Commit-and-Prove System for Vectors and Applications to Threshold Signing

Anja Lehmann and Cavit Özbay

Hasso Plattner Institute, University of Potsdam, Germany
{anja.lehmann,cavit.oezbay}@hpi.de

Abstract. Multi-signatures allow to combine several individual signatures into a compact one and verify it against a short aggregated key. Compared to threshold signatures, multi-signatures enjoy non-interactive key generation but give up on the threshold-setting. Recent works by Das et al. (CCS'23) and Garg et al. (S&P'24) show how multi-signatures can be turned into schemes that enable efficient verification when an *ad hoc* threshold – determined only at verification – is satisfied. This allows to keep the simple key generation of multi-signatures and support flexible threshold settings in the signing process later on. Both works use the same idea of combining BLS multi-signatures with inner-product proofs over committed keys. Das et al. give a somewhat generic proof from both building blocks, which we show to be flawed, whereas Garg et al. give a direct proof for the combined construction in the algebraic group model. In this work, we identify the common blueprint used in both works and abstract the proof-based approach through the building block of a *commit-and-prove system for vectors* (CP). We formally define a flexible set of security properties for the CP system and show how it can be securely combined with a multi-signature to yield a signature with ad hoc thresholds. Our scheme also lifts the threshold signatures into the multi-verse setting recently introduced by Baird et al. (S&P'23), which allows signers to re-use their long-term keys across several groups. The challenge in the generic construction is to express – and realize – the combination of homomorphic proofs and commitments (needed to realize flexible thresholds over fixed group keys) and their simulation-extractability (needed in the threshold signature security proof). We finally show that a CP instantiation closely following the ideas of Das et al. can be proven secure, but requires a new flexible-base DL-assumption to do so.

1 Introduction

Threshold signatures enable a set of distributed signers to jointly create a signature that can be verified using a compact verification key. Traditional threshold schemes support a structure where *t-out-of-n* individual signature contributions are enough, and necessary, to compute a valid signature for their joint verification key. The main disadvantage of traditional threshold signatures is the distributed key generation and key management, which strictly determines the

threshold structure. Supporting flexible threshold structures or signing groups requires dedicated secret keys for each setting.

In both aspects, multi-signatures with key aggregation offer more flexibility as signers can generate their keys independently and combine their individual signatures for arbitrary and ad hoc generated groups. The aggregated key thereby serves as their joint and compact verification key. However, multi-signatures fall short in providing threshold structure and only support *n-out-of-n* structure.

Building Threshold from Multi-Signatures. In the last year, several new schemes emerged that combine the best of both worlds and propose threshold-signatures that are built from multi-signatures. Baird et al. [4] introduced *multiverse threshold signatures* (MTS), where signers only have a single long-term (multi-signature) secret key, and can create aggregated public keys and signatures for arbitrary groups and threshold structures. Follow-up works [30, 17, 29] propose threshold signature schemes that support *arbitrary thresholds* within a fixed signing group. That is, keys are generated for a fixed group but are independent of a concrete threshold. Any set of individual signatures can get combined, and the amount of contributions becomes verifiable information. Only when the signature gets verified, the threshold get chosen explicitly and provided to the verification algorithm – and only combined signatures that meet the threshold will be deemed valid. In fact, both works even go beyond *ad hoc thresholds*, and provide *weighted* thresholds as well. That is, each signer can have a different weight and the threshold must be met through their weighted contributions. These schemes are denoted as *ad hoc threshold signatures* (ATS).

Commit&Prove Blueprint. The works of Das et al. [17] and Garg et al.[30] have a very similar approach to building their ATS schemes. Both constructions rely on BLS multi-signatures and let all signers provide additional secret-key dependent information during the (joint) key generation. These values are a short verification key vk that is a vector commitment to all secret keys, and an aggregation key ak which are n inner-product proofs π_j of the secret keys w.r.t to some base vector. The aggregation key later allows the combiner algorithm to compute a proof π_σ that the combined multi-signature is indeed a combination of a subset of thr' signers whose keys are in vk. Verification of a combined signature for a given threshold thr then simply checks the multi-signature, the proof π_σ and that $\mathsf{thr}' \geq \mathsf{thr}$. The main purpose of π_σ is not hiding any information, but making the combined signature compact and independent of the number of contained signers – which is why SNARKs are used here. How these compact proofs can be generated based on publicly available information only, is the core technical contribution of both works. The work [29] follows the main ideas of [30, 17], but they rely on a commitment scheme that runs on public instead of secret keys. This has the impact, that the signature size depends on the number of signers. Thus, for the rest of the discussion, we will focus on [17, 30].

The overall generic(-ish) structure of the recent line of threshold constructions – combining a proof system with BLS multi-signatures – also facilitates further extension. Both [17, 30] mentioned extensions such as ad hoc groups (as in [4])

privacy (along the lines of accountable and privacy threshold signatures [8]), or more advanced access structures that can be easily realized with their approach: intuitively, mainly the proof statement for π_σ needs to be adapted.

Challenges of Generic Construction. Since these schemes already enjoy a composed structure, ideally their construction and security analysis can rely on abstract building blocks with well-defined properties. Such a generic approach makes the analysis and constructions more versatile, as not the entire proof has to be re-done for every modification. As we will see, this is far from trivial here.

Das et al. [17] give their construction in a somewhat generic way by combining an inner-proof argument (IPA) with BLS signatures, and proving security from the knowledge soundness of the IPA and the (co)-CDH-assumption underlying the unforgeability of BLS signatures. We show that the security proof in [17] is flawed with no immediate way to be fixed. The challenge is that the ATS key generation requires generating commitments and proofs that depend on the secret keys of the multi-signature scheme. This entanglement makes it difficult to prove security via two fully separate security properties of the commitments/proofs and multi-signature.

Garg et al. [30] have a different proof strategy from [17]. They seem aware of the issue of not being able to run independent reductions from the knowledge soundness of the proof system and the unforgeability of the underlying signature scheme. Their work analyzes the construction as a whole instead of using a generic proof system. Thus, while not suffering from the problem in the security proof, their construction and analysis give up on the generic approach.

MPC-based Signing: more generic – but less efficient. Our target constructions [17, 30] involve creating a commitment to a set of secret keys during key generation. When signing a message, the signers then prove that an aggregated public key is correctly computed from a subset of the committed secret keys. Essentially, this can be viewed as a multi-party computation where signers collaboratively prove the correct key aggregation in the signing protocol.

Capturing this through generic multi-party computation, would simplify our definitions for the proof scheme, as it avoids making advanced properties such as homomorphism and simulation extractability explicit. However, what makes the previous works [17, 30] so interesting is their efficiency in terms of communication cost, which is achieved by using the homomorphic proofs. This approach effectively converts the communication overhead of generic multi-party computation into the storage overhead of handling the aggregation keys. This is particularly attractive in scenarios that have a large number of signers, and [17, 30] show the suitability of their approach for real-world applications, such as Byzantine-fault-tolerant protocols and decision making in decentralized autonomous organizations.Therefore, our work follows the homomorphic proof approach and aims to make it more accessible.

1.1 Our Contributions

Our work strongly builds upon [17, 30] and makes the internal blueprint available in a more generic and provably secure manner. To this end, we propose succinct *commit-and-prove* systems (CP) for vectors as the core building block. We formally define their properties, show how to build a provably secure threshold signature scheme from them, and give a concrete instantiation for the CP system. In more detail, we make the following contributions:

Security Flaw in Generic Approach (Section 2): We start by analyzing the common structure of [17, 30] and present the high-level blueprint used in both works. We then show that [17]'s security proof is flawed due to the conflicting requirements in terms of simulatability vs. knowledge-soundness vs. unforgeability and required homomorphic properties.

Commit-and-Prove System (Section 4): Building upon Section 2, we identify *commit-and-prove* systems (CP) for vectors as the common building block. The CP system allows to commit on vectors first, and then provide proofs of statements over committed vectors later on. To be useful in the desired protocol design, the commitments and proofs must provide sufficient homomorphisms. We then formally define the properties of *f-zero-knowledge*, *policy-based simulation-extractability*, and *policy-based simulation-soundness*, where the f-relaxation is needed when deterministically derived values from the witness must be simulated, and the policy-limitations enable to reason about simulation-extractability despite having homomorphic proofs.

Generic Multiverse Ad Hoc Threshold Signatures (Section 5): By using our CP system, we build a generic *and* provably secure threshold signature from multi-signatures. This construction closely follows [17, 30] but is built and proven in a modular way, relying on the unforgeability of the multi-signature scheme and assuming that the CP scheme satisfies our introduced properties for well-defined leakage f and policies. We state our construction for the class of *DL-based* multi-signature schemes, which makes it easier to express the key structures that our CP can rely on. Apart from the generic approach, our construction also extends the ATS schemes to the *multiverse* setting. That is, not only are the thresholds set dynamically during verification only, but the signers can join many signing groups with the same long-term key. We call this new type of signatures *multiverse ad-hoc threshold signatures* (MATS), combining the ad hoc thresholds from [17, 30] with the multiverse concept of [4], and propose the first provably secure scheme of that type.

CP Instantiation (Section 6): We realize the CP system using essentially the construction from [17], extended with a few additional pairing checks. The main contribution here is that we show that this CP instantiation achieves the necessary policy-based simulation-extractability, finally closing the gap from the original analysis. To prove this property, we need to rely on a new assumption: the *flexible-base* (n_1, n_2)*-DLOG* problem which we show that it holds in the algebraic group model if (n_1, n_2)-DLOG problem is hard.

Potential of our Abstraction. The abstraction and the CP system defined in our work originate from the desire to improve the understanding of complex recent threshold constructions, and have helped to discover and fix a flaw in [17]. We believe that our abstraction is useful in several other ways too:

Generic Protocol Design. Both previous works [17, 30] propose extensions to their schemes by relying on the functionalities of underlying SNARKs such as hiding the signers and hiding the threshold; accountability through the binding of commitments; or "multiverse" threshold signatures, allowing different ad hoc groups from the same long-term keys.

The construction and security analysis of these extensions would require re-evaluating their complex security proofs. With our generic construction, designing and proving different flavors of such schemes becomes much simpler, as it abstracts the complexity of SNARK schemes. In fact, in Section 5, our generic construction for MATS formally examines ideas proposed by previous works.

Flexibility in Instantiations. The previous works rely on vector commitments on Lagrange-basis polynomials to realize ATS. However, there are different vector commitment instantiations in the literature that could offer the proofs for the required relation, inner-product proofs and bit vector proofs. Using different instantiations may result in instantiations of the signature schemes that have different efficiency metrics, or are easier to compose with other protocols. The vector commitments with monomial basis [12, 42], or Libert-Yung vector commitments [36, 35] are examples to such vector commitments and proofs. If these constructions are shown to securely instantiate our CP-SNARK, they can immediately be used in our framework to yield threshold signatures.

Applicability Beyond Signatures. The common property of [17, 30] that got abstracted in our framework is the implicit use of CP-SNARKs to generate and aggregate keys. This approach is not limited to threshold signatures, and has indeed been applied for encryption too: the recent work of [31] on threshold encryption uses a similar design for their key generation technique. Using our abstraction as a blueprint for this type of encryption scheme could enable a generic design that is easily amenable to extensions too. For instance, the current CPA security is claimed to be upgradable to CCA2 security by adding a straight-line simulation-extractable proof to the ciphertexts, which could be more easily expressed (and analyzed) through a generic construction that determines the necessary simulation/extraction policies.

Related Work. The definitional framework for the CP system, as well as the ATS and CP constructions, build upon a large body of existing works. We discuss them in the dedicated sections to give a more clear comparison to our contributions. There are several other works that build threshold signatures with ad hoc thresholds or groups [38, 13, 3, 39]. However, these constructions often result in verification keys or signature sizes that grow dependent on the number of signers

in the group or the threshold value. Appendix A contains a wider discussion of these signature schemes and other related contexts.

2 Analysis and Challenges of Generic ATS Approach

To better understand both the need for, and the design choices of our generic Commit-and-Prove system, we first analyze the recent ATS schemes by Das et al. [17] and Garg et. al [30]. We sketch the common blueprint we have identified in both constructions and discuss the unforgeability proof of the ATS scheme in [17] and its shortcomings.

2.1 ATS Construction Blueprint

We first recap the intuition behind the ATS constructions of [17] and [30], which rely on BLS multi-signatures and a clever combination of non-interactive inner-product proofs over committed secret keys.

BLS multi-signatures with key aggregation [7] work on individual key pairs $pk_i := g^{sk_i}$. BLS signatures can simply be multiplied together and be verified against a short aggregated key $apk := \prod_{i\in[n]} pk_i$. The key aggregation internally sets an aggregated secret key, $ask := \sum_{i\in[n]} sk_i$ in the exponent. [17, 30] independently propose the same strategy to lift the *n-out-of-n* setting to a threshold structure. Therein a single and short verification key vk can be used to verify whether a signature was created by a certain – yet ad hoc – threshold of the signers. Both schemes support *weighted* thresholds, but for the sake of simplicity, we omit this feature in the description here.

Idea in a Nutshell. The high-level idea is as follows: for generating a verification key vk for n signers, each signer is assigned to a position in an n-dimensional secret key vector $\mathbf{sk}$ and first generates its own BLS key pair (sk_i, pk_i). Then, they jointly compute vk as a vector commitment to the (implicitly defined) $\mathbf{sk}$.

Later on, when a signer subset $S \subseteq [n]$ wants to sign a message, the multi-signature and the aggregated public key apk for the set S are computed from the individual contributions as usual. Additionally, the combiner creates a proof that shows that apk is computed correctly. To do so, it creates a commitment to the bit vector $\mathbf{b}$ where $b_i = 1$ if and only if $i \in S$. The proof then shows that this commitment is indeed a bit vector, the sum of b_i values satisfies a threshold thr', and the inner-product of $\mathbf{b}$ with $\mathbf{sk}$ is equal to the aggregated secret key ask such that $apk = g^{ask}$. To create these proofs, the key generation must also produce some auxiliary information denoted as the aggregation key ak.

An In-Depth Look. We will now take a closer look at how these keys and proofs are generated. We use the following abstract relation $\mathsf{IP}_{\mathbb{G}}$ to denote an inner-product proof that verifies the result in the exponent of g. The algorithm VfCom is used to represent the commitment verification informally.

$$\mathsf{IP}_{\mathbb{G}} := \{(\mathbf{u}, \mathbf{v}) : \mathsf{VfCom}(c_{\mathbf{u}}, \mathbf{u}) \ \wedge \ \mathsf{VfCom}(c_{\mathbf{v}}, \mathbf{v}) \ \wedge \ g^{\mathbf{u}\cdot\mathbf{v}} = Y\}$$

Key Generation. After standard BLS key generation with individual keys (sk_i, pk_i), we need to commit to the vector of all secret keys $\mathbf{sk}$ to set a verification key $vk := \mathsf{Com}(\mathbf{sk})$. This is done by letting each signer create a homomorphic commitment $U_i := \mathsf{Com}(sk_i \cdot \mathbf{e}_i)$ for the base vector $\mathbf{e}_i$ of the i'th dimension in $\mathbb{Z}_p^n$. By combining all homomorphic commitments, we get a commitment to $\mathbf{sk}$.

When signatures get combined for a concrete aggregated key apk later on, we need to compute an inner-product proof to show that $g^{\mathbf{sk}\cdot\mathbf{b}} = apk$. This proof must be computable without the secret keys $\mathbf{sk}$. To this end, key generation also outputs an aggregation key ak. Although the prior works do not state it explicitly, we observe that what they do to is just computing homomorphic proofs: the aggregation key ak consists of n proofs π_j for $j \in [n]$ which show $g^{\mathbf{sk}\cdot\mathbf{e}_j} = pk_j$, i.e., π_j is an $\mathsf{IP}_{\mathbb{G}}$ proof between $\mathbf{sk}$ and $\mathbf{e}_j$. Looking ahead, the bit vector $\mathbf{b}$ used in the signature combination can be represented with respect to the base vectors $\{\mathbf{e}_j\}$'s. Thus, using the homomorphic properties of the π_j proofs in ak, the combiner can compute the proof π_{Agg} that shows $g^{\mathbf{sk}\cdot\mathbf{b}} = apk$.

The proofs in the aggregation key ak are computed collaboratively by the signers as follows: The signer i, together with the commitment U_i creates n $\mathsf{IP}_{\mathbb{G}}$ proofs $\pi_{i,j}$ between $\mathbf{sk}_i := sk_i \cdot \mathbf{e}_i$ and $\mathbf{e}_j$ for $j \in [n]$. Each signer sends the proofs $\pi_{i,j}$'s to the other signers. Using the homomorphic properties of the proofs $\pi_{i,j}$'s, each signer can compute the $\mathsf{IP}_{\mathbb{G}}$ proofs π_j's between $\mathbf{sk}$ and $\mathbf{e}_j$ for $j \in [n]$ which constitutes the aggregation key $ak := (\pi_1, ..., \pi_n)$.

Signature Combination. The signature combination aggregates individual BLS signatures of a subset of thr' signers S, and later allows verification w.r.t. to an ad hoc threshold $\mathsf{thr}' \geq \mathsf{thr}$. This is done in two main steps. First, the BLS multi-signature σ' is computed and the corresponding bit vector $\mathbf{b}$ is set for S. Then, an $\mathsf{IP}_{\mathbb{G}}$ proof π_{Agg} is generated that σ' verifies for an apk such that $apk = g^{\mathbf{sk}\cdot\mathbf{b}}$.

This is done using the homomorphic proofs π_j in ak. The combiner also computes a proof π_{thr} to show that $\mathbf{b} \cdot \mathbf{1} = \mathsf{thr}'$, which is done by using the $\mathsf{IP}_{\mathbb{Z}_p}$ relation below to create an inner-product proof between $\mathbf{b}$ and $\mathbf{1}$.

$$\mathsf{IP}_{\mathbb{Z}_p} := \{(\mathbf{u}, \mathbf{v}) : \mathsf{VfCom}(c_{\mathbf{u}}, \mathbf{u}) \ \wedge \ \mathsf{VfCom}(c_{\mathbf{v}}, \mathbf{v}) \ \wedge \ \mathbf{u} \cdot \mathbf{v} = y\}$$

That the vector $\mathbf{b}$ is only inside a commitment $c_{\mathbf{b}}$ creates another challenge: the verifier must be able to check that the committed $\mathbf{b}$ is indeed a *bit* vector. Otherwise, a malicious combiner could use other values than bits, and fool a verifier that the signature satisfies a higher threshold than it actually does. Thus, the combiner must compute another proof π_{BIT} that shows $c_{\mathbf{b}}$ indeed commits to a bit vector $\mathbf{b} \in \mathbb{Z}_2^n$. We notate the relation of this additional proof as follows.

$$\mathsf{BIT} := \{(\mathbf{u}) : \mathsf{VfCom}(c_{\mathbf{u}}, \mathbf{u}) \ \wedge \ u \in \mathbb{Z}_2^n\}$$

Using the three proofs π_{Agg}, π_{thr}, and π_{BIT}, the combiner forms a proof π_σ for the relation in Eq. 1. The proof π_σ, the used apk, the implicit threshold thr', and the combined multi-signature σ' form the threshold signature $\sigma = (apk, \sigma', \pi_\sigma, \mathsf{thr}')$.

$$\begin{aligned} R_{\mathsf{TS}} := \{&(\mathbf{sk}, \mathbf{b}) : \mathsf{VfCom}(vk, \mathbf{sk}) \ \wedge \ \mathsf{VfCom}(c_{\mathbf{1}}, \mathbf{1}) \ \wedge \ \mathsf{VfCom}(c_{\mathbf{b}}, \mathbf{b}) \\ &\wedge \ g^{\mathbf{sk}\cdot\mathbf{b}} = apk \ \wedge \ \mathbf{b} \in \mathbb{Z}_2^n \ \wedge \ \mathbf{b} \cdot \mathbf{1} = \mathsf{thr}'\} \end{aligned} \tag{1}$$

Verification. To verify a threshold signature $\sigma = (apk, \sigma', \pi_\sigma, \mathsf{thr}')$ for an ad hoc threshold thr against the key vk, the verifier checks that $\mathsf{BLS.Vf}(apk, \sigma', m) = \mathsf{true}$, the proof for the above relation holds for apk and that $\mathsf{thr}' \geq \mathsf{thr}$.

Proofs for Succinctness. A crucial observation here is that none of the values in π_σ need to be hidden for secrecy. In particular, revealing $\mathbf{b}$ as part of the signature would not have an impact on unforgeability – but yield signatures of size $O(n)$. Proving the statement instead of revealing all (public) inputs with *succinct* proofs (SNARKs) is the reason we get *compact* signatures.

This observation is helpful when designing, and understanding, the security properties for the commit-and-prove system: In Section 4 we define a parameterized zero-knowledge property that in our MATS construction is allowed to leak most of the witnesses or deterministic computations thereof. This is sufficient, as the MATS construction never demands any secrecy from the computed proofs. The main purpose of the zero-knowledge property will be to simulate the additional key material in vk and ak, within the security reduction from a MATS forgery to a multi-signature forgery.

2.2 Flawed Proof in [17]

[17] claim that their ATS construction is unforgeable, if the (co-)CDH assumption holds and the non-interactive proof scheme for R_{TS} is knowledge-sound. Although no formal definition of knowledge-soundness is given, they informally describe that a partial extractor to extract the bit vector $\mathbf{b}^*$ from R_{TS} is required.

Proof Strategy in [17]. The main proof strategy is to build a CDH adversary $\mathcal{A}_{\mathsf{CDH}}$ using an ATS forger $\mathcal{A}_{\mathsf{ATS}}$. $\mathcal{A}_{\mathsf{CDH}}$ takes a CDH challenge $(X = g^x, Y = g^y)$, and simulates the ATS unforgeability game against $\mathcal{A}_{\mathsf{ATS}}$ by using X as an honest signer's public key $pk_i := X$. The (co-)CDH-related part of $\mathcal{A}_{\mathsf{CDH}}$'s reduction, following Boneh et al. [7]'s security proof of the PoP-based BLS multi-signature, is given in detail. However, the simulation of $\mathcal{A}_{\mathsf{ATS}}$ view related to the non-interactive proof scheme is not clearly argued. At the end of the game, $\mathcal{A}_{\mathsf{CDH}}$ runs the knowledge-soundness extractor on the non-interactive proof that was a part of the forged signature σ^*, and extracts a bit vector $\mathbf{b}^*$ which it then uses to solve the CDH challenge. The knowledge-soundness of the non-interactive proof for $\mathbf{b}^*$ is proven independently to hold based on the q-SDH assumption.

