Verifiable Encryption from MPC-in-the-Head

Akira Takahashi[1] and Greg Zaverucha[2]

[1] Aarhus University
takahashi@cs.au.dk
[2] Microsoft Research
gregz@microsoft.com

December 31, 2021

Abstract. Verifiable encryption (VE) is a protocol where one can provide assurance that an encrypted plaintext satisfies certain properties. It is an important buiding block in cryptography with many useful applications, such as key escrow, group signatures, optimistic fair exchange, etc. However, a 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 the MPC-in-the-head zero-knowledge proof systems (Ishai et al. STOC 2007). Our generic compiler can turn a large class of MPC-in-the-head ZK proofs into secure VE protocols for any CPA secure public-key encryption (PKE) schemes 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 prover's work can be focused on proving properties (i.e. relation) about the encrypted data, not the proof of plaintext knowledge. Hence, our approach allows for instantiation with various combinations of properties about encrypted data and encryption functions. As concrete applications we describe new approaches to verifiably encrypting discrete logarithms in any prime order group and AES private keys.

Table of Contents

1 Introduction . . . 3
1.1 Our contributions . . . 4
1.2 Comparison with previous verifiable encryption schemes . . . 5
1.3 Other related work . . . 6
2 Preliminaries . . . 7
2.1 Public key encryption . . . 8
2.2 Extractable commitment . . . 8
2.3 Verifiable encryption . . . 9
2.4 Interactive oracle proofs . . . 10
2.5 MPC-in-the-head . . . 11
3 Our Transform . . . 14
3.1 Extractable commitment from undeniable public-key encryption . . . 14
3.2 Compiling MPCitH-IOP into verifiable encryption with extractable commitments . . . 15
3.3 Compiling Banquet and KKW . . . 19
3.4 Applying Fiat–Shamir . . . 19
4 Methods for Compressing Ciphertexts . . . 19
4.1 The random subset method . . . 20
4.2 The equality proof method . . . 21
5 Concrete Instantiations . . . 23
5.1 Verifiable encryption of discrete logarithms in any prime order group . . . 23
5.2 Verifiable encryption of AES private keys . . . 26
6 Camenisch–Damgård Verifiable Encryption with Imperfect Correctness . . . 27
6.1 The Camenisch–Damgård framework [CD00] . . . 27
6.2 Committing encryption is required for [CD00] . . . 27
6.3 Fixing the [CD00] security analysis . . . 29
A Bitwise Commitment and Encryption . . . 33
B Attacking [CD00] Instantiated with LWE-based Encryption Schemes . . . 33
B.1 LWE encryption scheme . . . 33
B.2 Attacks breaking undeniability . . . 34
B.3 Attacking validity of [CD00] . . . 34
C Undeniability and Binding of Fujisaki–Okamoto Transform . . . 35
C.1 PKE_1 [HHK17] . . . 35
C.2 PKE_2 [FO99] . . . 35
C.3 PKE_3 [HHK17] . . . 36
D Proof for Theorem 2 . . . 37
E KKW as an IOP . . . 39
F Banquet as an IOP . . . 40

1 Introduction

A verifiable encryption (VE) scheme is a public-key encryption scheme where one party (called a *prover* $\mathcal{P}$) can encrypt data w with a public key pk (of which the corresponding decryption key sk is owned by the *receiver* $\mathcal{R}$), and convince a third party (called the *verifier* $\mathcal{V}$) that the data satisfies some relation, i.e., $R(x, w) = 1$ with respect to some 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, $\mathcal{V}$ accepts after interacting with $\mathcal{P}$ and $\mathcal{R}$ with knowledge of sk obtains a plaintext w satisfying $R(x, w) = 1$.
- **Zero knowledge**: $\mathcal{V}$ without decryption key sk learns nothing about the plaintext.
- **Validity**: If $\mathcal{V}$ accepts after interacting with potentially malicious $\mathcal{P}^*$, $\mathcal{R}$ is guaranteed to obtain a correct plaintext w such that $R(x, w) = 1$.

Our motivating example for verifiable encryption is the *verifiable backup problem*, where a cryptographic device (such as a hardware security module (HSM)) that is entrusted to store key material must securely export it for backup in case of hardware failure. These backups must be encrypted with the public key of another device, so that the plaintext keys are never exposed outside of the secure hardware. The adminsitrator 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 different (backup) cloud provider greatly expands the trust boundary. Then even if the import operation succeeds, the admin must still test the imported key somehow, which typically requires using it to create a test signature or decryption, adding to use of the key requiring logging to enable auditing. 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 an ECDSA signing key x for a public verification key y" and the administrator should be convinced as long as $y = g^x$ 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 (and others that we review below).

Verifiable Encryption Despite being introduced more than two 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 limitied. 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]). We also want to minimize the additional assumptions required, ideally not requiring any new assumptions; for example if an AES key is to be exported, encrypted under an RSA key, we should not need to make assumptions in elliptic curve groups (perhaps with a pairing), as might be the case if certain SNARK proof systems were used for verifiability [Gro16,MBKM19,BBB+18,LCKO19]. We also want flexibility in the reciever's PKE, again to minimize new assumptions, but also to support security goals like threshold decryption or post-quantum security, rather than have a VE scheme dictate the PKE the receiver must use (as in [CS03,NRSW20,LN17]). Finally, performance must be good enough for use in practice, which excludes using fully general proof systems (e.g., [Mic00, GMW87, GOS06]). In summary, we desire a construction that is as general as possible, introduces no new assumptions, and is performant enough to be practical.

There are multiple applications of verifiable encryption in the literature. Some early examples are publicly verifiable secret sharing [Sta96], and verifiable encryption of signatures for optimistic fair exchange [ASW98, Ate99]. 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 is still present in cryptographic voting systems today, see for example [EG2, 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 R, typically expressed as binary or arithmetic circuits. We also give an MPCitH protocol to prove knowledge of a discrete logarithm (and use our results to verifiably encrypt discrete logs), showing that the paradigm extends beyond circuits as well. Second, if the MPC protocol is information theoretically secure, converting it to a ZK proof only requires a secure commitment scheme, which can be instantiated with a cryptographic hash function, so that the proof system requires minimal assumptions, and is post-quantum secure. Finally, the performance of MPCitH proof systems in terms of prover and verification costs and proof sizes are practical, and have been steadily improving as has been demonstrated in the area of post-quantum signatures. To use the AES-128 circuit as an example, proof sizes went from 209 KB [GCZ16] to 32 KB [dDOS19] to 13 KB [BdK+21] in the past five years, and the running time of the prover and verifier is roughly 50ms (see the implementation benchmarks in [BdK+21]). Taken together, these properties will allow us to construct verifiable encryption schemes that are very general, make minimal assumptions, achieve PQ security and are efficient enough for practical use.

1.1 Our contributions

Our results apply to a broad class of MPCitH proofs: those that can be viewed as an interactive oracle proof (IOP). This class captures the original class from [IKOS07] as well as many more recent MPCitH proofs aimed at concrete efficiency, such as [GMO16, KKW18, BdK+21, BN20, dSGOT21, Bd20, Beu20]. The IOP framework [BCS16] allows a modular design, and comes with definitions of zero-knowledge, and straight-line extractability that we need to prove our results.

Generic compiler for MPC-in-the-head-based verifiable encryption 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. We describe MPCitH-VE as a public-coin three-round interactive protocol, which can be made non-interactive using the standard Fiat-Shamir transform [FS87]. An abstract protocol MPCitH-IOP captures several three-round protocols, including [IKOS07], ZKBoo [GMO16], ZKB++ [CDG+17], and our new DKG-in-the-head protocol described below. We also discuss how to compile KKW-IOP and Banquet-IOP (IOP versions of [KKW18] and [BdK+21], respectively) using essentially the same idea.

The other input to the compiler is a public key encryption (PKE) scheme, such as Elgamal, RSA-OAEP or PQ options like Frodo or Kyber. 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, 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 PQC schemes are usually not perfectly correct. In Appendix C we prove that existing variants of the Fujisaki-Okamoto transform [FO99, FO13, HHK17] can upgrade any statistically correct PKE schemes to obtain undeniability, making our construction compatible with many existing schemes. It implies FO-transformed encryption schemes can be used as secure commitments without any modification, which might be of independent interest.

Our framework is versatile: because the circuit proven by the MPC-in-the-head prover is decoupled from a complex encryption function, the prover's work can be focused on proving properties (i.e. relation R) about the encrypted data, not the proof of plaintext knowledge. Hence, our approach allows for instantiation with various combinations of properties about encrypted data and encryption functions. To illustrate the core idea of our transform, Fig. 1 describes an example VE scheme constructed from ZKBoo. Essentially, we replace commitments in the original proof system with public-key encryption functions. The verifier still learns nothing about the encrypted data w since one of its additive shares is kept encrypted. By contrast, the receiver $\mathcal{R}$ with knowledge of the decryption key sk can decrypt the unopened ciphertext (or commitment) to obtain the remaining share, from which the plaintext w can be recovered using revealed shares in the public transcript.

Methods for compressing ciphertext In our compiler, essentially the *transcript* itself is output as a ciphertext, after the prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact in MPCitH-VE. While the size of transcript is proportional to the number of parallel repetitions τ guaranteeing negligible soundness error, the receiver $\mathcal{R}$ only needs one of them in case the prover behaves honestly. To close this gap, in Section 4 we give two methods 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 τ is large. If

τ is already small, it is also possible to trade ciphertext size for τ, which might be desirable depending on the application. The second approach, 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 highlight improving compression as an interesting direction for future work.

Concrete instantiations In Section 5 we describe concrete approaches to verifiably encrypting discrete logarithms in any prime order group and AES private keys.

The former is realized by our new non-interactive ZK protocol for relation $R = \{(y, x) : y = g^x\}$, called *distributed key generation in the head (DKG-in-the-head)*. In this protocol, the prover emulates "in the head" a protocol where parties perform DKG to obtain $y = g^x$. 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 MPC parties.

The latter, verifiable encryption of AES private keys, is derived from the underlying interactive ZK protocol of $\mathsf{Banquet}$ [BdK+21], where the prover proves knowledge of AES private key used for generating an AES ciphertext from some public plaintext, i.e., it is specialized for relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$. Prior to this work, there has been no VE scheme for verifiably encrypting AES private keys, while it may find interesting applications in the post-quantum setting when instantiated with quantum-resilient public-key encryption functions.

Revisiting the Camenisch-Damgård verifiable encryption As a separate contribution, we revisit the existing verifiable encryption of Camanisch and Damgård [CD00] in Section 6 and show that it fails to retain the validity property when instantiated with *statistically correct* IND-CPA PKE schemes. We describe concrete attacks in which a malicious prover can convince the verifier, while the output ciphertext does not correctly decrypt to the data satisfying a claimed property. Finally, we show that by additionally assuming the undeniability property their construction can also be securely instantiated with statistically correct PKE schemes.

1.2 Comparison with previous verifiable encryption schemes

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

Camenisch–Shoup scheme Camenisch and Shoup [CS03] proposes protocols for efficient verifiable encryption and decryption of discrete logarithms. However, it only works for discrete logarithms in a group where Paillier's decision composite residuosity (DCR) assumption holds, and the PKE is fixed to (a variant of) Paillier's scheme as well. The scheme is not suitable for encrypting an ECDSA private key, one of our motivating examples.

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

Nick et al. [NRSW20] gives a construction, which can encrypt a discrete logarithm in an elliptic curve group, using a special PRF called Purify. The scheme does allow, e.g., encryption of an ECDSA private key without any trusted setup assumption thanks to the use of Bulletproofs [BBB+18], but requires that encryption be done with an Elgamal-like PKE. As we compare in Table 1, their ciphertext and proof are more compact than those of our DKG-in-the-head VE scheme, while ours has shorter prover time. A complication related to implementation of the Purify PRF is that one must choose an additional pair of

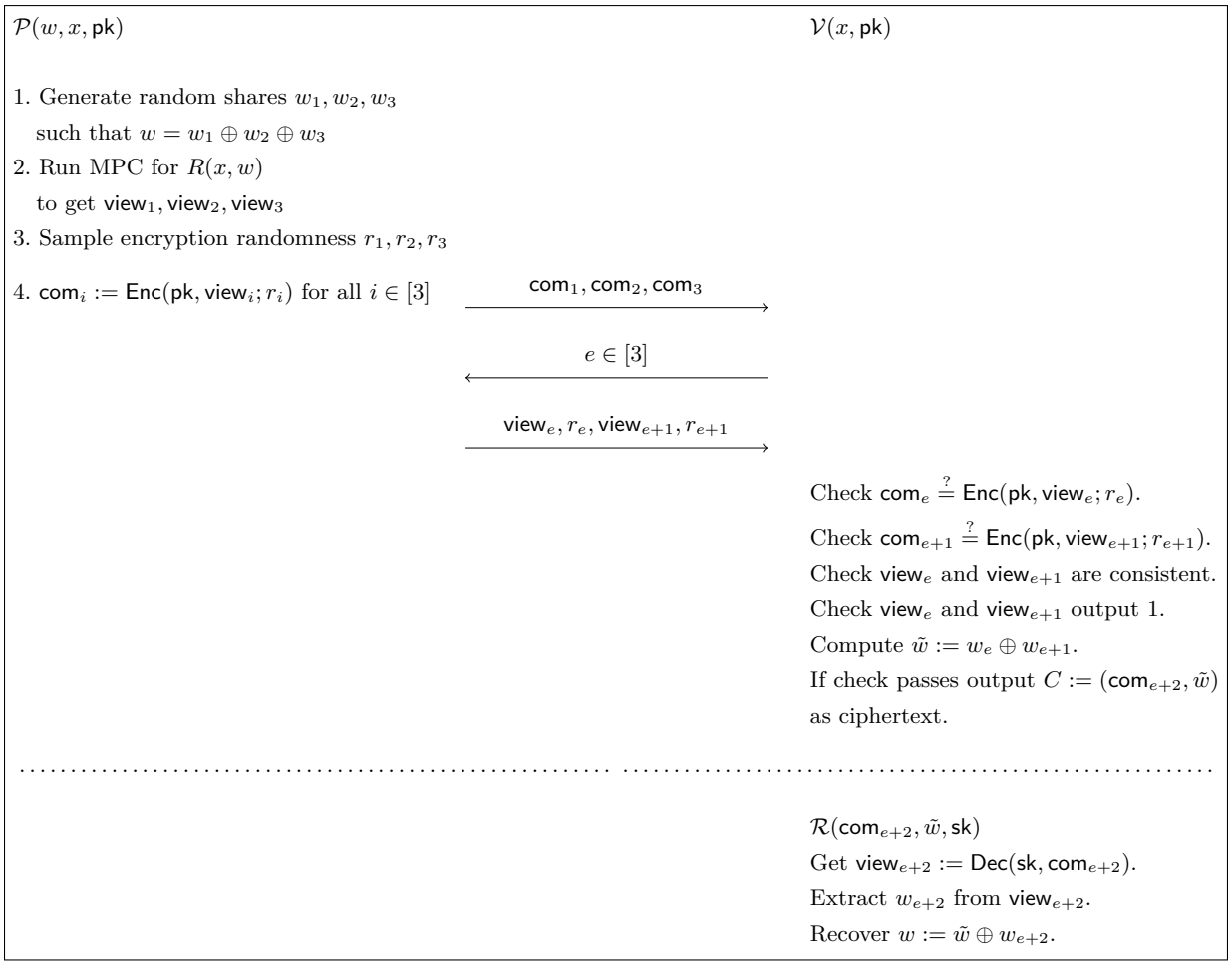

Fig. 1: High-level overview of our transform, applied to ZKBoo

elliptic curves, related to the group order of the curve where the discrete logarithm is defined, such that the DDH assumption holds. In contrast, our framework does not introduce any additional assumption other than IND-CPA and undeniability of PKE (already satisfied by perfectly correct schemes and many statistical ones as we analyze).

