Witness Encryption for Succinct Functional Commitments and Applications

Matteo Campanelli[1], Dario Fiore[2], and Hamidreza Khoshakhlagh[3]

[1] Protocol Labs matteo@protocol.ai
[2] IMDEA Software Institute, Madrid dario.fiore@imdea.org
[3] Concordium hk@concordium.com

Abstract. Witness encryption (WE), introduced by Garg, Gentry, Sahai, and Waters (STOC 2013) allows one to encrypt a message to a statement x for some NP language $\mathcal{L}$, such that any user holding a witness for $\mathsf{x} \in \mathcal{L}$ can decrypt the ciphertext. The extreme power of this primitive comes at the cost of its elusiveness: a construction from established cryptographic assumptions is currently out of reach.

In this work we introduce and construct a new notion of encryption that has a strong flavor of WE and that, crucially, we can build from well-studied assumptions (based on bilinear pairings) for interesting classes of computation. Our new notion, *witness encryption for (succinct) functional commitment*, takes inspiration from a prior weakening of witness encryption introduced by Benhamouda and Lin (TCC 2020). In a nutshell, theirs is a WE where: the encryption statement consists of a (non compressible) commitment cm, a function G and a value y; the decryption witness consists of a (non succinct) NIZK proof about the fact that cm opens to v such that $y = G(v)$. Benhamouda and Lin showed how to apply this primitive to obtain MPC with non-interactive and reusability properties—dubbed mrNISC—replacing the requirement of WE in existing round-collapsing techniques. Our new WE-like notion is motivated by supporting both commitments of a *fixed* size and *fixed* decryption complexity, independent of the size of the value v—in contrast to the work by Benhamouda and Lin where this complexity is linear. As a byproduct, our efficiency requirement substantially improves the offline stage of mrNISC protocols.

From a technical standpoint, our work shows how to solve additional challenges arising from relying on computationally binding commitments and computational soundness (of functional commitments), as opposed to statistical binding and unconditional soundness (of NIZKs), used in Benhamouda and Lin's work. In order to tackle them, we need not only to modify their basic blueprint, but also to model and instantiate different types of projective hash functions as building blocks. Our techniques are of independent interest and may highlight new avenues to design practical variants of witness encryption.

As an additional contribution, we show that our new WE-flavored primitive and its efficiency properties are *versatile*: we discuss its further applications and show how to extend this primitive to better suit these settings.

Keywords: Witness encryption; Secure multiparty computation; Functional commitments; Smooth projective hash functions

1 Introduction

Witness Encryption (WE) [GGSW13] is an encryption paradigm that allows one to encrypt a message under a hard problem—a statement x of an NP language $\mathcal{L}$—so that anyone knowing a solution to this problem—a witness w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathcal{L}}$—can decrypt the ciphertext in an efficient manner. Witness encryption generalizes the classical notion of public-key encryption, where a user can encrypt a message m to any user who knows the (secret) decryption key $\mathsf{w} = \mathsf{sk}$ associated to some (public) encryption key $\mathsf{x} = \mathsf{pk}$.

A general-purpose WE, one for all NP, is a powerful tool: it can be used to construct several cryptographic primitives [DH76,Sha84,BF03,SW05]. Yet, currently, all its general constructions rely on strong assumptions, such as multilinear maps [GGSW13,GLW14] or indistinguishability obfuscation (iO) [GGH$^+$13], that currently are not very well understood.

Recent work has shown how some applications of witness encryption can be obtained through *weaker* primitives, which, while still maintaining a strong "WE flavor" can at the same time be built through well-studied cryptographic assumptions. We are referring to the work of Benhamouda and Lin [BL20], which applies the round-collapsing techniques in [GLS15] to obtain powerful secure-computation protocols, *multi-party reusable non-interactive secure computation* (or mrNISC), a type of MPC that requires no interaction among subsets of users, provided that users had earlier committed to their input on a public bulletin board (this offline stage is called "input encoding stage"). Before [BL20], similar techniques had required full-blown WE. Benhamouda and Lin, on the other hand, use a different type of WE with a deterministic angle to it: the statement w.r.t. which we encrypt is a *commitment* to a certain string v and a claim on a function evaluation on v; the decrypting party can use a NIZK proof (non-interactive zero-knowledge [BFM88]). That is, the encryption statement is (cm, G, y) and in order to decrypt we require a NIZK proof π which proves that the evaluation of G on the value v committed in cm outputs y, i.e., "$\mathsf{cm} = \mathsf{Commit}(v)$ and $y = G(v)$". We refer to the WE-like primitive in [BL20] as $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$ for short.

Above, both the commitment and the proof size—and hence decryption time—grow linearly in the size of v. The latter represents a piece of potentially *large* data and whose commitment is *publicly shared* at an earlier time. We refer concisely to a construction not having this dependence in efficiency as having "input-independent (decryptor's) complexity". A scheme with input-independent complexity would be interesting to further minimize the communication complexity of applications of this type of WE. This can be relevant, for example, in the input encoding phase of mrNISC (as well as in other applications, see section 7): commitments are stored on a bulletin board (e.g., a blockchain) forever and thus their size significantly affects its growth over time. On the other hand, we cannot just remove this dependence by straightforwardly changing any "localized piece" in the approach in [BL20]. This dependency is to a certain extent entangled with the techniques in [BL20]. These techniques rely on statistically binding commitments and cryptographic *proofs* (rather than *arguments*, see also footnote 7), which cannot be compressed.

1.1 Our Work: WE For Succinct Functional Commitments

The work from [BL20] is encouraging: we may be able to use more familiar assumptions to obtain useful variants of witness encryption. Our work is motivated by pushing this avenue further, both practically and theoretically. We ask:

What are other weak-but-useful variants of WE that remain "as simple as possible" in terms of assumptions to build them and that can achieve input-independent complexity?

In order to address this question, we move towards defining and constructing witness encryption for *succinct commitments that support succinct arguments*. That is, where commitments are of fixed size—independent of the input's length—and so are the arguments about the correctness of computations on the committed inputs. At the same time, we turn our gaze towards approaches requiring simpler and plausibly strictly weaker primitives than general non-deterministic arguments.

We propose a generalization of $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$, the primitive in [BL20]. Our definition relies on the notion of *functional commitments* of Libert, Ramanna and Yung [LRY16]. We call this primitive *"WE for functional commitments"* ($\mathrm{WE}^{\mathrm{FC}}$). Functional commitments allow a party to commit to a value v and later open the commitment to $y = G(v)$ for some functions G, by generating an opening proof π. Notably, in FCs both the commitment and the opening proofs can be succinct (in particular, throughout this work we always use the term 'functional commitments' to mean succinct ones). In terms of security, an FC should be *evaluation binding* and *hiding*. The former means that an adversary cannot open the commitment to two distinct outputs $y \neq y'$ for the same function G, while the latter is the standard hiding property of commitments. In addition, in our work we require FC to be *zero-knowledge*, which informally states that the opening proof π should not reveal any information about the committed value v.

Functional commitments can be seen as a simple version of succinct commit-and-prove NIZK arguments for polynomial-time computations. Their main difference from a NIZK is that soundness is replaced by evaluation binding, which is a falsifiable notion since it requires the adversary to provide valid proofs for two contradicting statements. Thanks to this, large classes of computations support functional commitments that can be realized from falsifiable assumptions, as shown in the existing constructions of Libert et al. [LRY16] for linear functions and of Lipmaa and Pavlyk [LP20] for sparse polynomials. For our goal of extending $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$ to succinct commitments, functional commitments are therefore a suitable candidate as they minimally express the functionalities we need.

1.2 Our Contributions

Modeling: We introduce the notion of witness encryption for functional commitments ($\mathrm{WE}^{\mathrm{FC}}$) and formally define it. This notion can be seen as a—still highly useful—weakening of witness encryption with nice communication complexity properties and that can actually be built from common cryptographic assumptions. We elaborate on these features in the rest of this section.

Construction and techniques: We propose a construction of WE^{FC} based on bilinear pairings. Our construction combines a functional commitment with a linear verification procedure over a bilinear group, together with an extractable projective hash function. Informally, linear verification of an FC means that its openings proofs are group elements that are not paired together using the bilinear map. Interestingly, the two FC schemes in [LRY16,LP20] satisfy this property and can be used to instantiate our WE^{FC}.

While our construction follows the blueprint of [BL20] (i.e., combining a proof system with an SPHF for its verification language), we had to solve substantial challenges due to our shift from the "soundness against any adversary" of NIZKs to the "computational binding" of functional commitments. In particular, the $\text{WE}^{\text{ZK-CM}}$ definition of [BL20] used the notion of statistically binding commitments and statistically sound NIZKs that allow one to classify a statement $\mathsf{x} = (\mathsf{cm}, G, y)$ as false. The latter property is crucial when building WE from an SPHF for the NIZK verification language: if x is false, then there is also no proof that makes the NIZK verification accept, and thus the smoothness of SPHF can be used to mask the message. If cm is succinct, this property may no longer hold. cm may contain collisions and always open to a v' such that $G(v') = y$ (and a similar issue arises for the fact that openings are arguments). To solve this challenge, we develop two main technical contributions. The first one is finding a useful variant of projective hash function for our purposes. We define the notion of *extractable* PHFs and construct this primitive secure in the algebraic group model [FKL18]. The second one is a different methodology for building WE from extractable projective hash functions that resorts to the Goldreich-Levin technique [GL89] in order to turn a good distinguisher into an algorithm that computes the hash function. We refer to our technical overview in section 1.3 for further details.

A construction of mrNISC from WE^{FC}: We show how our WE^{FC}notion can be used to build mrNISC. The latter is a class of secure multiparty computation protocols in which parties work with minimal interaction. In a first round, each party posts an encoding of its inputs in a public bulletin board. This is done once and for all. Next, any subset of parties can compute a function of their private inputs by sending only one message each. This second phase can be repeated many times for different computations and different subsets of parties. Our construction for mrNISC confirms that our notion is not losing expressivity compared to $\text{WE}^{\text{ZK-CM}}$ from [BL20] and can in addition yield protocols with a succinct input encoding phase. We also discuss properties of concrete MPC protocols that make them amenable to be compiled into a mrNISC through our approach.

Other applications of WE^{FC}: We provide additional applications beyond mrNISC where WE^{FC}can be useful. As a first application, we show how WE^{FC}can be used for a simple construction of a variant of *targeted broadcast*. In targeted broadcast [GPSW06] we want a certain message to be conveyed only to authorized parties. An authorized party is one holding attributes satisfying a certain property (specified at encryption time). As an example, a streaming service may want to broadcast an encryption of a movie so that only users having purchased certain packages would be able to decrypt (and watch) it. There exist ways to build this primitive non-naively while satisfying basic desiderata of the application domain[4], for example through ciphertext-policy ABE [GPSW06]. We show how we can achieve targeted broadcast in a new (and simple) manner through WE^{FC}. We observe that our construction achieves some interesting properties absent in previous approaches: it achieves *flexible and secret attestation* and *without any master secret*. This means that decryption attributes may be granted to a user through different methods, that the latter can be kept secret and that there is no single party holding a key that can decrypt all messages in the system. We provide further details and motivation in section 7.

As a second application, we show how, through WE^{FC}, we can achieve simple and non-interactive *contingent payment for services* [CGGN17] ("contingent payment" for short[5]). In a contingent payment a *payer* wants to provide a reward/payment to another user conditional to the user having performed a certain service. For example, a user may want to pay a cloud service conditionally to them storing their data. Ideally this protocol should require no interaction. We describe a simple way to instantiate the above through WE^{FC}. Our solution can be used, for example, to incentivize, in a fine-grained manner, portions of large committed data (for instance incentivizing storage of specific pages of Wikipedia or the Internet Archive of particular importance on IPFS[6]) [dec22]. Compared to other approaches [CGGN17], our solution is simple (e.g., does not require a blockchain with special properties or smart contracts) and

[4] For example, sometimes a desideratum in such systems is that the broadcaster should not have to refer to a database of user authorizations each time a different item is to be encrypted for broadcast.

[5] Notice that "contingent payment" can also refer to payment for *goods*, rather than *services*. In this paper we only refer to payment for services.

[6] `http://wikipedia.org`, `http://archive.org`, `http://ipfs.io`

is highly communication efficient. To achieve this solution we need to solve additional technical challenges: modeling and building an extractable variant of $\mathrm{WE}^{\mathrm{FC}}$. We provide further details in section 7.

1.3 Technical Overview

We start with an overview of the techniques in [BL20]. Their notion of witness encryption called "WE for NIZK of Commitments" ($\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$) is defined for an NP language whose statements are of the form $\mathsf{x} = (\mathsf{cm}, G, y)$ such that cm is a commitment, G is an arbitrary polynomial-size circuit, and y is a value (additionally, this language is parametrized by the common reference string, or crs, of the NIZK). The type of commitment assumed in [BL20] is perfectly binding; therefore, a statement (cm, G, y) is true if there exists a NIZK proof π (as a witness) which proves w.r.t. crs that G evaluates to y on the value v committed in cm.

The definition of $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$ states that semantic security property should hold for ciphertexts created with respect to false claims (that is, commitments whose opening v is such that $G(v) \neq y$). To achieve this property, the idea in [BL20] relies on applying smooth projective hash function (SPHF) on the verification algorithm of the NIZK. For the sake of this high-level overview, the reader can think of an SPHF as a form of WE itself and which we know how to realize for simple languages. The crux of the construction in [BL20], then is that, if the NIZK verification algorithm is "simple enough", then we can leverage it to build $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$. In more detail, let $\mathbf{\Theta} = \mathbf{M}\pi$ be the linear equation corresponding to the verification of a NIZK for a statement $\mathsf{x} = (\mathsf{cm}, G, y)$, such that $\mathbf{\Theta}$ and $\mathbf{M}$ depend on x and crs, and hence are known at the time of encryption. To encrypt a message, one can use an SPHF for this relation such that only those who can compute the hash value using a valid witness π (i.e., π such that $\mathbf{\Theta} = \mathbf{M}\pi$) can retrieve the message. The work in [BL20] instantiates the above through Groth-Sahai NIZKs, which can be reduced to a linear verification for committed inputs (this is true for only a restricted class of computations which then [BL20] shows how to extend to all of P through randomized encodings). The commitments they rely on are statistically binding and thus not compressing.

We now discuss how to go from this idea to our solutions. Recall that our goal is to have a type of witness encryption that works on functional commitments. This implies that both the commitments and opening proofs for functional evaluation on them are compressing. This efficiency requirement is the main point of divergence between $\mathrm{WE}^{\mathrm{FC}}$ and $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$.

Moving from [BL20] to our approach is not unproblematic. In [BL20], in order to *(i)* effectively reduce the original relation ($G(v) = y$ for a correct opening v) to the verification of the NIZK, and *(ii)* to maintain semantic security at the same time—in order to simultaneously achieve these two points—it is crucial that the NIZK proof has unconditional soundness and that the underlying commitments are perfectly binding[7]. At a very high level, the switch from [BL20] to our work consists of the switch from a NIZK *proof* system [GS08], with *linear proof size*, to a *succinct certificate*, a succinct functional commitment. Simple as it may sound, however, this switch is not immediate and requires solving several new challenges on the way.

Brush up on functional commitments: The notion of functional commitments introduced by [LRY16] allows a committer to commit to a value v and later open the commitment to $y = G(v)$ by outputting an opening proof π. Note that G can be chosen by the verifier at the time of opening. Functional commitments should satisfy standard properties of *hiding* and *evaluation binding* (we will sometimes refer to the computational flavor of soundness of evaluation binding as "argument" to make immediate the connection with argument systems). In this work we also require a stronger property, namely a *zero-knowledge* property which informally states that the opening proof π should not reveal any information about the committed value v. Finally our main interest is in *succinct* functional commitments, where both the commitment and the opening proof are succinct. We choose succinct functional commitments for our design, intuitively, because they are a minimal primitive providing the properties we hope for. We discuss further in the related work sections.