Missing Simulation of Long-Term Values. A subtle but crucial part of the proof is missing in [17] – namely how the view towards $\mathcal{A}_{\mathsf{ATS}}$ is simulated. The challenge hereby is that the ATS scheme not only contains the individual BLS public keys (for which the reduction simply sets $pk_i{:=}X$), but also generates further values that contain the same secret exponent. These are the verification key vk – which we abstracted away above as the commitment to the individual secret keys – as well as the aggregation key ak that contains $\mathsf{IP}_{\mathbb{G}}$ proofs π_j that show $g^{\mathbf{sk}\cdot\mathbf{e}_j} {=} pk_j$. All values are part of the overall public key of the threshold scheme, i.e., output with key generation and are static throughout the construction and game.

Simulation vs. Soundness. The commitment vk and the $\mathsf{IP}_{\mathbb{G}}$ proofs in ak contain group elements in the form of KZG commitments, $g^{P(\tau)\cdot s_i}$ and $h^{P(\tau)\cdot s_i}$ with g and h both being generators of $\mathbb{G}$, P denoting a polynomial and τ being the q-SDH trapdoor. In the reduction to CDH, the secret key s_i is not known, and thus the only way to simulate these values directly is if τ can be chosen by the simulator. However, τ is the secret trapdoor on which the soundness of the proof system (related to $\mathbf{b}^*$) relies. Thus, we cannot let $\mathcal{A}_{\mathsf{CDH}}$ choose τ, and at the same time rely on the soundness, which requires secrecy of this value. Extractability of KZG commitments in the existence of a proof simulator was also discussed in other works [22, 26, 27] and simulating KZG commitments without using the trapdoor τ is a general challenge beyond [17].

Simulation-soundness does not help (out of the box). A natural idea might be to rely on some form of simulation-soundness, which requires that the soundness of zero-knowledge proofs holds in the presence of simulated proofs, that can be retrieved through a simulation oracle, too. There is no way to directly apply this here – both on a definitional and constructional level. First, simulation-soundness typically only covers the direct proof statement, whereas here also the secret key commitments in vk and ak need to be simulated. The fact that both values are in fact *deterministic* derivations from the keys further makes achieving a standard zero-knowledge definition infeasible.

Second, vk and ak are re-used in every signature proof, and a secure realization of simulation-sound proof systems crucially relies on the non-malleability of simulated values. This is necessary to guarantee that the extractor never gets run on a simulated input. However, the ATS construction relies on *homomorphic* properties of the $\mathsf{IP}_{\mathbb{G}}$ proofs π_j which allows the combiner to compute inner-product proofs on unknown keys. Thus, even if we'd have an extended simulation-soundness definition that could produce these additional vk, ak values, their needed properties for the ATS construction will make it impossible to argue with simulation-soundness in the security proof.

In summary, there is no immediate way to fix the proof in [17] in the *generic* manner they intended, as we are missing the definitional and constructional tools to do so. Which is exactly what we aim to solve with our work.

3 Preliminaries

We present the building blocks and the notation we use. For space reasons, some parts are referred to Appendix B: There we define type-3 pairings, give the (n_1, n_2)-DLOG Assumption [6]in Definition 10, and recap polynomial encodings of vectors and show the Hadamard/inner-product relations on these encodings in Lemmas 1 and 2 from [40].

Type-Based Commitments. We recap the type-based commitments below which were first defined by [20] and were also used by Campanelli et al. [10] and Escala and Groth [20] to design CP systems. The only change to the original definition

is inputs of the Setup algorithm: Setup takes a public parameter description pp to define different schemes on the same public parameters and a size bound n to paremeterize the message space of the commitments for vectors of messages. The correctness property is given in Appendix B.

Definition 1 (Type-based Commitment). *A type-based commitment scheme* $\mathsf{CS} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{VfCom})$ *is a tuple of algorithms such that:*

$\mathsf{Setup}(pp, n) \to ck$: On input public parameters $pp \leftarrow \mathsf{ParGen}(1^\lambda)$ and a size bound $n \in \mathbb{N}$, it generates a commitment key ck. ck is an implicit input to the other algorithms and specifies a type space $\mathcal{T}$, and for all $t \in \mathcal{T}$, message, opening, and commitment spaces, $\mathcal{M}_{ck,t}$, $\mathcal{O}_{ck,t}$, and $\mathcal{C}_{ck,t}$.

$\mathsf{Com}(t, m; o) \to c$: Given a type $t \in \mathcal{T}$, a message $m \in \mathcal{M}_{ck,t}$, and an opening $o \in \mathcal{O}_{ck,t}$ returns a commitment c such that $c \in \mathcal{C}_{ck,t}$.

$\mathsf{VfCom}(t, c, m, o) \to b \in \{\mathsf{true}, \mathsf{false}\}$**:** Verifies the commitment $c \in \mathcal{C}_{ck,t}$ for the message $m \in \mathcal{M}_{ck,t}$, and the opening $o \in \mathcal{O}_{ck,t}$. Returns the result $\mathsf{true}/\mathsf{false}$.

Multi-Signatures with Proof-of-Possession (PoP). Following [16], we define the multi-signatures in the PoP model. We employ a single-round signing algorithm for simplicity, and it can easily be generalized to multiple signing rounds. The full descriptions of multi-signatures and their properties are in Appendix B. A multi-signature contains the following algorithms. $\mathsf{Pg}(1^\lambda) \to pp_{\mathsf{MS}}$ sets public parameters. $\mathsf{Kg}(pp) \to (sk, pk, pop)$ generates a key pair and a proof-of-possession for the secret key and $\mathsf{KeyVf}(pk, pop) \to b$ verifies PoP. $\mathsf{KAg}(PK) \to apk$ computes an aggregated group public key. $\mathsf{MulSign}(sk_i, PK, m) \to ps_i$ and $\mathsf{Combine}$ $(PK, \{ps_i\}_{pk_i \in PK}) \to \sigma$ algorithms compute partial and final signatures. Finally, $\mathsf{Vf}(apk, \sigma, m) \to b$ verifies final signatures with respect to an aggregated key apk.

Notation. We denote the vector of zeros and ones by $\mathbf{0}$ and $\mathbf{1}$. $\mathbf{e}_i$ corresponds to the base vector of i'th dimension in $\mathbb{Z}_p^n$ (n is clear from the context all the time). Inner-product and Hadamard-Product of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$ are denoted as $\mathbf{u} \cdot \mathbf{v}$ and $\mathbf{u} \circ \mathbf{v}$. Similarly, for $\mathbf{U} \in \mathbb{G}^n$, $\mathbf{U}^{\mathbf{v}} := \prod_{i=1}^n U_i^{v_i}$. The scalar multiplication/exponentiation of vectors with scalars are defined as $u \cdot \mathbf{v} := [u \cdot v_1, ..., u \cdot v_n]$, $U^{\mathbf{v}} := [U^{v_1}, ..., U^{v_n}]$, and $\mathbf{U}^v := [U_1^v, ..., U_n^v]$.

A relation R is a set of tuples $(x, \omega) \in \mathcal{D}_x \times \mathcal{D}_\omega$. A universal relation $\mathcal{R}$ over a set of relations Φ is defined as $(R, x, \omega) \in \mathcal{R}$ if $R \in \Phi$ and $(x, \omega) \in R$. If a relation R is contained in a universal relation, we denote this by, $R \in \mathcal{R}$. We denote a parameterized relation R_{pt} for a parameter pt and the definition of R_{pt} potentially depends on pt. For example, a relation that defines the statement space as a group can be parameterized for the underlying group.

4 Commit-And-Prove System

This section introduces our core building block for the flexible and provably secure design of advanced threshold signatures: the *commit-and-prove (*CP*)* system. This is an established approach [20, 10, 15, 11] for a modular protocol

design, and we closely follow the syntactical definition of [10] for commit-and-prove SNARKs (CP-SNARKs). Our new contributions are the security definitions that we define here and are necessary to realize the required features in the proof-based threshold signature schemes.

Type-based Commitments. The high-level idea for building threshold signatures from commit-and-prove systems is to commit to the secret keys as part of the overall public key, and later prove various statements on the keys. Thus, we want to create commitments in different commitment spaces, e.g., committing the secret key in both source groups of a pairing group or committing to different parts of a witness separately. We use *Type-based commitments* which were introduced by Escala and Groth [20] to support this feature in our construction.

Requirements for the Proof Systems. We need a CP system that allows proving different relations on the same commitment in a modular way. As motivated in Section 2, the security proof requires a careful combination of extractability and simulatability properties from such a proof system, that are also compatible with the static (and non-zero-knowledge) values that are revealed as public keys and still provide the necessary (homomorphic) operations needed in the protocol design. We provide a definitional framework that captures these seemingly contradictory requirements. In short, we define four properties for our commit-and-prove system that will allow us to securely build a threshold scheme (Section 5) and that can be efficiently instantiated as shown in Section 6:

f-zero-knowledge: As threshold signatures typically reveal some deterministic and long-term values to re-use across proofs, we need a relaxed zero-knowledge notion that allows leaking some information about the witness.

Policy-based simulation extractability: This property captures the desired simulation-extractability but restricts the simulation queries that an adversary can make in order to guarantee the existence of an extractor; as well as captures flexible extraction scenarios such as partial extractors.

Policy-based simulation-sound binding: Our third security notion can be seen as the classical binding property expressed in the context of simulation-extractable commit-and-prove systems. It guarantees that the commitments are binding even in the presence of simulators.

Homomorphic properties: In addition to the security properties we also require the commitment and proof system to allow for homomorphic operations, which we make explicit in our definition.

Most of these properties have been defined in some way before, but either still needed extensions/variations or had only been defined for either commitments or proofs, but not the combined system. For each property, we briefly explain the existing concepts and how and why we extend them. First, we define the adapted syntax of the commit-and-prove SNARKs we rely on.

4.1 Commit-and-Prove SNARKs (CP-SNARK) [10]

We closely follow the definition of [10] for commit-and-prove SNARKs. The main idea is as follows: They define a relation R over some $(\mathcal{D}_x, \mathcal{D}_w)$ where $\mathcal{D}_x$ is the statement space and $\mathcal{D}_w$ is the witness space. It is required that $\mathcal{D}_w := \mathcal{D}_m \times \mathcal{D}_\omega$ can be split into two parts. The first part, $\mathcal{D}_m$ corresponds to the part of the witness that we will create commitments for, and the second part is the part where we don't need commitments. They further require that $\mathcal{D}_m$ can be split over $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ where each *slot* corresponds to a commitment. Lastly, the CP-SNARK is defined over a relation that contains the commitments within the statements and openings to the commitments within the witness.

The changes we make over the [10]'s definition are as follows: First, we define the CP-SNARKs over parameterized relations $R_{pp,n}$ where public parameters that can be generated using a parameter generator $pp \in \mathsf{ParGen}(1^\lambda)$, and a size bound $n \in \mathbb{N}$. While defining $R_{pp,n}$ over the vector of secret and public keys, the public parameters pp and the size bound n help us to parameterize the key spaces and the vector size. Note that we drop the parameter indexes from the relations for readability in the rest of the paper whenever the parameters are clear from the context. Second, we make type-based commitments explicit in the definition. This change is not technical but rather aims to improve readability. [10] defines CP-SNARKs on a regular commitment scheme and then fits type-based commitment schemes into this scheme by assuming that the types of a commitment and a message can be inferred from itself. The third change we make is removing the *specializable crs* feature. The main motivation of specializable crs is performance optimization, and we omit this feature for simplicity. The forth change is that we rely on commitment-only crs, i.e., the crs value can be derived from the commitment key deterministically. This assumption has already been used in [10] and allows removing the Kg for simplicity. The definitions of completeness and succinctness are in Appendix C.

Definition 2 (CP-SNARKs). *Let $\{\mathcal{R}_{pp,n}\}_{pp\in\mathsf{ParGen}(1^\lambda),n\in\mathbb{N}}$ be a family of universal relations over relations $R_{pp,n}$ on the space $\mathcal{D}_x \times \mathcal{D}_m \times \mathcal{D}_\omega$ such that $\mathcal{D}_m$ splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let $\mathsf{CS} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{VfCom})$ be a type-based commitment scheme as in Definition 1 such that for $i \in [\ell]$, CS has a type t_i where $\mathcal{D}_i \subset \mathcal{M}_{ck,t_i}$. A commit and prove zkSNARK for $\{\mathcal{R}_{pp,n}\}_{pp\in\mathsf{ParGen}(1^\lambda),n\in\mathbb{N}}$ is a zkSNARK for a family of universal relations $\{\mathcal{R}^{\mathsf{CS}}_{ck}\}_{ck\in\mathsf{Setup}(pp,n)}$ such that:*

- *every $R^{\mathsf{CS}}_{ck} \in \mathcal{R}^{\mathsf{CS}}_{ck}$ is represented by a pair $(ck, R_{pp,n})$ where $ck \in \mathsf{Setup}(pp, n)$, $pp \in \mathsf{ParGen}(1^\lambda)$, $n \in \mathbb{N}$, and $R_{pp,n} \in \mathcal{R}_{pp,n}$;*
- *R^{CS}_{ck} is over pairs $(\hat{x}, \hat{w})$ where the statement is $\hat{x} := (x, (t_j, c_j)_{j\in[\ell]}) \in \mathcal{D}_x \times \mathcal{C}^\ell$, the witness is $\hat{w} := ((t_j, m_j)_{j\in[\ell]}, (t_j, o_j)_{j\in[\ell]}, \omega) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$ and the relation R^{CS}_{ck} holds if and only if*

$$\bigwedge_{j\in[\ell]} \mathsf{VfCom}(t_j, c_j, m_j, o_j) \wedge (x, (m_j)_{j\in[\ell]}, \omega) \in R_{pp,n}$$

$\mathsf{Prove}(R_{ck}^{\mathsf{CS}}, \hat{x}, \hat{w}) \to \pi$**:** Takes relation $R_{ck}^{\mathsf{CS}} \in \mathcal{R}_{ck}^{\mathsf{CS}}$, a statement $\hat{x}$ and a witness $\hat{w}$ such that $(\hat{x}, \hat{w}) \in R^{\mathsf{CS}}$ and outputs a proof π.

$\mathsf{Vf}(R_{ck}^{\mathsf{CS}}, \hat{x}, \pi) \to b \in \{\mathsf{true}, \mathsf{false}\}$**:** Takes relation, a statement, the corresponding proof, checks if the proof holds for the statement and outputs the result.

4.2 Properties of CP-SNARKs

We present the required properties of CP-SNARKs for our MATS scheme.

Homomorphic CP Systems. We further require the CP to be homomorphic in the commitments and proofs, which will be necessary in the protocol design. That is, we require a commitment homomorphism EvalCom that outputs a commitment on a message m which is a function F of n messages, and corresponding openings, using only the commitments to the n messages. Similarly, a proof homomorphism EvalProof computes a proof for a function X of n statements, using only their corresponding individual proofs. We give the straightforward definitions of these homomorphic properties of CP in Appendix C.1.

***f*-Zero-Knowledge.** When using CP systems as building blocks for signature protocols, requiring a full zero-knowledge property is neither necessary nor possible – when relying on building blocks with deterministic commitments. Thus, we define the flexible notion of f-zero-knowledge which allows to leak certain knowledge about the witness and is given directly for the combined CP system.

Leaky Zero-Knowledge. The additional information that can be leaked is expressed through a function f_{Prove} and providing $f_{\mathsf{Prove}}(\hat{w})$ as an input to the simulator. The regular zero-knowledge property is the special case that $f_{\mathsf{Prove}}(\hat{w}) := \perp$ for all $\hat{w}$'s. A similar zero-knowledge variant was used by [22], too where the additional input to the simulator is called *leakage*. Note that the leakage has a different role than the statement even if both are public values. The statement is necessary to verify a proof, whereas the leakage is not.

An immediate application of this approach for a CP-SNARK is providing the openings of some commitments from the statement as leakage to the simulator. For example, our threshold signature application commits to the signer subset in a signer group as a bit vector $\mathbf{b}$ only for succinctness. While there is no harm of extending the statement with $\mathbf{b}$ to the security of the signatures, we would lose the efficiency by doing that. Thus, we leak this vector to the proof simulator when simulating the proofs on this commitment.

The main difference between our zero-knowledge definition and [22]'s is that we define the zero-knowledge property explicitly on CP-SNARKs and provide a commitment simulator additional to the proof's simulator. Our zero-knowledge definition allows for simulating commitments in addition to the proofs. Also, similar to the proof simulator, we employ a *flexible* commitment simulator that can get some information about the message to simulate a commitment on it as input. This information is modeled as an output of some function f_{Com} on the

message m. We note that Campanelli et al. [10] defined *somewhat-hiding polynomial commitments* which has a similar simulator specifically for the polynomial commitments where f_{Com} also takes a trapdoor as input.

CP-SNARK Simulator. More precisely, we define the following sub-algorithms of our zero-knowledge simulator on CP-SNARKs:

$\mathsf{SSetup}(pp, n) \to (ck, st_{\mathsf{S}})$**:** Outputs the simulated commitment key ck and the internal simulator state st_{S}.

$\mathsf{SCom}(st_{\mathsf{S}}, t, f_{\mathsf{Com}}(t, m)) \to (c, o, st_{\mathsf{S}})$**:** Outputs a simulated commitment and opening (c, o) of type t using state st_{S} and additional information $f_{\mathsf{Com}}(t,m)$.

$\mathsf{SProve}(st_{\mathsf{S}}, R^{\mathsf{CS}}, \hat{x}, f_{\mathsf{Prove}}(\hat{w})) \to (\pi, st_{\mathsf{S}})$**:** It outputs a simulated proof π for the statement $\hat{x}$, using state st_{S} and additional information $f_{\mathsf{Prove}}(\hat{w})$.

We can now define f-zero-knowledge as the indistinguishability of the real and simulated commitments and proofs as follows.

Definition 3 (f-Zero-Knowledge). *A CP-SNARK* CP *is* f*-zero-knowledge for* f*:*$(f_{\mathsf{Com}}, f_{\mathsf{Prove}})$ *if there exists a simulator* $(\mathsf{SSetup}, \mathsf{SCom}, \mathsf{SProve})$ *s.t. for all p.p.t. adversaries* $\mathcal{A}$*, for all* λ *and* $n \in poly(\lambda)$*:*

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{ParGen}(1^\lambda) \\ ck \leftarrow \mathsf{Setup}(pp, n) \\ b \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Com}, \mathcal{O}\mathsf{Prove}}(ck) \end{array} : b = \mathsf{true}\right] \approx \Pr\left[\begin{array}{l} pp \leftarrow \mathsf{ParGen}(1^\lambda) \\ (ck, st_{\mathsf{S}}) \leftarrow \mathsf{SSetup}(pp, n) \\ b \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{SCom}, \mathcal{O}\mathsf{SProve}}(ck) \end{array} : b = \mathsf{true}\right]$$

$\mathcal{O}\mathsf{Com}(t, m)$**:** *For* $m \in \mathcal{M}_{ck,t}$*, samples a random* $o \leftarrow \mathcal{O}_{ck,t}$*, runs* $c := \mathsf{Com}_{ck}(t, m, o)$*, and outputs* (c, o)*.*

$\mathcal{O}\mathsf{Prove}(R^{\mathsf{CS}}, \hat{x}, \hat{w})$**:** *It outputs* $\pi \leftarrow \mathsf{Prove}_{ck}(\hat{x}, \hat{w})$ *if* $R^{\mathsf{CS}} \in \mathcal{R}^{\mathsf{CS}}$ *and* $(\hat{x}, \hat{w}) \in R^{\mathsf{CS}}$*. Otherwise, outputs* $\perp$*.*

$\mathcal{O}\mathsf{SCom}(t, m)$**:** *If* $m \in \mathcal{M}_{ck,t}$*, it runs* $(c, o, st_{\mathsf{S}}) \leftarrow \mathsf{SCom}(st_{\mathsf{S}}, t, f_{\mathsf{Com}}(m))$ *and outputs* (c, o)*. Otherwise, it outputs* $\perp$*.*

$\mathcal{O}\mathsf{SProve}(R^{\mathsf{CS}}, \hat{x}, \hat{w})$**:** *If* $R^{\mathsf{CS}} \in \mathcal{R}^{\mathsf{CS}}$ *and* $(\hat{x}, \hat{w}) \in R^{\mathsf{CS}}$*, then it runs* $(\pi, st_{\mathsf{S}}) \leftarrow \mathsf{SProve}(R^{\mathsf{CS}}, st_{\mathsf{S}}, \hat{x}, f_{\mathsf{Prove}}(\hat{w}))$ *and outputs* π*. Otherwise, outputs* $\perp$*.*

Shared Simulator State. An obvious difference between our simulator and a regular zero-knowledge-proof simulator is the shared simulator state, which also contains the zero-knowledge trapdoor keys. Thus, the zero-knowledge property we define explicitly considers CP-SNARKs. Such zero-knowledge simulators with a shared state for the commitment and proof simulators on CP-SNARKs were also used by previous schemes [10, 11]. However, [11] does not model any "leaky" property, whereas [10] does but in a more specific way, as it models the extra leakage through a bounded number of evaluations of polynomials. Our notion is more generic, through the abstraction as an arbitrary function f which only becomes concrete when the construction is made.

Capturing Security for Deterministic Commitments. The motivation behind our simulation is to allow *deterministic* "commitments", such as g^x, and non-interactive proofs on them. These have been used in the current ATS constructions [17, 30], where the public key and further group elements derived from the same secret key serve as commitment values to the signers' keys.

In terms of simulation, we do not have to simulate an opening in deterministic schemes, but simulating the commitment itself would not be feasible. Unlike standard randomized commitment schemes, where any commitment can usually be validly opened to any message via the simulator, deterministic schemes require a unique commitment for each message that the simulator has to output right. Providing $f_{\mathsf{Com}}(m)$ to the simulator solves that problem. This feature is particularly useful when f_{Com} is a one-way function, e.g. the message m is a secret key and $f_{\mathsf{Com}}(m)$ is the corresponding public key. For example, in the proof of our generic construction, the additional commitment to the individual secret key sk_i will be simulated using this feature, despite only knowing the corresponding multi-signature public key pk_i.