Lattice-based construction Lyubashevsky and Neven [LN17] presents a verifiable encryption scheme for lattices, based on the hardness of the ring learning with errors (RLWE) problem. They give a proof of plaintext knowledge, secure in the ROM that does not use parallel repetition to boost soundness. Their scheme can be further extended to support CCA security. Our VE construction is only proven secure under the security definition of [CD00] and it is not "one-shot" as MPC-in-the-head proofs usually rely on parallel repetition or cut-and-choose unlike [LN17]. Their construction comes with multiple caveats. A malicious prover may create a ciphertext that takes variable time to decrypt. In particular decryption requires $O(q)$ time to decrypt, where q is the number of hash queries made by the prover. A receiver in our construction only needs $O(\tau)$ time to decrypt or even less if the compression techniques of Section 4 are applied, where τ is the number of parallel repetitions. Note that in both schemes decryption only takes $O(1)$ time if the prover is honest. Moreover, their VE prover can only prove "relaxed" linear relations about the encrypted data, while our framework naturally supports any NP relation as a corollary of [IKOS07].

1.3 Other related work

We briefly survey some of the many existing MPCitH-based proof systems, optimized for different relations, as these immediately give verifiable encryption schemes by applying our transform. [Beu20] gives an MPCitH-based proof of a solution of an SIS (short integer solution) instance. We can apply our transform to construct a verifiable encryption of SIS witnesses (here the witness is exact, not relaxed as in [LN17]). The other proof protocols in [Beu20] for other relations, such as the PKP and MQ problems, are also

compatible with our transform. [BN20] also gives multiple MPCitH-based proofs for lattice problems (SIS), which are also ameable to our transform, but are outperformed by the proofs of [Beu20]. Gjøsteen et al. [GHM+21] present verifiable *decryption* protocols from MPCitH proofs, by designing suitable distributed decryption protocols for Elgamal and BGV lattice-based encryption schemes.

Aurora [BCR+19] and Ligero [AHIV17] are non-interactive proof systems for R1CS that are constructed by defining an IOP, then making it non-interactive using the transform in [BCS16]. Both have short proofs for relations involving lattices, and Aurora has the shortest proofs for SIS, about 10x shorter than [Beu20, BN20].

Connection between straight-line extraction and verifiable encryption *Straight-line extractability (SLE)* (or sometimes called *online extractability*) is a special type of extractability, specialized to proof systems in the ROM or in the CRS model. The prover commits to witness-dependent strings via extractable commitments instantiated with the RO or PKE, and the extractor is given the statement, the transcript, and the prover's query history (in the ROM) or a secret trapdoor (in the CRS model) to extract a witness. In particular the straight-line extractor does not get any access to the prover, or ability to rewind them. SLE is especially crucial for security in the QROM, since rewinding techniques are generally prohibitively expensive in that setting. Numerous works achieve SLE of commit-and-open-type proof systems (including MPC-in-the-head) [Pas03, KKW18, DFMS21, HLR21], lattice-based ZK proof systems [Kat21], and straight-line extractable alternatives to the Fiat-Shamir transform [Fis05, Unr15]. A receiver $\mathcal{R}$ of our MPCitH-VE essentially behaves like a straight-line extractor for the MPC-in-the-head proof systems whose commitments are replaced with PKE. In this work, we formally draw a connection between the validity property of VE and SLE of IOP, a setting where commitments are idealized and thus SLE holds very naturally.

Committing Encryption Most natural public key encryption schemes are committing, and constructing a non-committing one (a deniable scheme) is challenging. Therefore, our attack on [CD00] is mostly academic. Further, the assumption required by [CD00] and our new construction is mild and satisfied by common encryption schemes.

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

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

[BDD20] recently proved that Pointcheval's IND-CCA PKE [Poi00] can be used as a secure commitment scheme as is, and it is thus plausible that their analysis can be adapted to show undeniability of the scheme as well. Our analysis of the FO transform also suggests that several CCA conversions are useful for obtaining undeniability and thus binding. It is an interesting follow-up question whether CCA security in general is sufficient for PKE to be committing and/or undeniable.

The opposite of what we need is called *deniable encryption* [CDNO97]. Here the scheme is carefully constructed so that the encryption is *not* a binding commitment to the message and randomness, allowing a sender of a ciphertext to later claim they sent a different message (hence denying the original message). After sending a ciphertext $c = \mathsf{Enc}(m; r)$ then sender can later claim they sent (m', r'), and anyone can check that $c = \mathsf{Enc}(m'; r')$ as well. In the terminology of [CDNO97], a malicious prover in the [CD00] scheme could cheat using a public-key, sender deniable encyrption scheme, where the sender decides on the real and fake messages at the time of encryption. Constructing such a scheme appears to be an epic challenge, the construction in [CDNO97] is only secure when the messages are very long, and only much later, using indistinguishability obfuscation was one succeessfully constructed [SW14]. Neither are at all practical, and their complexity can be viewed as evidence that most practical encryption schemes we use today (or will use in the future) will *not* be deniable.

2 Preliminaries

First we introduce some notation and conventions used throughout the paper. The security parameter is denoted λ, and for an integer x, $[x]$ is short for the set $\{1, \ldots, x\}$. Whenever we have a two-part adversary,

written as a pair, e.g, $(\mathbf{A}^*, \mathbf{P}^*)$, we assume that $\mathbf{A}^*$ and $\mathbf{P}^*$ share state, and do not explicitly write it as an output of $\mathbf{A}^*$ and an input to $\mathbf{P}^*$. For a set S, we denote by $x \leftarrow\!\!\$\, S$ sampling an element x from S uniformly at random.

2.1 Public key encryption

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

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

Definition 1. PKE *is* ϵ_{cpa}*-IND-CPA secure if for any PPT adversary* $(\mathcal{A}_1, \mathcal{A}_2)$

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

Throughout we assume PKE satisfies IND-CPA security.

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

Definition 2. *[HHK17] Let* q_{G} *be the number of queries to* G *made by an adversary. We say that* PKE *is* $\delta(q_{\mathsf{G}})$*-correct if for all (possibly unbounded) adversaries* $\mathcal{A}$

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

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

2.2 Extractable commitment

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

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

We require ECOM to satisfy hiding, binding and extractability.

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

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

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

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

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

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

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

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

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

2.2.1 Extractable commitment from perfectly correct public-key encryption In the following we show that most commonly used public-key encryption schemes give rise to perfectly binding and computationally hiding commitment schemes. A similar construction appears in [GH03], and is somewhat folklore, below we describe the exact construction we will use, and analyze its security. Let $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ be a public key encryption scheme. We construct a commitment scheme $\mathsf{ECOM} = (\mathsf{CGen},\mathsf{Commit},\mathsf{CExt})$ as follows. 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.

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

We now show this is a secure commitment scheme for perfectly correct, IND-CPA secure encryption schemes. The two most commonly used choices of PKE, RSA and Elgamal, both meet these requirements, and can be used as commitment schemes.

Lemma 1. *If* PKE *is perfectly correct and* ϵ_{cpa}*-IND-CPA secure, the above commitment scheme is perfectly extractable, perfectly binding and* ϵ_{hide}*-computationally hiding with* $\epsilon_{\mathsf{hide}} \leq \epsilon_{\mathsf{cpa}}$.

Proof. **Extractability** follows from perfect correctness, since we are guaranteed that for every honestly generated key pair $(\mathsf{pk},\mathsf{sk})$ and for every $(m,r) \in S_m \times S_r$, $m = \mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m,r))$.

Binding follows from perfect correctness. Suppose there exits a tuple (m,r,m',r',c) such that $m \neq m'$ and $c = \mathsf{Enc}(\mathsf{pk},m;r) = \mathsf{Enc}(\mathsf{pk},m';r')$. Recall that perfect correctness guarantees that $m = \mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m;r))$. If binding is not perfect, there exsits $c = \mathsf{Enc}(\mathsf{pk},m;r) = \mathsf{Enc}(\mathsf{pk},m';r')$, and c cannot be decrypted correctly for both of m and m', which contradicts perfect correctness. Hence, COM is perfectly binding.

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

Note that for encryption schemes that are not perfectly correct, there can exist (m,m',r,r') such that $\mathsf{Enc}(\mathsf{pk},m;r) = \mathsf{Enc}(\mathsf{pk},m';r')$. We will show two examples of such schemes, one based on decisional composite rediduosity (Section 6.2), and one based on the learning with errors (LWE) problem (Appendix B). In general, the base encryption scheme for of post-quantum latticed-based candidates like FrodoKEM [NAB+19] and Kyber [SAB+20] are CPA secure, but not perfectly correct.

2.3 Verifiable encryption

We define a secure verifiable encryption scheme by adapting [CD00]. The main difference with [CD00] is that we additionally consider a *compression* algorithm $\mathcal{C}$ that takes a transcript exchanged between a prover and a verifier, and outputs a corresponding ciphertext. In practice, a compression algorithm is also performed 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 postprocess accepting transcripts to produce a highly compressed ciphertext.

Definition 3 (Verifiable Encryption Scheme). *Let R be a relation and $L_R := \{x : \exists w : (x,w) \in R\}$. A secure verifiable encryption scheme VE_R for a relation R consists of a tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{R})$:*

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

We require VE to satisfy completeness, validity and honest verifier zero knowledge.

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

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

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

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

Computational Honest Verifier Zero-knowledge VE_R is ϵ_{zk}-HVZK if there exits a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$, all $(x,w) \in R$,

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

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

2.4 Interactive oracle proofs

We recall interactive oracle proofs (IOP) originally introduced by [BCS16]. As the MPC-in-the-head relevant to our work have public-coin verifiers that make non-adaptive queries (i.e., queries made by the verifier are solely determined by the verifier's randomness and inputs), we consider a slightly restricted class of IOPs satisfying those properties. This allows us to divide the protocol into three phases similar to those of the AHP framework [CHM+20] (although we do not require a preprocessing phase). 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., [dSGOT21, CHM+20, CFF+20]). 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 extractability.

Definition 4 (IOP). *Let R be a relation and $L_R := \{x : \exists w : (x,w) \in R\}$. A (public-coin) r-round interactive oracle proof for a relation R consists of a tuple $(\mathbf{P}, \mathbf{V})$. The protocol proceeds as follows.*

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

– ***Decision phase*** *Based on the responses from oracles,* $\mathbf{V}$ *accepts or rejects.*

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

Definition 5 (Straight-line extractability (SLE)). *An IOP* $(\mathbf{P}, \mathbf{V})$ *is* straight-line extractable *with knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}}$ *if there exists an efficient extractor* $\mathbf{E}$ *such that for all pairs of unbounded adversaries* $(\mathbf{A}^*, \mathbf{P}^*)$

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

Definition 6 (Honest-verifier zero knowledge (HVZK)). *An IOP* $(\mathbf{P}, \mathbf{V})$ *is* $\epsilon_{\mathsf{zk\text{-}iop}}$-statistical honest-verifier zero knowledge *if there exists a PPT simulator* $\mathbf{S}$ *such that for every* $(x, w) \in R$*, the statistical distance between* $\mathbf{S}(x)$ *and* $\mathbf{V}$*'s view of the honest interaction with* $\mathbf{P}$ *on input* x *and* w *is at most* $\epsilon_{\mathsf{zk\text{-}iop}}$*.*

Remark 1 (Security proofs for IOPs). In our security proofs, we assume that an IOP prover $\mathbf{P}$ gets to see the query strings q_i. This naturally models all concrete protocols we consider where the verifier queries are fixed or sent to the prover, and does not affect security because the query phase happens after the prover has sent all proof strings.

Also, since most protocols realize the oracles by commiting to the proofs strings π_i, all π_i are available to the extractor by using extractable commitments (Section 2.2) or reading the query history in the ROM. For examples of IOPs that follow this paradigm see Marlin [CHM+20] and Lunar [CFF+20].