In what follows, we go into the details of our construction in a step-by-step approach, by discussing the main challenges that will need to be addressed in each step.

Challenge 1: Proofs vs Arguments. The main challenge arises when using arguments (as opposed to proofs) as witness in the witness encryption scheme. Recall that $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$ constructs WE for the augmented relation $\mathcal{R}$ corresponding to the verification algorithm of the NIZK proof and, as mentioned above, switching to $\mathcal{R}$ still preserves semantic security. However, the same idea does not work when using

[7] Unconditional soundness of a proof system means: "for a false statement, *no* proof string will have a substantial probability of being accepted as valid". This is in contrast to the computational soundness of our building blocks: "for a false statement, no PPT adversary can produce a proof string with substantial probability of being accepted". The latter does not state that such proof string does not exist.

an argument system. This is because semantic security only guarantees security when the statement, under which the challenge ciphertext is generated, is *false*. Defining $\mathcal{R}$ as the relation specified by the verification of an argument system makes all statements potentially *true*. Hence, even though finding a witness (i.e, an argument) is computationally hard, semantic security holds vacuously and makes no guarantee about the encrypted message.

To solve this challenge, we observe that even though the relation is trivial here, finding the witness for a statement yields a contradiction to security properties of the commitment in use. To elaborate further, we note that the WE is constructed for the NP language corresponding to the verification algorithm of a functional commitment. Now, given a "false" statement $\bar{\mathsf{x}} = (\mathsf{cm}, G, y)$, where $G(v) \neq y$ for v committed in cm and chosen by the adversary, our construction is such that for any efficient adversary that distinguishes ciphertexts encrypted under the statement x corresponding to the verification circuit which (incorrectly) asserts the truth of $\bar{\mathsf{x}}$, there exists an efficient adversary that breaks the evaluation-binding property of the functional commitment by computing a valid opening proof op that satisfies the FC verification.

Challenge 2: Membership vs Knowledge. To build the above reduction, we make use of the Goldreich-Levin technique [GL89] by which we can transform a ciphertext distinguisher into an efficient algorithm that computes the hash value H used as a *one-time pad* to mask the message. While this part of the reduction is straightforward, one challenge is how to compute a valid opening proof op from H. To this end, we observe that the underlying SPHF is for the same language $\mathcal{L}$ that we build our WE and thus op plays the role of the witness for x by which one can compute H. Thus, it seems like we would need a type of SPHF with a strong notion of *extractable* security. Namely, a type of projective hash function (PHF) that guarantees the existence of an extractor such that for any adversary that is able to compute a valid hash, the extractor can compute a witness for the corresponding problem statement.

Unfortunately, there exists no construction of extractable PHF in literature, even based on non-standard assumptions. The closest work is that of Wee [Wee10] which suggests a similar notion but only for some relations not in NP that correspond to search problems. For the NP languages of our interest, it is easy to see that since the hash values are of constant size (usually consisting of one group element), one needs to rely on non-black-box techniques to provide extractability. We follow this approach and give a construction of extractable PHF in the algebraic group model [FKL18].

Challenge 3: Reusability. Replacing NIZK of commitments with a functional commitment as described above, and then following the same approach of [GLS15,BL20] yields a two-round MPC protocol. However, building a mrNISC protocol is more challenging as the construction may not necessarily provide reusability. To provide this property, we need functional commitment schemes that satisfy a strong form of zero-knowledge, wherein any number of opening proofs for a given commitment can be simulated. In other words, for a commitment cm broadcasted by a party in the first round of the protocol, running computation on different statements (cm, G_i, y_i) with the same commitment cm does not reveal any information about the committed value. This should be guaranteed by the existence of an efficient simulator that can generate simulated openings for any adversarially chosen computation.

Trusted Setup and Malicious Security: We note that both existing constructions of mrNISC from bilinear pairing groups [BL20] or from LWE [BJKL21] are in the plain model, whereas our construction requires a trusted setup. However, for security analysis of mrNISC construction in previous works, it is assumed that the corruption by the adversary is static. Further, the security in these works is only against semi-malicious adversaries where corrupted parties follow the protocol specification, except they are allowed to select their input and randomness from arbitrary distributions. This has been justified by the fact that providing stronger notion of malicious security for MPC in two rounds in the plain model is impossible and hence one should use either a trusted setup assumption or overcome this impossibility by relying on super-polynomial time simulation (See [FJK21] for the second approach). We thus see the use of trusted setup in our construction, in a sense, at no cost as it is crucial for achieving malicious security [8].

1.4 Related Work

The first candidate construction of witness encryption was proposed by the seminal work of Garg et al. [GGSW13] based on multilinear maps. In a line of research, several other works [GGH+13,GLW14,GLW14,GKW17] proposed constructions from similar strong assumptions; i.e., multilinear maps as in [GGSW13], or indistinguishability obfuscation (iO). Recently, Barta et al. [BIOW20] showed a witness encryption scheme based on a coding problem called *Gap Minimum Distance Problem* (GapMDP). However, they left it as an open

[8] Achieving malicious security by using NIZK proofs in the trust model is a folklore technique and has been used in many classical MPC works (e.g., See Lemma 7.5 in [BL20]). We thus omit details on malicious security and similarly to previous works focus only on semi-malicious security.

problem whether their version of GapMDP is NP-hard. Another recent proposal based on new unexplored algebraic structures and with conjectured security is that in [CVW18].

The work of [BL20] defines a restricted flavour of witness encryption called *WE for NIZK of commitments* wherein parties first commit to their private inputs once and for all, and then later, an encryptor can produce a ciphertext so that any party with a NIZK showing that the committed input satisfies the relation can decrypt. Their construction relies on the SXDH assumption in bilinear pairings and Groth-Sahai commitments and NIZKs. Using NIZK proofs as the decryption key provides a "delegatability" property in [BL20], where the holder of a witness can delegate the decryption by publishing a NIZK proof for the truth of the statement. Recently, [CDK$^+$21] formalize a similar notion but without delegation property, and give more efficient instantiations based on two-party Multi-Sender Non-Interactive Secure Computation (MS-NISC) protocols. The recent work of [Kho22] also defines a similar notion of *Witness Encryption with Decryptor Privacy* that provides zero-knowledge, but not delegation property. Our approach is a follow-up to the work of [BL20]. Finally, we note that constructions with a flavor of witness-encryption-over-commitments [BL20,CDK$^+$21] are also a viable solution to this problem, but with the caveat of commitments having to be as large as the data (which is problematic if the data is large). This is not the case in our constructions.

If we turn our attention to NIZKs and succinct commitments, one may wonder whether one can adapt the results of [BL20] to work with (commit-and-prove) SNARKs. Although we cannot exclude this option, we argue this may be an overkill for two reasons. First, in terms of assumptions this approach would inherently require the use of non-falsifiable assumptions due to the impossibility result of Gentry and Wichs [GW11]. In particular, the semantic security definition of $\mathrm{WE}^{\mathrm{ZK\text{-}CM}}$ is falsifiable and thus could in principle be realized without these strong assumptions. Second, in terms of efficiency, if we want to rely on the SPHF construction framework we would need a SNARK with a linear verification over bilinear groups, but such schemes are likely impossible, as shown by Groth [Gro16].

The primitive that we propose in this work is closely tied to functional commitments, first formalized by Libert et al. [LRY16]. The functional commitment schemes in the state of the art support a variety of functions classes, which include linear maps [LRY16,LM19], sparse polynomials [LP20], constant-degree polynomials [CFT22,ACL$^+$22], and NC1 circuits [CFT22]. Also, very recent works [BCFL22,dCP22] propose FC schemes for virtually arbitrary computations. As we mentioned earlier, our construction of $\mathrm{WE}^{\mathrm{FC}}$ relies on FCs whose verification algorithm is a 'linear' pairing-based equation. This property is achieved by the FC schemes for linear maps [LRY16,LM19] and sparse polynomials [LP20], which means we can obtain instantiations of $\mathrm{WE}^{\mathrm{FC}}$ for these classes of functions. The recent and more expressive constructions that are based on pairings [CFT22,BCFL22] unfortunately do not support this linear verification, as they need to pair elements of the proof. Finding new FCs with a linear pairing-based verification and supporting more expressive functions, such as circuits, is an interesting open problem motivated by our work.

2 Preliminaries

Notation. We use DPT (resp. PPT) to mean a deterministic (resp. probabilistic) polynomial time algorithm. We denote by $[n]$ the set $\{1,\dots,n\} \subseteq \mathbb{N}$. To represent matrices and vectors, we use bold upper-case and bold lower-case letters, respectively. We denote the security parameter by $\lambda \in \mathbb{N}$. For an algorithm $\mathcal{A}$, $\mathsf{RND}(\mathcal{A})$ is the random tape of $\mathcal{A}$ (for a fixed choice of λ), and $r \leftarrow\!\!\$\ \mathsf{RND}(\mathcal{A})$ denotes the random choice of r from $\mathsf{RND}(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathsf{x}; r)$ we denote that $\mathcal{A}$, given an input x and a randomizer r, outputs y. By $x \leftarrow\!\!\$\ \mathsf{D}$ we denote that x is sampled according to distribution D or uniformly randomly if D is a set. Let $\mathsf{negl}(\lambda)$ be an arbitrary negligible function.

Pairings. A pairing is defined by a tuple $\mathsf{bp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are (additive) groups of order p, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient, non-degenerate bilinear map. In particular, $\hat{e}(a \cdot g_1, b \cdot g_2) = (ab) \cdot \hat{e}(g_1, g_2)$ for any $a, b \in \mathbb{Z}_p$. We denote $[a]_t := a \cdot g_t$ for $t \in \{1, 2, T\}$ where we define $g_T = \hat{e}(g_1, g_2)$. The same notation naturally extends to matrices $[M]_t$ for $M \in \mathbb{Z}_p^{n \times m}$.

Algebraic (Bilinear) Group Model. Essentially, in the AGM [FKL18], one assumes that every PPT algorithm $\mathcal{A}$ is algebraic in the sense that $\mathcal{A}$ is allowed to see and use the structure of the group, but she is required to also output a representation of output group elements as a linear combination of the inputs. While the definition of AGM in [FKL18] only captures regular groups, here we require an extension that captures asymmetric pairings as well. To formalize this notion, we use the following definition that is taken from [CH20], but adjusted for our setting where $\mathcal{A}$ only outputs target group elements.

Definition 1 (Algebraic Adversaries). *. Let* $\mathsf{bp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ *be a pairing group and* $[\mathbf{x}]_1 = ([x_1]_1, \ldots, [x_n]_1) \in \mathbb{G}_1^n$*,* $[\mathbf{y}]_2 = ([y_1]_2, \ldots, [y_m]_2) \in \mathbb{G}_2^m$*,* $[\mathbf{z}]_T = ([z_1]_T, \ldots, [z_l]_T) \in \mathbb{G}_T^l$ *be vectors in* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_T$*, respectively. An algorithm* $\mathcal{A}$ *with input* $[\mathbf{x}]_1, [\mathbf{y}]_2, [\mathbf{z}]_T$ *is called algebraic if in addition to its output*

$$\mathbf{S} = \left([S_1]_T \ldots, [S_{l'}]_T\right) \in \mathbb{G}_T^{l'}$$

$\mathcal{A}$ *also provides a vector*

$$\mathbf{s} = \left((A_{ijk})_{\substack{1 \le i \le l' \\ 1 \le j \le n \\ 1 \le k \le m}}, (B_{ij})_{\substack{1 \le i \le l' \\ 1 \le j \le l}} \right) \in \mathbb{Z}_p^{\zeta} \qquad \text{with } \zeta = l' \cdot (l + n \cdot m)$$

such that

$$S_i = \sum_{j=1}^{n} \sum_{k=1}^{m} \hat{e}\left(x_j, y_k\right)^{A_{ijk}} + \sum_{j=1}^{l} B_{ij} z_i \qquad \text{for } i \in \{1, \ldots, l'\}$$

2.1 Functional Commitment Schemes

Let $\mathcal{D}$ be some domain and $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^{\kappa}$ be a circuit. In a functional commitment scheme for $\mathcal{C}$, the committer first commits to a vector $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$, obtaining a functional commitment cm, and then later can open C to $\mathbf{y} = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathcal{D}^{\kappa}$, where $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$.

Definition 2 (Functional Commitments). *For a class* $\mathbf{CC}$ *of circuits* $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^{\kappa}$*, a functional commitment scheme* FC *consists of four polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$*, where*

Setup. $\mathsf{Setup}(1^\lambda, \mathcal{C})$ *is a probabilistic algorithm that given a security parameter* $\lambda \in \mathbb{N}$*, and a circuit* $\mathcal{C} \in \mathbf{CC}$*, outputs a commitment key* ck *and a trapdoor key* td*. For simplicity of notation, we assume that* ck *contains the description of* 1^λ *and* $\mathcal{C}$*.*

Commitment. $\mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)$ *is a probabilistic algorithm that on input a commitment key* ck*, a message* $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$*, and randomness* r*, outputs* $(\mathsf{cm}, \mathsf{d})$*, where* cm *is a commitment to* $\boldsymbol{\alpha}$ *and* d *is a decommitment information.*

Opening. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{d}, \boldsymbol{\beta})$ *is a deterministic algorithm that on input* ck*, a commitment* cm *(to* $\boldsymbol{\alpha}$*), a decommitment* d*, and a vector* $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$*, outputs an opening* $\mathsf{op}_\mathbf{y}$ *to* $\mathbf{y} = \mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := (\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))_{i=1}^{\kappa}$*, where public function* $\mathcal{F}_i$ *is the i-th output value of* $\mathcal{C}$*.*

Verification. $\mathsf{Verify}(\mathsf{ck}, \mathsf{cm}, \mathsf{op}_\mathbf{y}, \boldsymbol{\beta}, \mathbf{y})$ *is a deterministic algorithm that on input* ck*, a commitment* cm*, an opening* $\mathsf{op}_\mathbf{y}$*, a vector* $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$*, and* $\mathbf{y} \in \mathcal{D}^{\kappa}$*, outputs 1 if* $\mathsf{op}_\mathbf{y}$ *is a valid opening for* cm *and outputs 0 otherwise.*

Security. We require two security properties for functional commitments, *zero-knowledge* and *evaluation-binding.* The zero-knowledge property can be seen as a simulation-based definition of hiding property, considerably stronger than the definition given in [LRY16] [9]. Further, compared to the zero-knowledge definition of [LP20], our definition is stronger in the sense that the commitment and simulated openings are not generated at the same time. In other words, to make commitments reusable for our mrNISC application, we need two simulators $\mathcal{S}_1, \mathcal{S}_2$, where $\mathcal{S}_1$ generates a simulated commitment, and $\mathcal{S}_2$—given the simulated commitment—can produce any number of simulated openings for different adversarially chosen functions.