Restrictions on Leakage? The choice of f determines how strong the f-zero-knowledge property is. The two extremes are either to set $f_{\mathsf{Com}} := \perp$ and $f_{\mathsf{Prove}} := \perp$, giving the common zero-knowledge notion, or to set f_{Com} and f_{Prove} to the identity function which allows simulators running the original Com and Prove algorithms internally. In our MATS construction, will use a mix of leaking parts of the witnesses and only a one-way function thereof. This gives the needed strength (but also not more) in the security proof, yet is weak enough to be efficiently realizable via deterministic commitments.

Simulation Extractability. Soundness of a proof system that yields a *proof-of-knowledge* is expressed through an *extractor* that can efficiently compute the witness from an adversarially provided proof. As argued in Section 2, we will need some sort of extractability in the presence of simulated proofs to securely construct threshold signatures along the blueprint from [17, 30]. Relying on the stronger notion of *simulation extractability* [32], that guarantees such an extractor even when the malicious prover is given access to simulated proofs, is not an option though: Simulation extractability the non-malleability of the created proofs, but we need homomorphic proofs for the signature construction. Our definition is inspired by other simulation extractability notions [14, 34] which allow specific malleabilities, and we refer to Appendix A for a detailed comparison.

Policy-based Simulation Queries. Luckily, we do not need full-fledged simulation extractability, but a weaker version along the lines of Faonio et al.'s policy-based simulation extractability notion [22] will be sufficient. [22] defines policies that can put restrictions on the type of simulation queries or the proof forgeries that the adversary is allowed to make. For example, our policy for the generic threshold construction will exploit the fact that we only need to simulate $\mathsf{IP}_{\mathbb{G}}$ proofs $\pi_{i,j}$ for an honest signer i and be set accordingly.

We modify the definition of [22] in several ways. The first change is on how we formulate a policy. Faonio et al. require that a policy contains two functions $\Phi := (\Phi_0, \Phi_1)$. Φ_0 is run before any simulation queries, and it is used to set certain parameters in the policy by sampling from a distribution. We simplify our definition by removing Φ_0 since we do not need to sample values for our policies. [22] defines Φ_1 as a predicate that contains the main controls over the game's view and checks whether the adversary followed the policy. We define two dedicated predicates Φ_{sim} and Φ_{ext} that check the rules of the policy related to simulators and extraction, respectively. [22] already does such separation in their concrete policies, but we lift this distinction to the definition level. This will be useful to define simulation-extractable proofs and simulation-sound commitments on the same simulation policies.

Partial Extractors. We further generalize the definition to allow for partial extraction, i.e., relaxing the extractor to output not the full witness itself, but only some function of the witness. This is mainly for convenience in our security proof, as part of the witnesses will be publicly known anyway (recall that the commitments $c_{\mathbf{b}}$ and $c_{\mathbf{1}}$ in Equation 1 are used for compactness, not for any secrecy requirements). We extend the policy-based simulation extractability definition by first allowing the adversary to output some additional information aux, which is also given to the extractor. When compliance with the extraction policy is checked in the winning condition, this aux is given in addition to the extracted value $\hat{w}$. In our application, we are going to use this generalization to only demand for extraction of the bit vector $\mathbf{b}$ from the proofs π_{Agg}, π_{thr}, and π_{BIT}. The witnesses of these proofs also contain the opening for the commitments of the public keys $\mathbf{pk}$, and the weight vector $\mathbf{w}$ – but these openings are public anyway, so there is no need to extract them again in the security proof.

Our formal definition of policy-based simulation extractability is presented below. Note that the leakage function f is explicitly parameterized just to be compatible with the simulator definitions.

Definition 4 (Policy-Based Simulation Extractability). *A CP-SNARK* CP *is f-Φ-simulation extractable (f-Φ-SE) if there exists an efficient extractor such that for all efficient adversaries $\mathcal{A}$,* $\Pr\left[\mathsf{Exp}^{f\text{-}\Phi\text{-}SE}_{\Pi,\mathcal{A},\mathcal{E}}(1^\lambda) = \mathsf{true}\right] \leq \mathsf{negl}_{SE}(\lambda)$.

Simulation-Sound Binding. Let us recall where we would need the binding property in the first place. CP-SNARKs allow proving various relations on the same commitment. The most common use case is proving the conjunctions of relations, which can be trivially done by the concatenation of the two individual proofs [11]. The corresponding knowledge-extractor can be built by using the individual extractors and relying on the binding property of the commitment, ensuring that both extracted values are identical.

The same argument cannot be made when simulation extractability is needed: the binding property assumes that the Setup algorithm is run honestly, not via a simulator, and it also does not capture that the adversary can see simulated

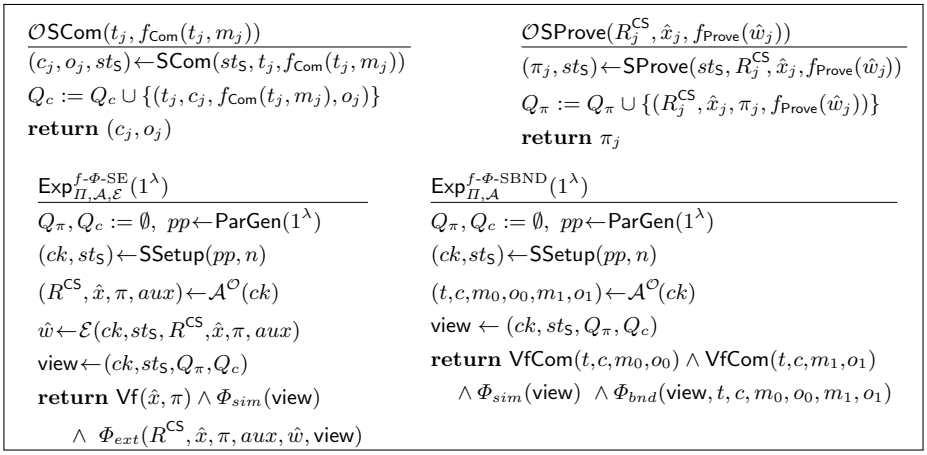

Fig. 1. Definitions of Φ-Simulation Extractability and Φ-Simulation-Sound Binding games which use the $\mathcal{O}\mathsf{SCom}$ and $\mathcal{O}\mathsf{SProve}$ oracles.

commitments or proofs. Thus, the binding property does not necessarily hold in the existence of commitment and proof simulators, and we cannot use it to argue that several extracted values in the presence of such a simulator are equivalent.

f-Φ-Simulation-Sound Binding. We introduce the notion of simulation-sound binding which requires that the binding property holds even when the adversary has access to commitment/proof simulators. We define the property on *policy-based* simulations for a policy $\Phi := (\Phi_{sim}, \Phi_{bnd})$. The adversary gets access to simulation oracles that can only be queried in accordance with Φ_{sim}, and eventually outputs two messages m_0, m_1 and openings o_0, o_1 that lead to the same commitment c. The binding and the simulation-extractability properties can be used in combination by setting a common Φ_{sim}. The binding policy Φ_{bnd} specifies when the adversary's output is valid. It must enforce that $m_0 \neq m_1$, and can impose further limits, e.g., that the commitment c must be of a specific type.

A similar simulation-sound binding property was defined on trapdoor commitment schemes by first [28], but in a commitment-only setting, explored more extensively in Appendix A. We are not aware of any works that define this property on commit-and-prove systems by also considering the simulated proofs.

Definition 5 (f-Φ-Simulation-Sound Binding). *A CP-SNARK* CP *is f-Φ-simulation-sound binding (f-Φ-SBND) if for all efficient adversaries $\mathcal{A}$,* $\Pr\left[\mathsf{Exp}^{f\text{-}\Phi\text{-}SBND}_{\Pi,\mathcal{A}}(1^\lambda) = \mathsf{true}\right] \leq \mathsf{negl}_{SBND}(\lambda)$.

5 Multiverse Ad Hoc Threshold Signatures (MATS)

We present our construction of a MATS scheme. A MATS extends the concept of ad hoc (weighted) threshold signatures – where each signer is assigned a weight and the threshold is defined ad hoc during verification over the sum of the signer

weights – by *ad hoc groups* (the multiverse). In the multiverse, signers can re-use their long-term secret keys across several signing groups.

Das et al. [17] and Garg et al. [30] only support fixed group of signers, but also sketch how their ATS constructions can be extended to the MATS setting. However, there was neither a formal definition of such a scheme nor a security proof of the sketched construction. We present a definition to cover both ad hoc groups and ad hoc thresholds, and a generic construction that provably satisfies that notion. It is worth to note that the use cases for ATS schemes which were proposed by [17, 30] – such as proof-of-stake consensus algorithms [41], decentralized autonomous organizations [21], and decentralized oracle networks [19] – allow their users to join and leave the system dynamically, necessitating ad hoc signing groups. Our construction uses a multi-signature scheme and a CP-SNARK for proving the correct key aggregation. It closely follows and extends the core ideas of the prior works [17, 30], but provides the construction in a more generic manner, and addressing the issues discussed in Section 2.

5.1 MATS Definition

Our MATS definition is mainly based on the definition from the original multiverse threshold scheme (MTS) by Baird et al. [4]. We adapt their model to our notation and extend it to cover ad hoc thresholds – incorporating the necessary parts of the models from [17, 30]. We also make the model more flexible to allow for generic constructions from any multi-signature as defined in Def. 11, instead of tailoring it to BLS-signatures. Our first change to the syntax by [4] is that we define a Setup algorithm that takes a maximum size for a signer group n, as an input. This aligns with vector commitment setup algorithms from [17, 30], where specifying vector size is necessary. Just as in [4], we provide an individual key generation algorithm Kg that outputs a key pair $(sk^{(u)}, pk^{(u)})$ for a signer u. Note that we use indexing by $sk^{(u)}$ as a way to index a signer in the system and sk_i as a way to index a signer in a vector of signers.

Universe Generation. In [4], generation of group public keys, called universe generation, involves two steps. UGen_1 is run by each signer to create a partial key to finalize the group public keys. Our UGen_1 algorithm takes the vector of signer public keys $\mathbf{pk}$ and the secret key corresponding to the index i in the vector, sk_i and outputs partial key prk_i for the signer. In the construction level, as also noted by [30], the UGen_1's output will not depend on $\mathbf{pk}$, but knowing only the index i is enough. However, requiring the vector $\mathbf{pk}$ as an input is the most straightforward way in the real world to determine the index i for a signer, so we keep $\mathbf{pk}$ as an input. Unlike [4], our UGen_1 algorithm doesn't require the weight vector and threshold as inputs. Instead, multiple thresholds are supported, and the weight vector is only needed in the second step, UGen_2, so signers can adjust weights without using their secret key.

UGen_2 is the final step to form group verification and aggregation keys. As in [4]'s definition, it takes the partial key vector $\mathbf{prk}$, the public key vector $\mathbf{pk}$, and the weight vector $\mathbf{w}$ as input to output the verification and aggregation keys.

Again, the threshold thr is not an input to this algorithm since we aim for ad hoc tresholds, that are only determined during verification.

Generic Syntax. Another minor change is that our definition is made more generic to allow instantiations from any suitable multi-signature, while [17] and [30] included some behaviour that is typical for BLS-based approaches in their syntax. More precisely, we give PSign the signer subset PK as optional input in our definition, which in turn allows to consider PSign as an interactive protocol. Finally, we have the threshold thr as an additional input to [4]'s signature verification algorithm so that we can use the same group public key to verify the signatures with different thresholds.

Definition 6 (Multiverse Ad Hoc Threshold Signature (MATS)). *A multiverse ad hoc threshold signature scheme is a tuple of algorithms such that:*

$\mathsf{Setup}(1^\lambda, n) \to pp_{\mathsf{MATS}}$**:** Outputs the public parameters pp_{MATS} for security parameter 1^λ and the maximum group size n. We only make pp_{MATS} an explicit input to Kg and assume that it is an input to all algorithms, implicitly.

$\mathsf{Kg}(pp) \to (sk^{(u)}, pk^{(u)})$**:** Outputs a secret and public key pair $(sk^{(u)}, pk^{(u)})$.

$\mathsf{UGen}_1(sk_i, \mathbf{pk}) \to prk_i$**:** For an index i of a vector of public keys $\mathbf{pk}$ and corresponding secret key sk_i, outputs a partial key prk_i that can be used to compute group verification and aggregation keys.

$\mathsf{UGen}_2(\mathbf{prk}, \mathbf{pk}, \mathbf{w}) \to (vk, ak)$**:** Given partial keys, individual public keys, and the corresponding weight vectors, outputs a signature verification key vk and a signature aggregation key ak.

$\mathsf{PSign}(sk^{(u)}, PK, m) \to ps^{(u)}$**:** Given a secret key $sk^{(u)}$, a message m and optionally a subset of signers $PK \subseteq [n]$, outputs the partial signature $ps^{(u)}$.

$\mathsf{Combine}(\{ps_i\}_{i\in S}, ak) \to \sigma$**:** It takes the partial signatures of signers that corresponds to the set of indexes S on $\mathbf{pk}$ and the aggregation key ak as an input. It outputs the combined signature σ.

$\mathsf{Vf}(vk, \sigma, m, \mathsf{thr}) \to b \in \{\mathsf{true}, \mathsf{false}\}$**:** Checks if σ is a valid signature on the message m for the verification key vk that satisfies the threshold thr.

The correctness is defined in Appendix D.1 and follows the definition of [4].

MATS *Unforgeability.* Our unforgeability definition for MATS is based on [4], but is adapted to the changes we made in the syntax. One additional difference is that the adversary in our setting starts by choosing the maximum group size n which is necessary to perform the setup accordingly. Then, just as in [4]'s definition, the adversary gets access to the oracles for creating honest signers, requesting an honest signer's partial key on a universe and partial signatures on message of its choice, as well as corrupting an initially honest signer. Similar to [4], the adversary can create malicious parties by itself, without needing to invoke any oracle. Finally, the adversary must output a non-trivial and valid forgery $(m^*, \sigma^*, \mathsf{thr}^*, \mathbf{pk}, \{prk_i\}_{pk_i\in\mathbf{pk}\setminus H}, \mathbf{w})$. The second part of the forgery $(\{prk_i\}_{pk_i\in\mathbf{pk}\setminus H}, \mathbf{w})$ is used to derive the group verification and aggregation keys via UGen_2. Note that we abuse the notation by using the intersection

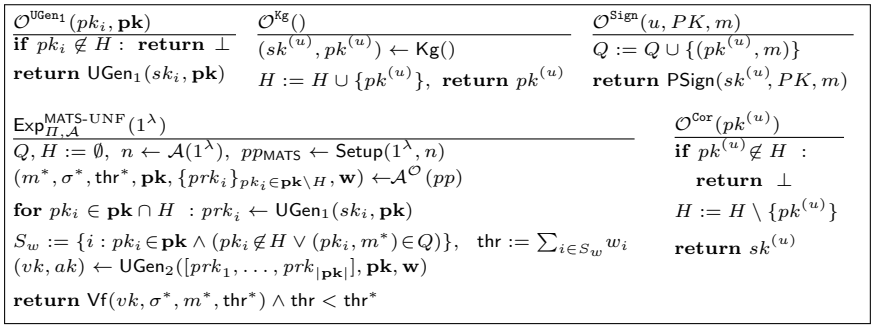

Fig. 2. MATS Unforgeability Game. u is a system-wide index for $sk^{(u)}$ and i is the index for sk_i in a vector of signers.

of a vector and set, $\mathbf{pk} \setminus H$ by treating the vector $\mathbf{pk}$ as a set without an order, for simplicity. The adversary outputs the partial keys of the malicious signers in the vector only and the partial keys of honest parties are computed by the challenger. A non-trivial forgery means that $(m^*, \sigma^*, \mathsf{thr}^*)$ from the forgery must be valid against the computed group verification key and the adversary must not control signers that reach the threshold or is a trivial combination of honest signature contributions. The detailed game is given in Figure 2.

Definition 7 (MATS Unforgeability). *A* MATS *scheme is unforgeable if for all efficient adversaries* $\mathcal{A}$, $\Pr\left[\mathsf{Exp}^{\mathsf{MATS}\text{-}UNF}_{\Pi,\mathcal{A}}(1^\lambda) = \mathsf{true}\right] \leq \mathsf{negl}_{\mathsf{MATS}\text{-}UNF}(1^\lambda)$.

5.2 Construction of MATS for DL-based Key Pairs

We now provide a MATS construction relying on a CP-SNARK as defined in Section 4 and a multi-signature scheme. Since we need to prove relations and statements over keys of the multi-signature scheme, we need to be able to express them properly. For this, we rely on a DL-based setting and assume that the multi-signature has keys $sk \in \mathbb{Z}_p$ and $pk := g^{sk}$.

Restriction to DL (and how to avoid it). Our "generic" construction only supports DL-based key pairs, thus limiting its general applicability to multi-signatures with such keys. We opted for this limitation, to make the definitions of the needed homomorphic properties over the key spaces of the signatures simpler. As our DL-based blueprint still captures the most prominent multi-signature schemes, BLS and Schnorr-based, [7] we consider this as a reasonable trade-off. Our construction can be generalized over more abstract mathematical structures, along the following ideas: The underlying multi-signature scheme *1)* must have a vector space $\mathbb{F}^m$ as the secret key space over some field $\mathbb{F}$, *2)*, there should be a deterministic mapping $map(sk) = pk$ between the corresponding secret and public keys, and *3)* there should be a homomorphism $\star$ such that $map(sk_1 + sk_2) = pk_1 \star pk_2$. Then, the inner-product proof $\mathsf{IP}_{\mathbb{G}}$ we have for the

DL-based key pairs can be replaced with a matrix-vector product proof showing that the product of the m by n matrix of secret keys $[sk_1|\ldots|sk_n]$ and the vector $[b_1,\ldots,b_n]^T \in \mathbb{F}^n$ maps to apk: $map([sk_1|\ldots|sk_n]\cdot[b_1,\ldots,b_n]^T) = apk$. The BIT proof must show that all b_i values are equal to zero or identity element of $\mathbb{F}$.

Comparison to Previous Schemes. We build our MATS scheme from the ATS schemes [17, 30], following their common blueprint we informally described in Section 2. We first give a brief overview of how our work differs from theirs on the ATS level, i.e., ignoring the changes for the multiverse, and then sketch the extensions that turn the ATS scheme into a multiverse version.

In the existing works, the verification and aggregation keys were expressed through auxiliary and BLS-specific values. This missing separation between the BLS multi-signature and additional parts hindered a generic security analysis [30] or lead to an incorrect argumentation in the proof [17].

Our generic construction clearly separates both, and for each makes the functional and security properties explicit. During the group key generation (and setup), generic commitments to the secret keys and weight vector are generated. Computing the verification and aggregation keys is performed using the homomorphic properties of the underlying commitments and proofs, instead of relying on construction-specific structures. Likewise, in signature combination, we create a generic commitment for the bit vector, compute proofs on these commitments, and make the required homomorphisms and proven statements explicit. Ultimately, our construction makes a security proof that treats the underlying multi-signature as a black-box possible. In [17]'s flawed proof, the main problem was a missing argumentation on how to simulate the verification key vk and the aggregation key ak. Our construction describes vk as a commitment and ak as a tuple of non-interactive proofs, both are values that we can simulate via our CP system.

Finally, to turn their ATS schemes into a MATS scheme, we follow the observation that was already made by both works [17, 30]: repeating the group public key/universe generation multiple times with different signer groups does not harm the security of the construction.

Compatability Requirements. We require some basic compatibility from both building blocks, mainly requiring that their group structure and types are compatible, which is needed for the correctness of the generic construction. We summarize these requirements here and refer to Appendix D.2 for the formal definitions of *compiling multi-signatures* and *compiling CP-SNARKs*. The multi-signature must have key spaces as a DL-key pair on the pairing group $\mathbb{G}$ and the aggregation method must be the group multiplication, $\mathsf{MS.KAg}(PK) := \prod_{pk\in PK} pk$ for the signer public key set PK (Definition 20).

$$
\begin{aligned}
\mathsf{IP}_{\mathbb{G}} &:= \{\big(\gamma,(\mathbf{u},\mathbf{v})\big) \;:\; \big(\gamma,(\mathbf{u},\mathbf{v})\big) \in \mathbb{G}\times(\mathbb{Z}_p^n,\mathbb{Z}_p^n) \;\wedge\; \gamma = g^{\mathbf{u}\cdot\mathbf{v}}\}\\
\mathsf{IP}_{\mathbb{Z}_p} &:= \{\big(\gamma,(\mathbf{u},\mathbf{v})\big) \;:\; \big(\gamma,(\mathbf{u},\mathbf{v})\big) \in \mathbb{Z}_p\times(\mathbb{Z}_p^n,\mathbb{Z}_p^n) \;\wedge\; \gamma = \mathbf{u}\cdot\mathbf{v}\} \qquad (2)\\
\mathsf{BIT} &:= \{(\bot,\mathbf{u}) \;:\; \mathbf{u}\in\mathbb{Z}_p^n \;\wedge\; \mathbf{u}\in\mathbb{Z}_2^n\}
\end{aligned}
$$

Definition 21 gives the five requirements for the underlying CP-SNARK which needs to cover the relations $\mathsf{IP}_{\mathbb{G}}$, $\mathsf{IP}_{\mathbb{Z}_p}$, and BIT from Eq. 2. The definitions are parameterized with a size bound $n \in \mathbb{N}$ that defines the vector size and public parameters $pp := pp_{\mathsf{MS}} \in \mathsf{MS.Pg}(1^\lambda)$ which is assumed to describe a group $\langle g \rangle := \mathbb{G}$ with prime order p. Note that the relation BIT is trivial to prove as it is, but the CP-SNARK relation we define on it is a non-trivial relation to prove as it requires proving that a committed vector is a bit vector.

The five requirements in Definition 21 can be summarized as follows. First, the CP-SNARK must contain commitments with appropriate message spaces to commit to the vectors $\mathbf{sk}$, $\mathbf{w}$, and $\mathbf{b}$, denoted by $t_{\mathbb{G}}$, t_1, t_2, such that it can compute the necessary proofs on these commitments.