Remark 2 (Uniqueness of extracted witness). While we are guaranteed that the witness w' output by extractor algorithm $\mathbf{E}$ of Definition 5 satisfies $(x, w') \in R$, w' might not be the same witness used by $\mathbf{P}$ when creating the proof, if there are multiple valid witness per statement. For example, when proving knowledge of a symmetric key that relates a given plaintext-ciphertext pair (as Banquet does for AES) it may be easy to find keys k_1, k_2 such that $E_{k_1}(p) = E_{k_2}(p)$ where p is a fixed plaintext block. In the context of our verifiable encryption construction, where decryption invokes the IOP extractor, it will be important that R is such that x is a binding commitment to w.

2.5 MPC-in-the-head

2.5.1 Three-round MPC-in-the-head as an IOP We describe the blueprint of generic MPC-in-the-head protocols characterized as a single-round IOP. See $\mathsf{MPCitH\text{-}IOP}_R$ in Protocol 1. The prover proves knowledge of a witness w such that $R(x, w) = 1$, where Π_f is an MPC protocol computing f that uses additive secret sharing over some finite field $\mathbb{F}$, and $R(x, w) := (f(w) \stackrel{?}{=} x)$. This protocol is similar to the one from [IKOS07] relying on the "idealized commitment functionality", but modified to cover MPC protocols with a broadcast functionality, so the prover may open $2 < t < N$ parties' views instead of two. We also employ the IOP framework following more recent MPC-in-the-head protocols such as Ligero [BFH+20] and Limbo [dSGOT21]. As we shall see below, as an IOP protocol it is straightforward to prove straight-line extractability of $\mathsf{MPCitH\text{-}IOP}_R$. This will allow a smooth transition to SLE of the MPCitH proof systems we compile (with suitable commitment schemes), then to the validity of the resulting verifiable encryption schemes.

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

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

1. The outputs of the opened parties (determined by their views) are 1, and
2. The opened views are consistent with each other, with respect to x and Π_f,

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

We also introduce the notion of k-consistency, which essentially guarantees N views correspond to an honest run of Π_f as long as for any k distinct subsets of party indices, the corresponding parties' views are consistent with each other. This generalizes the notion of "consistency" introduced for pairwise views $(k = 2)$ [IKOS07, Def. 2.2].

Protocol 1: MPCitH-IOP$_R$

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

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

Query phase

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

Decision phase: $\mathbf{V}$ accepts if and only if $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$.

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

Definition 7 (k-consistency). *A single repetition of the protocol* MPCitH-IOP$_R$ *has* k-consistency *if for any x, for any set of views $(V_1, \ldots, V_N)$ and for any subset of the the challenge space $S \subseteq \mathsf{Ch}$ such that $|S| \geq k$, the following two conditions are equivalent:*

1. *for every $\mathbf{e} \in S$, $\mathsf{CheckView}(x, (V_i)_{i \in \mathbf{e}}) = 1$;*
2. *$(V_1, \ldots, V_N)$ form an honest execution of Π_f on a public input x and the corresponding per-party private inputs $w_i = \mathsf{GetW}(V_i)$ such that $f(\sum_{i \in [N]} w_i) = x$.*

Remark 3. The above notion captures several different instantiations of MPC-in-the-head protocols. For example, the original protocol from [IKOS07, §3] opens 2-out-of-N parties (i.e., corresponding to the case where $t = N - 2$ in MPCitH_R) and satisfies $\binom{N}{2}$-consistency because their Lemma 2.3 only guarantees the validity of N views as long as every possible pair of the views is consistent. ZKBoo and ZKB++ are essentially a special case of that protocol with $N = 3$ and therefore they have 3-consistency. Looking ahead, our DKGitH protocol in Section 5.1 works with N parties and the challenge set Ch is all subsets of $[N]$ of size $N - 1$.[3] We will show it satisfies 2-consistency thanks to the use of a broadcast functionality.

Definition 8 (Canonical extractor). *An extractor $\mathbf{E}$ for* MPCitH-IOP$_R$ *is called* canonical *if on input x and $\pi = (V_1, \ldots, V_N)$, it works as follows: $\mathbf{E}$ obtains witness shares via $w_i = \mathsf{GetW}(V_i)$ for $i \in [N]$ and then outputs a candidate witness $w := \sum_{i \in [N]} w_i$.*

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

Lemma 2. *If a single repetition of* MPCitH-IOP$_R$ *has k-consistency, then it is SLE with respect to the canonical extractor $\mathbf{E}$ with knowledge error $\epsilon_{\mathsf{sle\text{-}iop}} \leq \frac{k-1}{|\mathsf{Ch}|}$.*

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

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

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

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

Lemma 3. *If one repetition of* $\mathsf{MPCitH\text{-}IOP}_R$ *has* k*-consistency, then* τ *parallel repetitions are SLE with knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}} \leq \left(\frac{k-1}{|\mathsf{Ch}|}\right)^\tau$ *with the following extractor* $\mathbf{E}^\tau$*: on receiving* $(x, (V_1^{(j)}, \ldots, V_N^{(j)})_{j\in[\tau]})$ *it obtains* $w^{(j)} = \mathbf{E}(x, (V_1^{(j)}, \ldots, V_N^{(j)}))$ *for* $j \in [\tau]$ *and outputs* $w^{(j)}$ *if* $(x, w^{(j)}) \in R$ *for some* j*. Otherwise it outputs* $\perp$*.*

Proof. This is a straightforward generalization of Lemma 2. For a set of views output by $\mathbf{P}^*$ in each parallel repetition $j \in [\tau]$, one can define $\mathsf{GoodCh}^{(j)} := \left\{\mathbf{e} \in \mathsf{Ch} \ : \ \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}}) = 1\right\}$. If there exists $j \in [\tau]$ such that $|\mathsf{GoodCh}^{(j)}| \geq k$, then the extractor $\mathbf{E}^\tau$ succeeds in extracting a valid witness $w^{(j)}$ due to k-consistency. If for all $j \in [\tau]$ $|\mathsf{GoodCh}^{(j)}| < k$, then the probability that the τ independently sampled challenges simultaneously fall into $\mathsf{GoodCh}^{(j)}$ for $j \in [\tau]$ is at most $\left(\frac{k-1}{|\mathsf{Ch}|}\right)^\tau$. □

Remark 4. We note that the above knowledge error is equivalent to the soundness error. For example, for ZKBoo and ZKB++ we have that $k = 3$ and $\mathsf{Ch} = \{\{1,2\},\{2,3\},\{3,1\}\}$ and therefore both the SLE knowledge error and soundness error are $(2/3)^\tau$.

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

Definition 9 (t-privacy). *The protocol* Π_f *is said to be* t*-private if there exists a PPT simulator* Sim *such that for every* $\mathbf{e} \in [N]$ *of size at most* t *and for every input* $(x, w_1, \ldots, w_N)$*, the joint view of parties in* $\mathbf{e}$ *is distributed ideintically to* $\mathsf{Sim}(\mathbf{e}, x, (w_i)_{i\in\mathbf{e}}, R(x, \sum_{i\in[N]} w_i))$*.*

Lemma 4. *If the MPC protocol* Π_f *is* t*-private, then* $\mathsf{MPCitH\text{-}IOP}_R$ *is perfectly HVZK.*

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

2.5.2 Protocols without k-consistency While the notion of k-consistency has some generality and gives a convenient way to prove SLE of some three-round protocols, many MPC-in-the-head proof systems such as KKW and Banquet have challenge spaces not limited to party indices and therefore do not have k-consistency. However, we remark that they are easily checked to be straight-line extractable since $\mathbf{P}$ outputs per-party views that include the shares of the witness in the first round of the committing phase. The existing soundness analysis thus implies SLE of the corresponding IOP protocols.

KKW [KKW18] is an MPC-in-the-head proof system that produces much more compact proofs than [IKOS07] thanks to the *proprocessing phase*. In this paradigm, the prover first runs an offline protocol (also "in the head") to compute correlated randomness. Then it proceeds by executing the corresponding online phase taking the secret witness as input. For completeness, we present KKW-IOP in Protocol 5. Notice that in the first message $\mathbf{P}$ includes per-party states $(\mathsf{st}_i)_{i\in[N]}$ and the masked witness $\hat{w}$ for each MPC execution. If the function GetW is defined such that it outputs the ith share of witness mask λ_i^w on input st_i, a witness is recovered by computing $w = \sum_{i\in[N]} \lambda_i^w \oplus \hat{w}$. Hence, an extractor $\mathbf{E}$ for a single repetition similar to the canonical extractor can be defined assuming it takes per-party states and a masked witness as input. The following claim is implicit in [KKW18].

Lemma 5. KKW-IOP *is SLE with knowledge error*

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

where the parameters (N, M, τ) *are as defined in Protocol 5 and an extractor* $\mathbf{E}^M$ *proceeds as follows: on receiving* x *and* $\pi = (\mathsf{sd}^{(j)}, (\mathsf{st}_i^{(j)}, \mathsf{msgs}_i^{(j)})_{i\in[N]}, \hat{w}^{(j)})_{j\in[M]}$*, it outputs* $w^{(j)} = \mathbf{E}(x, (\mathsf{st}_i^{(j)})_{i\in[N]}, \hat{w}^{(j)})$ *if* $(x, w^{(j)}) \in R$ *for some* $j \in [M]$*. Otherwise, it outputs* $\perp$*.*

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

Banquet [BdK+21] is a recent MPC-in-the-head proof system that allows a prover to prove knowledge of an AES key with a relatively short proof. The base MPC protocol for computing the AES circuit uses secret-sharings over the field $\mathbb{F}_{2^8}$. For completeness, we present Banquet-IOP in Protocol 6. The protocol is highly optimized for relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) : \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$, where $(\mathsf{ct}, \mathsf{pt})$ is a public ciphertext-plaintext pair and $K = w$ is a witness. Notice that in the first message $\mathbf{P}$ includes commitments to per-party random seeds $(\mathsf{sd}_i)_{i\in[N]}$ and the offset Δw. If the function GetW is defined such that it outputs the ith witness share w_i on input sd_i, a witness can be easily recovered by computing $w = \sum_{i\in[N]} w_i + \Delta w$. Hence, an extractor $\mathbf{E}$ for a single repetition similar to the canonical extractor can be defined assuming it takes per-party seeds and the offset as input. The following claim is implicit in [BdK+21].

Lemma 6. Banquet-IOP *is SLE with knowledge error*

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

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

Likewise, the above probability corresponds to the soundness error of interactive Banquet.[4] In case of "bad inputs", it must be that the cheating prover guessed at least one of three challenges for each parallel repetition to pass the verification checks. The p_1, p_2 and p_3 above correspond to the probabilities that $\mathbf{P}^*$ correctly guesses first, second, and third challenge, respectively.

3 Our Transform

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

3.1 Extractable commitment from undeniable public-key encryption

Given $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we consider the extractable commitment scheme $\mathsf{ECOM} := (\mathsf{CGen}, \mathsf{Commit}, \mathsf{CExt})$ as defined in Section 2.2.1. As we shall see in later sections, IND-CPA security of PKE is not sufficient for guaranteeing validity of the resulting verifiable encryption, if the correctness is imperfect. The reason is that a malicious prover may be able craft a ciphertext c^* that can be correctly opened to plantext m^* such that it passes validity checks performed by a verifier, while c^* decrypts to junk during the recovery phase. To prevent this attack, we require an additional property called *undeniability*. Intuitively, undeniability forces an adversary to open any ciphertext to the plaintext identical to the result of decryption.

Definition 10 (Undeniability). *We say that a public-key encryption scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ϵ_{und}-undeniable *if for any PPT adversary* $\mathcal{A}$*:*

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

The following utility lemma guarantees that an undeniable IND-CPA encryption scheme can be used as a secure extractable commitment with the simple construction given in Section 2.2.1.

Lemma 7. *If* PKE *is* ϵ_{und}*-undeniable and* ϵ_{cpa}*-IND-CPA secure, then the commitment scheme* ECOM *constructed from* PKE *is* ϵ_{cext}*-extractable with* $\epsilon_{\mathsf{cext}} \leq \epsilon_{\mathsf{und}}$, ϵ_{bind}*-binding with* $\epsilon_{\mathsf{bind}} \leq \epsilon_{\mathsf{und}}$ *and* ϵ_{hide}*-hiding with* $\epsilon_{\mathsf{hide}} \leq \epsilon_{\mathsf{cpa}}$.

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

Proof. We prove the three properties separately.

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

Binding follows from undeniability. Suppose there exists an adversary that outputs tuple (m, r, m', r', c) such that it breaks binding with non-negligible probability, i.e., $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$.

Given such an efficient adversary $\mathcal{A}$ against the binding game, we construct another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break undeniability as follows.

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ as input, $\mathcal{B}$ forwards it to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs (c, m, r, m', r') such that $c = \mathsf{Enc}(\mathsf{pk}, m; r) = \mathsf{Enc}(\mathsf{pk}, m'; r')$ and $m \neq m'$, the $\mathcal{B}$ first decrypts c: $\tilde{m} = \mathsf{Dec}(\mathsf{sk}, c)$ and proceeds as follows.
   (a) If $\tilde{m} \neq m$, then $\mathcal{B}$ outputs (c, m, r) in the undeniability game.
   (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 proof for Lemma 1, since it only relies on the IND-CPA security of PKE. □

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

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

$$\mathsf{Enc}'(\mathsf{pk}, m; r) := \mathsf{Enc}(\mathsf{pk}, m||r; \mathsf{G}(m||r)). \tag{2}$$

A crucial observation is that cheating provers are now forced to derive encryption randomness uniformly by querying the random oracle G. This makes it difficult to craft a malicious ciphertext c from biased randomness, which decrypts to some junk 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 unideniability. Details are deferred to Appendix C.