Definition 3 (Perfect zero-knowledge). *A functional commitment scheme* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *for circuit class* $\mathbf{CC}$ *is perfectly zero-knowledge if there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$*, such that for all* λ*, all* $\mathcal{C} \in \mathbf{CC}$*,* $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C})$*, the following distributions are identical.*

$$\left\{\mathcal{A}^{\mathsf{O}_{\mathsf{Open}}}(\mathsf{st}) = 1 : (\mathsf{st}, \boldsymbol{\alpha}) \leftarrow \mathcal{A}(\mathsf{ck}), r \leftarrow\!\!\$\ \mathsf{RND}_\lambda(\mathsf{Commit}), (\mathsf{cm}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)\right\}\Big\}$$

$$\left\{\mathcal{A}^{\mathsf{O}_{\mathcal{S}}}(\mathsf{st}) = 1 : (\mathsf{st}, \boldsymbol{\alpha}) \leftarrow \mathcal{A}(\mathsf{ck}), (\mathsf{cm}, \mathit{aux}) \leftarrow \mathcal{S}_1(\mathsf{td})\right\}$$

where $\mathsf{O}_{\mathsf{Open}}(\boldsymbol{\beta}) := \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{d}, \boldsymbol{\beta})$ *and* $\mathsf{O}_{\mathcal{S}}(\boldsymbol{\beta}) := \mathcal{S}_2(\mathsf{td}, \mathit{aux}, \boldsymbol{\beta}, \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}))$*.*

We now define evaluation-binding property. Our definition is weaker than the definition of evaluation-binding given in [LRY16,LP20] as the adversary here is required to output the committed input and random coins, rather than the commitment.

[9] The definition of hiding in [LRY16] guarantees that the commitment does not reveal any information about $\boldsymbol{\alpha}$.

Definition 4 (Computational evaluation-binding). *A functional commitment scheme* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *for circuit class* $\mathbf{CC}$ *is computationally evaluation-binding if for any* λ*, any* $\mathcal{C} \in \mathbf{CC}$*, and PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{bind}_{\mathsf{FC},\mathcal{C},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*, where* $\mathsf{Adv}^{bind}_{\mathsf{FC},\mathcal{C},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} \boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta} \ \wedge \ \mathbf{y} \in \mathcal{D}^{\kappa} \ \wedge & (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C}) \\ \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \neq \mathbf{y} \ \wedge & : \ (\boldsymbol{\alpha}, r, \boldsymbol{\beta}, \mathbf{y}, \mathsf{op}_\mathbf{y}) \leftarrow \mathcal{A}(\mathsf{ck}) \\ \mathsf{Verify}(\mathsf{ck}, \mathsf{cm}, \mathsf{op}_\mathbf{y}, \boldsymbol{\beta}, \mathbf{y}) = 1 & \mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r) \end{array}\right]$$

Lastly, we say that FC is succinct if the length of commitments and openings are polylogarithmic in $|\boldsymbol{\alpha}|$ and $|\boldsymbol{\beta}|$.

2.2 Smooth Projective Hash Functions

Let $\mathcal{L}_{\mathsf{lpar}}$ be a NP language, parametrized by a language parameter lpar, and $\mathcal{R}_{\mathsf{lpar}} \subseteq \mathcal{X}_{\mathsf{lpar}}$ be its corresponding relation. A Smooth projective hash functions (SPHFs, [CS02]) for $\mathcal{L}_{\mathsf{lpar}}$ is a cryptographic primitive with this property that given lpar and a statement x, one can compute a hash of x in two different ways: either by using a projection key hp and $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$ as $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}, \mathsf{w})$, or by using a hashing key hk and $\mathsf{x} \in \mathcal{X}_{\mathsf{lpar}}$ as $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$. The formal definition of SPHF follows.

Definition 5. *A SPHF for* $\{\mathcal{L}_{\mathsf{lpar}}\}$ *is a tuple of PPT algorithms* $(\mathsf{PGen}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$*, which are defined as follows:*

$\mathsf{PGen}(1^\lambda)$**:** *Takes in a security parameter* λ *and generates the global parameters* pp *together with the language parameters* lpar*. We assume that all algorithms have access to* pp*.*

$\mathsf{hashkg}(\mathsf{lpar})$**:** *Takes in a language parameter* lpar *and outputs a hashing key* hk*.*

$\mathsf{projkg}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a projection key* hp*, possibly depending on* x*.*

$\mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$**:** *Takes in a hashing key* hk*,* lpar*, and a statement* x *and outputs a hash value* H*.*

$\mathsf{projhash}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}, \mathsf{w})$**:** *Takes in a projection key* hp*,* lpar*, a statement* x*, and a witness* w *for* $\mathsf{x} \in \mathcal{L}$ *and outputs a hash value* pH*.*

To shorten notation, we sometimes denote "$\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathsf{lpar})$; $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$" by $(\mathsf{hp}, \mathsf{hk}) \leftarrow \mathsf{kgen}(\mathsf{lpar}, \mathsf{x})$. A SPHF must satisfy the following properties:

Correctness. It is required that $\mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x}) = \mathsf{projhash}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}, \mathsf{w})$ for all $\mathsf{x} \in \mathcal{L}$ and their corresponding witnesses w.

Smoothness. It is required that for any lpar and any $\mathsf{x} \notin \mathcal{L}$, the following distributions are statistically indistinguishable:

$$\{(\mathsf{hp}, \mathsf{H}) : (\mathsf{hp}, \mathsf{hk}) \leftarrow \mathsf{kgen}(\mathsf{lpar}, \mathsf{x}), \mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})\}$$
$$\{(\mathsf{hp}, \mathsf{H}) : (\mathsf{hp}, \mathsf{hk}) \leftarrow \mathsf{kgen}(\mathsf{lpar}, \mathsf{x}), \mathsf{H} \leftarrow \Omega\} \ .$$

where Ω is the set of hash values.

Remark 1. For our application, we need a type of SPHF where hp depends on the statement. This type of SPHF with such "non-adaptivity" in the smoothness property was formally defined by Gennaro and Lindell in [GL06] and was later named GL-SPHF in [Ham16]. Throughout this work, we always mean GL-SPHF when talking about SPHFs.

In the above definition, smoothness is only guaranteed for false statements. Hence, for trivial languages where all statements are true, such notion of smoothness is vacuous. To argue security in this case, a stronger notion of knowledge-smoothness is required which guarantees that if an adversary can compute the hash value with non-negligible probability, it must *know* a witness for the statement used in the hash computation. More formally,

Knowledge Smoothness. A projective hash function $(\mathsf{PGen}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ for $\{\mathcal{L}_{\mathsf{lpar}}\}$ is knowledge smooth if for any λ, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT extractor $\mathsf{Ext}_\mathcal{A}$ such that

$$\Pr\left[\begin{array}{l|l} \mathsf{H} = \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x}) & (\mathsf{pp}, \mathsf{lpar}) \leftarrow \mathsf{PGen}(1^\lambda); \mathsf{x} \leftarrow \mathcal{A}_1(\mathsf{lpar}) \\ & : \ (\mathsf{hp}, \mathsf{hk}) \leftarrow \mathsf{kgen}(\mathsf{lpar}, \mathsf{x}); \mathsf{H} \leftarrow \mathcal{A}_2(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}) \\ \wedge\, (\mathsf{x}, \mathsf{w}) \notin \mathcal{R}_{\mathsf{lpar}} & \mathsf{w} \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

A PHF with knowledge smoothness is called *extractable projective hash function* (EPHF).

Remark 2. The notion of extractable projective hash functions was first proposed by Wee [Wee10]. While our definition of extractability bears similarities with Wee's definition, it is different in that no dual mode property is required.

3 Extractable PHF

Since all the existing constructions of SPHFs over groups are based on a framework called *diverse vector space* (DVS), here we first recall this framework and then show that DVS-based SPHFs are knowledge smooth in the AGM [FKL18].

3.1 Diverse Vector Space (DVS).

Intuitively, a DVS [BBC$^+$13,ABP15,Ham16] is a way to represent a language $\mathcal{L} \subseteq \mathcal{X}$ as a subspace $\hat{\mathcal{L}}$ of some vector space of some finite field. In the seminal work [CS02], Cramer and Shoup showed that such languages automatically admit SPHFs.

To briefly recap the notion of DVS, let $\mathcal{R} = \{(\mathsf{x}, \mathsf{w})\}$ be a relation with $\mathcal{L} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}$. Let pp be system parameters, including say the description of a bilinear group. A (pairing-based) DVS $\mathcal{V}$ is defined as $\mathcal{V} = (\mathsf{pp}, \mathcal{X}, \mathcal{L}, \mathcal{R}, n, k, \mathbf{M}, \boldsymbol{\Theta}, \boldsymbol{\Lambda})$, where $\mathbf{M}(\mathsf{x})$ is an $n \times k$ matrix, $\boldsymbol{\Theta}(\mathsf{x})$ is an n-dimensional vector, and $\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$ a k-dimensional vector. In this work, we consider the case that the matrix $\mathbf{M}(\mathsf{x})$ may depend on x (i.e., GL-DVS similarly to GL-SPHF). Moreover, as long as the equation $\boldsymbol{\Theta}(\mathsf{x}) = \mathbf{M}(\mathsf{x}) \cdot \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$ is consistent, it could be that different coefficients of $\boldsymbol{\Theta}(\mathsf{x})$, $\mathbf{M}(\mathsf{x})$, and $\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$ belong to different algebraic structures. The most common case is that for a given bilinear group $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, these coefficients belong to either $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ as long as the consistency is preserved. A DVS $\mathcal{V}$ satisfies the following properties [Ham16]:

- *coordinate-independence of groups:* the group in which each coordinate of $\boldsymbol{\Theta}(\mathsf{x})$ lies is independent of x.
- *perfect completeness:* for any $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, $\boldsymbol{\Theta}(\mathsf{x}) = \mathbf{M}(\mathsf{x}) \cdot \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$.
- *statistical ε-soundness:* $\forall \mathsf{x} \in \mathcal{X} \setminus \mathcal{L}$, $\Pr[\boldsymbol{\Theta}(\mathsf{x}) \in \mathrm{colspace}(\mathbf{M}(\mathsf{x}))] \leq \varepsilon$.

3.2 Extractable PHF in the AGM.

Given a GL-DVS for $\mathcal{L}$ over some cyclic group $\mathbb{G}_1$, one can construct an efficient GL-SPHF for $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\Theta}, \boldsymbol{\Lambda})$ as follows. First, sample a hashing key $\mathsf{hk} = \boldsymbol{\alpha} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and then define the projection key as $\mathsf{hp} = [\boldsymbol{\gamma}(\mathsf{x})]_1 \leftarrow \mathsf{projkg}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x}) = \boldsymbol{\alpha}^\top [\mathbf{M}(\mathsf{x})]_1 \in \mathbb{G}_1^{1 \times k}$. The projection hash is $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}, \mathsf{w}) = [\boldsymbol{\gamma}(\mathsf{x})]_1 \cdot \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w}) = \boldsymbol{\alpha}^\top [\mathbf{M}(\mathsf{x})]_1 \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w}) \in \mathbb{G}_1$. For $[\boldsymbol{\Theta}(\mathsf{x})]_1 = [\mathbf{M}(\mathsf{x})]_1 \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w}) \in \mathbb{G}_1^n$, the hash is $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x}) = \mathsf{hk}^\top \cdot [\boldsymbol{\Theta}(\mathsf{x})]_1 = \boldsymbol{\alpha}^\top [\mathbf{M}(\mathsf{x})]_1 \boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w}) \in \mathbb{G}_1$. Thus, if $\mathsf{x} \in \mathcal{L}$, then $\mathsf{H} = \mathsf{pH}$. It is known that if the underlying GL-DVS is 0-sound, then this is a perfectly smooth GL-SPHF, see Theorem 3.1.11 in [Ham16].

Different from above construction over cyclic groups, our SPHF construction deals with more complex languages over bilinear groups. Namely, we are interested in SPHFs for languages $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\Theta}, \boldsymbol{\Lambda})$ such that $\boldsymbol{\Theta}(\mathsf{x})$ is in the target group, the coefficients of $\mathbf{M}(\mathsf{x})$ (resp. $\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$) belong to the group $\mathbb{G}_\iota$ (resp. $\mathbb{G}_{3-\iota}$) for some $\iota \in \{1, 2\}$, and that $[\boldsymbol{\Theta}(\mathsf{x})]_T = [\mathbf{M}(\mathsf{x})]_\iota [\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota}$. The construction of SPHF for such languages called $\mathsf{HF}_{\mathsf{dvs}}$ is depicted in fig. 1.

- $\mathsf{hashkg}(\mathsf{lpar})$: sample $\boldsymbol{\alpha} \leftarrow\!\!\$\ \mathbb{Z}_p^n$, and output $\mathsf{hk} \leftarrow \boldsymbol{\alpha}$;
- $\mathsf{projkg}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$: $[\boldsymbol{\gamma}]_\iota^\top \leftarrow \boldsymbol{\alpha}^\top [\mathbf{M}(\mathsf{x})]_\iota \in \mathbb{G}_\star^{1 \times k}$; return $\mathsf{hp} \leftarrow [\boldsymbol{\gamma}]_\star$;
- $\mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$: return $[\mathsf{H}]_T \leftarrow \boldsymbol{\alpha}^\top [\boldsymbol{\Theta}(\mathsf{x})]_T$;
- $\mathsf{projhash}(\mathsf{lpar}, \mathsf{hp}, \mathsf{x}, \mathsf{w})$: return $[\mathsf{pH}]_T \leftarrow [\boldsymbol{\gamma}]_\iota^\top [\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota}$;

Fig. 1: DVS-based SPHF construction $\mathsf{HF}_{\mathsf{dvs}}$ for $\mathcal{L}_{\mathsf{lpar}}$ with $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\Theta}, \boldsymbol{\Lambda})$.

In the following theorem, we prove that $\mathsf{HF}_{\mathsf{dvs}}$ in fig. 1 is extractable against algebraic adversaries under the discrete logarithm assumption. To that end, we restate the definition of knowledge smoothness for the languages of our interest, namely $\{\mathcal{L}_{\mathsf{lpar}}\}$, where $\mathsf{lpar} = (\mathbf{M}, \boldsymbol{\Theta}, \boldsymbol{\Lambda})$, and

$$\mathcal{L}_{\mathsf{lpar}} = \left\{ [\boldsymbol{\Theta}(\mathsf{x})]_T : \exists [\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota} \text{ s.t } [\boldsymbol{\Theta}(\mathsf{x})]_T = [\mathbf{M}(\mathsf{x})]_\iota [\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota} \right\}$$

Note that in this case, the extractor is supposed to extract only $[\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota}$ (and not w) such that $([\boldsymbol{\Theta}(\mathsf{x})]_T, [\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_{3-\iota}) \in \mathcal{R}_{\mathsf{lpar}}$.The security guarantee is that for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can compute a valid hash value for an adversarially chosen statement, there exists an efficient extractor that can extract a valid witness for the statement.

In our application, we also need to make sure that the statement chosen by $\mathcal{A}$ satisfies some predicate [10]. To this end, we let $\mathcal{A}_1$ to select the statement by revealing the random coins aux of the statement, instead. The actual statement is then generated by a deterministic instance generator IG that takes aux as input and returns an instance x if the predicate holds.