The second and third requirements define the homomorphic properties to compute vk and ak. During the universe generation, each signer will create a commitment to its own secret key under the correct position in the vector. Then, relying on the commitment homomorphism $F_{t_{\mathbb{G}},n}$, we can compute a commitment to the secret key vector $\mathbf{sk}$. Similarly, the proof homomorphism $X_{t_{\mathbb{G}},n}$ will be used to combine the individual signers' proofs on their individual keys, to get the proofs on the vector $\mathbf{sk}$ we homomorphically computed. These proofs will become a part of the aggregation key ak.

The fourth and fifth requirements define the necessary homomorphic properties that we will rely on to create the final $\mathsf{IP}_{\mathbb{G}}$ proof in the $\mathsf{Combine}$ algorithm. In more detail, the fourth requirement gives the commitment homomorphism $F_{t_2,n}$ on type t_2 commitments for the vector addition. The third requirement provides the proof homomorphism $X_{t_2,n}$ for proofs that contain the same $t_{\mathbb{G}}$ commitment, but different t_2 commitments to compute the proof corresponding to the inner product of the same $t_{\mathbb{G}}$ commitment and the sum of t_2 commitments.

CP-based MATS Construction. Our generic construction is formally described in Figure 3. We start by describing how the intuitive blueprint from Section 2 is expressed via our CP-SNARK abstraction.

Setup. The Setup algorithm creates the commitment key ck and the public parameters for the multi-signature scheme. It further sets some default openings for the base vectors $\mathbf{e}_j$'s and the weight vector $\mathbf{w}$. All signers need to commit to these vectors using the same opening during the key generation, and we do not need these commitments to be hiding. Thus, we simply set some openings as global parameters. Another opening is set for the vector of zero's which will become clear together with the key generation part. We note that these openings could simply be set to 0 bits. (The CP instantiation presented in Section 4 uses deterministic commitments, so there are no openings anyway.)

Universe Generation. Each signer creates their individual key pairs as a multi-signature key pair (sk_i, pk_i) together with the corresponding proof-of-possession pop_i. Universe generation contains two algorithms and we implicitly assume in these algorithms that the input vector has size n, the maximum group size, for

notational simplicity. The potential empty slots in the weight or secret key vector for smaller size groups can be filled using zero. In UGen_1, a signer commits to its secret in the vector form as $sk_i \cdot \mathbf{e}_i$ and computes inner product proofs for all base vectors $\mathbf{e}'_j s$. These proofs will be used to set the aggregation key. Finally, the signer outputs the public key, the commitment, and the corresponding proofs as part of the prk_i to be used in UGen_2.

In UGen_2, a signer checks if prk_k's of other signers are well-formed and sets the verification key and the aggregation key. The first check is verifying the *pop* for the pk_k to prevent maliciously chosen pk_k's as regular. The second check is verifying all inner-product proofs, $\mathsf{IP}_{\mathbb{G}}$, for the commitment U_k that corresponds to pk_k. If any of these checks do not hold, pk_k must be excluded from the group key. This is done by setting a dummy key $(sk_k := 0, pk_k := 1_{\mathbb{G}})$, creating the dummy commitment $U_k := \mathsf{Com}(t_{\mathbb{G}}, \mathbf{0}, o_{\mathbf{0}})$ and the weight $w_k := 0$. As the opening of U_k is known, the signer can also compute the dummy inner-products for the new pk_k and form a dummy prk_k. After checking all prk_k's and making the necessary changes, the signer evaluates the commitment to the vector of all secret keys $\mathbf{sk}$, U. Similarly, the signer evaluates the individual proofs that came from all other signers and forms the inner-product proofs of $\mathbf{sk}$ with all base vectors $\mathbf{e}_j$'s. While U forms the verification key together with the commitment to the weight vector W, the evaluated proofs form the aggregation key ak. Note that the elimination of ill-formed keys as above was also applied in [30].

Signature Generation & Verification. The algorithm PSign simply runs the $\mathsf{MS.MulSign}$ algorithm and outputs the partial signature. The $\mathsf{Combine}$ algorithm combines the multi-signature σ' using the partial signatures, evaluates the homomorphic proof π_{Agg} using π_j's in ak, and also computes the proofs π_{BIT} and π_{thr}. The final signature $\sigma := (\sigma', apk, B, \pi_{\mathsf{Agg}}, \pi_{\mathsf{BIT}}, \pi_{\mathsf{thr}}, \mathsf{thr}')$ contains the multi-signature, aggregated key apk, commitment to the bit vector B, the proofs, and the satisfied threshold thr'. We note that if the key and signature size of the underlying multi-signature is independent of the group size and the underlying commit-and-prove system is succinct, we get compact MTS signatures.

A signature verifier checks the validity of the proofs, verifies the multi-signature σ' for the aggregated key apk, and checks if $\mathsf{thr} \geq \mathsf{thr}'$.

Correctness. The correctness easily follows from Definitions 20 and 21.

Theorem 1. *If* MS *is a compiling multi-signature scheme and* CP *is a compiling CP-SNARK, then the construction in Figure 3 is a correct* MATS.

Leakage & Policy. As motivated in Section 2 it will be impossible to assume full zero-knowledge properties from the CP system, due to several deterministically derived values as part of key generation; or to rely on unrestricted simulation-extractability properties due to the homomorphisms required from the commitments and proofs. Thus, we must define the leakage function $f := (f_{\mathsf{Com}}, f_{\mathsf{Prove}})$ used to express the relaxed simulation guarantees, as well as the policies $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ for the simulation extractability/binding properties.

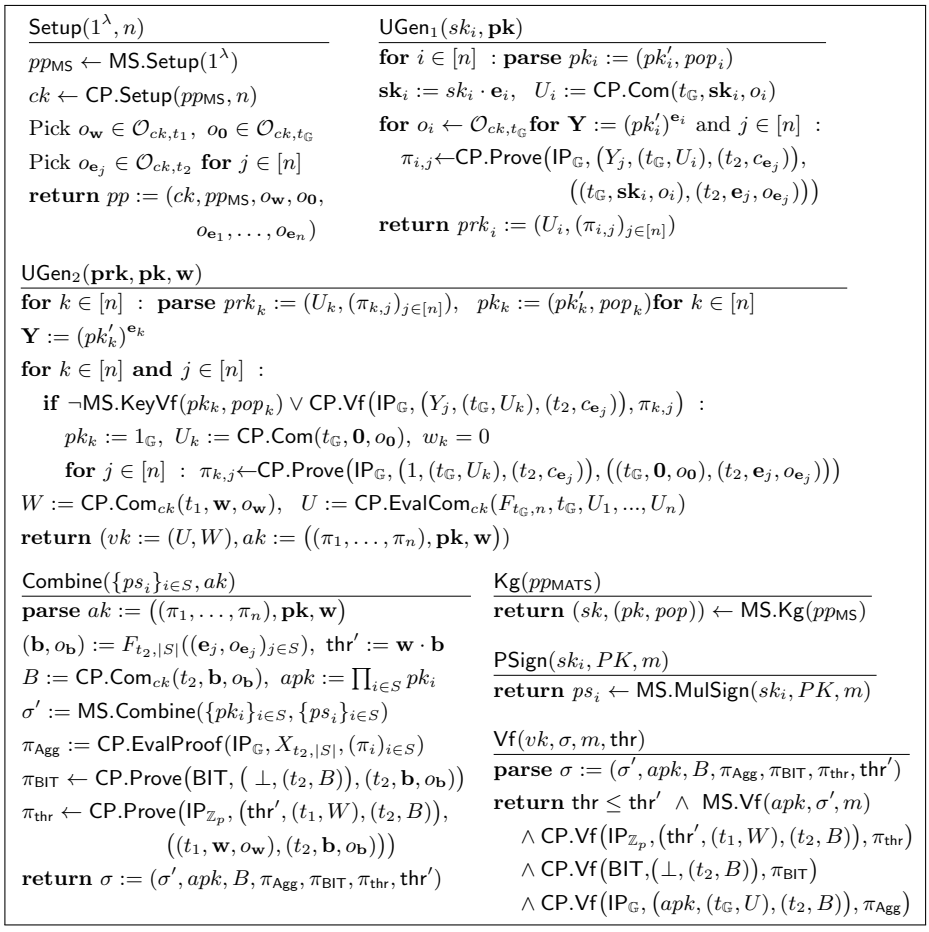

Fig. 3. Generic MATS construction.

Zero-knowledge Leakage f. The witness of the proven relations contain $\mathbf{sk}$, $\mathbf{b}$, and $\mathbf{w}$. Proving statements over the vectors $\mathbf{b}$, and $\mathbf{w}$ is done to achieve succinctness rather than to hide them. Thus, the leakage function leaks all the information about the openings of t_1 and t_2 commitments while for the $t_{\mathbb{G}}$ commitments (which contain the secret keys), it only leaks the vector of $\mathbb{G}$ elements with the $t_{\mathbb{G}}$ vector in the exponent and corresponds to $\mathbf{pk}$ for the committed $\mathbf{sk}$.

Policy Definition. Finally, in Figure 4 we define the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ which specifies the allowed simulation queries, the part that requires extraction, and the condition for the binding definition. Writing Φ_h denotes that the policy is parameterized with $h \in \mathbb{G}$. In the unforgeability proof, we will need to simulate an honest signer and only know the public key of the honest signer. The group element h serves as a parameter that we will replace with the honest signer's public key in the unforgeability proof.

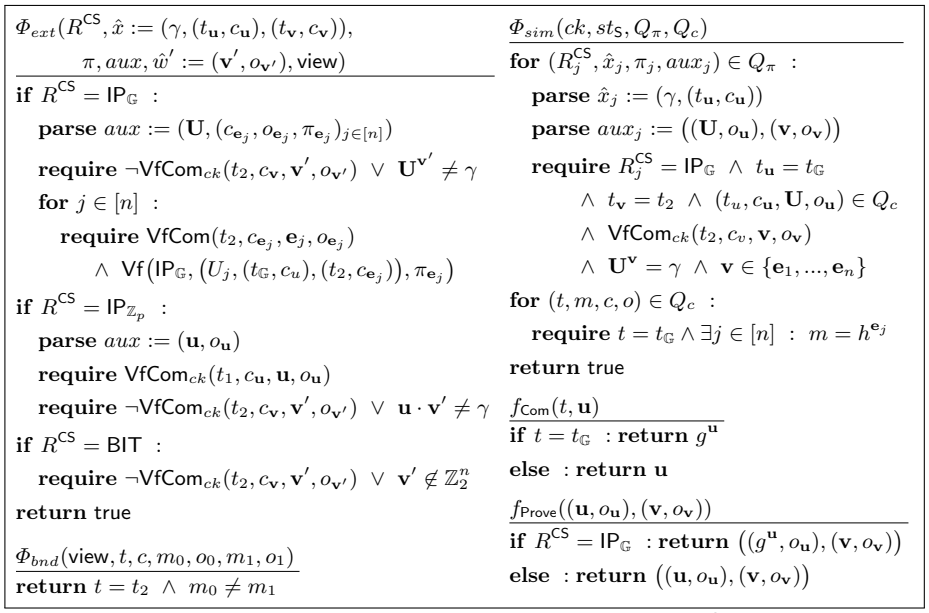

Fig. 4. Leakage functions and policies for generic MATS scheme.

In our simulation policy Φ_{sim}, we restrict simulation queries to only the essential ones for the unforgeability proof to keep our definition weak. For readability, we define $\Phi_{sim} := \Phi_c \wedge \Phi_\pi$ which are sub-policies related to the commitment simulator and the proof simulator, respectively. The commitment simulator will only be used to simulate a commitment on $sk_i \cdot \mathbf{e}_i$ in Kg_1 where we do not know sk_i. Thus, Φ_c allows commitment simulator queries only with the $t_{\mathbb{G}}$ commitments and for the verifying messages that are vectors in the form of $h^{\mathbf{e_j}}$ for some j.

For the proof simulator, we only need to simulate $\mathsf{IP}_{\mathbb{G}}$ proofs $\pi_{i,j}$'s in Equation 2 which are between $\mathbf{sk}_i$ and $\mathbf{e}_j$. Hence, we only permit simulation queries for $R^{\mathsf{CS}} = \mathsf{IP}_{\mathbb{G}}$ on relevant vectors. We check that the queried vectors satisfy the restricted form by using the bookkeeping from the commitment simulator, Q_c, and verifying the commitment opening of $c_{\mathbf{v}}$, $(\mathbf{v}, o_{\mathbf{v}})$.

The extractability policy must ensure that we can extract the bit vector $\mathbf{b}$ from the forged signature for the unforgeability proof. In the unforgeability proof, we will run the extractor on proofs that we already have some information about the openings of $t_{\mathbb{G}}$ and t_1 commitments which correspond to the vector of public keys and weights. Thus, the extracted $\mathbf{b}$ value must match to $\mathbf{pk}$ and $\mathbf{w}$.

One way to ensure this match could be asking the extractor to output candidate vectors of public keys and weights together with the bit vector for the corresponding proofs and argue that they must be identical to the original $\mathbf{pk}$ and $\mathbf{w}$. We could easily argue that the weight vectors must be identical by the binding property of t_1 commitments. However, for $t_{\mathbb{G}}$ commitments, we cannot rely on a binding property using the public key vectors as the message space of $t_{\mathbb{G}}$ commitments is $\mathbb{Z}_p^n$. The only remaining option would be relying on the

soundness of $\mathsf{IP}_{\mathbb{G}}$ proofs π_j's in ak which creates some sort of circular argument that we rely on the proof scheme's soundness to show the extractability property.

Instead of dealing with all these details, we supply all available information about $t_{\mathbb{G}}$ and t_1 commitments as auxiliary information to the extractor using the extractor policies and require extracting a valid $\mathbf{b}$ vector based on these values.

Lastly, the binding policy Φ_{bnd} only checks whether the commitment collision is found for a type-t_2 commitment. This is sufficient, as we only require binding for the bit-commitments in our proof.

5.3 Security Analysis of the MATS Scheme

Finally, we can show the unforgeability of the MATS construction in Figure 3.

Theorem 2 (Unforgeability). *If* MS *is an unforgeable multi-signature,* CP *is f-zero-knowledge, f-Φ_h-simulation extractable, and f-Φ_h-simulation-sound binding for the function f and the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ in Fig. 4 for all $h \in \mathbb{G}$, then the* MATS *construction in Fig. 3 is unforgeable according to Def. 7.*

An adversary breaking the unforgeability of the MATS scheme must come up with a non-trivial forgery $\sigma := (\sigma', apk, B, \pi_{\mathsf{Agg}}, \pi_{\mathsf{thr}}, \pi_{\mathsf{BIT}}, \mathsf{thr}')$. Soundness ensures that the proofs $\pi_{\mathsf{Agg}}, \pi_{\mathsf{thr}}, \pi_{\mathsf{BIT}}$ in the forgery guarantee that $\mathbf{w} \cdot \mathbf{b}^* = \mathsf{thr}' \ \wedge \ \mathbf{b}^* \in \mathbb{Z}_2^n \ \wedge \ \mathbf{pk}^{\mathbf{b}^*} = apk$ holds, which in turn allows to use σ' for the multi-signature forgery. The proof is rather simple, as we can rely on the convenient properties defined for the CP system.

Proof (Sketch). Given an adversary $\mathcal{A}_{\mathsf{MATS}}$ against our MATS scheme, we construct an adversary $\mathcal{A}_{\mathsf{MS}}$ against the underlying multi-signature. $\mathcal{A}_{\mathsf{MS}}$ receives pk^*, pop^* and uses these values for a randomly chosen signer in the MATS context. $\mathcal{A}_{\mathsf{MS}}$ has oracle access to sk^* and must output a forgery (σ', m^*, PK^*). σ' is already part of our MATS signatures, which will also contain the message m^*. Thus, we only need to compute an appropriate set of public keys PK^* out of the MATS forgery that will create a MS forgery. We achieve that by extracting the bit vector $\mathbf{b}^*$ that represents the signers from the forged MATS signature.

The first challenge for this proof strategy is to simulate the UGen_1 queries without knowing the MS secret key. The UGen_1 algorithm of MTS explicitly uses the individual secret keys to commit to them and provide the inner-product proofs $\pi_{i,j}$. Here we rely on the f-zero-knowledge property to simulate both the commitments (U_i, o_i) and $\pi_{i,j}$. The commitment simulator SCom gets the public key pk^* as additional input, where SProve gets pk^* and the openings to the base vector commitments. Note that we comply with our policy Φ_{sim} here, which allows us to use the extractability feature next.

The second step is extracting the multi-signature forgery from the cumulative signature. We rely on the policy-based f-Φ-SE property that we can extract the necessary part of the witness – the signer vector $\mathbf{b}^*$ – even in the presence of f-zero-knowledge simulators. Here we also need the Φ-simulation-sound binding property, to argue that the extracted values from the three individual proofs are identical. Now, knowing $\mathbf{b}^*$ we can carve out the signer set PK^* which completes $\mathcal{A}_{\mathsf{MS}}$'s forgery. The full proof is given in Appendix E.

6 CP-SNARK Instantiation for BLS-based MATS

We now present a concrete instantiation of the CP-SNARK for a BLS-based MATS scheme as needed in Figure 3. We note that this scheme could be combined with any other pairing-based multi-signature scheme with a DL-based key pair as well. The CP construction is a minor modification of the one from [17] that we express in terms of our commit-and-prove system. The main contribution here is that we prove that this CP construction achieves the desired properties needed for the MATS scheme. The biggest insight thereby is that we require a new assumption to do so. The challenging property is the simulation-extractability, where we need to balance the needed co-existence of a simulator and a (partial) knowledge extractor. To this end, we introduce the flexible-base (n_1, n_2)-discrete logarithm problem. We show that this assumption holds under the (n_1, n_2)-DLOG problem (Def. 10) in the algebraic group model. Finally, we prove the security of CP construction under our new assumption.

6.1 Adapted CP Instantiation from [17]

Our CP-SNARK CP-Pair is given in Figure 5. We rely on the CP instantiation from [17]. Note that this construction requires pairings, and we must now switch to an appropriate (type-3) pairing group. We assume that the pairing group definition can be parsed from the input multi-signature public parameters pp_{MS}. Thus, by exploiting the notation, we just present the pairing group description $pp_{\mathbb{G}}$ as input to the Setup algorithm. One of the differences is that we give the construction directly for type-3 pairings, whereas [17] used symmetric ones. This requires to add further pairing checks in the construction which is highlighted via the dashed boxes in Figure 5. Our policy requires that the extractor can output a valid type t_2 commitment for each relation so that we can rely on the binding property of the type t_2 commitments to get a conjunction proof. The extra checks ensure that type t_2 commitments are degree $\leq n-1$ polynomial commitments, so there will be a valid vector commitment opening to these commitments. Other differences between our adaption and [17] are mostly different efficiency optimizations, and we refer to Appendix F.1 for a detailed comparison [17] and to Appendix F.2 for an efficiency analysis. The main insight of this section is the security proof we give for the CP construction.

Realizing Commitments. As we only use deterministic commitments, we drop the openings from the notation for readability. The commitment key ck is nothing but a $(n-1, n)$-DLOG tuple. A possible optimization is to add frequently used values as part of the commitment key or CRS [17, 12]. The commitments to the vectors are polynomial commitments. We use the following notation in the construction. Let $\mathbb{H} := \{\mathsf{h}_1, ..., \mathsf{h}_n\}$ be a subgroup of $\mathbb{Z}_p^*$. The vanishing polynomial and Lagrange basis polynomials of $\mathbb{H}$ are defined as $z_{\mathbb{H}}(X) := \prod_{j\in[n]}(X - \mathsf{h}_j)$ and $\mathcal{L}_i(X) := \prod_{j\in[n]\setminus\{i\}} \frac{X-\mathsf{h}_j}{\mathsf{h}_i-\mathsf{h}_j}$. The commitment verification algorithm simply re-runs the committing algorithm for the corresponding type and checks whether

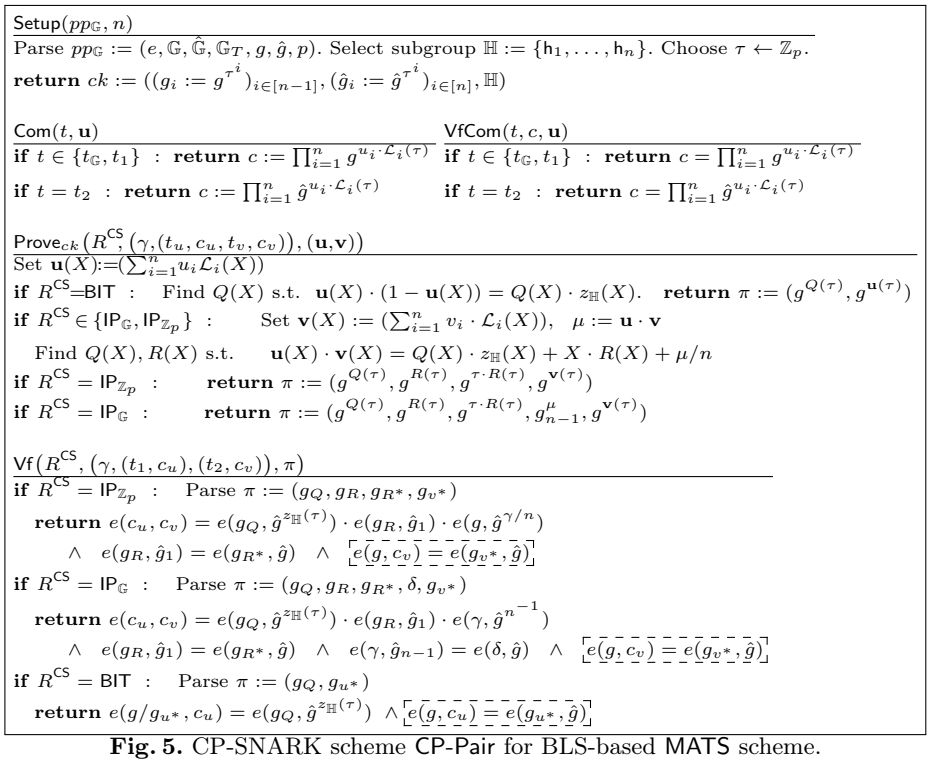

Fig. 5. CP-SNARK scheme CP-Pair for BLS-based MATS scheme.

the resulting value is equal to the commitment. We use the following notation in the construction.

Realizing Proofs. The proofs follow the prior constructions [17, 30, 12, 40] for the inner-product proofs and the hadamard-product proofs. To recap, while $\mathsf{IP}_{\mathbb{G}}$ and $\mathsf{IP}_{\mathbb{Z}_p}$ proofs are computed and verified by relying on the inner-product polynomial relation from Lemma 2, the BIT proofs rely on the hadamard-product polynomial relation from Lemma 1.