3.2 Compiling MPCitH-IOP into verifiable encryption with extractable commitments

The following simple observation explains the intuition for our construction MPCitH-VE. The details are given in Protocol 2. As we observed in Section 2.5, 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 [KKW18, ZCD+20]: the extractor can reconstruct a witness

Protocol 2: MPCitH-VE$_R$

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

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

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

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

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

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

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

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

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

1. For $j \in [\tau]$, extract the t witness shares $w_i^{(j)} = \mathsf{GetW}(V_i^{(j)})$ for $i \in \mathbf{e}^{(j)}$ and partially reconstruct the witness $\tilde{w}^{(j)} = \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)}$.
2. Output the compressed ciphertext $C = (\tilde{w}^{(j)}, (\mathsf{com}_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in [\tau]}$.

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

1. For $j \in [\tau]$ and $i \notin \mathbf{e}^{(j)}$, extract the unopened parties' views $\hat{V}_i^{(j)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{com}_i^{(j)})$ and computes the corresponding witness shares $\hat{w}_i^{(j)} = \mathsf{GetW}(\hat{V}_i^{(j)})$. Let $w^{(j)} = \tilde{w}^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}$ be jth candidate witness.
2. If there exits some $j \in [\tau]$ such that $(x, w^{(j)}) \in R$, output $w^{(j)}$. Otherwise, output $\perp$.

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

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

Now we formally prove security of our construction.

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

Proof. We prove both properties separately.

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

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

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

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

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

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

$$\mathsf{fail} = \Pr\left[\begin{array}{r}\mathsf{CheckView}(x,(V_i)_{i\in\mathbf{e}}) = 1 \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{com}_i \\ \wedge (x, \sum_{i\in\mathbf{e}} w_i + \sum_{i\notin\mathbf{e}} \hat{w}_i) \notin R\end{array}\right] \tag{3}$$

$$= \Pr\left[\begin{array}{r}\mathsf{CheckView}(x,(V_i)_{i\in\mathbf{e}}) = 1 \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, V_i; r_i) = \mathsf{com}_i \\ \wedge (x, \sum_{i\in\mathbf{e}} w_i + \sum_{i\notin\mathbf{e}} \hat{w}_i) \notin R \\ \wedge \exists i \in \mathbf{e} : V_i \neq \hat{V}_i\end{array}\right] \tag{4}$$

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

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

If event (4) happens, then the extractability of ECOM is broken. Hence (4) is bounded by ϵ_{cext}.

We now give an upper bound for (5). First, it can be bounded as follows

$$(5) = \Pr\left[\begin{array}{r}\mathsf{CheckView}(x,(\hat{V}_i)_{i\in\mathbf{e}}) = 1 \wedge \forall i \in \mathbf{e} : \mathsf{Commit}(\mathsf{pk}, \hat{V}_i; r_i) = \mathsf{com}_i \\ \wedge (x, \sum_{i\in\mathbf{e}} \hat{w}_i + \sum_{i\notin\mathbf{e}} \hat{w}_i) \notin R \\ \wedge \forall i \in \mathbf{e} : V_i = \hat{V}_i\end{array}\right] \tag{6}$$

$$\leq \Pr\left[\mathsf{CheckView}(x,(\hat{V}_i)_{i\in\mathbf{e}}) = 1 \wedge (x, \sum_{i\in[N]} \hat{w}_i) \notin R\right] \tag{7}$$

because we are guaranteed that $\hat{V}_i = V_i$ and thus $\hat{w}_i = w_i$ for $i \in \mathbf{e}$. Given a cheating prover $(\mathcal{A}^*, \mathcal{P}^*)$ that causes event (7) to occur, we construct an adversary pair $(\mathbf{A}^*, \mathbf{P}^*)$ against SLE of $\mathsf{MPCitH\text{-}IOP}_R$.

1. $\mathbf{A}^*(1^\lambda)$ invokes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^\lambda)$ and forwards $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}^*$.
2. On receiving x from $\mathcal{A}^*$, $\mathbf{A}^*$ forwards x to the IOP-SLE game.
3. $\mathbf{P}^*(x)$ runs $\mathcal{P}^*$ on input x and plays an MPCitH-VE verifier $\mathcal{V}$.
4. On receiving $(\mathsf{com}_i)_{i\in[N]}$ from $\mathcal{P}^*$, $\mathbf{P}^*$ extracts the views by invoking $\hat{V}_i = \mathsf{CExt}(\mathsf{sk}, \mathsf{com}_i)$ for $i \in [N]$. Then $\mathbf{P}^*$ constructs the proof string $\pi = (\hat{V}_i)_{i\in[N]}$ and outputs in the SLE-IOP game.
5. On receiving an oracle query $\mathbf{e}$ from $\mathbf{V}$ in the SLE-IOP game, $\mathbf{P}^*$ forwards $\mathbf{e}$ to $\mathcal{P}^*$.

Since the canonical extractor $\mathbf{E}$ for $\mathsf{MPCitH\text{-}IOP}_R$ (see Definition 8) also constructs a witness candidate via $w = \sum_{i\in[N]} \mathsf{GetW}(\hat{V}_i) = \sum_{i\in[N]} \hat{w}_i$, it must be that $\mathbf{E}$ fails to extract a valid witness if and only if the receiver $\mathcal{R}$ fails to decrypt correctly, i.e.,

$$(7) = \Pr\left[\mathsf{CheckView}(x,(\hat{V}_i)_{i\in\mathbf{e}}) = 1 \wedge (x, \mathbf{E}(x,(\hat{V}_i)_{i\in[N]}) \notin R\right] \leq \epsilon_{\mathsf{sle\text{-}iop}} \tag{8}$$

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

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

Moreover, following [KKW18], in case the MPC protocol Π_f relies on a broadcast channel and thus $N-1$ out of N vies are revealed, we can decouple broadcast messages $(\mathsf{msgs}_i)_{i\in[N]}$ from per-party views to reduce the communication complexity, where each msgs_i consists of messages broadcast by party i. That is, the prover $\mathcal{P}$ first generates a root seed sd^* to derive per-party seeds $(\mathsf{sd}_i)_{i\in[N]}$ with a binary tree construction. $\mathcal{P}$ now only commits to each seed sd_i used for deriving a witness share and a random

tape of party i using ECOM, and sends $h = \mathsf{H}((\mathsf{msgs}_i)_{i\in[N]})$. On receiving challenge $\bar{i} \in [N]$ from $\mathcal{V}$, indicating the index of unopened party, $\mathcal{P}$ reveals $\mathsf{msgs}_{\bar{i}}$ and $\lceil \log_2(N) \rceil$ nodes in the tree, which are sufficient to compute $(\mathsf{sd}_i)_{i\in[N]\setminus\{\bar{i}\}}$. From such information, $\mathcal{V}$ can reconstruct the remaining broadcast messages, check h against broadcast messages sent by all N parties, and check that $N-1$ parties on input $(\mathsf{sd}_i)_{i\in[N]\setminus\{\bar{i}\}}$ lead to a correct output "1" with respect to x and $\mathsf{msgs}_{\bar{i}}$.

Our DKG-in-the-head protocol in Section 5.1 benefits from these optimizations.

3.3 Compiling Banquet and KKW

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

To compile Banquet-IOP, it is sufficient to have the VE prover $\mathcal{P}$ commit to the per-party seeds $(\mathsf{sd}_i)_{i\in[N]}$ with an extractable commitment scheme during the first round. The second and third round operations are identical to the original Banquet-IOP protocol, and the VE verifier $\mathcal{V}$ proceeds by following the decision phase of Banquet-IOP and accepts iff $\mathbf{V}$ 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 witness offset Δw is added by $\mathcal{C}$ when creating a partially reconstructed witness $\tilde{w}$. Since the receiver tries to decrypt by using the SLE extractor algorithm defined in Lemma 6, the compiled protocol has ϵ_{val}-validity with $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{cext}} + \epsilon_{\mathsf{sle}}$, assuming ϵ_{cext}-extractability of ECOM and ϵ_{sle}-SLE of Banquet-IOP.

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

3.4 Applying Fiat–Shamir

Following the standard Fiat–Shamir transform [FS87], we can make our verifiable encryption protocol MPCitH-VE non-interactive in the random oracle model, by hashing the first prover messages together with x and pk to obtain challenge $\mathbf{e} \in \mathsf{Ch}$. Since the base interactive protocol has three rounds, the FS transform introduces a multiplicative factor of q security loss in validity, where q is the number of random oracle queries made by a non-interactive cheating prover. Note that this loss is well-known in (knowledge) soundness analysis for FS-NIZK and EUF-KOA security of signatures constructed from canonical identification schemes [KMP16]. Banquet-based verifiable encryption however requires a separate concrete analysis dedicated to the non-interactive version, since it has 7 rounds of interaction. Because the EUF-KOA security analysis of Banquet as a signature scheme [BdK+21, Theorem 2] already evaluates the probability that the witness (i.e., secret signing key) extraction fails, we expect their analysis can be reused in large part to derive the concrete validity error of non-interactive Banquet-VE, although a formal analysis is left for a future version of this work.

4 Methods for Compressing Ciphertexts

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

In this section we give two methods to compress the verfiable encryption ciphertexts output by schemes created with our transform. The first, called the random subset method, is very simple, incurs no computational overhead, and can reduce ciphertext size by a factor of three when τ is large. The second approach, the equality proof method, is optimal as it acheives 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 practial construction. We highlight improving compression as an interesting direction for future work.

4.1 The random subset method

Protocol 3: Random subset ciphertext compression

A description of our random subset ciphertext compression method. The differences with MPCitH-VE$_R$ are highlighted. $\mathcal{G}$, $\mathcal{P}$, and $\mathcal{V}$ are unchanged.

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

1. It samples a subset $S \subset [\tau]$ of size $n < \tau$ uniformly at random.
2. For $j \in S$, it extracts witness shares $w_i^{(j)} = \mathsf{GetW}(V_i^{(j)})$ for $i \in \mathbf{e}^{(j)}$ and computes a partially reconstructed witness $\tilde{w}^{(j)} = \sum_{i \in \mathbf{e}^{(j)}} w_i^{(j)}$.
3. It outputs a compressed ciphertext $C = (\tilde{w}^{(j)}, (\mathsf{com}_i^{(j)})_{i \in \mathbf{e}^{(j)}})_{j \in S}$.

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

1. For $j \in S$ and $i \notin \mathbf{e}^{(j)}$, it extracts the unopened parties' views $\hat{V}_i^{(j)} = \mathsf{CExt}(\mathsf{sk}, \mathsf{com}_i^{(j)})$ and computes the corresponding witness shares $\hat{w}_i^{(j)} = \mathsf{GetW}(\hat{V}_i^{(j)})$. Let $w^{(j)} = \tilde{w}^{(j)} + \sum_{i \notin \mathbf{e}^{(j)}} \hat{w}_i^{(j)}$ be jth candidate witness.
2. If there exits some $j \in S$ such that $(x, w^{(j)}) \in R$, it outputs $w^{(j)}$. Otherwise, it outputs $\perp$.

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. Protocol 3 formally describes optimized compression and receiver algorithms. Upon receiving a verifiable encryption proof with our transform, the verifier has a set of τ ciphertext components, corresponding to the τ parallel repetitions used to produce the proof. The verifier chooses a subset of the ciphertexts to keep at random, and discards the others. The size of the subset is denoted n, and is a parameter of the method.

We stress that soundness of the proof is unchanged, since the entire is proof is communicated to the verifier. Only the analysis of the validity error must be updated, since the receiver now has only n ciphertexts.

Let s be the number of ciphertexts in the initial set of size τ that are bad, meaning they do not decrypt to the witness. For the proof systems we consider, having $s > 0$ is quite easy, as it only requires guessing a small part of the challenge. Note that s must be at least n, otherwise the attack against compression never succeeds, since 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

$$\begin{aligned}
&\Pr\left[\mathcal{C} \text{ selects } n \text{ of } s \text{ bad ciphertexts } \wedge\ \mathcal{P}^* \text{ convinces } \mathcal{V} \text{ while creating a proof with } s \text{ bad ciphertexts}\right] \\
=&\frac{\#\text{subsets with } n \text{ bad ciphertexts}}{\#\text{ of subsets}} \cdot \Pr\left[\mathcal{P}^* \text{ convinces } \mathcal{V} \text{ while creating a proof with } s \text{ bad ciphertexts}\right] \\
=&\frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot (\epsilon_{\mathsf{sle\text{-}iop}}(s) + \epsilon_{\mathsf{cext}})
\end{aligned}$$

where $\epsilon_{\mathsf{sle\text{-}iop}}(s)$ is the probability that an IOP prover wins the SLE-IOP game with s parallel repetitions.

Formally, the random subset method updates the validity error of Theorem 1 as follows.

Theorem 2. *Let* MPCitH-IOP$_R$ *be an MPC-in-the-head-based IOP in the class described by Protocol 1 with SLE knowledge error* $\epsilon_{\mathsf{sle\text{-}iop}}$. *Let* ECOM *be an extractable commitment scheme with* ϵ_{cext}*-extractability and* ϵ_{hide}*-hiding. If the compression algorithm* $\mathcal{C}$ *and the receiver* $\mathcal{R}$ *of* MPCitH-VE$_R$ *are replaced with Protocol 3, then* MPCitH-VE$_R$ *is* ϵ_{val}*-valid with validity error*

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

Proof is deferred to Appendix D.