Knowledge Smoothness. A projective hash function $\mathsf{PHF} = (\mathsf{PGen}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ for $\{\mathcal{L}_{\mathsf{lpar}}\}$ defined by $\mathsf{lpar} = (\mathbf{M}, \mathbf{\Theta}, \boldsymbol{\Lambda})$ is knowledge smooth if for any λ, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ such that $\Pr[\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{PHF},\mathsf{IG}}(\mathcal{A}, \lambda)] \leq \mathsf{negl}(\lambda)$, where $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{PHF},\mathsf{IG}}(\mathcal{A}, \lambda)$ is defined in fig. 2.

```
1. $(\mathsf{pp}, \mathsf{lpar}) \leftarrow \mathsf{PGen}(1^\lambda)$;
2. $\mathsf{aux} \leftarrow \mathcal{A}_1(\mathsf{lpar})$;
3. $\mathsf{x} \leftarrow \mathsf{IG}(\mathsf{aux})$; **if** $\mathsf{x} = \perp$, **return** $0$; **else** $\mathsf{x}' \leftarrow [\mathbf{\Theta}(\mathsf{x})]_T$;
4. $(\mathsf{hp}, \mathsf{hk}) \leftarrow \mathsf{kgen}(\mathsf{lpar}, \mathsf{x}')$;
5. $\mathsf{H} \leftarrow \mathcal{A}_2(\mathsf{lpar}, \mathsf{hp}, \mathsf{aux})$;
6. $\mathsf{H}' = \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x}')$;
7. $\mathsf{w}' \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{lpar}, \mathsf{hp})$;
8. **return** $\big((\mathsf{H} = \mathsf{H}') \wedge (\mathsf{x}', \mathsf{w}') \notin \mathcal{R}_{\mathsf{lpar}}\big)$;
```

Fig. 2: Knowledge smoothness experiment $\mathsf{Exp}^{\mathsf{KS}}_{\mathsf{PHF},\mathsf{IG}}(\mathcal{A}, \lambda)$

Theorem 1. *Let $\mathcal{L}_{\mathsf{lpar}}$ be a language defined by language parameter $\mathsf{lpar} = (\mathbf{M}, \mathbf{\Theta}, \boldsymbol{\Lambda})$. Under the discrete logarithm assumption, $\mathsf{HF}_{\mathsf{dvs}}$ in fig. 1 is an extractable PHF (EPHF) against all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_2$ is algebraic.*

Proof. We prove the theorem for $\iota = 1$; the other case goes exactly in the same way. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPT adversary against the knowledge smoothness of $\mathsf{HF}_{\mathsf{dvs}}$ and assume that $\mathcal{A}_2$ is algebraic. Let x be the statement output by $\mathcal{A}_1$ on input $\mathsf{lpar} = (\mathbf{M}, \mathbf{\Theta}, \boldsymbol{\Lambda})$. $\mathcal{A}_2$ returns a hash value $\mathsf{H} \in \mathbb{G}_T$, and by its algebraic nature, $\mathcal{A}_2$ also provides coefficients that "explain" these elements as linear combinations of the input. Let $[\mathbf{x}]_1 = [x_1, \mathbf{x}_I, \mathbf{x}_J]_1 = [1, \mathbf{M}(\mathsf{x}), \boldsymbol{\gamma}]_1$ and $[y]_2 = [1]_2$ be the input vectors in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $[\mathbf{z}]_T = [\mathbf{\Theta}(\mathsf{x})]_T$ be $\mathcal{A}_2$'s input vector in $\mathbb{G}_T$. The coefficients returned by $\mathcal{A}_2([\mathbf{x}]_1, [y]_2, [\mathbf{z}]_T)$ are $\mathbf{s} = \big((A_i)_i, (B_i)_i\big)$ such that

$$\mathsf{H} = \sum_{i \in [1+|I+J|]} \hat{e}\,(x_i, y)^{A_i} + \sum_{i \in [|\mathbf{z}|]} B_i z_i.$$

Let Ext be the extractor that runs the algebraic adversary $\mathcal{A}_2$ and returns $[\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})]_2 := [\mathbf{A}_J]_2$. We can show that this is a valid witness for $[\mathbf{\Theta}(\mathsf{x})]_T$ as long as the hash value H returned by $\mathcal{A}_2$ is a correct hash. In other words, if $\mathcal{A}_2$ can output H such that $\mathsf{H} = \boldsymbol{\alpha}^\top [\mathbf{\Theta}(\mathsf{x})]_T$, and $\mathbf{\Theta}(\mathsf{x}) \neq \mathbf{M}(\mathsf{x})\boldsymbol{\Lambda}(\mathsf{x}, \mathsf{w})$, we can construct an algorithm $\mathcal{B}$ that exploits $\mathcal{A}_2$ and breaks the discrete logarithm problem. To do this, $\mathcal{B}$ on challenge input $Z = [z]_\iota$ proceeds as follows. First, it uses $\mathsf{D}_{\mathsf{lpar}}$ to sample $\mathsf{lpar} = (\mathbf{M}, \mathbf{\Theta}, \boldsymbol{\Lambda})$. Second, it samples $\mathbf{r}, \mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and implicitly sets $\boldsymbol{\alpha} := z \cdot \mathbf{r} + \mathbf{s}$. Third, it computes $\mathsf{hp} = [\mathbf{M}(\mathsf{x})^\top \boldsymbol{\alpha}]_1$ and runs $\mathcal{A}_2(\mathsf{lpar}, \mathsf{hp}, \mathsf{x})$. Once received $\mathcal{A}_2$'s output H, $\mathcal{B}$ returns $z = (\boldsymbol{\alpha} - \mathbf{s})\mathbf{r}^{-1}$, where $\boldsymbol{\alpha}$ is computed as follows:

$$\begin{aligned}
\boldsymbol{\alpha}^\top \mathbf{\Theta}(\mathsf{x}) - \boldsymbol{\gamma}^\top \mathbf{A}_J &= A_1 + \mathbf{M}(\mathsf{x})\mathbf{A}_I + \mathbf{\Theta}(\mathsf{x})\mathbf{B} \\
&\Rightarrow \boldsymbol{\alpha}^\top (\mathbf{\Theta}(\mathsf{x}) - \mathbf{M}(\mathsf{x})\mathbf{A}_J) = A_1 + \mathbf{M}(\mathsf{x})\mathbf{A}_I + \mathbf{\Theta}(\mathsf{x})\mathbf{B} \\
&\Rightarrow \boldsymbol{\alpha}^\top = \big(A_1 + \mathbf{M}(\mathsf{x})\mathbf{A}_I + \mathbf{\Theta}(\mathsf{x})\mathbf{B}\big)\big(\mathbf{\Theta}(\mathsf{x}) - \mathbf{M}(\mathsf{x})\mathbf{A}_J\big)^{-1}
\end{aligned}$$

□

[10] For example, for $\mathsf{x} = (\mathsf{cm}, G, y)$, the predicate checks if $G(v) \neq y$, where v is committed in cm.

4 WE for Functional Commitment

In this section we define our notion of witness encryption for functional commitments. In standard witness encryption we require semantic security for false statements; in our notion we require semantic security for false statements on committed inputs. The decryption algorithm requires an opening proof of the succinct functional commitment w.r.t. a function and output specified at encryption time. Like other variants of WE [BL20,CDK+21], loses the pure "non-deterministic" flavor of WE since it requires the existence of a commitment to the decryption witness. We refer to the introduction for further intuitions about the notion.

Definition 6 (Witness Encryption for Functional Commitments). *Let* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *be a functional commitment scheme for a circuit class* $\mathbf{CC}$*. A witness encryption for* FC*, denoted by* $\mathsf{WE}^{\mathsf{FC}}$*, for circuit class* $\mathbf{CC}$ *is a tuple of polynomial-time algorithms* $\mathsf{WE}^{\mathsf{FC}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Enc}, \mathsf{Dec})$*, where* $\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}$*, and* Verify *are defined by* FC *and*

Encryption. $\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{m})$ *is a probabilistic algorithm that takes as input the commitment key* ck *(including* $\mathcal{C} \in \mathbf{CC}$*), a statement* $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$*, and a bitstring* m*, and outputs an encryption of* m *under* x*.*

Decryption. $\mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{op})$ *is a deterministic algorithm that on input* ck*, a ciphertext* ct*, a statement* $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$*, and an opening proof* op*, decrypts* ct *into a message* m*, or returns* $\perp$*.*

We require two properties, *correctness* and *semantic security.*

(Perfect) Correctness. For all $\lambda \in \mathbb{N}$, $\mathcal{C} \in \mathbf{CC}$, $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C})$, message m, we have:

$$\Pr\left[\mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathsf{cm}, \boldsymbol{\beta}, \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}), \mathsf{op}) = \mathsf{m} \;:\; \begin{array}{l} (\mathsf{cm}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}), \mathsf{m}) \\ \mathsf{op} \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{d}, \boldsymbol{\beta}) \end{array}\right] = 1$$

Semantic Security. For any λ, any $\mathcal{C} \in \mathbf{CC}$, and PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{WE},\mathsf{FC},\mathcal{C},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{WE},\mathsf{FC},\mathcal{C},\mathcal{A}}(\lambda) :=$

$$\left|\Pr\left[b' = b \;:\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C}); (\boldsymbol{\alpha}, r, \boldsymbol{\beta}, \mathbf{y}, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_1(\mathsf{ck}) \\ (\mathsf{cm}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r); b \leftarrow\!\!\$\ \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{m}_b) \\ \textbf{if } \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{y} \textbf{ then } \mathsf{ct} := \perp; b' \leftarrow \mathcal{A}_2(\mathsf{ct}) \end{array}\right] - 1/2\right|$$

5 Our Construction

In this section we present our construction of WE for functional commitments and then discuss possible instantiations using state-of-the-art functional commitments.

5.1 Building $\mathbf{WE}^{\mathsf{FC}}$ from Extractable Projective Hash

Let $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ be a (succinct) functional commitment scheme for $\mathbf{CC}$, where the verification circuit is linear (i.e., of degree one) in the opening proof. Let $\mathsf{EPHF} = (\mathsf{PGen}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ be an extractable projective hash function. The key idea of the construction is to use EPHF for the language defined by the verification circuit of FC. Since this circuit is affine in the opening proof op, and we know how to construct PHF for affine languages, the witness encryption just uses EPHF in a straightforward way. Note that because the language is trivial, we need knowledge smoothness rather than standard smoothness.

Construction. Let $\mathsf{lpar} = \mathsf{ck}$ be the language parameter that defines $\mathcal{L}_{\mathsf{lpar}}$ corresponding to the verification circuit of FC as follows:

$$\mathcal{L}_{\mathsf{lpar}} = \{\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}) | \exists \mathsf{op} : \mathsf{Verify}(\mathsf{ck}, \mathsf{cm}, \mathsf{op}, \boldsymbol{\beta}, \mathbf{y}) = 1\}$$

Due to linearity of this circuit in the opening op, there exists a matrix $[\mathbf{M}_{\mathsf{lpar},\mathsf{x}}]_\star$ and a vector $[\boldsymbol{\Theta}_{\mathsf{lpar},\mathsf{x}}]_\star$ [11] such that

$$[\boldsymbol{\Theta}_{\mathsf{lpar},\mathsf{x}}]_T = [\mathbf{M}_{\mathsf{lpar},\mathsf{x}} \cdot \widetilde{\mathsf{op}}]_T$$

[11] The star $\star$ means that the elements are not necessarily in the same group.

where $\widetilde{\mathsf{op}}$ is derived from op by replacing its group elements with their discrete logarithms. Let $\sigma : G_T \to \{0,1\}^\ell$ be a generic deterministic injective encoding that maps group elements in $\mathbb{G}_T$ into ℓ-bit strings, and that has an efficient inversion algorithm σ^{-1}. Our WE for functional commitments $\mathsf{WE}^{\mathsf{FC}} = (\mathsf{Enc}, \mathsf{Dec})$ for $\mathcal{L}_{\mathsf{lpar}}$ can be described as follows:

$\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{m})$**.** Let $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$. To encrypt a bit message $\mathsf{m} \in \{0,1\}$, select a uniformly random vector $\mathsf{hk} \in \mathbb{Z}_p^{1\times\nu}$, where ν is the number of rows of $\mathbf{M}_{\mathsf{lpar,x}}$, sample a random $r \leftarrow\!\!\$\ \{0,1\}^\ell$, and compute the ciphertext $\mathsf{ct} = (\mathsf{hp}, r, \widehat{\mathsf{ct}})$, where

$$\mathsf{hp} = [\mathsf{hk}\cdot\mathbf{M}_{\mathsf{lpar,x}}]_\star, \qquad \mathsf{H} = [\mathsf{hk}\cdot\boldsymbol{\Theta}_{\mathsf{lpar,x}}]_T, \qquad \widehat{\mathsf{ct}} = \langle\sigma(\mathsf{H}), r\rangle \oplus \mathsf{m}$$

$\mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{op})$**.** On input a ciphertext $\mathsf{ct} = (\mathsf{hp}, r, \widehat{\mathsf{ct}})$, first compute $\mathsf{pH} = [\mathsf{hp}\cdot\widetilde{\mathsf{op}}]_T$, and then output the message $\mathsf{m} \in \{0,1\}$ computed as $\mathsf{m} = \langle\sigma(\mathsf{pH}), r\rangle \oplus \widehat{\mathsf{ct}}$.

Theorem 2. *Let* FC *be a functional commitment scheme for circuit class* $\mathbf{CC}$ *with computational evaluation-binding property. Let* EPHF *be an extractable projective hash function. The construction of* $\mathsf{WE}^{\mathsf{FC}}$ *described above is a* WE^{FC} *for* $\mathbf{CC}$.

Proof. Perfect correctness follows directly from correctness of FC and EPHF. To prove semantic security, we show a reduction from evaluation-binding of FC to semantic security of $\mathsf{WE}^{\mathsf{FC}}$. To do so, let us assume that $\mathsf{WE}^{\mathsf{FC}}$ is not semantically secure. By definition, there exists an efficient adversary $\mathcal{A}$ that, for a maliciously chosen (false) statement $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$ [12], where $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)$ (all known to $\mathcal{A}$), it can distinguish, with non-negligible advantage, encryptions of 0 and 1 under x. We first show how to construct an efficient algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to compute a hash value $\mathsf{H} = \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$.

Before giving the description of $\mathcal{B}$, let us first recall the classic Goldreich-Levin theorem [GL89] based on which we construct $\mathcal{B}$.

Theorem 3 (Goldreich-Levin). *Let* $\epsilon > 0$. *Fix some* $x \in \{0,1\}^n$ *and let* $\mathcal{A}_x$ *be a PPT algorithm such that* $\Pr[\mathcal{A}_x(r) = \langle r, x\rangle | r \leftarrow\!\!\$\ \{0,1\}^n] \geq 1/2 + \epsilon$. *There exists a decoding algorithm* $D^{\mathcal{A}_x(\cdot)}$ *with oracle access to* $\mathcal{A}_x$ *that runs in* $\mathsf{poly}(n, 1/\epsilon)$*-time and outputs a list* $L \subseteq \{0,1\}^n$ *such that* $|L| = \mathsf{poly}(n, 1/\epsilon)$ *and* $x \in L$ *with probability at least* $1/2$.

The fact that $\mathcal{A}$ can distinguishes ct_0 and ct_1 under $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$ with non-negligible advantage implies that

$$\Pr\left[b' = b \ : \begin{array}{l} b \leftarrow\!\!\$\ \{0,1\}; \mathsf{hk} \leftarrow\!\!\$\ \mathbb{Z}_p^{1\times\nu}; r \leftarrow\!\!\$\ \{0,1\}^\ell; \\ \mathsf{hp} = [\mathsf{hk}\cdot\mathbf{M}_{\mathsf{lpar,x}}]_\star; \mathsf{H} = [\mathsf{hk}\cdot\boldsymbol{\Theta}_{\mathsf{lpar,x}}]_T; \widehat{\mathsf{ct}} = \langle\sigma(\mathsf{H}), r\rangle \oplus b; \\ b' \leftarrow \mathcal{A}(\mathsf{hp}, r, \widehat{\mathsf{ct}}) \end{array} \right] \geq 1/2 + \epsilon$$

for some $\epsilon = 1/p(\lambda)$, where p is a polynomial. We first construct an algorithm $\bar{\mathcal{B}}$ that on input (hp, r) for $r \leftarrow\!\!\$\ \{0,1\}^\ell$, it uses $\mathcal{A}$ to predict the value of $\langle r, \sigma(\mathsf{H})\rangle$. $\bar{\mathcal{B}}$ proceeds as follows: on input (hp, r), it samples $b \leftarrow\!\!\$\ \{0,1\}$ and runs $\mathcal{A}$ on input (hp, r, b). If $\mathcal{A}$ correctly guesses b, $\bar{\mathcal{B}}$ outputs 0, and otherwise 1. By construction, it is easy to see that $\bar{\mathcal{B}}$ outputs $\langle r, \sigma(\mathsf{H})\rangle$ with probability at least $1/2 + \epsilon$. Using $\bar{\mathcal{B}}$ and Goldreich-Levin decoding algorithm $D^{\bar{\mathcal{B}}(\mathsf{hp},\cdot)}$ in theorem 3, we now construct $\mathcal{B}$ that on input lpar, hp and x, computes $\sigma(\mathsf{H})$ as follows:

– Runs $D^{\bar{\mathcal{B}}(\mathsf{hp},\cdot)}$ so that to answer an oracle query $r \in \{0,1\}^\ell$, $\mathcal{B}$ outputs $\bar{\mathcal{B}}(\mathsf{hp}, r)$.
– Let $L \subseteq \{0,1\}^\ell$ be the list that $D^{\bar{\mathcal{B}}(\mathsf{hp},\cdot)}$ outputs. $\mathcal{B}$ returns $\sigma(\mathsf{H}) \leftarrow\!\!\$\ L$.