Omitted Properties. We already described CP-Pair explicitly with the relations and commitment types that match the completeness requirements of a compiling CP-SNARK. We omit the completeness proofs as they easily follow from the previous work. Similarly, succinctness proof CP-Pair is straightforward. The homomorphism proofs mainly follow [12]'s results on homomorphic linear-map vector commitments and are in Appendix F.3. The perfect f-Zero-Knowledge property CP-Pair can be shown easily by using the trapdoor τ as it has been done for the proofs relying on KZG polynomial commitments before and is in Appendix F.4.

6.2 Flexible-Base Discrete Logarithm Problem

We need a new assumption to prove the f-Φ-SE property of the CP-Pair. Recall that not realizing that such a property is needed was the main gap in [17]. In the simulation extractability game, we must be able to answer commitment and proof simulators without using the trapdoor τ. CP-Pair creates a peculiar case that we need to simulate g^{τ^i}, $\hat{g}^{\tau^i}$, and h^{τ^i} for a given h. To satisfy this need, we define an adaptive variant of the (n_1, n_2)-DLOG assumption(Def. 10) [6] where the adversary is allowed to choose vectors of generators from both source groups to get τ powers of them.

New Flexible-base n-DLOG Assumption. We define *flexible-base-(n_1, n_2)-DLOG* (FB-(n_1, n_2)-DLOG) assumption according to the observations we made above. In the FB-(n_1, n_2)-DLOG problem, the adversary first provides the vector of generators $\mathbf{H} := [H_1, ..., H_\xi] \in (\mathbb{G}^*)^\xi$ and $\hat{\mathbf{H}} := [\hat{H}_1, ..., \hat{H}_{\hat{\xi}}] \in (\hat{\mathbb{G}}^*)^{\hat{\xi}}$. Then the adversary gets $(\mathbf{H}_i := [H_1^{\tau^i}, ..., H_\xi^{\tau^i}])_{i\in[n_1]}$ and $(\hat{\mathbf{H}}_i := [\hat{H}_1^{\tau^i}, ..., \hat{H}_{\hat{\xi}}^{\tau^i}])_{i\in[n_2]}$ back. The adversary's task will be finding a polynomial that has τ as a root, but we need to introduce further notation to define it formally.

Let $\mathbf{f} := [f_1(X), \ldots, f_\xi(X)]$. Then we define $\mathbf{H}^{\mathbf{f}(\tau)} := \prod_{i=1}^{\xi} H_i^{f_i(\tau)}$. Furthermore, $\mathbf{f}_\mathbf{H}(X) := \sum_{i=1}^{n} \theta_i \cdot f_i(X)$ for $H_i := g^{\theta_i}$. If we do not know the θ_i values, we cannot compute $\mathbf{f}_\mathbf{H}(X)$ in the clear. However, we can easily compute the coefficients of $\mathbf{f}_\mathbf{H}(X)$ in the exponent. In more detail, $\mathbf{f}_\mathbf{H}(X) := \sum_{j=0}^{m} a_j \cdot X^j = \sum_{j=0}^{m}(\sum_{i=1}^{\xi} \theta_i \cdot a_{j,i}) \cdot X^j$ for some m and $a_{j,i}$'s. Thus, the j'th coefficient of $\mathbf{f}_\mathbf{H}(X)$, a_j can be computed in the exponent of the generator g as $g^{a_j} = \prod_{i=1}^{\xi} H_i^{a_j}$. This observation plays a crucial role in defining FB-(n_1, n_2)-DLOG problem. Using this equation, we can compare the equivalence of the coefficients for a term of two polynomials one by one as $g^{a_j} = g^{a'_j}$ if and only if $a_j = a'_j$. We can perform a polynomial identity check on two polynomials by checking the equality of the coefficients for all terms in those polynomials. Similarly, we can compute the degree of a polynomial $\mathbf{f}_\mathbf{H}(X)$ using the vector of polynomials $\mathbf{f}$ and the vector of group elements $\mathbf{H}$. By finding the $g^{a_j} \neq 1_\mathbb{G}$ for the maximum j, we learn the degree of $\mathbf{f}_\mathbf{H}(X)$. The degree check on these polynomials is written down as $\mathbf{deg}(\mathbf{H}, \mathbf{f})$. The degree check over $\mathbf{f}_\mathbf{H}$ becomes useful in the security proofs.

The formal definition of the flexible-base-(n_1, n_2)-DLOG assumption is presented below. In the winning condition, we check that $\mathbf{H}^{\mathbf{f}(\tau)} = 1_\mathbb{G}$, which means $\mathbf{f}_\mathbf{H}(\tau) = 0$, and $\mathbf{f}_\mathbf{H}(X) \neq 0$, so τ must be a root of the polynomial $\mathbf{f}_\mathbf{H}(X)$. A discussion of the relation between the FB-(n_1, n_2)-DLOG assumption and similar assumptions in the literature is given in Appendix F.5.

Definition 8 (Flexible-Base (n_1, n_2)-DLOG Assumption). *For all $\lambda \in \mathbb{N}$, p.p.t. adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} pp_\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda),\ (\mathbf{H}, \hat{\mathbf{H}}) \leftarrow \mathcal{A}(pp_\mathbb{G}) \\ \tau \leftarrow \mathbb{Z}_p,\ \mathbf{f} \leftarrow \mathcal{A}((\mathbf{H}^{\tau^i})_{i\in[n_1]}, (\hat{\mathbf{H}}^{\tau^i})_{i\in[n_2]}) \end{array} : \begin{array}{c} \mathbf{f}_\mathbf{H}(X) \neq 0 \\ \wedge\ \mathbf{H}^{\mathbf{f}(\tau)} = 1_\mathbb{G} \end{array}\right] \leq \mathsf{negl}_{FB\text{-}DLOG}(\lambda)$$

Below, we show that our FB-(n_1, n_2)-DLOG assumption holds in AGM [24] if (n_1, n_2)-DLOG holds. The full proof of Theorem 3 is in Appendix F.5.

Theorem 3. *FB-(n_1, n_2)-DLOG assumption in Definition 8 holds in AGM under the (n_1, n_2)-DLOG assumption.*

Necessity of New Assumption. We claim that (n_1, n_2)-DLOG was not suitable to show the f-Φ_h-SE property of CP-Pair and introduce a new assumption. One may think that if we can show that this assumption can be reduced from (n_1, n_2)-DLOG assumption in AGM, then we could show that CP-Pair is f-Φ_h-SE under (n_1, n_2)-DLOG assumption in AGM easily. This is not the case. AGM allows assuming adversaries are algebraic, but in the f-Φ_h-SE game, h is a hard-coded value in the game and not obtained from the adversary. Thus, we do not necessarily know an algebraic representation of h on other group elements. This means that we still have the problem of computing h^{τ^i} for given g^{τ^i} and h. Thus, we still need the extra power of FB-(n_1, n_2)-DLOG assumption.

6.3 f-Φ-Simulation Extractability and Binding

Finally, we can show that CP-Pair is simulation-extractable and simulation-sound binding under the policy Φ_h. The full proofs are in Appendix G and sketched below.

Theorem 4. CP-Pair *in Figure 5 is f-Φ_h-SE and f-Φ_h-SBND for the function f and the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ in Figure 4 for all $h \in \mathbb{G}$ in AGM if FB-$(n-1, n)$-DLOG assumption holds.*

The first part of the proof shows how we can simulate the commitments and proofs that are allowed in the policy, without knowing the n-DLOG trapdoor τ. Simulation is fairly simple due to the adaptive step in the FB-(n_1, n_2)-DLOG problem instance, which we use to set $\mathbf{H} := [g, h]$ and $\hat{\mathbf{H}} := [\hat{g}]$. The resulting challenge values are then used to derive the necessary values.

Then, we show that we can extract the required witnesses from the proofs of the different relations. This part uses a similar strategy to the existing works that implicitly or explicitly uses the Lemmas 1 and 2 to build inner-product or Hadamard-product proofs [12, 30, 40, 10]. The main argument in these proofs is that either a certain polynomial identity holds, so the required values can be extracted or the challenger knows a polynomial that has the trapdoor τ as a root. By finding the roots of the polynomial, τ can be computed. We follow a similar reasoning, but, we must make these arguments on $\mathbf{f}_{\mathbf{H}}(X)$, which we do not know all the coefficients in the clear. Using the FB-(n_1, n_2)-DLOG assumption, though, we do not need to compute such polynomial in the clear but even computing such polynomial vector, $\mathbf{f}$ is assumed to be hard.

Note that our proof requires the AGM to extract a witness with size n from a succinct prof that has a size independent of n as was done in many previous SNARK constructions [12, 40, 10]. By extending FB-(n_1, n_2)-DLOG to a *knowledge assumption* [5], one could try to give a proof in the standard model.

Acknowledgments. This research was partially funded by the HPI Research School on Systems Design. It was also supported by the German Federal Ministry of Education and Research (BMBF) through funding of the ATLAS project under reference number 16KISA037.

References

1. Acar, T., Nguyen, L.: Revocation for Delegatable Anonymous Credentials. In: Public Key Cryptography – PKC 2011. pp. 423–440 (2011)
2. Ananth, P., Deshpande, A., Kalai, Y.T., Lysyanskaya, A.: Fully homomorphic NIZK and NIWI proofs. In: Theory of Cryptography - (TCC) 2019. pp. 356–385 (2019)
3. Attema, T., Cramer, R., Rambaud, M.: Compressed σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In: Advances in Cryptology – ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV. p. 526–556. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-92068-5_18, https://doi.org/10.1007/978-3-030-92068-5_18
4. Baird, L., Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: Threshold signatures in the multiverse. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 1454–1470 (2023)
5. Bauer, B., Farshim, P., Harasser, P., Kohlweiss, M.: The uber-knowledge assumption: A bridge to the AGM. IACR Communications in Cryptology **1**(3) (2024). https://doi.org/10.62056/anr-zoja5
6. Bauer, B., Fuchsbauer, G., Loss, J.: A Classification of Computational Assumptions in the Algebraic Group Model. In: Advances in Cryptology – CRYPTO 2020. pp. 121–151 (2020)
7. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Asiacrypt 2018. pp. 435–464. Springer (2018)
8. Boneh, D., Komlo, C.: Threshold signatures with private accountability. In: CRYPTO 2022 -Annual International Cryptology Conference. pp. 551–581 (2022)
9. Boyen, X.: The Uber-Assumption Family. In: Pairing-Based Cryptography – Pairing 2008, vol. 5209, pp. 39–56 (2008). https://doi.org/10.1007/978-3-540-85538-5_3, http://link.springer.com/10.1007/978-3-540-85538-5_3, iSSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science
10. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In: Advances in Cryptology–ASIACRYPT 2021. pp. 3–33 (2021)
11. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. In: ACM CCS 2019. pp. 2075–2092 (Nov 2019). https://doi.org/10.1145/3319535.3339820, https://dl.acm.org/doi/10.1145/3319535.3339820
12. Campanelli, M., Nitulescu, A., Ràfols, C., Zacharakis, A., Zapico, A.: Linear-map vector commitments and their practical applications. In: Asiacrypt 2022. pp. 189–219 (2022)
13. Chaidos, P., Kiayias, A., Reyzin, L., Zinovyev, A.: Approximate lower bound arguments. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 55–84. Springer Nature Switzerland, Cham (2024)

14. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Advances in Cryptology–EUROCRYPT 2012. pp. 281–300 (2012)
15. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile Verifiable Computation. In: 2015 IEEE Symposium on Security and Privacy. pp. 253–270. IEEE (May 2015). https://doi.org/10.1109/SP.2015.23, https://ieeexplore.ieee.org/document/7163030/
16. Crites, E., Komlo, C., Maller, M.: How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Paper 2021/1375 (2021), https://eprint.iacr.org/2021/1375, https://eprint.iacr.org/2021/1375
17. Das, S., Camacho, P., Xiang, Z., Nieto, J., Bünz, B., Ren, L.: Threshold Signatures from Inner Product Argument: Succinct, Weighted, and Multi-threshold. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 356–370. CCS '23, Association for Computing Machinery, New York, NY, USA (Nov 2023). https://doi.org/10.1145/3576915.3623096, https://dl.acm.org/doi/10.1145/3576915.3623096
18. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous Identification in Ad Hoc Groups. In: Advances in Cryptology - EUROCRYPT 2004, vol. 3027, pp. 609–626. Berlin, Heidelberg (2004)
19. Ellis, S.: A decentralized oracle network steve ellis, ari juels, and sergey nazarov (2017), https://research.chain.link/whitepaper-v1.pdf
20. Escala, A., Groth, J.: Fine-Tuning Groth-Sahai Proofs. In: Public-Key Cryptography – PKC 2014. pp. 630–649 (2014)
21. ethereum.org: What are daos? (2025), https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/
22. Faonio, A., Fiore, D., Kohlweiss, M., Russo, L., Zajac, M.: From polynomial iop and commitments to non-malleable zksnarks. In: Rothblum, G., Wee, H. (eds.) Theory of Cryptography - TCC 2023 (2023)
23. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Annual Cryptology Conference - Crypto 2015. pp. 233–253. Springer (2015)
24. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology–CRYPTO 2018. pp. 33–62. Springer (2018)
25. Fujisaki, E.: New Constructions of Efficient Simulation-Sound Commitments Using Encryption and Their Applications. In: Topics in Cryptology – CT-RSA 2012. pp. 136–155. Springer (2012). https://doi.org/10.1007/978-3-642-27954-6_9
26. Ganesh, C., Khoshakhlagh, H., Kohlweiss, M., Nitulescu, A., Zajac, M.: What makes fiat–shamir zksnarks (updatable srs) simulation extractable? In: International Conference on Security and Cryptography for Networks - SCN 2022. pp. 735–760 (2022)
27. Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable snarks. In: Eurocrypt 2023. pp. 315–346 (2023)
28. Garay, J.A., MacKenzie, P., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. In: Advances in Cryptology — EUROCRYPT 2003. pp. 177–194 (2003). https://doi.org/10.1007/3-540-39200-9_11
29. Garg, S., Goel, A., Wang, M.: How to prove statements obliviously? In: Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part X. p. 449–487. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-68403-6_14, https://doi.org/10.1007/978-3-031-68403-6_14

30. Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: hints: Threshold signatures with silent setup. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 3034–3052 (2024). https://doi.org/10.1109/SP54263.2024.00057
31. Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024. pp. 352–386. Springer Nature Switzerland, Cham (2024)
32. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: Advances in Cryptology–ASIACRYPT 2006: 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006. Proceedings 12. pp. 444–459. Springer (2006)
33. Herold, G., Hoffmann, M., Klooß, M., Ràfols, C., Rupp, A.: New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1547–1564. CCS '17, Association for Computing Machinery (Oct 2017). https://doi.org/10.1145/3133956.3134068, https://dl.acm.org/doi/10.1145/3133956.3134068
34. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., abhi shelat, Shi, E.: C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Paper 2015/1093 (2015), https://eprint.iacr.org/2015/1093, https://eprint.iacr.org/2015/1093
35. Libert, B.: Vector Commitments with Proofs of Smallness: Short Range Proofs and More. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024, vol. 14602, pp. 36–67. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-57722-2_2, https://link.springer.com/10.1007/978-3-031-57722-2_2, series Title: Lecture Notes in Computer Science
36. Libert, B., Yung, M.: Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In: Theory of Cryptography – TCC 2010. pp. 499–517 (2010)
37. MacKenzie, P., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Advances in Cryptology - EUROCRYPT 2004. pp. 382–400. Springer (2004)
38. Micali, S., Reyzin, L., Vlachos, G., Wahby, R.S., Zeldovich, N.: Compact certificates of collective knowledge. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 626–641 (2021). https://doi.org/10.1109/SP40001.2021.00096
39. Qiu, T., Tang, Q.: Predicate aggregate signatures and applications. In: Asiacrypt - 2023. pp. 279–312. Springer (2023)
40. Ràfols, C., Zapico, A.: An algebraic framework for universal and updatable snarks. In: Crypto 2021. pp. 774–804. Springer (2021)
41. Smith, C.: Proof-of-stake (pos) (Sep 2024), https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/
42. Szepieniec, A., Zhang, Y.: Polynomial iops for linear algebra relations. In: Conference on Public-Key Cryptography - PKC 2022. pp. 523–552. Springer (2022)

A Further Related Work

In the following, we discuss further related concepts. Throughout the paper, we discuss closely related works to our definitions and constructions. In this section, to provide a perspective, we present some prior works that are not crucial, but still indirectly related to our work.

Ad Hoc Key Management. Ad hoc threshold signatures aim to get rid of costly interactive key generation protocols by supporting flexible threshold values per single verification key. We refer to other works with similar aims that support ad hoc signing groups per a single individual secret key here.

Multi-signatures with key aggregation and other threshold signatures with long-term secret keys that we mentioned are fairly new concepts. However, there is an old concept that supports ad hoc groups and group key aggregation, *ad hoc group identification* (AGI) which was proposed by Dodis et al. [18]. AGI provides a key aggregation method for an identification scheme such that the group members can prove their membership to the group using the group public key. Note that, AGI construction of [18] can easily be turned into an *ad hoc group signature* (AGS). This signature scheme only supports *1-out-of-n* structure.

Micali et al. [38] proposes *compact certificates* in which a prover tries to convince a verifier that a group of signers with a sufficient weight have certificated certain data. They set a Merkle tree of signer public keys and weights as a verification key which is a specific form of vector commitment and gives compact verification keys. Then, the prover tries to show that she knows valid signatures for a set of public keys such that she can show Merkle tree openings with sufficient cumulative weight. While this approach can be applied on top of any signature scheme and supports ad hoc thresholds and groups, the signature size is dependent on the group size.

Chaidos et al. [13] designs an efficient *approximate lower bound argument* (ALBA) which can be used to show that the size of a set exceeds a certain threshold. One of the applications they propose is threshold signatures with long-term keys. They also provide a generic construction of such a signature scheme using ALBA and any unique signature scheme. Similar to [29], the resulting signature scheme creates longer signatures. [13] shows the black box nature of their construction as a reason for the longer signatures.

Attema et al. [3] construct threshold signatures with long-term secret keys which is a similar concept to the multiverse ad hoc threshold signatures. They define an environment that signers generate their individual key pairs for a digital signature and share their public keys in a public bulletin board. While this process gets rid of an interactive key generation setup, it results in a group verification key with size $O(n)$ for n signers. Their signing protocol involves a combiner to generate a proof that shows she owns thr valid digital signature out of n public keys in the verification key. They use a *compressed Σ-protocol* for this proof and the resulting signature has $O(\log n)$ size.

Qiu et al. [39] proposes *predicate aggregate signatures* that extends the aggregate signatures to a scheme that shows the signers of messages satisfy a certain policy, e.g. a threshold. Thus, their scheme can be used to build an ad hoc threshold signature scheme. The rely on aggregate BLS signatures and SNARK proof which is computed by a combiner to show that aggregate signature satisfies the policy. Their verification key and signatures for an ad hoc threshold signature application have sizes $O(\log n)$.

Soundness Notions in Proof Systems. Simulation-sound extractability was first defined by Groth [32] which requires a knowledge extractor even when the adversary has access to a zero-knowledge simulator. One immediate result of such a strong notion is that it is not achievable by any malleable proof system. Thus, it is not a good choice for the use cases which exploit the malleability features of proofs, but also need a stronger extractability notion than the regular knowledge soundness.

[34] defines *weak simulation extractability* notion that tolerates randomizable proofs. The main point to achieve such a notion is changing the freshness requirement in the simulation extractability game so that the game will ask for a fresh statement $x \notin Q$ instead of a fresh statement and proof tuple $(x, \pi) \notin Q$. By doing so, we get a relaxed notion of simulation extractability that allows randomizable proof. Weak simulation extractability still does not allow homomorphic proofs, so it is not enough for our use case.

Randomizability is only a type of malleability and there are still other types of malleable proofs that do not satisfy [34]'s weak simulation extractability property. In particular, [34]'s definition only allows to randomize a proof. When we have a proof homomorphism, such as a malleability that allows forging proofs on a statement $x' := T_x(x)$ using a proof of the statement x, there is no hope to satisfy [34] property. Chase et al. define *controlled malleable NIZK* [14] that allows unary transformations on the proofs of the corresponding statements like T_x above. They also define *Controlled-malleable simulation sound extractability* where the extractor either outputs the witness of the statement that the adversary outputs, or a transformation that shows the proof was *transformed* from the simulated proofs. Their definition only considers unary homomorphic operations and generalizing it to arbitrary ℓ-ary homomorphic operations seems non-trivial.

Another way to weaken the simulation extractability property is by limiting the allowed simulation queries. *True-simulation extractability* only guarantees the extractor of a valid witness if all proof simulation queries are made for valid statements.

Faonino et al.'s policy-based simulation extractability definition can cover prior notions such as weak simulation extractability and true-simulation extractability. However, it cannot cover the [14]'s simulation extractability notion as [22]'s definition strictly asks the extractor to output a valid witness, but [14]'s definition allows the extractor to output a transformation which shows that the chosen proof by the adversary was mauled from a simulated proof. Regardless of the simulation extractability, [22]'s winning condition also does not allow partial extractors either.

Homomorphic Proofs. As described in Section 4.2, a homomorphic zero-knowledge proof scheme for a relation R allows obtaining a proof π for a valid statement $x := X(x_1, ..., x_n)$ from the proofs $\pi_1, ..., \pi_n$ where $x_1, ..., x_n$ are valid statements and X is a binary relation over $\mathcal{D}_x^n \to \mathcal{D}_x$. We provide further examples of how prior works defined these mappings for their specific aims.

[2] aims for fully homomorphic proofs and defines X as any possible circuit; [14] defines X as an n-ary transformation from an allowed set of transformations;

[1] only considers a specific type of R by requiring that R must be a relation over a group and defines X as the group operation. Linear-map vector commitment scheme [12] computes proofs of the evaluation of linear functions on vectors. They define X as the linear combinations of these linear functions.