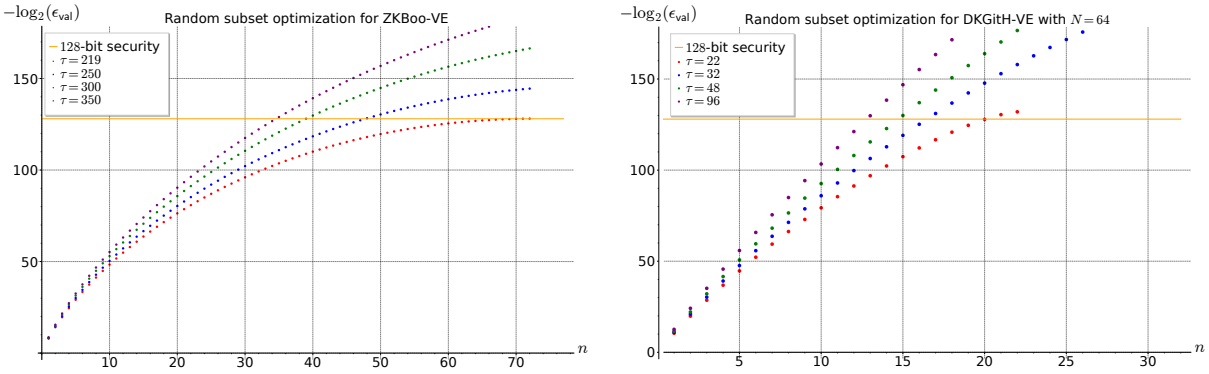

Fig. 2: Approximate minimum cost of breaking validity of ZKBoo-based VE (left) and DKGitH-based VE (right) with a random subset of size n. The parameter τ denotes the number of parallel repetitions. The number of parties N is fixed to 3 for ZKBoo and 64 for DKGitH, respectively. Note that $\tau = 219$ corresponds to the `picnic-L1` parameter from the Picnic spec [ZCD+20].

4.1.1 Application to IKOS/ZKBoo/ZKB++

We consider interactive IKOS-style protocols, such as ZKBoo and ZKB++. For each repetition of the protocol, they have $\binom{N}{2}$-consistency, where N is the number of parties. As ZKBoo and ZKB++ have $N = 3$ and $\mathsf{Ch} = \{1, 2, 3\}$ they have 3-consistency and thus are SLE with knowledge error $\epsilon_{\text{sle-iop}}(s) \leq 2/3$ from Lemma 2. Hence, if the verifier outputs all τ ciphertexts, the validity error is $\epsilon_{\text{val}}(\tau) \leq (2/3)^\tau + \epsilon_{\text{cext}}$. If the random subset optimization is applied, however, this will give cheating provers an extra strategy to break validity: by breaking soundness only in s executions and performing the remaining $\tau - s$ runs honestly using the genuine witness, the receiver in the validity game still fails to obtain the right witness if the subset of size n is only selected from s "bad" instances. Hence, now the validity error can be calculated as

$$\epsilon_{\text{val}}(\tau, n) = \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\left(\frac{2}{3} \right)^s + \epsilon_{\text{cext}} \right).$$

In Fig. 2 we show the costs of breaking validity for different combinations of τ and n assuming $\epsilon_{\text{cext}} = 0$.

4.1.2 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 $1 - 1/\binom{N}{2}$. Hence, for τ parallel repetitions we have

$$\epsilon_{\text{val}}(\tau, n, N) = \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \left(\left(\frac{1}{N} \right)^s + \epsilon_{\text{cext}} \right). \tag{9}$$

In Fig. 2 we show the costs of breaking validity for different combinations of τ and n assuming $\epsilon_{\text{cext}} = 0$.

4.2 The equality proof method

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

In an honestly generated proof, all component ciphertexts are valid, and decryption will always succeed on the first attempt. Therefore, our transform produces VE ciphertexts that are very redundant. If after the VE protocol, the prover were able to additionally prove that the receiver would produce 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 the receiver function outputs the same value for any component ciphertext – *and is not requiring that we prove the relation.* The crux of the receiver function for MPCitH protocols is recombining additive shares of the witness, a comparatively simple operation.

However one of the shares is encrypted, meaning we are back to proving something about encrypted data. We describe one instantiation of the idea to show that this is possible without resorting to general methods, by using PKE in a non-black-box way.

Theorem 3. *Let Π be an MPCitH-based IOP in the class given by Protocol 1 with $t = N - 1$, for a relation R where $|w| = \lambda$. Then there exists a VE scheme Π' with a compression algorithm that produces $O(\lambda)$ ciphertexts for Π', assuming Paillier's encryption scheme is IND-CPA secure.*

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

Next, Π' is instantiated with extractable commitments constructed from the Paillier encryption scheme. Paillier is IND-CPA secure under the decisional composite residuosity assumption [Pai99], and encryption is perfectly correct, so it is a secure commitment scheme by Lemma 1. Further, each bit of the witness share is encrypted separately which will allow bitwise operations using the homomorphic properties of Paillier encryption.

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

1. Run the compression algorithm $\mathcal{C}$ from Protocol 2, to get $C = (\tilde{w}^{(j)}, \widehat{\mathsf{com}}^{(j)})_{j \in [\tau]}$, where $\widehat{\mathsf{com}}^{(j)}$ is the unopened commitment for jth parallel execution. Since $t = N - 1$ there is only one commitment per parallel repetition.
2. Recall that $\widehat{\mathsf{com}}^{(j)} = \mathsf{Enc}(\mathsf{pk}, \hat{w}^{(j)})$ and $w = \hat{w}^{(j)} \oplus \tilde{w}^{(j)}$. Using the additive homomomorphic property of encryption, compute $C' = (\mathsf{Enc}(\hat{w}^{(1)} \oplus \tilde{w}^{(1)}), \ldots, \mathsf{Enc}(\hat{w}^{(\tau)} \oplus \tilde{w}^{(\tau)}))$ as described in Appendix A. This is possible because $\tilde{w}^{(j)}$ are public constants, and there is only one unopened party, so we only need to compute the XOR of one public and one encrypted value.
3. Convert the set C' of bitwise encryptions of w, to the set C'' of encryptions of w as described in Appendix A. This is again possible using the homomorphic property, by computing $w = \sum_{i=0}^{\lambda} 2^i w_i$ (converting from binary to integer), and choosing paramters such that λ-bit strings fit in the plaintext space of Enc.
4. Prove all ciphertexts in C'' have the same plaintext. This step can be realized, e.g., with a standard generalization of Schnorr's proof of knowledge of a discrete logarithm (details in Appendix A). A non-interactive equality proof π is output by $\mathcal{P}$ and sent to $\mathcal{V}$.

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

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

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

The construction has drawbacks that keep it from being practical, and it would interesting to address them. Because we require some (relatively weak) homomorphic properties, we lose the flexibility in the choice of PKE, and do not know of a PQ-secure instantiation short of FHE. Then the cost of creating and communicating of the proof soars because we require bitwise encryptions of witness shares, meaning the prover must compute $O(\tau N \lambda)$ individual Paillier encryptions. In practice this cost could be significantly reduced by using binary Elgamal, however then the final ciphertext would have to remain in the bitwise

representation (to allow efficient decryption) meaning the ciphertext would have size $O(\lambda^2)$, rather than $O(\lambda)$.

5 Concrete Instantiations

5.1 Verifiable encryption of discrete logarithms in any prime order group

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 is amenable to our transform, and can therefore be used to verifiably encrypt discrete logs.

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.

Related Work As a baseline for comparison, we use the Camenisch-Damgård protocol [CD00] for a generic Σ protocol, combined with Schnorr's Σ-protocol [Sch91] for discrete logs with binary challenges, to verifiably encrypt a discrete logarithm. This 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] paper has k parallel repetitions, and the verifier selects a subset to form the output, and audits the encryption step of the $k-u$ other repetitions (and the verifier checks all repetitions have a valid transcript for the Σ protocol with one challenge). No parameters are given for concerete, non-interactive security – we found that for λ-bit security, (k, u) must be chosen so that $\binom{k}{u} \geq 2^\lambda$. Then there are multiple possible choices for (k, u), which trade ciphertext size for computation: we can have a small decrease in ciphertext size, for a large increase in comptuation 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 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 as well. The scheme is not suitable for encrypting an ECDSA private key, one of our motivating examples.

5.1.1 The proof protocol: DKG-in-the-head

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

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

The prover 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 selects one party to remain unopened, having index $\bar{i}$. In the response, the prover sends the views of the other $N-1$ parties, along with $y_{\bar{i}}$, and a commitment to $x_{\bar{i}}$. Based on the revealed values, the verifier $\mathcal{V}$ 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.

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

Along with the core idea, the full protocol in Protocol 4 uses two ideas (originating in [KKW18]) that are now standard in protocols of this type. First, the shares of the parties are computed by reading random values from their tapes, and the first share is corrected with an auxiliary value that depends on the secret. Second, the tapes are derived from a per-iteration seed with a binary tree construction, so that the $N-1$ revealed seeds can be opened by revealing $\lceil \log_2(N) \rceil$ seeds.

Remark 5. Although Protocol 4 already incorporates the optimizations described in Section 3.2.1, the underlying MPCitH-IOP protocol instantiated with Π_f does satisfy the requirements from Section 2.5 so that our general compiler theorem applies: the challenge space is $\mathsf{Ch} = \left\{\mathbf{e} \subset [N] \ : \ |\mathbf{e}| = N-1\right\}$; the party i's view V_i consists of $(x_i, (y_{i'})_{i' \neq i})$; the function $\mathsf{GetW}(V_i)$ outputs x_i; the function $\mathsf{CheckView}(y, (V_i)_{i \in \mathbf{e}})$ parses V_i as $(x_i, (y_{i',i})_{i' \neq i})$ and checks $y \stackrel{?}{=} g^{x_i} \prod_{i' \neq i} y_{i',i}$ for all $i \in \mathbf{e}$ and $y_{i',i} \stackrel{?}{=} g^{x_{i'}}$ for $i \in \mathbf{e}$ and $i' \in [N] \setminus \{i, \bar{i}\}$, where $\bar{i} \notin \mathbf{e}$ is an index of the unopened party. Proving 2-consistency (Definition 7) and thus SLE with $\epsilon_{\mathsf{sle\text{-}iop}} = 1/N$ (Lemma 2) is straightforward. Showing 1. from 2. is trivial. To show the converse, let $\mathbf{e}, \mathbf{e}'$ be two distinct challenges. If $\mathsf{CheckView}$ outputs 1 w.r.t. both challenges, then for some i such that $i \in \mathbf{e} \cap \mathbf{e}'$, it must be that $y = g^{x_i} \prod_{i' \neq i} g^{x_{i'}} = g^{x_1 + \ldots + x_N}$. Hence, $(V_1, \ldots, V_N)$ form an honest execution of Π_f on y and witness shares $(x_i)_{i \in [N]}$ as inputs.

Remark 6. The idea of DKG-in-the-head can be generalized to proving knowledge of a preimage under a one-way group homomorphism. For example, if the homomorphism is $\phi : m \mapsto m^e \bmod n$ with $n = p \cdot q$, one can design a simple MPCitH protocol for knowledge of an RSA preimage: the parties share m multiplicatively, $m = m_1 \cdots m_N \pmod n$ then broadcast $\phi(m_i) = m_i^e$, and then check that $c = \prod m_i^e \pmod n$.

5.1.2 Applying our transform

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

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

1. For each execution $j \in [\tau]$
   (a) Recompute $x_i^{(j)}$ as in Verify, then compute $\tilde{x}^{(j)} = \sum_{i \neq \bar{i}_j} x_i^{(j)}$.
   (b) Parse $\mathsf{com}_{\bar{i}_j}$ as the Elgamal ciphertext (c_1, c_2). Set $C^{(j)} = (c_1, c_2 + \tilde{x}^{(j)})$
2. Output $C = (C^{(j)})_{j \in [\tau]}$ as the ciphertext.

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

1. For each repetition $j \in [\tau]$,
   (a) Decrypt $C^{(j)}$ using sk to get $x' \in \mathbb{Z}_p$
   (b) If $y = g^{x'}$ output x'.

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

We now point out another optimization that reduces both proof and ciphertext sizes. For each ciphertext the prover computes, we use $y_i^{(j)} = g^{x_i^{(j)}}$ as the c_1 component of the Elgamal ciphertext. This reduces the ciphertext size in the proof to a single element of $\mathbb{Z}_p$ and saves one exponentiation, reducing the prover's total number of exponentiations by τN. The size of the output ciphertext C stays the same though, since the value $g^{x_{\bar{i}}^{(j)}}$ must be present for decryption.

Parameter choices As an interactive protocol, it's easy to see that the soundness error is $1/N$. When executing τ repetitions in parallel, we therefore need to set τ so that $(1/N)^\tau < 2^{-\lambda}$, or equivalently $\tau \log_2(N) \geq \lambda$. This offers a range of choices for (N, τ) for each choice of λ, with a different balance of proof size and running time.

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

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

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

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

Commit

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

Challenge

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

Response and output

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

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

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

Efficiency Starting with the proof, without our transform, when Commit is implemented with a hash function, the size is

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

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

Examples and comparison We give some concrete parameters showing various time-speed trade offs, and compare to related work in Table Table 1. We give three parameter sets, and estimate the size in bytes of the transcript tr, the VE ciphertext $|C|$ (both with and without the random subset (RS) optimization), as well as the computational costs of the prover and verifier. For the costs we count the number of exponentiations, and also give an estimated time in milliseconds (ms) by using the timings given in [NRSW20] (based on their benchmarks from an Intel i7-7820HQ system pinned to 2.90 GHz).

The options for our scheme offer short ciphertexts (480–640 bytes), at the expense of higher prover and verifier times, or much lower times, but with larger ciphertexts (1536 bytes) and proof sizes, or somewhere in the middle.

When comapred to [NRSW20], there the ciphertext size is a regular Elgamal ciphertext, the proof size is about 1KB, but the prover and verifier times are 943ms and 50ms respectively. Notably, the prover time is much less, about 10-24x faster with the parameters above.

We also compare to the [CD00] scheme with Schnorr's Σ-protocol for discrete logs, using the same hashed Elgamal scheme. It has proof size $\lambda + k\ell_p + (k-u)\lambda + u\ell_C$ and ciphertext Size: $u(\ell_C + \ell_p + 1)$. Prover and verifier must compute $4k$ exponentiations. In terms of proof and VE ciphertext size, our scheme always outperforms [CD00]. The running time of the prover and verifier are nearly the same.