To analyze the success probability of $\mathcal{B}$, let K be the set of hashing keys $\mathsf{hk} \in \mathbb{Z}_p^{1\times\nu}$ such that for $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$, and $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{lpar}, \mathsf{hk}, \mathsf{x})$,

$$\Pr[\bar{\mathcal{B}}(\mathsf{hp}, r) = \langle r, \sigma(\mathsf{H})\rangle | r \leftarrow\!\!\$\ \{0,1\}^\ell] \geq 1/2 + \epsilon/2.$$

By an averaging argument, the probability that a random $\mathsf{hk} \leftarrow\!\!\$\ \mathbb{Z}_p^{1\times\nu}$ is in K is at least ϵ. This indicates that with probability at least ϵ, the hashing key hk chosen in the knowledge smoothness experiment of EPHF lies in K and hence the oracle $\bar{\mathcal{B}}(\mathsf{hp}, \cdot)$ satisfies the requirement in theorem 3. This subsequently indicates that the list L returned by $D^{\bar{\mathcal{B}}(\mathsf{hp},\cdot)}$ contains $\sigma(\mathsf{H})$ with probability at least 1/2. Therefore, $\mathcal{B}$ computes $\sigma(\mathsf{H})$, and thus H with probability at least $\epsilon\cdot\frac{1}{2}\cdot\frac{1}{|L|}$ which is $\frac{1}{q(\lambda)}$ for some polynomial q. Due to extractability of the EPHF, there should exist an efficient extractor $\mathsf{Ext}_{\mathcal{B}}$ for $\mathcal{B}$ such that for $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)$ and $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}, \mathbf{y})$, $\mathsf{Ext}_{\mathcal{B}}$ can extract a valid witness $\mathsf{w} = \mathsf{op}$ such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}_{\mathsf{lpar}}$ with probability at least $\frac{1}{q(\lambda)}$. The above reduction can subsequently be invoked by a computational evaluation-binding adversary to break this property with non-negligible probability by outputting $(\boldsymbol{\alpha}, r, \boldsymbol{\beta}, \mathbf{y}, \mathsf{op})$. This completes the proof. ☐

[12] Note that x is false in the sense that for $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)$, we have $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \neq \mathbf{y}$. With respect to the language $\mathcal{L}_{\mathsf{lpar}}$ corresponding to the verification of functional commitments, such statements are always true however.

5.2 Instantiations

Let $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^\kappa$ be a polynomial-time circuit that on input $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, outputs $\mathbf{y} = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. In a SFC for $\mathcal{C}$, $\boldsymbol{\alpha}$ can be seen as the committer's secret input, and $\boldsymbol{\beta}$ as the verifier's public input. In the following, we give two instantiations of functional commitments that satisfy the required properties.

Libert et al.'s FC. The seminal work of Libert et al. [LRY16] constructs FC for linear functions $\mathcal{C} : \mathbb{Z}_p^{\mu_\alpha} \times \mathbb{Z}_p^{\mu_\beta} \to \mathbb{Z}_p^\kappa$ where $\mu_\alpha = \mu_\beta = n$ and $\kappa = 1$, and the linear function $\mathcal{C}(\cdot, \boldsymbol{\beta})$ defined as $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^n \boldsymbol{\alpha}_i \boldsymbol{\beta}_i$. While the original construction is over bilinear groups of composite order, here to keep it consistent with the algebraic structure of EPHF in section 3.2, we describe the construction over prime order groups (see [LM19]).

Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing with group order p. The construction is as follows.

$\mathsf{Setup}(1^\lambda, \mathcal{C})$ samples $u \leftarrow\!\!\$\ \mathbb{Z}_p$ and returns ck as

$$\mathsf{ck} := \left(\{[u^j]_1\}_{j \in [2n] \setminus \{n+1\}}, \{[u^j]_2\}_{j \in [n]} \right).$$

The trapdoor key is defined as $\mathsf{td} := [u^{n+1}]_1$.

$\mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r)$ returns $\mathsf{cm} = [r]_1 + \sum_{j \in [n]} \boldsymbol{\alpha}_j \cdot [u^j]_1$ and $\mathsf{d} = (\boldsymbol{\alpha}, r)$.

$\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{d}, \boldsymbol{\beta})$ parses d as $\mathsf{d} = (\boldsymbol{\alpha}, r)$ and for $y = \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$, returns $\mathsf{op}_y = \sum_{i \in [n]} \boldsymbol{\beta}_i \cdot W_i$, where

$$W_i = [u^{n-i+1} r]_1 \cdot \sum_{j \in [n], j \neq i} \boldsymbol{\alpha}_j \cdot [u^{n+1-i+j}]_1.$$

$\mathsf{Verify}(\mathsf{ck}, \mathsf{cm}, \mathsf{op}_y, \boldsymbol{\beta}, y)$ returns 1 if

$$\hat{e}(\mathsf{cm}, \sum_{i \in [n]} \boldsymbol{\beta}_i \cdot [u^{n+1-i}]_2) \stackrel{?}{=} \hat{e}(\mathsf{op}_y, [1]_2) \cdot \hat{e}([u]_1, [u^n]_2)^y$$

It is clear that the verification is linear in the opening proof. To show the construction provides perfect ZK property of definition 3, an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ can be constructed as follows: $\mathcal{S}_1(\mathsf{td})$ first generates $\mathsf{cm} := \mathsf{Commit}(\mathsf{ck}, \mathbf{0}; r)$ and defines $\mathsf{aux} := r$. Now, for any adversarially chosen vector $\boldsymbol{\alpha}$, and any query $\boldsymbol{\beta}$, $\mathcal{S}_2(\mathsf{td}, \mathsf{aux}, \boldsymbol{\beta}, y = \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle)$ returns $\mathsf{op} = (r \sum_{i \in [n]} \boldsymbol{\beta}_i \cdot [u^{n+1-i}]_1) \cdot ([u^{n+1}]_1)^{-y} \in \mathbb{G}_1$.

Lipmaa and Pavlyk's FC. Recently, Limpaa and Pavlyk [LP20] proposed a SFC for a class of circuits $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^\kappa$ based on the SNARK construction of Groth16 [Gro16] for $\mathcal{C}^*$—a compiled version of $\mathcal{C}$. In Groth's SNARK, the argument consists of three group elements $\pi = ([A]_1, [B]_2, [C]_1)$. The key idea in SFC of [LP20] is as follows: first, express the first two elements $[A]_1, [B]_2$ as sums of two elements where the first depends only on the secret data and the second depends only on the public data (i.e., $\boldsymbol{\beta}$ and the output $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$). That is, $[A]_1 = [A_s]_1 + [A_p]_1$ and $[B]_2 = [B_s]_2 + [B_p]_2$. Next, write $[C]_1$ as $[C_{sp}]_1 + [C_p]_1$, where $[C_p]_1$ depends only on the public data and $[C_{sp}]_1$ depends on both the public and the secret data. Now, a functional commitment to $\boldsymbol{\alpha}$ is $\mathsf{cm} = ([A_s]_1, [B_s]_2)$ and the opening to $\mathcal{C}(\cdot, \boldsymbol{\beta})$ is $\mathsf{op} = [C_{sp}]_1$. To verify an opening op, the verifier computes $[A_p]_1$, $[B_p]_2$, and $[C_p]_1$, and then runs the SNARK verifier on the argument $([A_s]_1 + [A_p]_1, [B_s]_2 + [B_p]_2, [C_{sp}]_1 + [C_p]_1)$. The construction is shown to be perfectly zero-knowledge as defined in [LP20], but it is not hard to show that it satisfies our stronger definition (i.e., definition 3) as well. In fact, given $\mathsf{td} = (u, v)$ as the trapdoor of the commitment key, $\mathcal{S}_1(\mathsf{td})$ generates the commitment as the first step of the SNARK simulation in [LP20] and defines aux as the discrete logarithm of the commitment. Now, $\mathcal{S}_2$ can utilises aux and answer oracle queries for different circuits $\mathcal{C}(\cdot, \boldsymbol{\beta})$ by performing the rest of the SNARK simulation. Given that the verification in the SNARK of [Gro16] is linear in the opening $[C_{sp}]_1$ makes this functional commitment an appropriate instantiation for our construction of WE^{FC}.

6 From WE^{FC} to Reusable Non-Interactive MPC

6.1 Preliminaries on mrNISC

Here we first recall the definition of mrNISC schemes in [BL20] and their construction based on $\text{WE}^{\text{ZK-CM}}$. We then show how our notion of WE^{FC} can be used as a replacement of $\text{WE}^{\text{ZK-CM}}$ in their construction.

There are two rounds in mrNISC-style variant of secure multiparty computation protocols, *input encoding phase* and *evaluation phase*. In the first round, parties publish encodings of their secret inputs on a public bulletin board, without any coordination with other parties. This happens once and for all. Next, in the second round, any subset of parties can compute a function on their inputs by publishing only one message each. More formally, a mrNISC scheme is defined by the following three algorithms:

Input Encoding $(\hat{x}_i, s_i) \leftarrow \mathsf{Commit}(1^\lambda, x_i)$ by which a party P_i encodes its private input x_i and publishes the encoding $\hat{x}_i$.

Computation Encoding $\eta_i \leftarrow \mathsf{Encode}(z, \{\hat{x}_j\}_{j\in J}, s_i)$ by which each party P_i among a subset of parties $\{P_j\}_{j\in J}$ generates and publishes a computation encoding η_i. This allows parties in J to compute a functionality f described by z (i.e., $f(z,\star)$) on their private inputs.

Output $y = \mathsf{Eval}(z, \{\hat{x}_j\}_{j\in J}, \{\eta_j\}_{j\in J})$ which deterministically computes the output y (required to be $f(z, \{\hat{x}_j\}_{j\in J})$ by the correctness property).

The construction of mrNISC in [BL20] is based upon the work of [GLS15], where they follow the *round collapsing* approach for constructing 2-round MPC protocols used in [GGHR14]. Let $\prod$ be an L-round MPC protocol. The round collapsing approach collapses $\prod$ into a 2-round protocol $\overline{\prod}$ as follows. For $\ell \in [L]$, let m_i^ℓ denote the message published by party P_i in round ℓ of $\prod$. Let x_i and r_i be respectively the secret input and random tape of P_i used to execute $\prod$. In the first round of $\overline{\prod}$, each party P_i commits to its private input (x_i, r_i) and broadcasts the resulting commitment cm_i. In the second round, each party P_i garbles its next-step message function F_i^ℓ in $\prod$ for each round $\ell \in [L]$. Note that the resulting garbled circuit, denoted by $\hat{F}_i^\ell$, should take as input all the messages $\mathbf{m}^{<\ell} = \{m_j^\ell\}_{l<\ell, j\in[n]}$ of all parties up to round $\ell-1$, and outputs the next message m_i^ℓ of P_i in $\prod$. To do so, each P_i should provide a way for other parties to compute the labels of $\hat{F}_i^\ell$ that correspond to the correct messages in $\prod$, where a message m_j^l is correct if it is computed from P_i's committed messages (x_i, r_i) in the first round. To this end, [GLS15] suggests the following mechanism: let k_0 and k_1 be two labels for an input wire in P_i's garbled circuit $\hat{F}_i^\ell$. Suppose that $\hat{F}_i^\ell$ takes as input the t'th bit $y = m_{j,t}^l$ of a message from P_j (where m_j^l is output by P_j's garbled circuit $\hat{F}_j^l$), and provides a way for all parties to obtain the valid label k_y. The key idea in [GLS15] is to use a general-purpose WE to produce a ciphertext $\mathsf{ct}_y \leftarrow \mathsf{WE.Enc}(\mathsf{x}_y, k_y)$ for $y \in \{0,1\}$ under the statement x_y that "there exists a NIZK proof π_y that proves $y = m_{j,t}^l$ is computed correctly". Again, correct computation here means that y is computed from P_j's committed messages (x_i, r_i) in the first round, and in accordance to the partial transcript of messages $\mathbf{m}^{<l}$. The two ciphertexts $(\mathsf{ct}_0, \mathsf{ct}_1)$ are part of what P_i in the garbled circuit $\hat{F}_i^{l-1}$ outputs. Furthermore, to allow all parties to (publicly) obtain the correct label k_y, P_j's garbled circuit $\hat{F}_j^l$ additionally outputs a NIZK proof π_y that $y = m_{j,t}^l$ is correctly computed. Correctness of $\overline{\prod}$ follows from correctness of WE. Security also follows from the fact that k_{1-y} remains hidden by the soundness of NIZK and semantic security of WE. Furthermore, the ZK property of NIZK guarantees the privacy of parties.

The main problem in the above construction of [GLS15] is the lack of general-purpose WE from standard assumptions. Benhamouda and Lin [BL20] overcome this problem by observing that not a WE for general NP language, but a WE scheme for a particular language corresponding to the verification circuit of a NIZK proof that proves the correctness of computation over committed information suffices to realize the above construction. This variant of WE, denoted by $\mathrm{WE}^{\text{ZK-CM}}$ in this work, consists of a triple $\mathsf{WE}^{\mathsf{ZK\text{-}CM}} = (\mathsf{COM}, \mathsf{NIZK}, \mathsf{WE})$ and is defined for a NP language $\mathcal{L}$ such that a statement $\mathsf{x} = (\mathsf{cm}, G, y)$ is in $\mathcal{L}$ iff there exists an accepting NIZK proof π (as the witness for x) w.r.t. crs that proves cm is a commitment of some value v and that $G(v) = y$. As provided in the construction of [GLS15] (but based on stronger assumption of general-purpose WE), $\mathrm{WE}^{\text{ZK-CM}}$ should support all polynomial computations; i.e., it should be that G in the statements $\mathsf{x} = (\mathsf{cm}, G, y)$ can be any arbitrary polynomial-sized circuit. Moreover, the commitments in $\mathrm{WE}^{\text{ZK-CM}}$ should be reusable in the sense that generating unbounded number of NIZK proofs and WE ciphertexts w.r.t. commitments should not reveal any information about the committed (secret) values, except what is revealed by the statements. Equipped with this property then allows to make the construction of [GLS15] reusable by replacing r_i with a PRF seed s_i that generates pseudo-random tapes for an unbounded number of computations. The key idea in the construction of $\mathrm{WE}^{\text{ZK-CM}}$ in [BL20] is to use a NIZK proof system that has a linear-decision verification. Given such NIZK is then sufficient to realize $\mathrm{WE}^{\text{ZK-CM}}$ using a WE for linear languages which can be constructed efficiently based on smooth projective hash functions (SPHFs). In more details, let $\mathbf{\Theta} = \mathbf{M}\pi$ be the linear equation corresponding to the verification of NIZK for a statement $\mathsf{x} = (\mathsf{cm}, G, y)$, such that $\mathbf{\Theta}$ and $\mathbf{M}$ only depend on x and thus are known at the time of encryption. One can now encrypt a message straightforwardly by using an SPHF for this relation such that only one who can compute the hash value using a valid witness π (i.e., $\mathbf{\Theta} = \mathbf{M}\pi$) can retrieve the message.