Simulation-Sound Binding Commitments. Trapdoor commitments allow to equivocate/simulate an opening to a commitment for any message using a trapdoor key. Simulation-sound binding trapdoor commitments which were first proposed by [28] are trapdoor commitment schemes that the adversary cannot win the traditional binding game even after seeing the equivocated/simulated commitment openings. While there are several works on simulation-sound trapdoor commitments [37, 25], none of them considers the binding property of the commitment where the commitment scheme shares a simulation trapdoor with a proof simulator and the adversary can see both simulated commitments and the proofs.

B Further Preliminaries

We present the rest of the preliminary information for our paper.

Bilinear Pairings. The formal definition we use for the pairings is presented below. We note that we use type-3 pairings throughout the paper.

Definition 9 (Bilinear Pairing). *For $\langle g\rangle = \mathbb{G}_1$, $\langle \hat{g}\rangle = \mathbb{G}_2$ and $\mathbb{G}_T$ which are groups of prime order p, $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is a bilinear pairing if it is efficiently computable and* bilinear*: $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} = e(g^b, \hat{g}^a)\ \forall a, b \in \mathbb{Z}_p$, and* non-degenerate*: $\langle e(g, \hat{g})\rangle = \mathbb{G}_T$, so $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$. A bilinear group generator* BGGen *is a p.p.t. algorithm which outputs a bilinear pairing description $\mathcal{BG} = (e, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g, \hat{g}, p)$ such that $\lceil \log_2 p \rceil = \lambda$ and the requirements above hold.*

(n_1, n_2)-DLOG Assumption. As we mention (n_1, n_2)-DLOG throughout the paper, we present the formal definition of the assumption below.

Definition 10 ((n_1, n_2)-DLOG Assumption [6]). *For all $\lambda \in \mathbb{N}$, p.p.t. adversaries $\mathcal{A}$, and for a bilinear group $\mathcal{BG} := (e, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g, \hat{g}, p) \leftarrow \mathsf{BGGen}(\lambda)$,*

$$\Pr\left[\begin{array}{c} pp_{\mathbb{G}} := (e, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g, \hat{g}, p) \leftarrow \mathsf{BGGen}(1^\lambda),\ \tau \leftarrow \mathbb{Z}_p \\ (\tau^*) \leftarrow \mathcal{A}(pp_{\mathbb{G}}, (g^{\tau^i})_{i\in[n_1]}, (\hat{g}^{\tau^i})_{i\in[n_2]}) \end{array} : \tau^* = \tau \right] \leq \mathsf{negl}_{DLOG}(\lambda)$$

Deferred Definitions of Multi-Signatures. We provide the full description of multi-signatures and their properties.

Definition 11 (MS with PoP). *A multi-signature scheme* MS *is a tuple of algorithms* $(\mathsf{Pg}, \mathsf{Kg}, \mathsf{KeyVf}, \mathsf{KAg}, \mathsf{MulSign}, \mathsf{Combine}, \mathsf{Vf})$ *such that:*

$\mathsf{Pg}(1^\lambda) \to pp_{\mathsf{MS}}$**:** Outputs public parameters pp_{MS} for security parameter 1^λ. We only make pp_{MS} explicit in key generation and assume it to be an implicit input to all other algorithms.

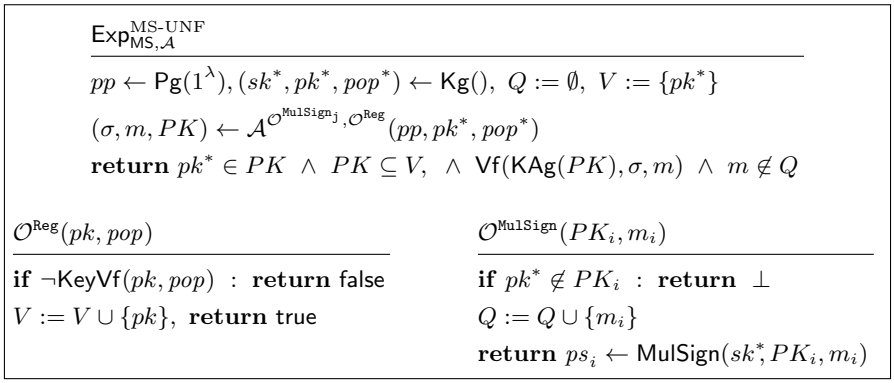

Fig. 6. Unforgeability for MS schemes with key aggregation.

$\mathsf{Kg}(pp) \to (sk, pk, pop)$**:** Probabilistic key generation, outputs key pair (sk, pk) along with a proof pop.

$\mathsf{KeyVf}(pk, pop) \to b \in \{\mathsf{true}, \mathsf{false}\}$**:** Verifies the proof of possession pop for the public key pk.

$\mathsf{KAg}(PK) \to apk$**:** Deterministic key aggregation, that on input a set of public keys $PK = \{pk_i\}$, outputs an aggregated public key apk.

$\mathsf{MulSign}(sk_i, PK, m) \to ps_i$**:** (Possibly interactive) algorithm, that on input the secret key sk_i, message m, and *optionally* a set of public keys $PK = \{pk_i\}$, outputs a partial signature s.

$\mathsf{Combine}(PK, \{ps_i\}_{pk_i \in PK}) \to \sigma$**:** On input a set of public keys $PK = \{pk_i\}$ and set of shares $\{ps_i\}_{pk_i \in PK}$ outputs a combined signature σ for PK.

$\mathsf{Vf}(apk, \sigma, m) \to b \in \{\mathsf{true}, \mathsf{false}\}$**:** Verifies if σ is a valid signature on m for apk.

The correctness of a MS scheme is defined as follows.

Definition 12 (MS-Correctness). *A multi-signature scheme* MS *is correct if for all λ, for all $pp \leftarrow \mathsf{Pg}(1^\lambda)$, for all messages m, for all n, for all $(sk_i, pk_i) \leftarrow \mathsf{Kg}()$ for $i \in [n]$, for all $ps_i \leftarrow \mathsf{MulSign}(sk_i, \{pk_i\}_{i\in[n]}, m)$,*

$$\mathsf{Vf}(\mathsf{KAg}(\{pk_i\}_{i\in[n]}), \mathsf{Combine}(\{pk_i\}_{i\in[n]}, \{ps_i\}_{i\in[n]}), m) = \mathsf{true}$$

The unforgeability definition is presented below.

Definition 13 (MS Unforgeability). *A multi-signature scheme Π is unforgeable if for all PPT adversaries $\mathcal{A}$ $\Pr[\mathsf{Exp}^{MS\text{-}UNF}_{\Pi,\mathcal{A}}(1^\lambda) = \mathsf{true}] \leq \mathsf{negl}_{MS\text{-}UNF}(\lambda)$ for the experiment from Figure 6.*

Correctness of Type-based Commitments. The deferred correctness definition of type-based commitments is given below.

Definition 14 (Correctness of CS). *A type-based commitment scheme* CS *is correct if for all $\lambda \in \mathbb{N}$, for $n \in poly(\lambda)$, for all $ck \leftarrow \mathsf{Setup}(\mathsf{ParGen}(1^\lambda), n)$, $t \in \mathcal{T}$, $m \in \mathcal{M}_{ck,t}$, and $o \in \mathcal{O}_{ck,t}$, $\mathsf{VfCom}_{ck}(t, \mathsf{Com}_{ck}(t, m; o), o) = \mathsf{true}$.*

Polynomial Relations. We recap two polynomial relations to encode hadamard-product and inner-product relations of vectors which were presented in [40].

Lemma 1 (Hadamard-Product Polynomial Relation). *Let $n \in \mathbb{N}$, and $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{Z}_p^n$ with order n. Let further $\mathbf{u}(X) := \big(\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)\big)$, $\mathbf{v}(X) := \big(\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)\big)$, and $\mathbf{y}(X) := \big(\sum_{i=1}^{n} y_i \cdot \mathcal{L}_i(X)\big)$ for given vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$. Then, $\mathbf{u} \circ \mathbf{v} = \mathbf{y}$ if and only if there exists $Q(X) \in \mathbb{Z}_p[X]$ satisfying Equation 3,*

$$\mathbf{u}(X) \cdot \mathbf{v}(X) - \mathbf{y}(X) = z_{\mathbb{H}}(X) \cdot Q(X) \tag{3}$$

Lemma 2 (Inner-Product Polynomial Relation). *Let $n \in \mathbb{N}$, and $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{Z}_p^n$ with order n. Then, given vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$, $\mathbf{u} \cdot \mathbf{v} = \mu$ if and only if there exists $Q(X), R(X) \in \mathbb{Z}_p[X]$ satisfying Equation 4,*

$$\mathbf{u}(X) \cdot \mathbf{v}(X) - \mu/n = X \cdot R(X) + z_{\mathbb{H}}(X) \cdot Q(X) \tag{4}$$

where $\mathbf{u}(X) := \big(\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)\big)$, $\mathbf{v}(X) := \big(\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)\big)$, and $R(X)$ has degree at most $n-2$. Note that for $\mathbf{y}(X) := \big(\sum_{i=1}^{n} u_i \cdot v_i \cdot \mathcal{L}_i(X)\big)$, $R(X)$ and $Q(X)$ can be computed using the equations

$$R(X) = (\mathbf{y}(X) - \frac{\mathbf{u} \cdot \mathbf{v}}{n}) \cdot X^{-1} \quad Q(X) = (\mathbf{u}(X) \cdot \mathbf{v}(X) - \mathbf{y}(X)) \cdot z_{\mathbb{H}}^{-1}(X) \tag{5}$$

Lemma 3. *For a multiplicative subgroup $\mathbb{H}$ with order n, for all $i \in [n]$ and all $j \in [n]$ such that $j \neq i$,*

$$\mathcal{L}_i(X) \cdot \mathcal{L}_j(X) = 0 \mod z_{\mathbb{H}}(X) \tag{6}$$

$$\mathcal{L}_i^2(X) = \mathcal{L}_i(X) \mod z_{\mathbb{H}}(X) \tag{7}$$

$$\mathcal{L}_i(0) = 1/n \tag{8}$$

C Deferred Definitions of CP-SNARK

For space reasons, we present the definitions of some properties of CP-SNARKs here. These definitions mainly follow the existing works.

Succintness. We present the regular definition succinctness for a CP-SNARK for completeness.

Definition 15 (Succintness). *Let CP be a CP-SNARK on the family of relations $\mathcal{R}_{ck}^{\mathsf{CS}}$ for $ck \leftarrow \mathsf{Setup}(\mathsf{ParGen}(1^\lambda), n)$ and $n \in poly(\lambda)$. CP is succinct if for all $R_{ck}^{\mathsf{CS}} \in \mathcal{R}_{ck}^{\mathsf{CS}}$, $(\hat{x}, \hat{w}) \in R_{ck}^{\mathsf{CS}}$, and $\pi \leftarrow \mathsf{Prove}(R_{ck}^{\mathsf{CS}}, \hat{x}, \hat{w})$ the running time of $\mathsf{Vf}(R^{\mathsf{CS}}, \hat{x}, \pi)$ is $poly(\lambda + |\hat{x}| + \log|\hat{w}|)$ and the proof size is $poly(\lambda + \log|\hat{w}|)$.*

Completeness. Our completeness notion is similar to the *type-based completeness* notion of [10]. Definition 2 assigns commitment types to the commitment slots in the simple relation R arbitrarily in the sense that any commitment type with the appropriate message domain could be used to commit to the witness part in the commitment slot. However, the commitment scheme can have multiple commitment types with the same message space. In this case, we have to show that the scheme is complete for all possible commitment type-commitment slot matches when we want to show the completeness of a proof scheme for such a relation. This process would be impractical and unnecessary. Campanelli et al. overcome this issue by defining type-restricted completeness, where the completeness is shown for a specific commitment type-commitment slot match. We further relax this notion by also allowing to match a commitment type to a commitment slot for a relation $R \in \mathcal{R}_{pp,n}$. Showing that a CP-SNARK is type-relation-restricted complete for the same type tuple T and for all relations $R \in \mathcal{R}_{pp,n}$ means the CP-SNARK is type-restricted complete for [10]'s definition.

Definition 16 (Type-Relation-Restricted Completeness). *Let $\{\mathcal{R}_{pp,n}\}_{pp\in\mathsf{ParGen}(1^\lambda),n\in\mathbb{N}}$ be a family of universal relations over relations $R_{pp,n}$ on the space $\mathcal{D}_x \times \mathcal{D}_m \times \mathcal{D}_\omega$ such that $\mathcal{D}_m$ splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let $\mathsf{CS} := (\mathsf{Setup}, \mathsf{Com}, \mathsf{VfCom})$ be a type-based commitment scheme as in Definition 1 such that for $T \in \mathcal{T}^\ell$ and $T := (t_1, \ldots, t_\ell)$, CS has a type t_i where $\mathcal{D}_i \subset \mathcal{M}_{ck,t_i}$. A CP-SNARK CP is T-$R_{pp,n}$-restricted complete if for $R_{pp,n} \in \{\mathcal{R}_{pp,n}\}_{pp\in\mathsf{ParGen}(1^\lambda),n\in\mathbb{N}}$, $ck \in \mathsf{Setup}(pp, n)$, and $\big((x, (c_j)_{j\in[\ell]}), \hat{w}\big) \in R_{ck}^{\mathsf{CS}}$ such that c_j's type is t_j for $j \in [\ell]$, it holds:*

$$\Pr\big[\, \pi \leftarrow \mathsf{Prove}(R_{ck}^{\mathsf{CS}}, \hat{x}, \hat{w}) \;:\; \mathsf{Vf}(R_{ck}^{\mathsf{CS}}, \hat{x}, \hat{w}) \,\big] = 1$$

C.1 Homomorphic Commit-and-Prove Systems

Finally, we define homomorphic properties which will be necessary when constructing our signature scheme. We first define the commitment homomorphism which allows to create a commitment on a message m which is a function of n messages using only the commitments to the n messages. Similarly, we define a proof homomorphism that computes a proof for a function of n statements, using only their corresponding individual proofs.

Commitment Homomorphism. We define commitment homomorphism through an algorithm $c \leftarrow \mathsf{EvalCom}(t, F, c_1, ..., c_n)$ where F is the function to evaluate on the messages and/or openings of the commitments $c_1, ..., c_n$ with the commitment type t. Formally, $F(m_1, o_1, ...m_n, o_n)$ takes both messages and randomness part and output some (m, o) such that $\mathsf{VfCom}_{ck}(t, c, m, o)$. For simplicity, we will sometimes exploit the notation and write $m \leftarrow F(m_1, o_1, ...m_n, o_n)$ or $o \leftarrow F(m_1, o_1, ...m_n, o_n)$.

Definition 17 (Type-Commitment Homomorphism). *Π is type-commitment homomorphic for the commitment type t and family of functions $\mathcal{F}_{ck,t}$ if*

there exists an algorithm $\mathsf{EvalCom}$ *such that for any* $F \in \mathcal{F}_{ck,t}$*, and commitments* $c_i := \mathsf{Com}_{ck}(t, m_i; o_i)$ *for* $i := 1, \ldots, n$*, and for* $(m, o) := F(m_1, o_1, \ldots, m_n, o_n)$*,*

$$\mathsf{VfCom}_{ck}(t, \mathsf{EvalCom}_{ck}(F, t, c_1, ..., c_n), m, o) = \mathsf{true}$$

Proof Homomorphism. A homomorphic zero-knowledge proof scheme for a relation R allows obtaining a proof π for a valid statement $x := X(x_1, ..., x_n)$ from the proofs $\pi_1, ..., \pi_n$ where $x_1, ..., x_n$ are valid statements and X is a binary relation over $\mathcal{D}_x^n \to \mathcal{D}_x$. In terms of allowed homomorphism relations X's, we define it using an allowed set of n-ary partial operations. An n-ary partial operation does not have to be defined for all $(x_1, ..., x_n) \in \mathcal{D}_x^n$. This flexibility will also be used in our generic construction, where we need to define homomorphic proofs on the statements for $\mathsf{IP}_{\mathbb{G}}$ proofs on the commitment tuples (U, V_1) and (U, V_2), but not for the ones with distinct U commitments. A detailed comparison of this definition to the prior works is in Appendix A.

Definition 18 (Relation-Proof Homomorphism). *Let* CP *be a CP-SNARK on the family of relations* $\mathcal{R}^{\mathsf{CS}}$ *and* $R^{\mathsf{CS}} \in \mathcal{R}^{\mathsf{CS}}$ *is a relation over pairs* $(\hat{x}, \hat{w})$ *as in Definition 2. Let* $\mathcal{X}$ *be the set of* n*-ary operations* $X : \mathcal{D}_{\hat{x}}^n \to \mathcal{D}_{\hat{x}}$*.* CP *is homomorphic on the relation* R^{CS} *with respect to* $\mathcal{X}$ *if there exists an algorithm* $\mathsf{EvalProof}$ *such that, for all* $X \in \mathcal{X}$*,* $\hat{x}_1, \pi_1, ..., \hat{x}_n, \pi_n$ *such that* $\mathsf{Vf}(R^{\mathsf{CS}}, \hat{x}_i, \pi_i)$ *and* X *is defined for* $(\hat{x}_1, ..., \hat{x}_n)$*,*

$$\mathsf{Vf}(R^{\mathsf{CS}}, X(\hat{x}_1, ..., \hat{x}_n), \mathsf{EvalProof}(R^{\mathsf{CS}}, X, \pi_1, ..., \pi_n)) = \mathsf{true}$$

D Deferred Definitions of Generic MATS Construction - Section 5

We present the additional definitions that are related to Section 5.

D.1 Correctness of MATS

We first evaluate the correctness definitions of previous works. Although they follow a general pattern, they differ on whether they cover the correctness of the scheme under certain malicious behavior. [4, 30] specifically relies on BLS multi-signatures in their construction, which allows them to define a partial signature verification algorithm PVf and define PSign without defining the signer set PK as an input. They can define and guarantee the correctness with the malicious parties in the following way. They allow the adversary to interact with oracles as in the unforgeability game and the adversary must output a target signing group, corresponding partial keys, just as in the unforgeability game. The difference from the unforgeability game is that the adversary must output partial signatures for some signer subset S instead of a final signature. The challenger can verify the partial signatures one by one using PVf and compute the threshold that the valid partial signatures must satisfy. Then the challenger runs the signature combining algorithm and verifies the resulting signature against the threshold it computed

before. The signature scheme is said to be correct if the verification fails only with a negligible probability. Obviously, this definition requires supporting the PVf algorithm.

At the construction level, if the partial signatures of the underlying multi-signature scheme do not depend on the signer group PK, such as BLS multi-signatures, then the combiner can run the partial verification on the input multi-signatures and exclude the invalid ones from combining process. This is not possible when the partial signatures of the multi-signature scheme depend on the signer group PK, such as Schnorr, as it is not possible to change the intended signing group for the message in the combining process. All in all, by sacrificing some generic properties of the syntax and the construction, we could achieve malicious correctness. We choose to keep the syntax and the construction generic and define the basic correctness notion of the MATS scheme as follows.

Definition 19 (Correctness of MATS). *A multiverse ad hoc threshold signature scheme is correct if for all λ and n, for $pp_{\mathsf{MATS}} \leftarrow \mathsf{Setup}(1^\lambda, n)$, for all $\ell \leq n$, weight vectors with size ℓ $\mathbf{w}$, $((sk_i, pk_i) \leftarrow \mathsf{Kg}())_{i\in\ell}$, for*

$$(vk, ak) \leftarrow \mathsf{UGen}_2(\{\mathsf{UGen}_1(sk_i, \mathbf{pk})\}_{i\in[\ell]}, \mathbf{pk}, \mathbf{w})$$

for all thr and $S \subseteq [\ell]$ s.t. $\mathsf{thr} \leq \sum_{i\in S} w_i$, and for all m, let $ps_i \leftarrow \mathsf{PSign}(sk_i, S, m)$ for $i \in S$. Then,

$$\mathsf{Vf}(vk, \mathsf{Combine}(\{ps_i\}_{i\in S}, ak), m, \mathsf{thr}) = \mathsf{true}$$

D.2 MATS Compatibility

To make the generic construction more readable, we define the requirements on the underlying multi-signature and CP-SNARK separately. We note that we only cover requirements that are necessary for the correctness of the generic construction. Below we define the requirements for the multi-signature. It mainly restricts the key spaces and the type of multi-signature key aggregation methods we support.

Definition 20 (Compiling Multi-Signature). *A multi-signature scheme MS is a compiling multi-signature if for $pp_{\mathsf{MS}} \leftarrow \mathsf{Pg}(1^\lambda)$ it has the secret key space $\mathbb{Z}_p$ and public key space $\mathbb{G}$, where $pk = g^{sk}$ for a key pair (sk, pk) and $\mathsf{MS.KAg}(PK) := \prod_{pk\in PK} pk$ where $\langle g\rangle = \mathbb{G}$ is a group with prime order p.*

We list the required properties from the CP-SNARK scheme for the generic construction. For the CP-SNARK, we define the relations $R_{pp_{\mathsf{MS}},n}$'s as follows.

$$\begin{aligned}
\mathsf{IP}_{\mathbb{G}} &:= \{(\gamma, (\mathbf{u}, \mathbf{v})) \ : \ (\gamma, (\mathbf{u}, \mathbf{v})) \in \mathbb{G} \times (\mathbb{Z}_p^n, \mathbb{Z}_p^n) \ \wedge \ \gamma = g^{\mathbf{u}\cdot\mathbf{v}}\} \\
\mathsf{IP}_{\mathbb{Z}_p} &:= \{(\gamma, (\mathbf{u}, \mathbf{v})) \ : \ (\gamma, (\mathbf{u}, \mathbf{v})) \in \mathbb{Z}_p \times (\mathbb{Z}_p^n, \mathbb{Z}_p^n) \ \wedge \ \gamma = \mathbf{u}\cdot\mathbf{v}\} \\
\mathsf{BIT} &:= \{(\bot, \mathbf{u}) \ : \ \mathbf{u} \in \mathbb{Z}_p^n \ \wedge \ \mathbf{u} \in \mathbb{Z}_2^n\}
\end{aligned}$$

where a prime order group $\langle g\rangle := \mathbb{G}$ is defined in pp_{MS}.