Scheme	N	τ	n	k	u	|tr|	$\|C\|$	(RS)	$\mathcal{P}$ exp.	(ms)	$\mathcal{V}$ exp.	(ms)
DKGitH	64	48	15			7744	1536	(480)	6144	(239.62)	6048	(235.87)
	85	20	20			3584	640	(640)	3400	(132.60)	3360	(131.04)
	16	32	30			4160	1024	(960)	1024	(39.94)	960	(37.44)
	4	64	48			6208	2048	(1536)	512	(19.97)	384	(14.98)
[CD00]		712	20			35100	1922		2880	(112.32)	2880	(112.32)
		250	30			13500	2884		1000	(39.00)	1000	(39.00)
		132	64			9424	6152		528	(20.59)	528	(20.59)
[NRSW20]						1100	64		24823	(968.10)	1316	(51.32)

Table 1: Parameters and performance estimates for veriable encryption of a discrete logarithm. Our scheme is in the first part of the table, followed by the generic scheme from [CD00] (combined with Schnorr's proof protocol [Sch91]), followed by the Purify PRF-based construction of [NRSW20]. Sizes are given in bytes. The ciphertext size for our scheme when the Random Subset Section 4.1 compresssion method is used is given in parenthesis.

5.2 Verifiable encryption of AES private keys

As a corollary of 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 relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) \ : \ \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$, one can verifiably encrypt K with some PKE satisfying the relation R. To the best of our knowledge, no prior work proposed a verifiable encryption scheme for AES private keys. Notably, if PKE is instantiated with quantum-resilient schemes, such as lattice-based ones, one can obtain post-quantum verifiable encryption (in the sense that both the *encryption scheme* and *relation* to be proven about the plaintext may withstand quantum attacks). As we analyze in Appendix C, variants of the FO transform can be used for achieving undeniability and thus many efficient post-quantum PKE schemes, including Kyber [SAB+20] and FrodoKEM [NAB+19], are indeed compatible with our framework.

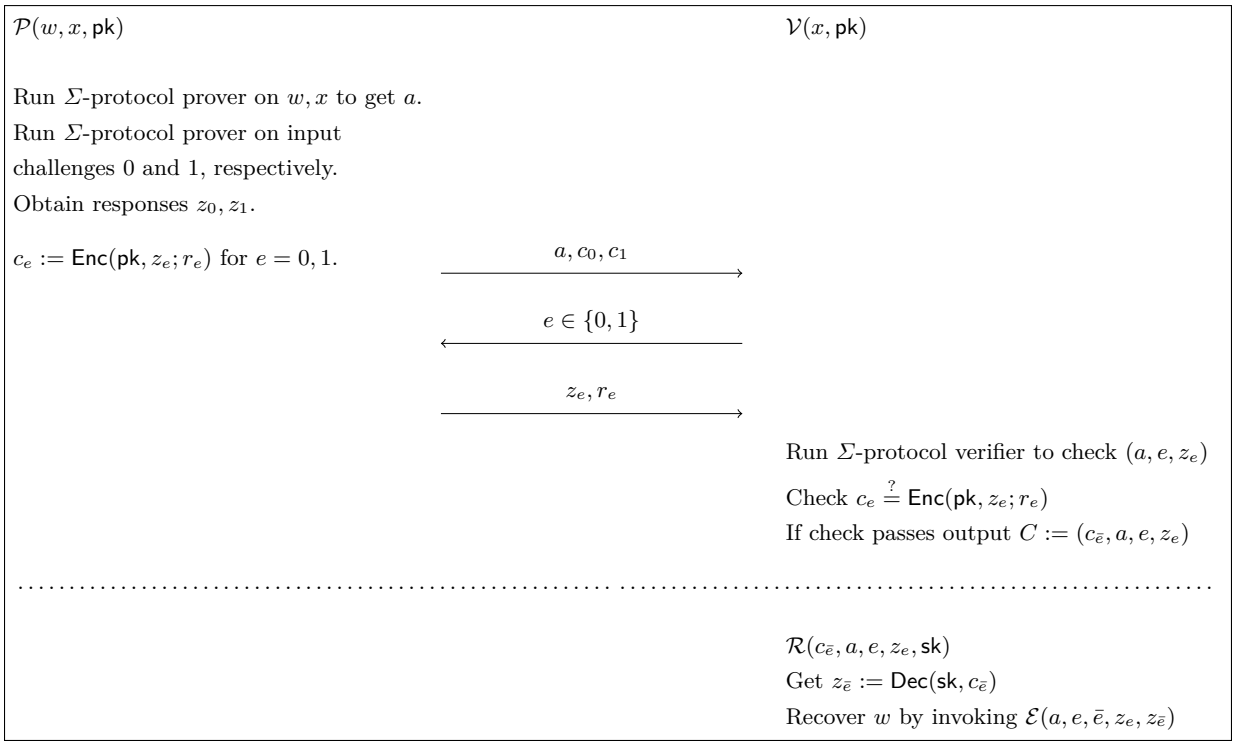

Fig. 3: Camenisch–Damgård verifiable encryption.

6 Camenisch–Damgård Verifiable Encryption with Imperfect Correctness

6.1 The Camenisch–Damgård framework [CD00]

Σ-protocol A Σ-protocol for relation R is an interactive proof system consisting of three rounds. In a Σ-protocol, the prover sends a message a, the verifier replies with a random bit string e, and the prover responds with z. The verifier decides to accept or reject based on the transcript (a, e, z). A Σ-protocol can be efficiently compiled into a non-interactive zero-knowledge proof of knowledge (in the random oracle model) through the Fiat-Shamir transform [FS87]. The usual requirements for a Σ-protocol are *special soundness* and *honest verifier zero knowledge*. In particular, special soundness implies existence of an efficient extractor $\mathcal{E}$ that outputs a valid witness, given two accepting transcripts (a, e, z) and (a, e', z') such that $e \neq e'$.

The transform See Fig. 3. We assume that there exists a Σ-protocol with one-bit challenge for relation R. On a high-level, the receiver $\mathcal{R}$ can obtain a witness by first decrypting the unopened response and then by invoking the extractor $\mathcal{E}$ of the underlying Σ-protocol.

6.2 Committing encryption is required for [CD00]

In the analysis of the VE scheme in Fig. 3, Camenisch and Damgård assume only that Enc is semantically secure, which means that ciphertexts are indistinguishable against chosen plaintext attacks (also called IND-CPA security, or CPA security for short). We first note that CPA security does not imply that an encryption scheme is committing *if its correctness is not perfect*, so the analysis of [CD00] suggests that this property is not required for the security of their VE scheme.

We provide two counterexamples to the security analysis of [CD00]. An example with LWE-based encryption is deferred to Appendix B. We describe encryption schemes that are semantically secure, and non-committing, in a way that allows a malicious prover, who knows the witness, to compute a proof and ciphertext guaranteed to decrypt to junk, rather than the witness, breaking the validity property of Definition 3.

Here we present the scheme that is essentially an instance of the basic construction of [CDNO97], and is much simpler than a fully deniable encryption scheme, as we only need the prover to be able to open ciphertexts by flipping bits in one direction. In our attack the unopened/junk encryptions are encyrptions of the all-ones string, and when we open one to a specific value, we must flip some of the ones to zero, but

we never need to change a zero to a one bit. In a more general deniable encryption scheme we would need both one-to-zero and zero-to-one, in order to open one arbitrary message to another arbitrary message.

6.2.1 A secure encryption scheme that is not a commitment We describe the encryption scheme now. Let λ be a security parameter and for an RSA modulus n, let $\mathbb{QR}_n$ be the set of quadratic residues in $\mathbb{Z}_n^*$.

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

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

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

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

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

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

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

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

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

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

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

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

6.2.2 An Attack on [CD00] verifiable encryption We now give a simple attack on the Camensich–Damgård verifiable encryption scheme, given in Fig. 3, when the recipient's public key encryption algorithm is our semantically secure encryption scheme given above.

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

[5] For simplicity we assume here that all z_e values are encoded to have the same length.

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

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

6.3 Fixing the [CD00] security analysis

By additionally assuming *undeniability* of PKE (Definition 10) we can prove validity of the scheme described in Fig. 3. Recall that a cheating prover $\mathcal{P}^*$ wins the validity game if the receiver $\mathcal{R}$ failed to decrypt a witness while the verifier $\mathcal{V}$ accepts. Similar to the validity analysis of Theorem 1, we consider two cases: (1) $z_e \neq \mathsf{Dec}(\mathsf{sk}, C_e)$ while $c_e = \mathsf{Enc}(\mathsf{pk}, z_e; r_e)$ and (2) $z_e = \mathsf{Dec}(\mathsf{sk}, C_e)$ while (a, e, z_e) is an accepting transcript, where the challenge bit e is chosen uniformly. In the former case, one can break undeniability of PKE using a cheating prover $\mathcal{P}^*$. In the latter case, due to special soundness, if the extractor $\mathcal{E}$ (internally invoked by $\mathcal{R}$) fails to obtain a valid witness, it must be that the unopened response $z_{\bar{e}} = \mathsf{Dec}(\mathsf{sk}, c_{\bar{e}})$ is non-accepting w.r.t. $(a, \bar{e})$. Since z_e and $z_{\bar{e}}$ are determined *before* a cheating prover $\mathcal{P}^*$ gets to see the challenge e, the probability that $\mathcal{P}^*$ can correctly guess e is at most 1/2. Overall, the validity error is $\epsilon_{\mathsf{val}} = \epsilon_{\mathsf{cext}} + 1/2$.

Acknowledgment

The authors are grateful to Ivan Damgård, Bernardo David, and Claudio Orlandi for helpful comments and insightful discussions.

References

AHIV17. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 2017*, pp. 2087–2104. ACM Press, 2017.

ASW98. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT'98*, vol. 1403 of *LNCS*, pp. 591–606. Springer, Heidelberg, 1998.

Ate99. G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *ACM CCS 99*, pp. 138–146. ACM Press, 1999.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pp. 315–334. IEEE Computer Society Press, 2018.

BCP03. E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT 2003*, vol. 2894 of *LNCS*, pp. 37–54. Springer, Heidelberg, 2003.

BCR+19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, pp. 103–128. Springer, Heidelberg, 2019.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, vol. 9986 of *LNCS*, pp. 31–60. Springer, Heidelberg, 2016.

Bd20. W. Beullens and C. de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pp. 130–150. Springer, Heidelberg, 2020.

BDD20. C. Baum, B. David, and R. Dowsley. (public) verifiability for composable protocols without adaptivity or zero-knowledge. Cryptology ePrint Archive, Report 2020/207, 2020. https://ia.cr/2020/207.

BdK+21. C. Baum, C. de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha. Banquet: Short and fast signatures from AES. In *PKC 2021, Part I*, vol. 12710 of *LNCS*, pp. 266–297. Springer, Heidelberg, 2021.

Beu20. W. Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In *EUROCRYPT 2020, Part III*, vol. 12107 of *LNCS*, pp. 183–211. Springer, Heidelberg, 2020.

BFH+20. R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang. Ligero++: A new optimized sublinear IOP. In *ACM CCS 2020*, pp. 2025–2038. ACM Press, 2020.

[6] Theorem 2 in the full version of the paper, the BRICS technical report

BG10. Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO 2010*, vol. 6223 of *LNCS*, pp. 1–20. Springer, Heidelberg, 2010.

BN20. C. Baum and A. Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In *PKC 2020, Part I*, vol. 12110 of *LNCS*, pp. 495–526. Springer, Heidelberg, 2020.

BS20. D. Boneh and V. Shoup. A graduate course in applied cryptography, 2020. Available online https://crypto.stanford.edu/~dabo/cryptobook/.

CCFG16. P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In *ACM CCS 2016*, pp. 1614–1625. ACM Press, 2016.

CD00. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 331–345. Springer, Heidelberg, 2000.

CDG+17. M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS 2017*, pp. 1825–1842. ACM Press, 2017.

CDNO97. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *CRYPTO'97*, vol. 1294 of *LNCS*, pp. 90–104. Springer, Heidelberg, 1997.

CF85. J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th FOCS*, pp. 372–382. IEEE Computer Society Press, 1985.

CFF+20. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. Cryptology ePrint Archive, Report 2020/1069, 2020. https://eprint.iacr.org/2020/1069.

CFQ19. M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In *ACM CCS 2019*, pp. 2075–2092. ACM Press, 2019.

CHM+20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 738–768. Springer, Heidelberg, 2020.

CS03. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 126–144. Springer, Heidelberg, 2003.

dDOS19. C. de Saint Guilhem, L. De Meyer, E. Orsini, and N. P. Smart. BBQ: Using AES in picnic signatures. In *SAC 2019*, vol. 11959 of *LNCS*, pp. 669–692. Springer, Heidelberg, 2019.

DFMS21. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280, 2021. https://ia.cr/2021/280.

DGRW18. Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage. Fast message franking: From invisible salamanders to encryptment. In *CRYPTO 2018, Part I*, vol. 10991 of *LNCS*, pp. 155–186. Springer, Heidelberg, 2018.

DJ01. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001*, vol. 1992 of *LNCS*, pp. 119–136. Springer, Heidelberg, 2001.

dSGOT21. C. D. de Saint Guilhem, E. Orsini, and T. Tanguy. Limbo: Efficient zero-knowledge mpcith-based arguments. Cryptology ePrint Archive, Report 2021/215, 2021. https://ia.cr/2021/215.

EG2. Electionguard specification. Available at https://www.electionguard.vote/, year = 2021.

Fis05. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, vol. 3621 of *LNCS*, pp. 152–168. Springer, Heidelberg, 2005.

FO99. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC'99*, vol. 1560 of *LNCS*, pp. 53–68. Springer, Heidelberg, 1999.

FO13. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, 2013.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 186–194. Springer, Heidelberg, 1987.

GCZ16. S. Goldfeder, M. Chase, and G. Zaverucha. Efficient post-quantum zero-knowledge and signatures. Cryptology ePrint Archive, Report 2016/1110, 2016. https://eprint.iacr.org/2016/1110.