6.2 Our mrNISC construction.

We now show how one can replace $\mathrm{WE}^{\text{ZK-CM}}$ with WE^{FC} in the aforementioned construction. Let FC be a succinct functional commitment for circuit class $\mathbf{CC}$, and $\mathsf{WE}^{\mathsf{FC}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Enc}, \mathsf{Dec})$ be a WE^{FC} for $\mathbf{CC}$ constructed as in section 5. Besides $\mathsf{WE}^{\mathsf{FC}}$, the construction uses the following building blocks:

Hardwired Values. $1^\lambda, \ell, i, z, \{\hat{x}_j = \mathsf{cm}_j\}_{j \in J}, s_i = (x_i, fk_i, \mathsf{d}_i), \mathbf{stE}_i^{\ell+1}, \{\mathbf{msgE}_{i,j}^{\ell+1}\}_{j \in J}$.

Circuit Inputs. $(\mathbf{m}^{<\ell-1}, \mathbf{m}^{\ell-1})$, where $\mathbf{m}^{<\ell-1}$ are the protocol messages of the first $\ell - 2$ rounds with corresponding garble labels $\mathbf{stE}_i^\ell$, and $\mathbf{m}^{\ell-1}$ are the messages of the $\ell - 1$ round with corresponding garble labels $\{\mathbf{msgE}_{i,j}^\ell\}_{j \in J}$.

Procedure. 1. For $j \in J$ and $k \in [\nu_m]$, define the circuits G_j^ℓ and $G_{j,k}^\ell$ as follows:

$$G_j^\ell(x_j, fk_j) = \mathsf{Next}_j(z, x_j, \mathsf{PRF}(fk_j, z||[\nu_r]), \mathbf{m}^{<\ell-1}, \mathbf{m}^{\ell-1}) \ ; \qquad G_{j,k}^\ell := k\text{-th bit of } G_j^\ell$$

2. Compute the ℓ-th round message $m_i^\ell = m_{i,1}^\ell || \dots || m_{i,\nu_m}^\ell$ of P_i, and proofs of correct openings $\mathsf{op}_{i,k}^\ell$ for each bit $k \in [\nu_m]$:

$$m_i^\ell := G_i^\ell(x_i, fk_i) \ ; \mathsf{op}_{i,k}^\ell \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{cm}_i, s_i, G_{i,k}^\ell) \qquad \textbf{for } k \in [\nu_m].$$

3. For $j \in J$ and $k \in [\nu_m]$, encrypt labels $\mathbf{msgE}_{i,j}^{\ell+1}[k, b]$ so that the valid message m_j^ℓ can be used to obtain $\mathbf{msgE}_{i,j}^{\ell+1}[m_j^\ell] = \{\mathbf{msgE}_{i,j}^{\ell+1}[k, m_{j,k}^\ell]\}_{k \in [\nu_m]}$:

$$\mathsf{ct}_{i,j,k,b}^{\ell+1} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{cm}_j, G_{j,k}^\ell, b, \mathbf{msgE}_{i,j}^{\ell+1}[k, b])^a \qquad \textbf{for } b \in \{0, 1\}.$$

Circuit Output. $(\mathbf{stE}_i^{\ell+1}[\mathbf{m}^{<\ell-1}||\mathbf{m}^{\ell-1}], \{\mathsf{ct}_{i,j,k,b}^{\ell+1}\}_{j,k,b}, m_i^\ell, \{\mathsf{op}_{i,k}^\ell\}_k)$.

[a] The ciphertexts are set to be empty strings for $\ell = L$.

Fig. 3: Circuit $\mathbf{F}_i^\ell$ for the construction of mrNISC based on $\mathrm{WE}^{\mathrm{FC}}$

– A semi-malicious output-delayed simulatable L-round MPC protocol $\prod = (\mathsf{Next}, \mathsf{Output})$ for f (see definition 7).
– A garbled circuit scheme $\mathsf{GC} = (\mathsf{Gen}, \mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ for $\mathbf{CC}$ (see definition 10).

The construction is as follows:

Input Encoding. For a binary input x_i and PRF key $fk_i \leftarrow\$ \{0,1\}^\lambda$, party P_i commits to $x_i||fk_i$ as $(\mathsf{cm}_i, \mathsf{d}_i) \leftarrow \mathsf{Commit}(\mathsf{ck}, (x_i||fk_i); r_i)$. It then sets $\hat{x}_i := \mathsf{cm}_i$ and $s_i := (x_i, fk_i, \mathsf{d}_i)$.

Computation Encoding. To encode a computation $f(z, \star)$, each party P_i for $\ell \in [L]$ generates input labels $(\mathbf{stE}_i^\ell, \{\mathbf{msgE}_{i,j}^\ell\}_{j \in J}) \leftarrow \mathsf{Gen}(1^\lambda)$ and garbles the evaluation function $\mathbf{F}_i^\ell$ (defined in fig. 3) as $\hat{\mathbf{F}}_i^\ell \leftarrow \mathsf{Garble}((\mathbf{stE}_i^\ell, \{\mathbf{msgE}_{i,j}^\ell\}_{j \in J}), \mathbf{F}_i^\ell)$. Finally, it sets $\eta_i := \{\hat{\mathbf{F}}^\ell{}_i\}_{\ell \in [L]}$.

Output. The output is computed by recovering the input labels and then evaluating the garbled circuits on them in L iteration. That is, for $\ell = 1, \dots, L$:

1. For $i \in J$,

$$\left(\mathbf{stE}_i'^{\ell+1}, \{\mathsf{ct}_{i,j,k,b}^\ell\}_{j,k,b}, m_i^\ell, \{\mathsf{op}_{i,k}^\ell\}_k\right) := \mathsf{Eval}(\hat{\mathbf{F}}_i, (\mathbf{stE}_i'^\ell, \{\mathbf{msgE}_{i,j}^\ell[m_j^{\ell-1}]\}_{j \in J})).$$

2. If $\ell \neq L$, then for $i, j \in J$ and $k \in [\nu_m]$,

$$\mathbf{msgE}_{i,j}^{\ell+1}[m_j^\ell] := \left\{\mathsf{Dec}(\mathsf{ck}, \mathsf{ct}_{i,j,k,m_{j,k}^\ell}^{\ell+1}, \mathsf{cm}_j, G_{j,k}^\ell, m_{j,k}^\ell, \mathsf{op}_{i,j,k}^\ell)\right\}$$

After all the messages $\mathbf{m} = \{m_j^\ell\}_{j \in J, \ell \in [L]}$ of the inner MPC are recovered, the final output is computed as $y := \mathsf{Output}(z, \mathbf{m})$.

The correctness of the construction follows straightforwardly from the correctness of the underlying building blocks. For security, we refer to [BL20] as the proof is similar to the security of the mrNISC construction in [BL20]. Here, we only state the theorem.

Theorem 4. *Let* PRF *be a pseudorandom function,* GC *be a garbled circuit with simulatability property (see 10),* $\prod$ *be a semi-malicious output-delayed simulatable MPC protocol (see 7), and* $\mathsf{WE}^{\mathsf{FC}}$ *be a WE*^{FC} *for* **CC** *with semantic security (see 4). The* mrNISC *scheme described above is semi-maliciously private as defined in A.3.*

6.3 On the Instantiation of our mrNISC Construction

The main advantage of our mrNISC construction compared to the one in [BL20] is that our approach admits an input encoding phase with much shorter communication since we use succinct commitments. This is especially important since commitments are supposed to be stored in a public bulletin board to be re-used in several future computations. Next, we discuss the requirements to instantiate our scheme.

In order to instantiate our mrNISC construction, we notice that it is possible to use a $\mathrm{WE}^{\mathrm{FC}}$ for circuits in NC^1, rather than a $\mathrm{WE}^{\mathrm{FC}}$ for arbitrary polynomial-time computations.

To see this, we observe that in the concrete example of MPC protocols based on Beaver's technique for multiplication gates, it is possible to write the gate computations as a linear combination of some Beaver triples. Namely, the next-message function Next_j is a linear function over a finite field and can be computed in NC^1. Second, in order to define the function G_j^ℓ, one needs to compose Next_j with the function computing the PRF used in the mrNISC construction. One can use a PRF that is computable in NC^1. Finally, the last step consists into extracting the output bit-by-bit, which comes for free if G_j^ℓ is computed via a boolean circuit.

As observed, in Section 5.2, the state of the art of functional commitments that we can use to instantiate our $\mathrm{WE}^{\mathrm{FC}}$ construction (i.e., those having a pairing-based verification linear in the proof elements) can only support a class of *linearizable functions* [LP20] over a large finite field. These are functions where outputs can be written as $\sum_{i,j} \phi_i(\boldsymbol{\alpha})\psi_j(\boldsymbol{\beta})$ for some efficiently computable polynomials ϕ_i, ψ_j, but do not seem powerful enough to capture boolean circuits in the class NC^1. We leave the construction of such more expressive functional commitments based on pairings and with linear verification as an interesting open problem. Given the very recent progress in enhancing the expressivity of functional commitments (see Section 1.4), we believe it might be a feasible achievement.

7 Other Application Scenarios

In this section we show that our notion of $\mathrm{WE}^{\mathrm{FC}}$can be versatile; we describe how it can be used in other scenarios besides mrNISC.

7.1 Targeted Broadcast

As a first application scenario, we discuss how to apply $\mathrm{WE}^{\mathrm{FC}}$ to a targeted broadcast with "special properties". See last item in section 1.2 for a description of the problem, but a quick summary is: we aim at encrypting a message with respect to some attributes (not necessarily known before encryption time); only users holding those attributes can decrypt (we discuss later how they are granted).

This subsection proceeds in three parts. We first give a flavor of our approach template, which we call "commit-and-receive" since it involves a commitment to user attributes which allows them to decrypt to compatible messages. We then argue what properties make this approach interesting compared to the more standard targeted broadcast setting. Finally we compare to alternative approaches in more detail.

Our Approach: Commit-and-Receive. We now describe our general approach. To better provide an intuition for it, we start with a flavor of which *settings* it is suitable for; this is best introduced through a specific toy example. Consider a sophisticated programming contest where participants are asked to write a program solving a specific algorithmic problem. To evaluate each submission, it is common for the organizers to execute the program against several test cases (not public before submission deadline). If submission passes enough test-cases, the sender can receive instructions to move on to the next stage (or receive a digital prize, e.g. a full copy of TAOCP[13]). If the participants want to keep their code secret, can their program still be tested and receive the instructions/prize? There are arguably other natural settings besides this one [14].

[13] The Art Of Computer Programming (TAOCP) by Donald E. Knuth `https://www-cs-faculty.stanford.edu/~knuth/taocp.html`.

[14] Another straightforward example for our setting is that of *lotteries*. Each party commits to a lottery number (or through an identifier sampled in some manner), then a draw occurs and only the winner(s) can obtain a certain message, e.g., a digital prize or some other message. The lottery setting while simple is actually quite concretely practical, for instance in *proofs of stake* [DPS19]. The problem of commit-and-receive can be seen as a a more general version of the primitive "Encryption to the Current Winner" (ECW) defined in [CDK+21] in the context of proofs of stake. In fact the solution described in this section can be leveraged as a construction for ECW with short commitments.

We aim at providing a solution for a generalized version of the setting above with particular attention at *minimizing round interaction.* We call our approach "Commit-and-Receive"[15] and we show how it can be naturally built through our primitive $\mathrm{WE}^{\mathrm{FC}}$.

Our approach, described through the lens of the application example above: consider a party R interested in receiving some message (e.g. a digital good) from a sender S. The latter would like the message to be received by R only if some data D_R held by R satisfy a certain policy (e.g. the tests determined which programs will pass the contest or the drawn lottery number). The data D_R are committed beforehand and the policy is not chosen adaptively and thus possibly not known at commitment time. After each participant has published a commitment cm_i to their program, the organizers can broadcast $\vec{\mathsf{ct}} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ with

$$\mathsf{ct}_i \leftarrow \mathsf{WE}^{\mathsf{FC}}.\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}_i, F_{\mathrm{tests}}, m)$$

where m contains further instructions or a digital prize and F_{tests} is a function checking if tests are passed. The participants whose solutions do not pass the tests will not be able to decrypt their respective ciphertext.

Motivating our approach: more flexibility, more privacy, less trust. We argue that the approach to targeted broadcasting we just described is of interest because of three properties:

1) Flexible attestation What is left undiscussed above is how, in general, the list of commitments $(\mathsf{cm}_1, \mathsf{cm}_2, \ldots)$ available to the sender is updated or comes to be. Will a party with a special authority have to issue each of them or could commitments be registered through some form of consensus? A commit-and-receive approach is flexible in that respect because it enables different solutions in that spectrum. We call the process of updating this list *attestation* since, once in the list, commitments are close to handles for *some* user's identity, which becomes attested once part of the list.
In fig. 4 we give some intuitive examples of how this process may work, from the more centralizing/requiring trust in specific parties, to the more decentralized. We assume an abstract informal interface $\mathsf{AddUserComm}/\mathsf{VfyUpdUsers}$ for adding a user commitment to the list and verify&update such list.
The first approach (fig. 4a) shows the case where users may receive explicit attestation by one in a network of authorities. This is close to the approach common in ABE—standard ABE corresponds to the special case where there is only one authority. In some settings, we may do without an authority by allowing users to perform attestation by showing their data satisfy a minimal general property (fig. 4b), e.g. the format of an identification document (this is not the property required for decryption but one common to all committed values). Pushing this to extreme, users can also be allowed to register themselves (fig. 4c). We expand on application scenarios and other caveats in appendix C.

2) No key escrow Our approach does not require a party holding a global secret that allows decryption of all ciphertexts. In this respect, some of our examples in fig. 4 resemble the approach used in registration-based encryption which achieve the special case of IBE (rather than general attributes) without a master secret key [GHMR18].

3) Attribute-hiding Since the users' communication handles—their commitments—have hiding properties, they may be able to keep their content completely secret. This is true in self-attestation approaches (fig. 4b and fig. 4c) where no authority has access to their attributes through an attestation procedure where they explicitly show attributes or through a master secret that acts as a trapdoor.

Comparing Commit-and-Receive to Alternative Approaches

A naive solution based on zero-knowledge As a starting point of comparison, we observe that another simple solution to the problem could have the receiving parties publish a (possibly zero-knowledge) proof that their data satisfy the policy. The sender could then send the information only to the parties with a valid proof. However, this solution clearly requires additional rounds of interaction if the policy is not known at commitment time or is adaptively chosen. This is, for example, the case in a programming contest (test cases cannot be known in advance), lotteries or whenever we want to reuse that same commitment stage for multiple rounds. Notice that this solution is different than the one we propose in fig. 4b since there we are not proving the property required for decryption but a once-and-for-all simple property of the data D (e.g., the format of structure of the data).