Definition 21 (Compiling CP-SNARK). *A CP-SNARK* CP *is a compiling CP-SNARK if for a given size bound* n *and public parameters* pp_{MS} *that defines the group* $(g, \mathbb{G}, p)$ *and* $\mathsf{ParGen} := \mathsf{MS.Pg}$ *:*

1. CP has the commitment types $t_{\mathbb{G}}$, t_1, and t_2 with the message space $\mathbb{Z}_p^n$ such that CP is $(t_{\mathbb{G}}, t_2)$-complete for the relation $\mathsf{IP}_{\mathbb{G}}$, (t_1, t_2)-complete for the relation $\mathsf{IP}_{\mathbb{Z}_p}$, and (t_1)-complete for the relation BIT.
2. Type $t_{\mathbb{G}}$ commitments are homomorphic on $\mathbb{Z}_p^n$ for the family of functions $\mathcal{F}_{t_{\mathbb{G}}}$ where $F_{t_{\mathbb{G}},k} \in \mathcal{F}_{t_{\mathbb{G}}}$ if $F_{t_{\mathbb{G}},k}(\mathbf{u}_1, ..., \mathbf{u}_k) := \sum_{i\in[k]} \mathbf{u}_i$ for $k \leq n$.
3. Proofs for the relation $\mathsf{IP}_{\mathbb{G}}$ are homomorphic with respect to the set of operations $\mathcal{X}_{t_{\mathbb{G}}}$ where $X_{t_{\mathbb{G}},k} \in \mathcal{X}_{t_{\mathbb{G}}}$ if $X_{t_{\mathbb{G}},k}(\hat{x}_1, ..., \hat{x}_k) := \hat{x}$ is defined over $\hat{x}_i = (\gamma_i, c_{\mathbf{u},i}, c_{\mathbf{v}})$ for $i \in [k]$ and
$$\hat{x} := \big(\prod_{i\in[k]} \gamma_i, (t_{\mathbb{G}}, F_{t_{\mathbb{G}},k}(c_{\mathbf{u},1}, ..., c_{\mathbf{u},k})), (t_2, c_{\mathbf{v}}) \big)$$
4. Type t_2 commitments are homomorphic on $\mathbb{Z}_p^n$ for the family of functions $\mathcal{F}_{t_2}$ where $F_{t_2,k} \in \mathcal{F}_{t_2}$ if $F_{t_2,k}(\mathbf{u}_1, ..., \mathbf{u}_k) := \sum_{i\in[k]} \mathbf{u}_i$ for $k \leq n$.
5. Proofs for the relation $\mathsf{IP}_{\mathbb{G}}$ are homomorphic with respect to the set of operations $\mathcal{X}_{t_2}$ where $X_{t_2,k} \in \mathcal{X}_{t_2}$ if $X_{t_{\mathbb{G}},k}(\hat{x}_1, ..., \hat{x}_k) := \hat{x}$ is defined over $\hat{x}_i = \big(\gamma_i, (t_{\mathbb{G}}, c_{\mathbf{u}}), (t_2, c_{\mathbf{v},i})\big)$ for $i \in [k]$ and
$$\hat{x} := \big(\prod_{i\in[k]} \gamma_i, (t_{\mathbb{G}}, c_{\mathbf{u}}), (t_2, F_{t_2,k}(c_{\mathbf{v},1}, ..., c_{\mathbf{v},k})) \big)$$

The first item simply requires that the CP-SNARK contains commitments with appropriate message spaces to commit to the vectors $\mathbf{sk}$, $\mathbf{b}$, and $\mathbf{w}$ such that it can compute the necessary proofs on these commitments. The second and third requirements define the necessary homomorphic properties that we will rely on to create the final $\mathsf{IP}_{\mathbb{G}}$ proof in the $\mathsf{Combine}$ algorithm. In more detail, the second requirement provides the commitment homomorphism on type t_2 commitments for the vector addition. The third requirement provides the proof homomorphism for proofs that contain the same $t_{\mathbb{G}}$ commitment, but different t_2 commitments so that we can compute the proof corresponding to the inner product of the same $t_{\mathbb{G}}$ commitment and the sum of t_2 commitments. Similarly, the fourth and fifth requirements define the homomorphic properties to compute vk and ak. In the key generation protocol, each signer will create a commitment to its own secret key under the correct position in the vector. Then, relying on the homomorphic properties required by item 4, we can compute a commitment to the secret key vector of all signers $\mathbf{sk}$. The proof homomorphism in item 5 will be used to combine the proofs of the individual signers on their individual commitment keys, and we will get the proofs on the vector we homomorphically computed. These will become a part of the aggregation key ak.

E Security Proof of MATS Unforgeability - Theorem 2

Here we give detailed proof that our MATS construction from Figure 3 is an unforgeable multiverse ad hoc threshold signature scheme according to Definition 7, if MS is an unforgeable multi-signature scheme, CP is f-zero-knowledge,

f-Φ_h-simulation extractable, and f-Φ_h-simulation-sound binding for for the function f and the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ from Figure 4 for all $h \in \mathbb{G}$.

Proof. Recall that an adversary breaking the unforgeability of the MATS scheme must come up with a non-trivial forgery $(m^*, \sigma^*, \mathsf{thr}^*, \mathbf{pk}, \{prk_i\}_{pk_i \in \mathbf{pk} \setminus H}, \mathbf{w})$ where $\sigma^* := (\sigma', apk, B, \pi_{\mathsf{Agg}}, \pi_{\mathsf{thr}}, \pi_{\mathsf{BIT}}, \mathsf{thr}')$. Soundness ensures that the proofs $\pi_{\mathsf{Agg}}, \pi_{\mathsf{thr}}, \pi_{\mathsf{BIT}}$ in the forgery guarantee that $\mathbf{w} \cdot \mathbf{b}^* = \mathsf{thr}' \ \wedge \ \mathbf{b}^* \in \mathbb{Z}_2^n \ \wedge \ \mathbf{pk}^{\mathbf{b}^*} = apk$ holds, which in turn allows to use σ' for the multi-signature scheme.

Given an adversary $\mathcal{A}_{\mathsf{MATS}}$ against our MATS scheme, we construct an adversary $\mathcal{A}_{\mathsf{MS}}$ against the underlying multi-signature as follows.

A MS forger $\mathcal{A}_{\mathsf{MS}}$ receives pk^*, pop^* and uses these values for a randomly chosen honest signer in the MATS context. $\mathcal{A}_{\mathsf{MS}}$ has oracle access to sk^* and must output a forgery (σ', m^*, PK^*). σ' is already part of our MATS signatures, which will also contain the message m^*. Thus, we only need to compute an appropriate set of public keys PK^* out of the MATS forgery that will create a MS forgery. We achieve that by extracting the bit vector $\mathbf{b}^*$ that represents the signers from the forged MATS signature.

Game_1 (*Simulating Honest Signer*): We need to simulate an honest signer in MATS view by only knowing the public key of an honest MS signer. An honest MATS signer's secret key is used for creating commitment and proofs during the key generation and answering the signing oracle queries. We first change the universe generation process for a randomly chosen honest signer by simulating the commitment and proofs in UGen_1. In the final step of the proof, we are going to simulate the signing queries using MS unforgeability signing oracle, too.

This game aims to simulate the key generation process without using one of the secret keys. We are going to achieve this using the f-zero-knowledge property. We first change how we set public parameters. We run CP.SSetup instead of CP.Setup and store the resulting internal state st_{S}. Due to the f-zero-knowledge property, this change is indistinguishable. We also change how we set the challenge public key and the aggregation key. In particular, let q_{UGen_1} be number of $\mathcal{O}^{\mathtt{UGen}_1}$ queries that the adversary makes. For a random signer index $u \in [q_{\mathsf{UGen}_1}]$, during the partial key generation processes, instead of running, $U_i := \mathsf{Com}_{ck}(t_{\mathbb{G}}, \mathbf{sk}_i, o_i)$ we run $(U_i, o_i, st_{\mathsf{S}}) \leftarrow \mathsf{SCom}(st_{\mathsf{S}}, t_{\mathbb{G}}, \mathbf{pk}_i)$. Furthermore, instead of running the Prove algorithm during UGen_1 for $\mathbf{Y} := pk_i^{\mathbf{e}_i}$ and $j \in [n]$, we run

$$(\pi_{i,j}, st_{\mathsf{S}}) \leftarrow \mathsf{SProve}\Big(st_{\mathsf{S}}, \mathsf{IP}_{\mathbb{G}}, \big(Y_j, (t_{\mathbb{G}}, U_i), (t_2, c_{\mathbf{e}_j})\big), \big((\mathbf{pk}_i, o_i), (\mathbf{e}_j, o_{\mathbf{e}_j})\big)\Big)$$

Note that the index i of our honest signer differs according to the signing group and all simulation queries conform to the policy Φ_h. Furthermore, by f-zero-knowledge this change is indistinguishable. $|\Pr[W_1] - \Pr[W_0]| \leq \mathsf{negl}_{f\text{-}\mathrm{zk}}(\lambda)$.

Game_2 (*Extracting the Bit Vector*): Now we rely on the simulation extractability to extract $\mathbf{b}^*$ so that we can set a valid public key set PK against the MS unforgeability challenger.

We extract the signer vector $\mathbf{b}^*$ from π_{Agg}, π_{thr}, and π_{BIT} for the forged signature. If the extractor fails for either of the proofs or the extracted values are not identical, we abort. By relying on the f-Φ-SE property, we can extract $\mathbf{b}^*$ values. Note that we can rely on f-Φ-SE property as all simulation queries we made in Game_1 conform to the Φ_{sim} for $h = pk_i$. Furthermore, when we set up a verification key successfully, we know the valid auxiliary information to be provided to the extractor that Φ_{ext} requires. For $\mathsf{IP}_{\mathbb{Z}_p}$ proof, the weight vector $\mathbf{w}$ is the valid opening for t_1 commitment. For $\mathsf{IP}_{\mathbb{G}}$ proof, the proofs in the aggregation key ak are the valid auxiliary information that the extractor needs.

The extracted values must be identical by Φ-special-sound binding property of the commitment scheme as all π_{Agg}, π_{thr}, and π_{BIT} use the same commitment value for their $\mathbf{b}^*$ values and our simulation queries conform to the policy. Thus, the abort case only occurs with the negligible probability, $|\Pr[W_2] - \Pr[W_1]| \leq 3 \cdot \mathsf{negl}_{\mathrm{SE}}(\lambda) + \mathsf{negl}_{\mathrm{SBND}}(\lambda)$.

Game_3 (*Reduction From Multi-Signature Unforgeability*): Now we are ready to make our final argument against the MS unforgeability. We can simulate the MATS adversary's using the MS challenger as an honest signer, and we can extract the set of signers PK^* that MATS forgery is performed on. The last argument we must make is about a trivial check to build a successful MS forger: the MS challenge public key must be a member of PK^*. Game_3 satisfies this requirement with a polynomial loss and we conclude our proof.

In this game, we add an abort condition to guarantee that the signer index we guessed in Game_1 can be used to embed the multi-signature challenge public key. We require that

1. $pk^{(u)}$ must be in $\mathbf{pk}^*$ for the honest signer (u) that we choose in Game_1.
2. There must be no signing query for this signer on message m^*, $(pk^{(u)}, m^*) \notin Q$.
3. Let i be the index of $pk^{(u)}$ in $\mathbf{pk}$. Then the i'th bit of $\mathbf{b}^*$, $b_i^* = 1$.

If any of these conditions do not hold, we abort. By winning condition, we know that there has to be an honest signer who did not sign the message, but the corresponding position in $\mathbf{b}^*$ is equal to 1. As (u) was sampled randomly, $\Pr[W_3] = \Pr[W_2]/q_{\mathsf{UGen}_1}$.

Finally, we build a MS forger. We get a challenge MS public key pk^* and set $pk^{(u)} := pk^*$. In Game_3, we use the secret key $sk^{(u)}$ only for answering signing oracle queries and we overcome this issue by simulating $\mathcal{O}^{\mathtt{Sign}}$ queries using the $\mathcal{O}^{\mathtt{MulSign}}$ oracle of the MS unforgeability game. This change is perfectly indistinguishable from Game_3. Finally, the adversary outputs the forgery $\sigma^* := (\sigma', apk, B, \pi_{\mathsf{Agg}}, \pi_{\mathsf{thr}}, \pi_{\mathsf{BIT}})$. By Game_2 we know $\mathbf{b}^*$ and we can build the signer set PK^* using $\mathbf{b}^*$. We already know valid proof-of-possesion for all signers in $\mathbf{pk}^*$ by UGen_2, so we know pop_i for all indexes in PK^*. We register all public keys in PK^* to the MS forgery game using these proof-of-possesion values and we output (σ', m^*, PK^*) as a MS forgery. By Game_3, we know that MS unforgeability challenge public key $pk^* \in PK^*$ and we do not make a signing

query for m^* to the MS challenger. Also, by the winning condition of MTS unforgeability game, we know that $\mathsf{MS.Vf}(apk, \sigma', m^*)$, so (σ', m^*, PK^*) is a valid multi-signature forgery. Thus, $\Pr[W_3] \leq \mathsf{negl}_{\mathsf{MS\text{-}UNF}}(\lambda)$. In the end, we have

$$\Pr\left[\mathsf{Exp}^{\mathsf{MATS\text{-}UNF}}_{\Pi,\mathcal{A}}(\lambda)\right] \leq \mathsf{negl}_{f\text{-}\mathrm{zk}}(\lambda)+3\cdot\mathsf{negl}_{\mathrm{SE}}(\lambda)+\mathsf{negl}_{\mathrm{SBND}}(\lambda)+q_{\mathsf{UGen}_1}\cdot\mathsf{negl}_{\mathsf{MS\text{-}UNF}}(\lambda)$$

F Deferred Content of CP-SNARK Instantiation

We present the deferred content from Section 6 here.

F.1 Comparison of CP-Pair to the Das et al.'s Construction [17].

We compare CP-Pair scheme in Figure 5 to the proof-scheme of [17].

The first change is that we explicitly define the scheme on type-3 pairings while [17] defined their scheme on symmetric pairings. The second difference is on the crs. [17] adds additional parameters to the crs which are basically powers-of-τ in different bases. In addition to the powers on the generator g, they also create powers-of-tau on a generator h. Furthermore, they add another generator v to the crs. They use powers on h to perform the degree check by computing $h^{\tau \cdot R(\tau)}$ instead of $g^{\tau \cdot R(\tau)}$ for the relations $\mathsf{IP}_{\mathbb{G}}$ and $\mathsf{IP}_{\mathbb{Z}_p}$. Similarly, they use the generator v to perform the degree check on γ value for the relation $\mathsf{IP}_{\mathbb{G}}$. We perform all degree checks using powers-of-tau on g and $\hat{g}$ which gets rid of the extra values in the crs.

An additional check we perform is the degree checks on the type t_2 commitments for all relations. This check simply assures that the type t_2 commitments have a degree $\leq n-1$, so that we can extract valid openings to these commitments in the extractability proof. Note that we perform this degree check in a different way than we do for g_R's or γ. This is mainly because efficiency. The current degree check on type t_2 commitments also serves as a proof of equality between the commitments in groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ which becomes useful in the BIT relation proof. As we are in type-3 pairings, this check is necessary to build a commitment to the vector $\mathbf{1} - \mathbf{u}$ in group $\mathbb{G}$. In a concrete instantiation of $\mathsf{ATS}/\mathsf{MATS}$ with CP-Pair, the pairing equations in the dashed boxes become the identical equation which means performing it once is enough. We also note that this check would become necessary for an implementation of [17] on a type-3 pairing.

The last difference is that [17] performs an optimization on proof verification for their ATS by batching the verification of $\mathsf{IP}_{\mathbb{G}}$ and $\mathsf{IP}_{\mathbb{Z}_p}$ proofs. Such a batching technique could be added to our construction by relying on the homomorphic properties of the proofs as [12] did for the vector commitments or by using the generic techniques for the batched verification of pairing product equations [33].

F.2 Efficiency of the Instantiation

We present an efficiency comparison of the instantiation of our generic construction in Figure 3 with the CP-SNARK instantiation from Section 5 and BLS

multi-signatures with PoP [7]. We choose BLS multi-signatures as that is also used by the previous works [17, 30, 4]. Note that the signature schemes support different functionalities, in particular, [4] does not support flexible thresholds. The additional burden in the performance metrics of all other works, including ours, is the result of SNARKs in the signature to support ad hoc thresholds.

Our concrete instantiation has comparable efficiency to the previous works. Below we present two main efficiency metrics in Figure F.2, the signature size and verification cost. Both [17, 30] employ several optimizations on the underlying proofs which can also be applied to our scheme (inner-product proof aggregation [12] and batched pairing equation verification [33]). Note that [17]'s construction uses symmetric pairings, but their implementation uses type-3 pairings. We provide the numbers for both versions.

	Signature Size	Verification	ad hoc thresh.	ad hoc groups
[17] Asymmetric	$7\mathbb{G} + 2\hat{\mathbb{G}} + 1\mathbb{Z}_p$	$15P + 18\mathbb{G}$	✓	✗
[17] Symmetric	$8\mathbb{G} + 1\mathbb{Z}_p$	$13P + 2\mathbb{G}$	✓	✗
[30]	$9\mathbb{G} + 5\mathbb{Z}_p$	$10P + 1\mathbb{G}$	✓	✗
[4]	$2\mathbb{G} + 1\hat{\mathbb{G}}$	$4P$	✗	✓
Our Plain	$7\mathbb{G} + 5\hat{\mathbb{G}} + 1\mathbb{Z}_p$	$20P + 2\mathbb{G}$	✓	✓
Our Optimized	$6\mathbb{G} + 3\hat{\mathbb{G}} + 1\mathbb{Z}_p$	$13P + 4\mathbb{G}$	✓	✓

Fig. 7. The efficiency comparison. For the computational performance of the verification, $\hat{\mathbb{G}}$, and P correspond to the number of exponentiations in group $\mathbb{G}$, exponentiations in group $\hat{\mathbb{G}}$ and pairing operations, respectively. For $\|\text{sig.}\|$, $\mathbb{G}$, $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{Z}_p$ correspond to the number of $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{Z}_p$ elements, respectively.

F.3 Proofs of Homomorphism Properties

For commitment homomorphism, we give a single theorem as all types of commitments are homomorphic. The commitments are homomorphic for all linear functions on the vectors.

Theorem 5 (Type-Commitment Homomorphism.). *Let $\mathcal{F}_{ck,t}$ be the family of binary functions for $t \in \{t_1, t_2, t_{\mathbb{G}}\}$ such that $F_{t,\alpha,\beta} \in \mathcal{F}_{ck,t}$ is defined as $F_{t,\alpha,\beta}(\mathbf{u}, \mathbf{v}) := \alpha \cdot \mathbf{u} + \beta \cdot \mathbf{v}$ for $\alpha, \beta \in \mathbb{Z}_p$. Let further* $\mathsf{EvalCom}$ *be defined as:*

$$\mathsf{EvalCom}(F_{t,\alpha,\beta}, U, V) := U^{\alpha} \cdot V^{\beta}$$

Then, for all $F_{t,\alpha,\beta} \in \mathcal{F}_{ck,t}$, $t \in \{t_1, t_2, t_{\mathbb{G}}\}$ $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$, U s.t. $\mathsf{VfCom}(t, U, \mathbf{u}) =$ true, and V s.t. $\mathsf{VfCom}(t, V, \mathbf{v}) = \mathsf{true}$,

$$\mathsf{VfCom}(t, \mathsf{EvalCom}(F_{t,\alpha,\beta}, U, V), F_{t,\alpha,\beta}(\mathbf{u}, \mathbf{v})) = \mathsf{true}$$

Proof. For the commitment types t_1 and $t_{\mathbb{G}}$, we know by the definition that $U = g^{\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(\tau)}$ and $V = g^{\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(\tau)}$. Then,

$$c = U^{\alpha} \cdot V^{\beta} = (g^{\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(\tau)})^{\alpha} \cdot (g^{\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(\tau)})^{\beta} = g^{\sum_{i=1}^{n} (\alpha \cdot u_i + \beta \cdot v_i) \cdot \mathcal{L}_i(\tau)}$$

where the last equation corresponds to a valid commitment to $\alpha \cdot \mathbf{u} + \beta \cdot \mathbf{v}$. Got the commitment type t_2, the same reasoning holds in the exponent of $\hat{g}$ instead of g. □

The homomorphisms of commitments can be transposed to the $\mathsf{IP}_{\mathbb{G}}$ and $\mathsf{IP}_{\mathbb{Z}_p}$ proofs, but we only provide the theorems for the homomorphisms on $\mathsf{IP}_{\mathbb{G}}$ proofs as we do not need the $\mathsf{IP}_{\mathbb{Z}_p}$ proofs for the generic construction. We note that the homomorphic properties shown below cover more than our generic construction requires. First we describe the $\mathsf{EvalProof}$ algorithm for the $\mathsf{IP}_{\mathbb{G}}$ homomorphism related to the commitment homomorphism of t_2 commitments and show its homomorphic properties.