GH03. Y. Gertner and A. Herzberg. Committing encryption and publicly-verifiable signcryption. Cryptology ePrint Archive, Report 2003/254, 2003. https://eprint.iacr.org/2003/254.

GHM+21. K. Gjøsteen, T. Haines, J. Müller, P. Rønne, and T. Silde. Verifiable decryption in the head. Cryptology ePrint Archive, Report 2021/558, 2021. https://ia.cr/2021/558.

GLR17. P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. In *CRYPTO 2017, Part III*, vol. 10403 of *LNCS*, pp. 66–97. Springer, Heidelberg, 2017.

GMO16. I. Giacomelli, J. Madsen, and C. Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In *USENIX Security 2016*, pp. 1069–1083. USENIX Association, 2016.

GMP21. P. Grubbs, V. Maram, and K. G. Patterson. Anonymous, robust post-quantum public key encryption. NIST Third PQC Standardization Conference, 2021. https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/maram-anonymous-robust-pqc2021.pdf.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, vol. 263 of *LNCS*, pp. 171–185. Springer, Heidelberg, 1987.

GOS06. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 339–358. Springer, Heidelberg, 2006.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pp. 197–206. ACM Press, 2008.

Gro05. J. Groth. Cryptography in subgroups of zn. In *TCC 2005*, vol. 3378 of *LNCS*, pp. 50–65. Springer, Heidelberg, 2005.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 305–326. Springer, Heidelberg, 2016.

HHK17. D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *TCC 2017, Part I*, vol. 10677 of *LNCS*, pp. 341–371. Springer, Heidelberg, 2017.

HLR21. J. Holmgren, A. Lombardi, and R. D. Rothblum. Fiat-shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pp. 750–760. ACM, 2021.

IKOS07. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *39th ACM STOC*, pp. 21–30. ACM Press, 2007.

Kat21. S. Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In *CRYPTO 2021, Part II*, vol. 12826 of *LNCS*, pp. 580–610, Virtual Event, 2021. Springer, Heidelberg.

KKW18. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pp. 525–537. ACM Press, 2018.

KMP16. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO 2016, Part II*, vol. 9815 of *LNCS*, pp. 33–61. Springer, Heidelberg, 2016.

LCKO19. J. Lee, J. Choi, J. Kim, and H. 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. V. Lyubashevsky and G. Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT 2017, Part I*, vol. 10210 of *LNCS*, pp. 293–323. Springer, Heidelberg, 2017.

LPR10. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 1–23. Springer, Heidelberg, 2010.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, vol. 7881 of *LNCS*, pp. 35–54. Springer, Heidelberg, 2013.

Lyu20. V. Lyubashevsky. Basic lattice cryptography: Encryption and fiat-shamir signatures. Manuscript, 2020. https://drive.google.com/file/d/1JTdW5ryznp-dUBBjN12QbvWz9R41NDGU/view?usp=sharing.

MBKM19. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, pp. 2111–2128. ACM Press, 2019.

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

NAB+19. M. Naehrig, E. Alkim, J. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

NRSW20. J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In *ACM CCS 2020*, pp. 1717–1731. ACM Press, 2020.

NS98. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *ACM CCS 98*, pp. 59–66. ACM Press, 1998.

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, vol. 1592 of *LNCS*, pp. 223–238. Springer, Heidelberg, 1999.

Pas03. R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 316–337. Springer, Heidelberg, 2003.

Ped92. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, vol. 576 of *LNCS*, pp. 129–140. Springer, Heidelberg, 1992.

Poi00. D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC 2000*, vol. 1751 of *LNCS*, pp. 129–146. Springer, Heidelberg, 2000.

PS00. G. Poupard and J. Stern. Fair encryption of RSA keys. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 172–189. Springer, Heidelberg, 2000.

PVW08. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, vol. 5157 of *LNCS*, pp. 554–571. Springer, Heidelberg, 2008.

Reg05. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pp. 84–93. ACM Press, 2005.

SAB+20. P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

Sch91. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

Sta96. M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT'96*, vol. 1070 of *LNCS*, pp. 190–199. Springer, Heidelberg, 1996.

SW14. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *46th ACM STOC*, pp. 475–484. ACM Press, 2014.

Unr15. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, pp. 755–784. Springer, Heidelberg, 2015.

YY98. A. Young and M. Yung. Auto-recoverable auto-certifiable cryptosystems. In *EUROCRYPT'98*, vol. 1403 of *LNCS*, pp. 17–31. Springer, Heidelberg, 1998.

ZCD+20. G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

A Bitwise Commitment and Encryption

Our equality proof method (Section 4.2) to compress ciphertexts requires an extractable commitment scheme where the messages are bitstrings, and the commitments must be (somewhat) homomorphic with respect to the XOR operation. For a comitted value $c = \mathsf{Commit}(a)$ and a public value b, where a and b are both bitstrings, we must be able to compute $c' = \mathsf{Commit}(a \oplus b)$. In Section 4.2 our construction uses Paillier encryption, but what we describe here also applies to Elgamal encryption and can be adapted to Pedersen commitments. The setup is a group $\mathbb{G}$ with generator g and public key h. Encryption is defined as follows.

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

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

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

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

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

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

B.1 LWE encryption scheme

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

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

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

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

$$\mathbf{u}^T = \mathbf{r}^T \mathbf{A} + \mathbf{e}_2^T \mod q$$
$$v = \mathbf{r}^T \mathbf{t} + e_3 + \frac{q}{2} \cdot \mu \mod q$$

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

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

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

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

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

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

B.2 Attacks breaking undeniability

We present three scenarios, depending on the adversarial power.

- **Case 1. Adversary generates a key pair (i.e. breaking strong undeniability)** If the adversary has control over key generation then the attack is straightforward: by setting $\mathbf{s} = (-\beta, \ldots, -\beta)$ and $\mathbf{e}_1 = (\beta, \ldots, \beta)$, they encrypt $\mu = 1$ with randomness $\mathbf{r} = \mathbf{e}_2 = (\beta, \ldots, \beta)$ and $e_3 = \beta$. Clearly, the resulting ciphertext $(\mathbf{u}, v)$ decrypts to 0, since the noise term $\mathbf{r}^T\mathbf{e}_1 + e_3 - \mathbf{e}_2^T\mathbf{s}$ exceeds $q/4$ for the example parameter $(q, m, \beta) = (3329, 4096, 1)$.
- **Case 2. Key pair is generated honestly, but the adversary knows the decryption key (i.e. breaking undeniability as defined in Definition 10)** Even if a key pair is generated honestly, one may observe that the attack succeeds if the adversary sees the secret decryption key $(\mathbf{s}, \mathbf{e}_1)$ as in Definition 10. To maximize the norm of noise term, the adversary looks at each element of $\mathbf{e}_1$ and adaptively choose the corresponding position of $\mathbf{r}^T$. That is, $r_i = \beta$ if $\mathbf{e}_{1,i}$ is positive, and $r_i = -\beta$ otherwise. The $\mathbf{e}_2$ is chosen in the same fashion and e_3 doesn't matter as it has little impact on the resulting norm. For the example parameter $(q, m, \beta) = (3329, 4096, 1)$ this strategy succeeds with overwhelming probability (where the probability is taken over random coins used in key generation).
- **Case 3. Key pair is generated honestly, and the adversary only receives the public key (i.e. breaking a variant of undeniability weaker than Definition 10)** Even if the adversary does not get to see the decryption key, which may be the case in some practical scenarios, one can still significantly increase the chance of decryption failure. If the adversary deterministically uses the largest possible randomness $\mathbf{r}^T = \mathbf{e}_2^T = (\beta, \ldots, \beta)$ and $e_3 = \beta$, it amounts to evaluating the following probability.

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

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

B.3 Attacking validity of [CD00]

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

Remark 7. We remark that, unlike the counterexample presented in Section 6.2, the randomness submitted by the adversary when opening the plaintext looks somewhat suspicious: because the norm of revealed randomness is often large, the verifier may be able to detect that don't follow correct (uniform) distributions over the randomness space. However, it still serves as another counterexample to the security analysis of [CD00], because it does not specify how the verifier should check randomness when

the plaintext is revealed. Applying the FO transform is one way to circumvent the issue, as we observe in Appendix C. It would be an interesting follow-up question how the verifier should impose the norm bound on revealed randomness such that the scheme retains both correctness and binding when used as a commitment.

C Undeniability and Binding of Fujisaki–Okamoto Transform

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

Motivated by this we consider simple generic constructions of undeniable encryption from any CPA-secure scheme with statistical correctness. We analyze two variants of the Fujisaki–Okamoto transform [FO99, FO13, HHK17] and prove that both provide computational undeniability in the random oracle model. Note that from Lemma 7, binding of PKE when used as a commitment scheme is implied by undeniability. Hence, the result in this section also implies these variants of the FO transform can be used to construct a secure commitment scheme, which might be of independent interest. Throughout this section, we denote the number of queries to a random oracle G by q_{G}.

C.1 PKE_1 [HHK17]

We first present a simple transform that forms the basis of both conversions. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure scheme with statistical correctness, message space $S_m = \{0,1\}^k$, and randomness space $S_r = \{0,1\}^l$. Let $\mathsf{G} : \{0,1\}^* \to S_r$ be a random oracle. Then we define a deterministic encryption scheme $\mathsf{PKE}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ as follows.

- Gen_1 is identical to Gen
- $\mathsf{Enc}_1(\mathsf{pk}, M)$ takes $M \in S_m$ and outputs $c := \mathsf{Enc}(\mathsf{pk}, M, \mathsf{G}(M))$.
- $\mathsf{Dec}_1(\mathsf{sk}, c)$ first obtains $M := \mathsf{Dec}(\mathsf{sk}, c)$, and
  - if $M' = \bot$ or $c \neq \mathsf{Enc}(\mathsf{pk}, M'; \mathsf{G}(M'))$, outputs $\bot$;
  - otherwise outputs M'.

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

C.2 PKE_2 [FO99]

As a randomized variant of the previous conversion, we define $\mathsf{PKE}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ parameterized by bitlength k_0 of randomness as follows.

- Gen_2 is identical to Gen
- $\mathsf{Enc}_2(\mathsf{pk}, m; \rho)$ takes $m \in \{0,1\}^{k-k_0}$ and $\rho \in \{0,1\}^{k_0}$ such that $m||\rho \in S_m$ and outputs $c := \mathsf{Enc}_1(\mathsf{pk}, m||\rho) = \mathsf{Enc}(\mathsf{pk}, m||\rho, \mathsf{G}(m||\rho))$.
- $\mathsf{Dec}_2(\mathsf{sk}, c)$ first obtains $M' := \mathsf{Dec}_1(\mathsf{sk}, c)$, and
  - if $M' = \bot$, outputs $\bot$;
  - otherwise parses M' as $m'||\rho'$ and outputs m'.

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

Below we prove our new result about PKE_2.

Lemma 8. *If* PKE *is* δ*-correct, then in the random oracle model,* PKE_2 *is* ϵ_{und}*-undeniable with* $\epsilon_{\mathsf{und}}(q_{\mathsf{G}}) = q_{\mathsf{G}} \cdot \delta$.

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

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ in the correctness game, $\mathcal{B}$ forwards $(\mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}$.
2. Whenever $\mathcal{A}$ makes a query to G, the $\mathcal{B}$ forwards the same query to G in the correctness game.
3. When $\mathcal{A}$ outputs (c, m, ρ) such that $c = \mathsf{Enc}_2(\mathsf{pk}, m; \rho)$ and $m \neq \mathsf{Dec}_2(\mathsf{sk}, c)$, the $\mathcal{B}$ outputs $M := m||\rho$ in the correctness game w.r.t. PKE_1.

Note that the output M of $\mathcal{B}$ satisfies $c = \mathsf{Enc}_1(\mathsf{pk}, M)$, because of how Enc_2 works internally.

We argue that, as long as $m \neq \mathsf{Dec}_2(\mathsf{sk}, c)$, we have $M \neq \mathsf{Dec}_1(\mathsf{sk}, c)$ in the correctness game w.r.t. PKE_1. There are two cases where $\mathcal{A}$ wins the undeniability game w.r.t. PKE_2: (1) $\perp = \mathsf{Dec}_1(\mathsf{sk}, c)$, and (2) $m' \neq m$, where $m'||\rho' = \mathsf{Dec}_1(\mathsf{sk}, c)$. In case (1), $\mathcal{B}$ clearly breaks correctness of PKE_1; in case (2), $\mathsf{Dec}_1(\mathsf{sk}, c)$ outputs $M' := m'||\rho' \neq M$. Hence, whenever $\mathcal{A}$ successfully breaks undeniability, $\mathcal{B}$ successfully breaks correctness of PKE_1. Overall, we have $\epsilon_{\mathsf{und}} = \delta_1 = q_{\mathsf{G}} \cdot \delta$.

□

C.3 PKE_3 [HHK17]

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

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

- Gen_3 first obtains $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_1(1^\lambda)$ and then samples a secret random seed $s \leftarrow\!\!\$\, S_m$. It outputs $(\mathsf{pk}', \mathsf{sk}') := (\mathsf{pk}, (\mathsf{sk}, s))$.
- $\mathsf{Enc}_3(\mathsf{pk}, m; \rho)$ takes $m \in \mathcal{M}_{\mathrm{sym}}$ and $\rho \in S_m$. It computes $c_1 := \mathsf{Enc}_1(\mathsf{pk}, \rho)$, $K := \mathsf{H}(\rho, c_1)$, and $c_2 := \mathsf{SEnc}(K, m)$. It outputs (c_1, c_2).
- $\mathsf{Dec}_3((\mathsf{sk}, s), (c_1, c_2))$ first obtains $\rho' := \mathsf{Dec}_1(\mathsf{sk}, c_1)$, and
  - if $\rho' = \perp$, let $K' := \mathsf{H}(s, c_1)$;
  - otherwise let $K' := \mathsf{H}(\rho', c_1)$.