FHE Compact FHE [Gen09] could be used as a non-interactive solution, but at the price of significantly worse efficiency in some settings. For example, in the programming contest case, each participant can generate an FHE key-pair and encrypt their submission with respect to it. The organizers can then run $\mathsf{ct}_i \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, \mathsf{ct}^{(i)}_{\mathrm{subm}}, G^*_m)$ where $G^*_m(P)$ is a function that returns m if P passes the tests

[15] The name is a variant of "commit-and-prove" as used in [CFQ19,Lip16].

```
$\mathsf{AddUserComm}(D) \rightarrow (\mathsf{cm}', \sigma)$
    One of the valid authorities
    first produces and then signs
    a commitment $\mathsf{cm}'$ to the data
```

```
$\mathsf{VfyUpdUsers}(\mathsf{cm}, \sigma) \rightarrow \mathsf{pp}'_{\text{users}}$
    Signature of authority is checked;
    if valid, the commitment list
    is updated appropriately
```

(a) Attestation through authority. It assumes there is a pre-established agreement on which entities are authorized to approve attestation after seeing a commitment.

```
$\mathsf{AddUserComm}(D) \rightarrow (\mathsf{cm}', \mathsf{aux}_{\text{add}})$
    $\mathsf{cm}' \leftarrow \mathsf{FC.Commit}(\mathsf{ck}, D)$
    // ZK proof of valid structure of data
    $\pi_{\text{fmt}} \leftarrow \mathsf{ZK.Prove}(R_{\text{fmt}}, D, \dots)$
    return $(\mathsf{cm}', \pi_{\text{fmt}})$
```

```
$\mathsf{VfyUpdUsers}(\mathsf{cm}, \pi_{\text{fmt}}) \rightarrow \mathsf{pp}'_{\text{users}}$
    if ZK proof verifies then
        return $\mathsf{pp}_{\text{users}} \cup \{\mathsf{cm}'\}$
    else
        return $\mathsf{pp}_{\text{users}}$
```

(b) Self-attestation with minimal validation

```
$\mathsf{AddUserComm}(D) \rightarrow \mathsf{cm}'$
    return $\mathsf{cm}' \leftarrow \mathsf{FC.Commit}(\mathsf{ck}, D)$
```

```
$\mathsf{VfyUpdUsers}(\mathsf{pp}_{\text{users}}, \mathsf{cm}') \rightarrow \mathsf{pp}'_{\text{users}}$
    // no checks
    return $\mathsf{pp}_{\text{users}} \cup \{\mathsf{cm}'\}$
```

(c) Self-attestation

Fig. 4: Examples of flexibility in attestation (informally described). We assume all protocols in the interface take as input static public parameters such as commitment keys, etc. We denote by D the committed data. The set $\mathsf{pp}_{\text{users}}$ denotes the list of already registered commitments $= (\mathsf{cm}_1, \mathsf{cm}_2, \dots)$.

and $\perp$ otherwise. This solution satisfies the same security goal as the one based on $\mathrm{WE}^{\mathrm{FC}}$—as a circuit private FHE ensures that ct_i does not reveal information on m—but requires communication at least linear in the length of the committed data. In particular, in our example each participant must send an FHE encryption of P, which is at least $|P|$-bits long, whereas by using $\mathrm{WE}^{\mathrm{FC}}$ they only need to send a succinct commitment to their solution P.

ABE Our application is closely related to the "targeted broadcast" in [GPSW06] (based on ciphertext-policy ABE) and in general to "Decentralized" ABE [LW11]. Differences in approach and scope are the following. First, we want to account for a wider class of settings where it is acceptable for users to self-attest or the attesting could work through other mechanisms: by design, our approach keeps abstract the attribute registration stage (i.e., how user commitments are registered). This allows more flexibility than ABE and its variants where there is a clear structure of authority/authorities providing access keys. Second, differently from ABE, our solutions do not have secrets (e.g., the master secret key) that allow to decrypt all ciphertexts. *Tradeoffs in Communication Complexity:* The ciphertext size in our approach grows in the number of commitments that are of interest for a certain plaintext. This may not be practical in large networks and where there is no way to discriminate users (and respective commitments) of interest for a given plaintext. This, however, does not necessarily make this approach worse than other systems. In particular, (non-threshold) ABE systems have a ciphertext size that depends on the policy/attribute size. Our ciphertexts do not. They may then offer better bandwidth for the setting of large computations/data with a modest amount of users.

7.2 Simple Contingent Payment for Services

The next application setting we describe has to do with a form of conditional payments. Imagine we want to incentivize the availability of some large data (Internet Archive, Wikipedia, etc.). One approach to (publicly) check data availability uses some variant of this approach: for the data D there exists a public, *succinct* commitment (e.g., a Merkle Tree or a functional commitment compatible with $\mathrm{WE}^{\mathrm{FC}}$ in our case); once every epoch, a verifier samples random indices $r_1, \dots, r_m \leftarrow\!\$\ [|D|]$; a storage provider shows an opening (e.g., Merkle tree paths) to the values $D[r_1], \ldots, D[r_m]$. If carried out enough times and appropriately choosing m, this procedure can guarantee data availability with low communication [JK07]. Notice that the use of succinct commitments is essential in such an application: if verification requires the same amount of storage as the data D, one may be better off storing D.

There are several approaches to incentivizing availability without the need of interaction from the party interested in keeping the data available (which we call stakeholder in the remainder). Several of these approaches involve embedding incentives in the mining process in a blockchain (e.g., Filecoin) or letting a smart contract (e.g., on Ethereum) unlock a reward[16] if the verification process above succeeds. Other solutions apply threshold cryptography requiring a set of parties to be available and act as decryptor oracles [KAS⁺18]. Through $\mathrm{WE}^{\mathrm{FC}}$, we can achieve a simpler solution that does not rely on threshold networks, a specific blockchain architecture or smart contracts (convenient both in terms of gas costs, simplicity and communication complexity on chain) . The solution is as follows. The stakeholder produces a vector of random indices $\vec{r}$ as above and produces the ciphertext

$$\mathsf{ct} \leftarrow \mathsf{WE}^{\mathsf{FC}}.\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}_D, F_{\vec{r}}, \mathsf{k}_{\mathrm{ca\$h}})$$

[17] where cm_D is a commitment to the data, $F_{\vec{r}}$ is a selector function—$F_{\vec{r}}(D) := (D[r_1], \ldots, D[r_m])$—and $\mathsf{k}_{\mathrm{ca\$h}}$ is the message we are encrypting, that is a secret that allows access to the reward (e.g., a Bitcoin private key). Further subtleties of this approach are discussed in appendix D. We believe the solution above can be applied generically to other natural settings.

An important note is that for the approach above to work we need a stronger variant of $\mathrm{WE}^{\mathrm{FC}}$ in which (a) the encryptor does not need to know the output of the function used in the statement, and (2) security has an *extractability* flavor which ensures that a successful decryptor will actually know the output of $F_{\vec{r}}(D)$ for committed data D. In appendix B, we define this variant of $\mathrm{WE}^{\mathrm{FC}}$ and show that our same construction of Section 5 can be proven to have this stronger property.

References

ABP15. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 69–100. Springer, Heidelberg, April 2015.

ACL+22. Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based snarks: Publicly verifiable, preprocessing, and recursively composable. In *CRYPTO 2022*, 2022.

BBC+13. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

BCFL22. David Balbás, Dario Catalano, Dario Fiore, and Russell W. F. Lai. Functional commitments for circuits from falsifiable assumptions. Cryptology ePrint Archive, Report 2022/1365, 2022. `https://eprint.iacr.org/2022/1365`.

BF03. Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

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

BIOW20. Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 776–806. Springer, Heidelberg, August 2020.

BJKL21. Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 724–753. Springer, Heidelberg, October 2021.

BL20. Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, November 2020.

[16] See respectively `https://docs.filecoin.io/about-filecoin/what-is-filecoin/` and `https://thegraph.com/docs/en/about/`

[17] We use a slightly different syntax than the usual one, which we explain later (see also appendix B).

CDK$^+$21. Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders K. Kristensen, and Jesper Buus Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. *IACR Cryptol. ePrint Arch.*, page 1423, 2021.

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

CFT22. Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In *ASIACRYPT 2022*, 2022. `https://eprint.iacr.org/2022/1331`.

CGGN17. Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 229–243. ACM Press, October / November 2017.

CH20. Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 768–798. Springer, Heidelberg, August 2020.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

CVW18. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018.

dCP22. Leo de Castro and Chris Peikert. Functional commitments for all functions, with transparent setup. Cryptology ePrint Archive, Paper 2022/1368, 2022. `https://eprint.iacr.org/2022/1368`.

dec22. Decentralized storage. `https://ethereum.org/en/developers/docs/storage/`, 2022.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DPS19. Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 23–41. Springer, Heidelberg, February 2019.

FJK21. Rex Fernando, Aayush Jain, and Ilan Komargodski. Maliciously-secure mrnisc in the plain model. *IACR Cryptol. ePrint Arch.*, page 1319, 2021.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GGH$^+$13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGHR14. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014.

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

GHMR18. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 689–718. Springer, Heidelberg, November 2018.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

GL06. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information and System Security*, 9(2):181–234, 2006.

GLS15. S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015.

GLW14. Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 426–443. Springer, Heidelberg, August 2014.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

Ham16. Fabrice Ben Hamouda-Guichoux. *Diverse modules and zero-knowledge.* PhD thesis, École Normale Supérieure, Paris, France, 2016.

JK07. Ari Juels and Burton S. Kaliski Jr. Pors: proofs of retrievability for large files. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 584–597. ACM Press, October 2007.

KAS+18. Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Sandra Deepthy Siby, Nicolas Gailly, Linus Gasser, Philipp Jovanovic, Ewa Syta, and Bryan Ford. CALYPSO: Auditable sharing of private data over blockchains. Cryptology ePrint Archive, Report 2018/209, 2018. `https://eprint.iacr.org/2018/209`.

Kho22. Hamidreza Khoshakhlagh. (Commit-and-prove) predictable arguments with privacy. In Giuseppe Ateniese and Daniele Venturi, editors, *ACNS 22*, volume 13269 of *LNCS*, pages 542–561. Springer, Heidelberg, June 2022.

Lip16. Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 185–206. Springer, Heidelberg, April 2016.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019.

LP20. Helger Lipmaa and Kateryna Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 686–716. Springer, Heidelberg, December 2020.

LRY16. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

LW11. Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, Heidelberg, May 2011.

RSW96. Ron Rivest, Adi Shamir, and David Wagner. Time lock puzzles and timed release cryptography. Technical report, Technical report, MIT/LCS/TR-684, 1996.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

Wee10. Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, Heidelberg, August 2010.

Supporting Material

A Additional Preliminaries

A.1 Output-delayed Simulatable MPC

Here we give the formal definition of a special MPC protocol with *output-delayed simulatability* property which guarantees that all the messages except the last one can be simulated for all-but-one honest parties before knowing the output. This is required as the adversary in a mrNISC protocol learns the output only when all the honest parties agreed to provide a computation encoding.

Definition 7 (MPC Protocol). *Let $\boldsymbol{\mathcal{F}}$ be a class of functions. An L-rounds MPC scheme $\prod = (\mathsf{Next}, \mathsf{Output})$ for $\boldsymbol{\mathcal{F}}$ between n parties consists of two PPT algorithms:*

Next message. $m_i^\ell := \mathsf{Next}_i(1^\lambda, 1^n, z, x_i, r_i, \mathbf{m}^{<\ell})$ *is the message broadcasted by party P_i in round $\ell \in L$, on public input z, private input x_i, randomness $r_i \in \{0,1\}^{\nu_r}$, and received messages $\mathbf{m}^{<\ell} = \{m_j^{\tilde{\ell}}\}_{j\in[n],\tilde{\ell}<\ell}$, where $m_j^{\tilde{\ell}}$ is the message broadcasted by P_j on round $\tilde{\ell}$.*

Output. $y := \mathsf{Output}(1^\lambda, 1^n, z, \mathbf{m})$ *is the output of the MPC protocol based on public input z and the full transcript $\mathbf{m} := \mathbf{m}^{<L+1}$.*

We require an L-round MPC protocol to be *perfectly correct* and *semi-malicious output-delayed Simulatable* as defined below.

Definition 8 (Perfect Correctness). *An L-round MPC protocol $\prod = (\mathsf{Next}, \mathsf{Output})$ for $\boldsymbol{\mathcal{F}}$ is perfectly correct if for any $\lambda \in \mathbb{N}$, for any public input z, any inputs $(x_1, \ldots, x_n)$, and any $f \in \boldsymbol{\mathcal{F}}$,*

$$\Pr\left[\mathsf{Output}(1^\lambda, 1^n, z, \mathbf{m}) = f(z, x_1, \ldots, x_n) \ : \ r \leftarrow\!\!\$ \ \{0,1\}^{\nu_r \cdot n}\right] = 1,$$

where $r := (r_1, \ldots, r_n)$, and $\mathbf{m} := \mathbf{m}^{<L+1}$ such that $m_j^\ell = \mathsf{Next}_j(1^\lambda, 1^n, z, x_j, r_j, \mathbf{m}^{<\ell})$ for $j \in [n]$ and $\ell \in [L]$.

Definition 9 (Semi-Malicious Output-Delayed Simulatability). *An L-round MPC protocol $\prod = (\mathsf{Next}, \mathsf{Output})$ for $\boldsymbol{\mathcal{F}}$ is semi-malicious output-delayed simulatable, if there exists a PPT simulator $\mathcal{S}$, such that for any PPT adversary $\mathcal{A}$ and any $f \in \boldsymbol{\mathcal{F}}$, the view of $\mathcal{A}$ in the Ideal-Real experiments in fig. 5 are indistinguishable.*

A.2 Garbled Circuit

Definition 10 (Garbled Circuit). *Let $\mathbf{CC} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a polynomial-size class of circuits with input and output lengths n and l. A garbled circuit scheme GC for $\mathbf{CC}$ consists of four polynomial-time algorithms $\mathsf{GC} = (\mathsf{Gen}, \mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$:*

$\mathbf{E} \leftarrow \mathsf{Gen}(1^\lambda)$**:** *It generates input labels $\mathbf{E} = \{\mathbf{E}[i,b]\}_{i\in[n],b\in\{0,1\}}$, where the input label $\mathbf{E}[i,b]$ corresponds to the value b of the i-th input wire.*
$\widehat{C} \leftarrow \mathsf{Garble}(\mathbf{E}, C)$**:** *Given input labels and a circuit $C \in \mathcal{C}_\lambda$ as input, it outputs a garbled circuit $\widehat{C}$.*
$y \leftarrow \mathsf{Eval}(\widehat{C}, \mathbf{E}')$**:** *On input a garbled circuit $\widehat{C}$ and input labels $\mathbf{E}'$, it outputs $y \in \{0,1\}^l$.*
$(\widetilde{C}, \mathbf{E}') \leftarrow \mathsf{Sim}(1^\lambda, y)$**:** *Given the security parameter λ and a value $y \in \{0,1\}^l$, it outputs a garbled circuit $\widetilde{C}$ and input labels $\mathbf{E}'$.*

We require a garbled circuit to be *perfectly correct* and *simulatable* as defined below.

Definition 11 (Perfect Correctness). *For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, any input $x \in \{0,1\}^n$, any $\mathbf{E} \leftarrow \mathsf{Gen}(1^\lambda)$, and any $\widehat{C} \leftarrow \mathsf{Garble}(\mathbf{E}, C)$,*

$$\Pr[\mathsf{Eval}(\widehat{C}, \{\mathbf{E}[i, x_i]\}_{i\in[n]}) = C(x)] = 1,$$

Definition 12 (Simulatability). *The following two distributions are computationally indistinguishable:*

$$\left\{(\mathbf{E}, \widehat{C}) : \mathbf{E} \leftarrow \mathsf{Gen}(1^\lambda); \widehat{C} \leftarrow \mathsf{Garble}(\mathbf{E}, C)\right\}_{\lambda, C\in\mathcal{C}_\lambda, x\in\{0,1\}^n},$$
$$\left\{(\mathbf{E}', \widehat{C}) : (\mathbf{E}', \widehat{C}) \leftarrow \mathsf{Sim}(1^\lambda, C(x))\right\}_{\lambda, C\in\mathcal{C}_\lambda, x\in\{0,1\}^n}$$

$\underline{\mathsf{Exp}^{\mathsf{Real}}_{\mathcal{A},\mathcal{S}}(\lambda, f)}$

1. $\mathcal{A}$ chooses the number of parties n, the set of honest parties $H \subseteq [n]$, the public input z, the private inputs $\{x_i\}_{i\in[n]}$ of all the parties and the randomnesses $\{r_i\}_{i\in\bar{H}}$ of all the corrupt parties.
2. The challenger picks fresh randomness $\{r_i\}_{i\in H}$ of honest parties, runs the MPC protocol with the specified inputs and randomnesses, and sends the resulting transcript $(\mathbf{m}^{<L}, \{m_i^L\}_{i\in\bar{H}})$ except the last message of the honest parties to $\mathcal{A}$.
3. $\mathcal{A}$ can send queries $(\mathsf{Compute}, P_i)$ to the challenger for $i \in H$ and receive the last message m_i^L of P_i.

