 1, \dots, t$  and lets  $a_0 = x$ . Define a degree- $t$  polynomial  $a(X) = a_0 + a_1X + \dots + a_tX^t$ .
2. For  $i = 1, \dots, N$ , send to party  $i$  a Shamir secret share  $x_i = a(i)$  of  $x$ . Moreover, broadcast a commitment to the polynomial  $A_0 = g^x, A_1 = g^{a_1}, \dots, A_t = g^{a_t}$ .
3. Each party  $i$  checks the validity of its share:  $g^{x_i} \stackrel{?}{=} \prod_{j=0}^t A_j^{i^j}$ .

The corresponding VE protocol instantiated with the hashed Elgamal PKE is detailed in [TZ21]. 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  $\phi : m \mapsto m^e \pmod{n}$  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  $R = \{((ct, pt), K) : ct = AES_K(pt)\}$ , one can verifiably encrypt  $K$  satisfying the relation  $R$  with any PKE. The compiled VE scheme Banquet-NIVE and validity proof are detailed in [TZ21]. 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.

As we analyze in [TZ21], variants of the FO transform can be used for achieving undeniability and thus many efficient post-quantum PKE schemes, including Kyber [SAB<sup>+</sup>20] and FrodoKEM [NAB<sup>+</sup>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.<sup>3</sup> 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

<sup>3</sup>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	$ tr $	$ C $	$ C _{RS}$	$\mathcal{P}$ (ms)	$\mathcal{V}$ (ms)	$\mathcal{C}$ (ms)
DKGitH ( $N, \tau, n$ )	(64, 48, 15)	9 360	3 120	975	182.45	181.31	2.57
	(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 ( $N, t, n$ )	(132, 64, 67)	11 781	6 596	6 499	6.65	4.36	45.76
	(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] ( $k, u$ )	(712, 20)	52 968	1 300		42.29	37.88	
	(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 <sup>†</sup>	40.28 <sup>†</sup>	
AES-VE ( $N, \tau$ )	(16, 31)	40 396	24 894		12.41	12.13	0.31
	(57, 22)	29 400	17 676		23.63	22.81	0.77
	(256, 16)	21 920	12 864		67.24	66.43	2.16

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 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 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.

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<sup>+</sup>22], or those based only on symmetric-key primitives such as [KKW18, DRS18].

## Acknowledgments

The authors are grateful to Ivan Damgård, Bernardo David, Matthias Geihs, Yashvanth Kondi, and Claudio Orlandi for helpful comments and insightful discussions. We thank Matthias Geihs for bringing to our attention inconsistencies of benchmarks in earlier versions of this paper. We thank our anonymous referees for their thorough proof reading and constructive feedback. This paper was partially prepared while Akira Takahashi was affiliated with Aarhus University and the University of Edinburgh, where he was supported by the Protocol Labs Research Grant Program PL-RGP1-2021-064.

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. Liger: 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.
- [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, *EURO-CRYPT’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](https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#kms_keys).
- [BBB<sup>+</sup>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.
- [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.
- [BDK<sup>+</sup>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<sup>+</sup>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<sup>+</sup>20] Rishabh Bhaduria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkatasubramanian, Tiancheng Xie, and Yupeng Zhang. Liger++: 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.
- [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.

- [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](https://doi.org/10.1007/978-3-540-30574-3_11).
- [CCFG16] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: 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](https://doi.org/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](https://doi.org/10.1007/3-540-44448-3_25).
- [CDG<sup>+</sup>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](https://doi.org/10.1145/3133956.3133997).
- [CDK<sup>+</sup>22] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, 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](https://doi.org/10.1007/978-3-031-22969-5_6).
- [CF85] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th FOCS*, pages 372–382. IEEE Computer Society Press, October 1985. doi:[10.1109/SFCS.1985.2](https://doi.org/10.1109/SFCS.1985.2).
- [CFF<sup>+</sup>21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zk-SNARKs 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](https://doi.org/10.1007/978-3-030-92078-4_1).
- [CHM<sup>+</sup>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, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. doi:[10.1007/978-3-030-45721-1\\_26](https://doi.org/10.1007/978-3-030-45721-1_26).
- [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](https://doi.org/10.1007/978-3-540-45146-4_8).
- [DDOS19] Cyprien Delpéch 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](https://doi.org/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.
- [DHMW23] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wöhrig. 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.
- [DOT21] Cyprien Delpach 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.
- [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] Thibault 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>.

- [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<sup>+</sup>22] Tim Güneysu, Philip W. Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. Proof-of-possession for KEM certificates using verifiable generation. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1337–1351. ACM Press, November 2022. [doi:10.1145/3548606.3560560](https://doi.org/10.1145/3548606.3560560).
- [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](https://doi.org/10.1109/FOCS.2012.47).
- [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.
- [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](https://doi.org/10.1007/11761679_21).
- [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](https://doi.org/10.1007/978-3-662-49896-5_11).
- [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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1145/1250790.1250794).
- [Kal22] Daniel Kales. Implementation of bn++ and helium signatures, 2022. [https://github.com/IAIK/bnpp\\_helium\\_signatures](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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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, EuroS&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](https://doi.org/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>.
- [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](https://doi.org/10.1007/978-3-319-56620-7_11).
- [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](https://doi.org/10.1145/3319535.3339817).
- [Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.
- [NAB<sup>+</sup>19] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2019. available 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 Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020. doi:10.1145/3372297.3417236.
- [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>.
- [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.
- [SAB<sup>+</sup>20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede 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.
- [TZ21] Akira Takahashi and Greg Zaverucha. Verifiable Encryption from MPC-in-the-Head. Cryptology ePrint Archive, Paper 2021/1704, 2021. <https://eprint.iacr.org/2021/1704>. URL: <https://eprint.iacr.org/2021/1704>.
- [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](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<sup>+</sup>20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladimir 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>.