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

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

Below we prove our new result about PKE_3.

Lemma 9. *If PKE_1 is δ_1-correct and SKE is perfectly correct, then in the random oracle model, PKE_3 is ϵ_{und}-undeniable with $\epsilon_{\mathsf{und}} = \delta_1$.*

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

1. On receiving $(\mathsf{pk}, \mathsf{sk})$ in the correctness game, $\mathcal{B}$ samples a random seed $s \leftarrow\!\!\$\, S_m$ and forwards $(\mathsf{pk}, (\mathsf{sk}, s))$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs $((c_1, c_2), m, \rho)$ such that $(c_1, c_2) = \mathsf{Enc}_3(\mathsf{pk}, m; \rho)$ and $m \neq \mathsf{Dec}_3((\mathsf{sk}, s), (c_1, c_2))$, the $\mathcal{B}$ outputs ρ in the correctness game w.r.t. PKE_1.

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

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

PKE_1 is broken. Hence, whenever $\mathcal{A}$ successfully breaks undeniability, $\mathcal{B}$ successfully breaks δ_1-correctness of PKE_1.

□

D Proof for Theorem 2

Proof. We extend the proof for Theorem 1. From the description of $\mathsf{MPCitH\text{-}VE}_R$ and Protocol 3, the probability fail can be parsed as follows.

$$\mathsf{fail} = \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge \forall j \in S : (x, \sum_{i\in\mathbf{e}^{(j)}} w_i^{(j)} + \sum_{i\notin\mathbf{e}^{(j)}} \hat{w}_i^{(j)}) \notin R\end{array}\right] \tag{10}$$

$$= \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge S \subset \mathsf{BadRun}\end{array}\right] \tag{11}$$

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

$$= \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s\end{array}\right] \tag{13}$$

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

Now we split (13) into the two cases, depending on whether extractability of ECOM is broken or not.

$$(13) = \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \wedge \exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)}\end{array}\right] \tag{14}$$

$$+ \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge |\mathsf{BadRun}| = s \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : V_i^{(j)} = \hat{V}_i^{(j)}\end{array}\right] \tag{15}$$

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

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

$$(14) = \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r}\forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i\in\mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge \exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)}\end{array}\middle| \; |\mathsf{BadRun}| = s\right] \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{16}$$

$$\leq \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\exists j \in \mathsf{BadRun}, \exists i \in \mathbf{e}^{(j)} : V_i^{(j)} \neq \hat{V}_i^{(j)} \;\middle|\; |\mathsf{BadRun}| = s\right] \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{17}$$

$$\leq \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \epsilon_{\mathsf{cext}} \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \leq \max_{n\leq s\leq\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \epsilon_{\mathsf{cext}} \tag{18}$$

Likewise, the latter case can be bounded as follows.

$$(15) = \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r} \forall j \in [\tau] : \mathsf{CheckView}(x, (V_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in [\tau], \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, V_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : V_i^{(j)} = \hat{V}_i^{(j)} \end{array} \middle| \, |\mathsf{BadRun}| = s \right] \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{19}$$

$$\leq \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\begin{array}{r} \forall j \in \mathsf{BadRun} : \mathsf{CheckView}(x, (\hat{V}_i^{(j)})_{i \in \mathbf{e}^{(j)}}) = 1 \\ \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : \mathsf{Commit}(\mathsf{pk}, \hat{V}_i^{(j)}; r_i^{(j)}) = \mathsf{com}_i^{(j)} \\ \wedge \forall j \in \mathsf{BadRun}, \forall i \in \mathbf{e}^{(j)} : V_i^{(j)} = \hat{V}_i^{(j)} \end{array} \middle| \, |\mathsf{BadRun}| = s \right] \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{20}$$

$$\leq \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \Pr\left[\forall j \in \mathsf{BadRun} : \mathsf{CheckView}(x, (\hat{V}_i^{(j)})_{i \in \mathbf{e}}) = 1 \wedge (x, \sum_{i \in [N]} \hat{w}_i^{(j)}) \notin R \, \middle| \, |\mathsf{BadRun}| = s \right] \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \tag{21}$$

$$\leq \sum_{s=n}^{\tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \epsilon_{\mathsf{sle\text{-}iop}}(s) \cdot \Pr\left[|\mathsf{BadRun}| = s\right] \leq \max_{n \leq s \leq \tau} \frac{\binom{s}{n}}{\binom{\tau}{n}} \cdot \epsilon_{\mathsf{sle\text{-}iop}}(s) \tag{22}$$

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

Putting (18) and (22) together, we obtain the desired bound. □

E KKW as an IOP

The underlying interactive protocol of KKW can be characterized as a single-round IOP. See Protocol 5.

Protocol 5: KKW-IOP$_R$

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

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

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

1. Choose uniform $\mathsf{sd}^{(j)}$ and use it to generate per-party seeds $(\mathsf{sd}_i^{(j)})_{i\in[N]}$. $\mathbf{P}$ computes $\mathsf{aux}^{(j)}$ by running the offline phase of MPC on input $(\mathsf{sd}_i^{(j)})_{i\in[N]}$. For each $i \in [1, N-1]$, let $\mathsf{st}_i^{(j)} = \mathsf{sd}_i^{(j)}$ and let $\mathsf{st}_N^{(j)} = \mathsf{sd}_N^{(j)} || \mathsf{aux}^{(j)}$.
2. Compute the masked witness $\hat{w} = \lambda_1^w \oplus \ldots \oplus \lambda_N^w \oplus w$, where λ_i^w is party i's random share to mask the witness in jth MPC execution, and is read out from $\mathsf{st}_i^{(j)}$.
3. Emulate "in her head" the online phase of the N-party protocol for $f(w) \stackrel{?}{=} x$ by running the online phase of MPC on input x, $\hat{w}^{(j)}$ and $(\mathsf{st}_i^{(j)})_{i\in[N]}$. As a result the prover obtains per-party broadcast messages $(\mathsf{msgs}_i^{(j)})_{i\in[N]}$.

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

Query phase

1. $\mathbf{V}$ chooses a uniformly random subset $T \subset [M]$ of size τ and party indices $(\bar{i}_j)_{j\in T}$ where each $\bar{i}_j \in [N]$ is uniform. It queries the oracle for π with T and $(\bar{i}_j)_{j\in T}$.
2. The oracle returns $\mathsf{sd}^{(j)}$ and $(\mathsf{st}_i^{(j)})_{i\in[N]}$ for $j \notin T$ and $(\mathsf{st}_i^{(j)})_{i\neq\bar{i}_j}$, $(\mathsf{msgs}_i^{(j)})_{i\in[N]}$ and $\hat{w}^{(j)}$ for $j \in T$.

Decision phase:

1. For each $j \notin T$, $\mathbf{V}$ emulates the offline phase using $\mathsf{sd}^{(j)}$ to compute $(\tilde{\mathsf{st}}_i^{(j)})_{i\in[N]}$ as an honest prover would.
2. For each $j \in T$, $\mathbf{V}$ emulates the online phase using $(\mathsf{st}_i^{(j)})_{i\neq\bar{i}_j}$, masked witness $\hat{w}^{(j)}$ and $\mathsf{msgs}_{\bar{i}_j}^{(j)}$ to compute $(\tilde{\mathsf{msgs}}_i^{(j)})_{i\neq\bar{i}_j}$ and output bit $b^{(j)}$.
3. Accept iff
   - For $j \notin T$, the offline phases are computed correctly, i.e., $(\tilde{\mathsf{st}}_i^{(j)})_{i\in[N]} = (\mathsf{st}_i^{(j)})_{i\in[N]}$.
   - For $j \in T$, the online phases are computed correctly, i.e., $(\tilde{\mathsf{msgs}}_i^{(j)})_{i\neq\bar{i}_j} = (\mathsf{msgs}_i^{(j)})_{i\neq\bar{i}_j}$ and $b^{(j)} = 1$.

F Banquet as an IOP

The underlying interactive protocol of Banquet can be characterized as a 3-round IOP. See Protocol 6.

Protocol 6: Banquet-IOP for relation $R = \{((\mathsf{ct}, \mathsf{pt}), K) \ : \ \mathsf{ct} = \mathsf{AES}_K(\mathsf{pt})\}$

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

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

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

1. The prover picks at random seeds $\mathsf{sd}_1, \ldots, \mathsf{sd}_N$.
2. For each party $i \in [N]$:
   (a) Expand sd_i into tape_i
   (b) Sample witness share w_i from tape_i
3. It computes witness offset $\Delta w = w - \sum_i w_i$ and adjust first share $w_1 := w_i + \Delta w$.
4. For each S-box ℓ:
   (a) For each party $i \in [N]$, compute the local linear operations to obtain the share $s_{i,\ell}$ of the S-box of input s_ℓ.
   (b) Compute the S-box output $t_\ell = (\sum_i s_{i,\ell})^{-1}$.
   (c) For each party $i \in [N]$, sample the share of the output $t_{i,\ell}$ from tape_i
   (d) Compute output offset $\Delta t_\ell = t_\ell - \sum_i t_{i,\ell}$.
   (e) Adjust first share $t_{1,\ell} = t_{1,\ell} + \Delta t_\ell$
5. Broadcast each party's share ct_i of the output.
6. Send an oracle $\pi_1 = ((\mathsf{sd}_i, \mathsf{ct}_i)_{i\in[N]}, \Delta w, (\Delta t_\ell)_{\ell\in[m]})$

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

1. For each party $i \in [N]$ and S-box $\ell \in [m]$, lift $s_{i,\ell}$ and $t_{i,\ell}$ to $\mathbb{F}_{2^{8\lambda}}$.
2. For $i \in [N]$ and $j \in [m_1]$:
   (a) For $k \in [0, m_2 - 1]$, set $s'_{i,j,k} = r_j \cdot s_{i,j+km_1}$ and $t'_{i,j,k} = t_{i,j+km_1}$
   (b) Sample masking points $\bar{s}_{i,j}$ and $\bar{t}_{i,j}$ from tape_i.
   (c) Interpolate degree m_2 polynomials $S, T \in \mathbb{F}_{2^{8\lambda}}[X]$ such that

$$S_{i,j}(k) = s'_{i,j,k} \text{ for } k \in [0, m_2 - 1], \qquad S_{i,j}(m_2) = \bar{s}_{i,j} \tag{23}$$

$$T_{i,j}(k) = t'_{i,j,k} \text{ for } k \in [0, m_2 - 1], \qquad T_{i,j}(m_2) = \bar{t}_{i,j} \tag{24}$$

3. Compute product polynomial $P := \sum_{j\in[m_1]}(\sum_i S_{i,j}) \cdot (\sum_i T_{i,j})$
4. For $k \in [m_2, 2m_2]$, compute offset $\Delta P(k) = P(k) - \sum_i \mathsf{Sample}(\mathsf{tape}_i)$
5. For $i \in [N]$ interpolate ith share of degree $2m_2$ polynomial $P_i \in \mathbb{F}_{2^{8\lambda}}[X]$ such that

$$\text{For } k \in [0, m_2 - 1]: \qquad P_1(k) = \sum_j r_j, \tag{25}$$

$$P_i(k) = 0 \text{ for } i \neq 1 \tag{26}$$

$$\text{For } k \in [m_2, 2m_2]: \qquad P_1(k) = \mathsf{Sample}(\mathsf{tape}_1) + \Delta P(k), \tag{27}$$

$$P_i(k) = \mathsf{Sample}(\mathsf{tape}_i) \text{ for } i \neq 1 \tag{28}$$

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

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

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

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

Query phase

1. The verifier picks at random $\bar{i} \in [N]$ and queries π_1 with $\bar{i}$.
2. The oracle π returns $((\mathsf{sd}_i)_{i\neq\bar{i}}, \mathsf{ct}_{\bar{i}}, \Delta w, (\Delta t_\ell)_{\ell\in[m]})$.
3. The verifier queries π_2 and π_3 with empty strings
4. The oracle π_2 and π_3 return $(\Delta P(k))_{k\in[m_2,2m_2]}$ and $((a_{i,j}, b_{i,j})_{j\in[m_1]}, c_i)_{i\in[N]}$, respectively.

Decision phase

1. For $i \neq \bar{i}$:
   (a) Expand sd_i into tape_i
   (b) Sample witness share w_i from tape_i
   (c) If $i = 1$, adjust $w_1 = w_1 + \Delta w$
   (d) For each S-box ℓ:
      i. Compute local linear operations to obtain the share $s_{i,\ell}$
      ii. Sample output share $t_{i,\ell}$ from tape_i
      iii. If $i = 1$ adjust, $t_{1,\ell} = t_{1,\ell} + \Delta t_\ell$
   (e) Recompute output broadcast $\tilde{\mathsf{ct}}_i$ and missing $\tilde{\mathsf{ct}}_{\bar{i}} = \mathsf{ct} - \sum_{i\neq\bar{i}} \tilde{\mathsf{ct}}_i$
   (f) For $j \in [m_1]$, interpolate polynomials $S_{i,j}$ and $T_{i,j}$ as the prover would.
   (g) Interpolate product polynomial P_i using $(\Delta P(k))_{k\in[m_2,2m_2]}$ as the prover would.
   (h) For $j \in [m_1]$, compute $\tilde{a}_{i,j} = S_{i,j}(R)$ and $\tilde{b}_{i,j} = T_{i,j}(R)$. Compute $\tilde{c}_i = P_i(R)$.
2. Accept iff
   – For $i \neq \bar{i}$, $c_i = \tilde{c}_i$
   – For $i \neq \bar{i}$ and $j \in [m_1]$, $a_{i,j} = \tilde{a}_{i,j}$ and $b_{i,j} = \tilde{b}_{i,j}$
   – $\mathsf{ct}_{\bar{i}} = \tilde{\mathsf{ct}}_{\bar{i}}$
   – $\sum_i c_i = \sum_j (\sum_i a_{i,j}) \cdot (\sum_i b_{i,j})$

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