$\underline{\mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda, f)}$

1. $\mathcal{A}$ chooses the number of parties n, the set of honest parties $H \subseteq [n]$, the public input z, the private inputs $\{x_i\}_{i\in[n]}$ of all the parties and the random tapes $\{r_i\}_{i\in\bar{H}}$ of all the corrupt parties.
2. Given the inputs and randomnesses of the corrupt parties, $\mathcal{S}$ outputs a transcript $(\mathbf{m}^{<L}, \{m_i^L\}_{i\in\bar{H}})$ of all but the last message of the honest parties to $\mathcal{A}$.
3. $\mathcal{A}$ can send queries $(\mathsf{Compute}, P_i)$ to $\mathcal{S}$ for $i \in H$ and receive the last message m_i^L of P_i. If all the honest parties have been queried, $\mathcal{S}$ is additionally given $f(x_1, \ldots, x_n)$ before answering the query.

Fig. 5: Real and Ideal experiments for semi-malicious output-delayed simulatability

A.3 Security Definition of MrNISC

Definition 13 (Semi-malicious Privacy). *A mrNISC scheme for a function f is called semi-malicious private if there exists a PPT simulator $\mathcal{S}$ such that no PPT adversary $\mathcal{A}$ can distinguish the two experiments defined in fig. 6.*

B Output Extractable WE$^{\text{FC}}$

Here we discuss the output extractable variant of WE$^{\text{FC}}$mentioned in our application in section 7.2.

Model. The basic intuition of this primitive is that a party able to decrypt should know the output value $\mathbf{y} = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. We model this through extraction and dub it "*output*" extractability.

The syntax of this primitive is almost the same as that in definition 6 except for a few changes. The syntax of encryption is $\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathsf{m})$ instead of $\mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{m})$. The decryption algorithm additionally takes as input $\mathbf{y}$. Correctness stays the same mutatis mutandis.

We replace the security definition with the following one:

Output Extractability. For any λ, any $\mathcal{C} \in \mathbf{CC}$, any stateless PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and any polynomial $q(\cdot)$, there exists a PPT extractor $\mathcal{E}$ and a polynomial $p(\cdot)$, such that

$$\Pr\left[b \leftarrow \mathcal{A}_2(\mathsf{ck}, \mathsf{ct}) \;:\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C}); (\boldsymbol{\alpha}, r, \boldsymbol{\beta}, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_1(\mathsf{ck}) \\ (\mathsf{cm}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r); b \leftarrow\!\!\$\, \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathsf{m}_b) \end{array} \right] \geq \frac{1}{2} + \frac{1}{q(\lambda)}$$

$$\Rightarrow \Pr\left[\begin{array}{l} \mathbf{y} \leftarrow \mathcal{E}(\mathsf{ck}, \mathsf{ct}) \\ \wedge\, \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{y} \end{array} \;:\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{C}); (\boldsymbol{\alpha}, r, \boldsymbol{\beta}, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}_1(\mathsf{ck}) \\ (\mathsf{cm}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{\alpha}; r); b \leftarrow\!\!\$\, \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathsf{m}_b) \end{array} \right] \geq \frac{1}{p(\lambda)}$$

Above we assume the extractor has also access to the random coins of the adversary $\mathcal{A}_2$.

$\underline{\mathsf{Exp}^{\mathsf{Real}}_{\mathcal{A},\mathcal{S}}(\lambda, f)}$

$\mathcal{A}$ chooses the number of parties n, and the set of honest parties $H \subseteq [n]$. It then interacts with a challenger $\mathcal{C}$ for an arbitrary number of iterations until it terminates. $\mathcal{A}$ can submit one query of the following three types in every iteration.

1. `CORRUPT INPUT ENCODING.` Upon $\mathcal{A}$ sending a query $(\mathsf{input}, P_i, x_i, \rho_i)$ for a corrupt party $i \in \bar{H}$, $\mathcal{C}$ records the input encoding $\hat{x}_i$ generated as $(\hat{x}_i, s_i) = \mathsf{Commit}(x_i; \rho_i)$ using input x_i and randomness ρ_i.
2. `HONEST INPUT ENCODING.` Upon $\mathcal{A}$ choosing the input $(\mathsf{input}, P_i, x_i)$ for an honest party $i \in H$, $\mathcal{C}$ generates $(\hat{x}_i, s_i) = \mathsf{Commit}(x_i)$, and sends $\hat{x}_i$ to $\mathcal{A}$.
3. `HONEST COMPUTATION ENCODING.` Upon $\mathcal{A}$ querying $(\mathsf{compute}, P_i, z, I)$ for an honest party $i \in H \cap I$, $\mathcal{C}$ checks if the input encodings $\{\hat{x}_j\}_{j \in I}$ for all participants have been generated, then it sends the computation encoding $\alpha_i \leftarrow \mathsf{Encode}(z, \{\hat{x}_j\}_{j \in I}, s_i)$ to $\mathcal{A}$.

$\underline{\mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda, f)}$

The ideal experiment is the same as above, except for the following differences.

1. `CORRUPT INPUT ENCODING.` Additionally send query $(\mathsf{input}, P_i, x_i, \rho_i)$ to $\mathcal{S}$.
2. `HONEST INPUT ENCODING.` Upon $\mathcal{A}$ choosing the input $(\mathsf{input}, P_i, x_i)$ for an honest party $i \in H$, $\mathcal{C}$ sends query (input, P_i) to $\mathcal{S}$ and forwards the simulated input encoding $\tilde{x}_i$ to $\mathcal{A}$.
3. `HONEST COMPUTATION ENCODING.` Upon $\mathcal{A}$ querying $(\mathsf{compute}, P_i, z, I)$ for an honest party $i \in H \cap I$, if this is the last honest computation encoding, $\mathcal{C}$ sends the query $(\mathsf{compute}, P_i, z, I, y)$ with the output $y = f(z, \{x_t\}_{t \in I})$ to $\mathcal{S}$; otherwise, $\mathcal{C}$ sends the query $(\mathsf{compute}, P_i, z, I)$ without y. The challenger forwards the simulated computation encoding $\tilde{\alpha}_i$ to $\mathcal{A}$.

Fig. 6: Real and Ideal experiments for semi-malicious output-delayed simulatability

Construction. For the construction of output extractable $\mathrm{WE}^{\mathrm{FC}}$, we modify the language $\mathcal{L}_{\mathsf{lpar}}$ and $[\mathbf{\Theta}_{\mathsf{lpar},\mathsf{x}}]_T$ as follows:

$$\mathcal{L}_{\mathsf{lpar}} = \{\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta}) | \exists \mathsf{op}, \mathbf{y} : \mathsf{Verify}(\mathsf{ck}, \mathsf{cm}, \mathsf{op}, \boldsymbol{\beta}, \mathbf{y}) = 1\}$$

$$[\mathbf{\Theta}_{\mathsf{lpar},\mathsf{x}}]_T = [\mathbf{M}_{\mathsf{lpar},\mathsf{x}} \cdot (\widetilde{\mathsf{op}} || \mathbf{y})]_T$$

where $\widetilde{\mathsf{op}}$ is derived from op by replacing its group elements with their discrete logarithms. The new construction $\mathsf{xWE}^{\mathsf{FC}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}, \mathsf{xEnc}, \mathsf{xDec})$ can be described as follows:

$\mathsf{xEnc}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{\beta}, \mathsf{m})$. Let $\mathsf{x} = (\mathsf{cm}, \boldsymbol{\beta})$. To encrypt a bit message $\mathsf{m} \in \{0, 1\}$, select a uniformly random vector $\mathsf{hk} \in \mathbb{Z}_p^{1 \times \nu}$, where ν is the number of rows of $\mathbf{M}_{\mathsf{lpar},\mathsf{x}}$, sample a random $r \leftarrow_\$ \{0, 1\}^\ell$, and compute the ciphertext $\mathsf{ct} = (\mathsf{hp}, r, \widehat{\mathsf{ct}})$, where

$$\mathsf{hp} = [\mathsf{hk} \cdot \mathbf{M}_{\mathsf{lpar},\mathsf{x}}]_\star, \qquad \mathsf{H} = [\mathsf{hk} \cdot \mathbf{\Theta}_{\mathsf{lpar},\mathsf{x}}]_T, \qquad \widehat{\mathsf{ct}} = \langle \sigma(\mathsf{H}), r \rangle \oplus \mathsf{m}$$

$\mathsf{xDec}(\mathsf{ck}, \mathsf{ct}, \mathsf{cm}, \boldsymbol{\beta}, \mathbf{y}, \mathsf{op})$. On input a ciphertext $\mathsf{ct} = (\mathsf{hp}, r, \widehat{\mathsf{ct}})$, first compute $\mathsf{pH} = [\mathsf{hp} \cdot (\widetilde{\mathsf{op}} || \mathbf{y})]_T$, and then output the message $\mathsf{m} \in \{0, 1\}$ computed as $\mathsf{m} = \langle \sigma(\mathsf{pH}), r \rangle \oplus \widehat{\mathsf{ct}}$.

The proof of security is similar to the proof of theorem 2 with the observation that we can still rely on the extractability of the underlying EPHF to extract the witness $(\widetilde{\mathsf{op}} || \mathbf{y})$ even though part of the witness (i.e., $\mathbf{y}$) consists of field elements. This comes from the fact that the extractor in the proof of theorem 1 can always extract the representation of the witness as field elements efficiently. Below, we state the theorem.

Theorem 5. *Let* FC *be a functional commitment scheme for circuit class* **CC** *with computational evaluation-binding property. Let* EPHF *be an extractable projective hash function. The construction of* $\mathsf{xWE}^{\mathsf{FC}}$ *described above is an output extractable* $\mathit{WE}^{\mathit{FC}}$ *for* **CC**.

Instantiations. Both instantiations of functional commitments proposed in section 5.2 have the property that their verification procedure is linear even if the function output is part of the opening proof. In other words, the function output is not paired together with the actual opening proof in the verification. Hence, they can be used to instantiate our output extractable variant of $\mathrm{WE}^{\mathrm{FC}}$.

C More on Attestation Approaches

This section expands on the discussion in section 7.1 and fig. 4. We are motivated by providing a solution depending as little as possible on trusted parties. As already mentioned, one solution to the problem of targeted broadcast is CP-ABE (ciphertext-policy attribute-based encryption). The latter requires an authority providing a key for a certain set of attributes. In the main text, we mention various approaches where users can register/attest themselves the attributes that will allow them to decrypt for certain policies. Here we elaborate more on them.

Self-Attestation: Where and Why

It may seem counterintuitive that there exist cases where we do not need to involve in this attestation process. We observe, however, that there exist settings actually amenable to self-attestation.

Some level of self-attestation may be meaningful in settings where the opening allowing decryption:

– *shows knowledge of a not easily available piece of information.* For example, the information could be a proof of the Riemann hypothesis and the encryptor would like to send information readable only to those users holding such valid proof. The programming contest example also falls into this category.
– *has to do with something not necessarily referring to a ground truth.* An illustrative example are the features one can choose for their own character at the beginning of a role-playing game (RPG)[18]. The application could then for instance issue periodic messages, readable only by users with certain features, but not others.

Full self-attestation. We can then ask where it would make sense to apply *full* self attestation (fig. 4c). We notice that the first class of attributes described above can be completely self-attested: a key for that attributes would just consist of a commitment to the solution to a difficult puzzle (e.g., the proof of the Riemann Hypothesis, the solutions to all New York Times Sudokus of the last year, etc.). An adversary would need to be able to come up with the right attributes to decrypt but this would deny the assumption on their hardness.

A mitigation against Sybil attacks: one-time tokens. Not all self-attested settings have attributes that are hard to find (e.g. our RPG example or the lottery one from earlier). Here an adversary could run a Sybil attack and "spam" the system with commitments to different combinations of attributes. Potentially this strategy can allow them to decrypt all ciphertexts (because there would probably exist at least some of the adversarial commitments that open to data satisfying the ciphertext policy). To prevent such an attack we can consider one (or multiple) entity/entities that can issue tokens for attestation (fig. 7). The adversary can thus issue no more commitments than the tokens it received. Notice that such entity would be trusted only to properly issue tokens to users but would not be trusted in other ways—it would not be learned the opening of the commitment nor would be able to decrypt ciphertexts (as it is the case for ABE authorities). Such entities could also be distributed and their authority could be revoked in case of malfeasance.

```
$\mathsf{AddUserComm}(D, \mathsf{token}_{\text{add}}) \to \mathsf{cm}$
    $\mathsf{cm}' \leftarrow \mathsf{FC.Commit}(\mathsf{ck}, D)$
    return $(\mathsf{cm}', \mathsf{token}_{\text{add}})$
```

```
$\mathsf{VfyUpdUsers}(S_{\text{tokens}}) \to \mathsf{pp}'_{\text{users}}$
    if $\mathsf{token}_{\text{add}} \notin S_{\text{tokens}}$
        $S_{\text{tokens}} \leftarrow S_{\text{tokens}} \cup \{\mathsf{token}_{\text{add}}\}$
        return $\mathsf{pp}_{\text{users}} \cup \{\mathsf{cm}'\}$
    else
        return $\mathsf{pp}_{\text{users}}$
```

Fig. 7: Token-based attestation. Assumes an external mechanism for providing registration tokens and maintaining a set S_{tokens} of used ones.

[18] Such features would involve for example gender, species (elf, human, etc...), background, weapon held, etc.. These features are chosen once and for all and it is not important what they are. This is a toy example, but we believe natural higher-impact examples could be found.

Partially validating opening through zero-knowledge once. Another problem to be solved for attributes that are self-attesting is that they could have the wrong format. In the RPG example, we expect attributes to contain only one value per field (see Footnote 18). This requirement can be enforced by letting the user provide a zero-knowledge proof at the time of attestation that guarantees that the opening satisfies certain format requirements fig. 4b. We stress this would not be the same as providing a proof for the fact that the opening satisfies a decryption policy: we assume this offline validation to be simpler (e.g. just format based) and known in advance (in contrast to decryption policies which are learned dynamically).

D Additional Subtleties in Contingent Payment Applications

Here we discuss some subtleties worth mentioning for the application in section 7.2. We stress that our goal is to show that our primitive has the potential to be versatile, not to provide a full-fledged solution to a specific application setting.

- It is important that the indices are not revealed to the storage provider, otherwise they could just store the part of the file revealed by those indices. The indices themselves can be encrypted through a time-released encryption so that they are only revealed after a certain amount of time [RSW96]. The $\mathsf{WE}^{\mathsf{FC}}$ ciphertext itself should be time-release encrypted: there is no guarantee on hiding the function used for encryption and the indices could be leaked as a consequence.
- This payment can be performed many times by simply releasing a large number of timed-released encryptions as described above, each requiring more and more time to decrypt.
- We assume the payer is either trusted or there is a way to guarantee that a ciphertext contains a payment, e.g. through a zero-knowledge proof.
- If there are several providers, we ignore the issue of how to guarantee fairness ("who gets the reward first"). If the reward is a digital good naturally this problem does not occur as every honest party will be able to decrypt it.