Theorem 6 ($\mathsf{IP}_{\mathbb{G}}$-Proof Homomorphism on t_2.). *Let $\mathcal{X}$ be the set of binary partial operations on valid $\mathsf{IP}_{\mathbb{G}}$ statements with the same $t_{\mathbb{G}}$ commitments such that for $\hat{x}_1 := \big(Y_1, (t_{\mathbb{G}}, U), (t_2, V_1)\big)$ and $\hat{x}_2 := \big(Y_2, (t_{\mathbb{G}}, U), (t_2, V_2)\big)$ which are valid statements with the corresponding proofs π_1 and π_2, $X_{t_2,\alpha,\beta} \in \mathcal{X}$ is defined as $X_{t_2,\alpha,\beta}(\hat{x}_1, \hat{x}_2) := \big(Y_1^{\alpha} \cdot Y_2^{\beta}, (t_{\mathbb{G}}, U), (t_2, V_1^{\alpha} \cdot V_2^{\beta})\big)$ for $\alpha, \beta \in \mathbb{Z}_p$. Let further the $\mathsf{EvalProof}$ algorithm on $X_{t_2,\alpha,\beta}$ be defined as below. Then, for all $X_{t_2,\alpha,\beta} \in \mathcal{X}$, $\hat{x}_1$, $\hat{x}_1$, π_1, and π_2 as defined above,*

$$\mathsf{Vf}(\mathsf{IP}_{\mathbb{G}}, X_{t_2,\alpha,\beta}(\hat{x}_1, \hat{x}_2), \mathsf{EvalProof}(\mathsf{IP}_{\mathbb{G}}, X_{t_2,\alpha,\beta}, \pi_1, \pi_2)) = \mathsf{true}$$

$\mathsf{EvalProof}(\mathsf{IP}_{\mathbb{G}}, X_{t_2,\alpha,\beta}, \pi_1, \pi_2)$**:** For $\pi_1 := (g_{Q,1}, g_{R,1}, g_{R^*,1}, \delta_1, \hat{g}_{v^*,1})$ and $\pi_2 := (g_{Q,2}, g_{R,2}, g_{R^*,2}, \delta_2, \hat{g}_{v^*,2})$, output

$$\pi := (g_{Q,1}^{\alpha} \cdot g_{Q,2}^{\beta}, g_{R,1}^{\alpha} \cdot g_{R,2}^{\beta}, g_{R^*,1}^{\alpha} \cdot g_{R^*,2}^{\beta}, \delta_1^{\alpha} \cdot \delta_2^{\beta}, \hat{g}_{v^*,1}^{\alpha} \cdot \hat{g}_{v^*,2}^{\beta})$$

Proof. We only show why the first equation of the proof verification holds for the proof π that $\mathsf{EvalProof}$ and the other equations follow. As π_1 and π_2 are valid proofs, we know by the proof verification algorithm that

$$e(U, V_1) = e(g_{Q,1}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,1}, \hat{g}_1) \cdot e(\gamma_1, \hat{g}^{n^{-1}}) \tag{9}$$

$$e(U, V_2) = e(g_{Q,2}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,2}, \hat{g}_1) \cdot e(\gamma_2, \hat{g}^{n^{-1}}) \tag{10}$$

The first equation in the proof verification checks for the proof π that

$$e(U, V) = e(g_Q, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,}, \hat{g}_1) \cdot e(\gamma, \hat{g}^{n^{-1}})$$

$$e(U, V_1^{\alpha} \cdot V_2^{\beta}) = e(g_{Q,1}^{\alpha} \cdot g_{Q,2}^{\beta}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,1}^{\alpha} \cdot g_{R,2}^{\beta}, \hat{g}_1) \cdot e(\gamma_1^{\alpha} \cdot \gamma_2^{\beta}, \hat{g}^{n^{-1}})$$

$$\begin{aligned} e(U, V_1^{\alpha}) \cdot e(U, V_2^{\beta}) = {} & e(g_{Q,1}^{\alpha}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,1}^{\alpha}, \hat{g}_1) \cdot e(\gamma_1^{\alpha}, \hat{g}^{n^{-1}}) \\ & \cdot e(g_{Q,2}^{\beta}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,2}^{\beta}, \hat{g}_1) \cdot e(\gamma_2^{\beta}, \hat{g}^{n^{-1}}) \end{aligned}$$

$$[e(U,V_1)]^{\alpha} \cdot [e(U,V_2)]^{\beta} = [e(g_{Q,1}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,1}, \hat{g}_1) \cdot e(\gamma_1, \hat{g}^{n^{-1}})]^{\alpha} \cdot [e(g_{Q,2}, \hat{g}^{z_{\mathbb{H}}(\tau)}) \cdot e(g_{R,2}, \hat{g}_1) \cdot e(\gamma_2, \hat{g}^{n^{-1}})]^{\beta}$$

which holds by Equations 9 and 10. We can show that other equalities hold for the proof π by arranging the corresponding equalities of π_1 and π_2. □

Similarly, we describe the $\mathsf{EvalProof}$ algorithm for the $\mathsf{IP}_{\mathbb{G}}$ homomorphism related to the commitment homomorphism of $t_{\mathbb{G}}$ commitments and show its homomorphic properties. The only difference between the behavior of two $\mathsf{EvalProof}$ algorithms is their behavior to set g_{v^*}. As $X_{t_{\mathbb{G}},\alpha,\beta}$ keeps the t_2 commitment the same, the corresponding degree check g_{v^*} does not change.

Theorem 7 ($\mathsf{IP}_{\mathbb{G}}$-Proof Homomorphism on $t_{\mathbb{G}}$.). *Let $\mathcal{X}$ be the set of binary partial operations on valid $\mathsf{IP}_{\mathbb{G}}$ statements with the same t_2 commitments such that for $\hat{x}_1 := \big(Y_1, (t_{\mathbb{G}}, U_1), (t_2, V)\big)$ and $\hat{x}_2 := \big(Y_2, (t_{\mathbb{G}}, U_2), (t_2, V)\big)$ which are valid statements with the corresponding proofs π_1 and π_2, $X_{t_{\mathbb{G}},\alpha,\beta} \in \mathcal{X}$ is defined as $X_{t_{\mathbb{G}},\alpha,\beta}(\hat{x}_1, \hat{x}_2) := \big(Y_1^{\alpha} \cdot Y_2^{\beta}, (t_{\mathbb{G}}, U_1^{\alpha} \cdot U_2^{\beta}), (t_2, V)\big)$ for $\alpha, \beta \in \mathbb{Z}_p$. Let further the algorithm $\mathsf{EvalProof}$ be defined as below. Then for all $X_{t_{\mathbb{G}},\alpha,\beta} \in \mathcal{X}$, $\hat{x}_1$, $\hat{x}_1$, π_1, and π_2 as defined above,*

$$\mathsf{Vf}(\mathsf{IP}_{\mathbb{G}}, X_{t_{\mathbb{G}},\alpha,\beta}(\hat{x}_1, \hat{x}_2), \mathsf{EvalProof}(\mathsf{IP}_{\mathbb{G}}, X_{t_{\mathbb{G}},\alpha,\beta}, \pi_1, \pi_2)) = \mathsf{true}$$

$\mathsf{EvalProof}(\mathsf{IP}_{\mathbb{G}}, X_{t_{\mathbb{G}},\alpha,\beta}, \pi_1, \pi_2)$**:** For $\pi_1 := (g_{Q,1}, g_{R,1}, g_{R^*,1}, \delta_1, \hat{g}_{v^*,1})$ and $\pi_2 := (g_{Q,2}, g_{R,2}, g_{R^*,2}, \delta_2, \hat{g}_{v^*,2})$, output

$$\pi := (g_{Q,1}^{\alpha} \cdot g_{Q,2}^{\beta}, g_{R,1}^{\alpha} \cdot g_{R,2}^{\beta}, g_{R^*,1}^{\alpha} \cdot g_{R^*,2}^{\beta}, \delta_1^{\alpha} \cdot \delta_2^{\beta}, \hat{g}_{v^*,1})$$

F.4 Proof of CP-Pair's f-Zero-Knowledge Property

Theorem 8. CP-Pair *scheme in Figure 5 is perfectly f-zero-knowledge for the leakage function f in Figure 4.*

The leakage function f_{Com} leaks the message as it is for the commitment types t_1 and t_2, and similarly, the leakage function f_{Prove} leaks the witness as it is for the relations $\mathsf{IP}_{\mathbb{Z}_p}$ and BIT to the simulator. Thus, for these commitments and proofs, the simulator simply runs the Com and Prove algorithms.

For the commitment type $t_{\mathbb{G}}$ and the relation $\mathsf{IP}_{\mathbb{G}}$, this is not the case. Our proof will show that if keep τ in the st_{S} as a trapdoor key, we can compute these proofs. In a nutshell, for both of the commitment type $t_{\mathbb{G}}$ and the relation $\mathsf{IP}_{\mathbb{G}}$, the respective simulators will get $\mathbb{G}$ elements $U_i := g^{u_i}$ and they need to compute $g^{u_i \cdot P_i(\tau)}$ for some polynomials $P_i(X)$. We can compute them simply using τ as $g^{u_i \cdot P_i(\tau)} = U_i^{P_i(\tau)}$. The full proof of Theorem 8 is in Appendix F.4.

Proof. We set a SSetup that runs the Setup algorithm of f-zero-knowledge, but also outputs the trapdoor τ as part of the simulation state st_{S}. The function f leaks the witness as it is to the simulator for the relations $\mathsf{IP}_{\mathbb{Z}_p}$ and BIT. Thus, the simulator can run the original Prove algorithm for these relations. For

the relation $\mathsf{IP}_{\mathbb{G}}$, we can simulate proofs using the trapdoor τ as follows. Given $\mathbf{U} := [g^{u_i}, ..., g^{u_n}]$ and $\mathbf{v} := [v_1, ..., v_n]$, an $\mathsf{IP}_{\mathbb{G}}$ proof $\pi := (g_Q, g_R, g_{R^*}, \delta)$ can be simulated as

$$g_R := (\prod_{i=1}^{n} U_i^{v_i \cdot \mathcal{L}_i(\tau) - v_i/n})^{\tau^{-1}} \qquad g_{R^*} := g_R^{\tau} \qquad \hat{g}_{v^*} := g^{\mathbf{v}(\tau)}$$

$$g_Q := \big((\prod_{i=1}^{n} U_i^{\mathcal{L}_i(\tau)})^{\mathbf{v}(\tau)} / g^{\mathbf{y}(\tau)}\big)^{z_{\mathbb{H}}^{-1}(\tau)} \qquad \delta := (\prod_{i=1}^{n} U_i^{v_i})^{\tau^{n-1}}$$

Note that the computations only require $\mathbf{U}$, $\mathbf{v}$, and τ and the simulated proofs are identical to the honestly computed proofs as

$$\big((\prod_{i=1}^{n} U_i^{\mathcal{L}_i(\tau)})^{\mathbf{v}(\tau)} / g^{\mathbf{y}(\tau)}\big)^{z_{\mathbb{H}}^{-1}(\tau)} = (g^{(\mathbf{u}(\tau)\cdot\mathbf{v}(\tau) - \mathbf{y}(\tau))})^{z_{\mathbb{H}}^{-1}(\tau)} = g^{Q(\tau)} \tag{11}$$

$$(\prod_{i=1}^{n} U_i^{v_i \cdot \mathcal{L}_i(\tau) - v_i/n})^{\tau^{-1}} = g^{(\mathbf{y}(\tau) - \frac{\mathbf{u}\cdot\mathbf{v}}{n})\cdot\tau^{-1}} = g^{R(\tau)} \tag{12}$$

which can be verified using Equation 5 and simple group manipulations.

The commitment simulator can run the original Com algorithms for the commitment types t_1 and t_2. For the type $t_{\mathbb{G}}$, a valid commitment for a verifying message $\mathbf{U} := [U_1, \ldots, U_n]$ is $c_u := g^{\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(\tau)}$ for $U_i = g^{u_i}$. The commitment simulator answers the query as $\prod_{i=1}^{n} U_i^{\mathcal{L}_i(\tau)}$ which is identical to c_u. □

F.5 FB-DLOG Assumption

We first present the extra discussion on FB-DLOG assumption. Then we present the deferred proof of Theorem 3.

More Discussion on FB-DLOG Assumption. *Uber Assumption Family* [9] allows an adversary to ask the evaluation of any polynomials on a secret vector of $\mathbb{Z}_p$ element and return the evaluated values in the exponent of the group generator. In the end, the adversary needs to compute a group member and a polynomial such that the group member has the evaluation of the polynomial on the secret vector and the polynomial is linearly independent from the queried polynomials by the adversary. (n_1, n_2)-DLOG assumption only contains evaluations of univariate polynomials. However, Uber Assumption allows multivariate polynomials. Thus, one may question whether we could define an assumption from the Uber Assumption Family with bivariate polynomials such that $[\tau, \theta]$ will be the secret vector to evaluate the polynomials for $h := g^{\theta}$. Unfortunately, this is not possible since the Uber Assumption Family samples the secret value itself, but it's not an input.

Another candidate is *GeGenÜber Assumption* which is an extension of Uber Assumption that was defined by Bauer et al. [6]. In a nutshell, GeGenÜber Assumption extends the Uber Assumption such that the adversary can provide the

polynomial evaluation on the exponent of a generator of his choice instead of computing it strictly to the exponent of the generator g. For the Uber Assumption, we argued that h value can only be provided as an input to the challenger, but it cannot be output by the challenger. GeGenÜber Assumption allows the adversary to input the h value as a generator, but it does not return polynomial evaluations on this generator h which we need for h_i values.

Such an interactive definition that asks the adversary to choose the bases to build the problem instance on was also used by Fuchsbauer et al. [23]. They define an interactive variant of DDH where the adversary is asked to choose $(Q, \hat{Q})$ such that $e(Q, \hat{g}) = e(g, \hat{Q})$ and distinguish Q^{rs} from a random value when g^r, Q^r, g^s are given. Their assumption puts an extra requirement on the base vectors $\mathbf{H} := [Q]$ and $\hat{\mathbf{H}} := [\hat{Q}]$ to represent the same $\mathbb{Z}_p$ vector in the exponent by the pairing check. Our assumption does not require it. In fact, it allows even choosing the $\mathbf{H} := [Q]$ and $\hat{\mathbf{H}} := [\hat{Q}]$ with the different sizes. Another difference between our work and [23] on the allowed base vectors that an adversary can choose is that our work does not allow $1_{\mathbb{G}}$ or $1_{\hat{\mathbb{G}}}$ to be chosen as a base. This additional restriction is mainly due to defining a computational problem while [23] defines a decisional problem. In a decisional problem, using identity element as a base value instead of a generator does not give an extra advantage to the adversary. However, in a DLOG-like computational problem, using identity element as a base value usually makes the problem trivial to solve. [23] problem can be used as an example in the sense that the computational variant of their assumption would as for computing Q^{rs} and computing this value would be trivial if the adversary is allowed to choose $Q = 1_{\mathbb{G}}$. We leave the analysis of potential flexible-base assumption families and investigating the relation of decisional and computational flexible-base assumptions to each other as an open question.

Proof of Theorem 3. The proof relies on the fact that an algebraic adversary provides the algebraic representations of $\mathbf{H}$ and $\hat{\mathbf{H}}$. We embed a (n_1, n_2)-DLOG challenge into the FB-(n_1, n_2)-DLOG challenge using these representations. When the adversary comes up with a winning polynomial vector $\mathbf{f}$, we know that τ is a root of $\mathbf{f}_{\mathbf{H}}(X)$. The polynomial $\mathbf{f}_{\mathbf{H}}(X)$ can be computed in the clear using the algebraic representations of $\mathbf{H}$ and $\hat{\mathbf{H}}$ and we can easily find τ by computing $\mathbf{f}_{\mathbf{H}}(X)$'s roots.

Proof. The proof relies on the fact that an algebraic adversary provides the algebraic representations of $\mathbf{H}$ and $\hat{\mathbf{H}}$. As the adversary only knows the generators g and $\hat{g}$ from these groups, the representations of elements are in the form of $H_i := g^{\theta_i}$ and $\hat{H}_i := \hat{g}_i^{\hat{\theta}_i}$. We can take a (n_1, n_2)-DLOG problem instance $((g_i)_{i\in[n_2]}, (\hat{g}_i)_{i\in[n_2]})$, and compute the values to be returned to the adversary as,

$$(\mathbf{H}^{\tau^i})_{i\in[n_1]} = ([H_1, ..., H_\xi]^{\tau^i})_{i\in[n_1]} = ([g_i^{\theta_1}, ..., g_i^{\theta_\xi}])_{i\in[n_1]}$$
$$(\hat{\mathbf{H}}^{\tau^i})_{i\in[n_2]} = ([\hat{H}_1, ..., \hat{H}_{\hat{\xi}}]^{\tau^i})_{i\in[n_2]} = ([\hat{g}_i^{\hat{\theta}_1}, ..., \hat{g}_i^{\hat{\theta}_\xi}])_{i\in[n_2]}$$

which can be performed without the knowledge of τ. When the adversary returns a vector of polynomials $\mathbf{f}$ such that, $\mathbf{f}_{\mathbf{H}}(X) \neq 0$ and $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$, we know that τ is a root of the polynomial $\mathbf{f}_{\mathbf{H}}(X)$. We can compute the polynomial $\mathbf{f}_{\mathbf{H}}(X)$ in the clear as we already know the θ_i values. It means that we can compute τ in polynomial time by finding $\mathbf{f}_{\mathbf{H}}(X)$'s roots and we win (n_1, n_2)-DLOG game. $\square$

G Proof of Theorem 4

We present the proof of Theorem 4 separately for f-Φ_h-SE and f-Φ_h-SBND properties. The proofs of both properties apply the same game-hop change to answer the simulator queries. Then, they are specialized to have reductions from FB-(n_1, n_2)-DLOG assumption according to their own winning conditions.

G.1 Simulation Extractability

We start by showing that CP-Pair scheme is f-Φ_h-simulation-extractable.

Theorem 9. CP-Pair *scheme in Figure 5 is f-Φ_h-SE for for the function f and the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ in Figure 4 for all $h \in \mathbb{G}$ in AGM if FB-$(n-1, n)$-DLOG assumption in Definition 8 holds.*

Proof. Our proof strategy will contain two main parts for handling the queries to be simulated via zero-knowledge simulator and extracting the requested values related to the witnesses in the policy.

The policy Φ_h requires to simulate the inner-product proof queries for a vector of $\mathbb{G}$ elements $\mathbf{U}$ and vector of $\mathbb{Z}_p$ elements $\mathbf{v}$. It further allows $\mathbf{U}$ to contain h. The challenge to simulate the queries related to h is that the Prove algorithm needs $\mathbf{u}$ such that $\mathbf{U} = g^{\mathbf{u}}$ and we do not know the discrete logarithm θ of h, $h = g^{\theta}$. We are going to rely on multi-base $(n-1, n)$-DLOG assumption to simulate these queries. When we choose $\mathbf{H} := (g, h)$ and $\hat{\mathbf{H}} := (\hat{g})$, we get all elements in the commitment key ck in the form of g^{τ^i} and $\hat{g}^{\tau^i}$ and we also get extra elements $h^{\tau^i} = g^{\theta \cdot \tau^i}$. We need to compute $g^{\theta \cdot P(\tau)}$ for various $P(X)$ and all such values can be perfectly simulated as $h^{P(\tau)}$.

In the extraction part of the proof, we must show that the requested values in the policy Φ_h for each statement type in $\mathsf{IP}_{\mathbb{G}}$, $\mathsf{IP}_{\mathbb{Z}_p}$, and BIT. Our first observation will be that as we are in AGM, each $\hat{\mathbb{G}}$ value output by the adversary will have an algebraic representation on top of $\hat{g}^{\tau_i}$ values. Similarly, each $\mathbb{G}$ value output by the adversary will have an algebraic representation on top of g^{τ^i}, h^{τ^i}, and proofs that the simulator outputs. As we computed the simulated proofs only using g^{τ^i} and h^{τ^i} values, we can compute an algebraic representation of all $\mathbb{G}$ values that the adversary outputs. Relying on these representations which are on a multi-base $(n-1, n)$-DLOG challenge, we will provide a sequence of games with the reductions from multi-base $(n-1, n)$-DLOG assumption and show that our extractor only fails with a negligible probability at the end.

It is obvious that all of g_Q, g_R, and g_{R^*} can be computed using h_i values. Thus, SProve does not need the trapdoor τ anymore. As Game_1 outputs identical values to Game_0, $\Pr[W_1] = \Pr[W_0]$.

Game_1 (*Trapdoorless Simulation*): In this game, we change how the proof simulator SProve simulates proofs. The main aim of this change is to simulate the proofs without using the trapdoor τ in SProve so that we can change how we set the commitment key ck later on. There is a trivial case that we can simulate proofs easily, which is the case that $h = 1_{\mathbb{G}}$, so $\theta = 0$. According to the policy, θ is the only member of the vector $\mathbf{u}$ that we do not know. Hence, once we learn it, we can run the original Prove algorithm to simulate the proofs. In the rest of this game hop, we consider the non-trivial case where $h \neq 0$.

First, while running SSetup, we compute additional values $h_i := h^{\tau^i}$ for $i \in [n-1]$ for the internal use. Then, while computing the simulated proofs, we need to compute $g_Q := g^{Q(\tau)}$, $g_R := g^{R(\tau)}$, $g_{R^*} := g^{\tau \cdot R(\tau)}$ for the following $Q(X)$ and $R(X)$ polynomials. Note that we only need to simulate the queries that are allowed by the policy as an adversary making any other query would lose the game anyway. According to the policy Φ_h, the simulation queries can only be made for $\mathsf{IP}_{\mathbb{G}}$ statements. Furthermore, the leakage $(\mathbf{U}, \mathbf{v})$ that was defined in the leakage function f must be in the form of $\mathbf{U} = h^{\mathbf{e}_i}$ and $\mathbf{v} = \mathbf{e}_j$ for some i and j.

According to the limitations above, a simulated proof must compute $(g_Q := g^{Q(\tau)}, g_R := g^{R(\tau)}, g_{R^*} := g^{\tau \cdot R(\tau)}, \delta := g^{\Delta(\tau)}, g_{v^*} := \hat{g}^{\mathbf{v}^*(\tau)})$ for the polynomials defined in Figure 8. Note that the polynomials in Figure 8 can be reproduced easily by placing $(\mathbf{U}, \mathbf{v})$ values into the Equation 5 accordingly. Here, we show that these polynomials can be computed in the exponent by using only g_i and h_i values. For the polynomials to be computed on $\mathbb{G}$, we need to show that they have degree $\leq n-1$ and they contain θ with at most power 1. $Q(X)$ polynomial is computed using a polynomial division, so these division operations must have the remainder 0. For the cases that $i \neq j$ and $i = j$, we can show that the polynomial divisions with $z_{\mathbb{H}}$ have remainder 0 by relying on the Equations 6 and 7 of Lemma 3, respectively. Furthermore, $Q(X)$ polynomials have degree at most $n-2$, so g_Q values can be computed by only using h_i values. Similarly, the computation of $R(X)$ polynomial contains a polynomial division by X, and we can show that this division has remainder 0 by Equation 8 of Lemma 3. As $R(X)$ has degree $n-2$ at most, g_R values can be computed using h_i values. It is easy to observe that g_{R^*}, δ, g_{v^*} values can be computed, too.

It means that SProve does not need the trapdoor τ anymore, but it uses g_i and h_i values to simulate proofs. As Game_1 outputs identical values to Game_0, $\Pr[W_1] = \Pr[W_0]$.

Now, we define the different events that the adversary can win Game_1. We name the event E_T as the event that the adversary returns a statement $\hat{x}$ and a proof π from a relation R^{CS} for $R^{\mathsf{CS}} \in \{\mathsf{IP}_{\mathbb{G}}, \mathsf{IP}_{\mathbb{Z}_p}, \mathsf{BIT}\}$ to extract its witness. Then,

$$\Pr[W_1] \leq \Pr[E_{\mathsf{IP}_{\mathbb{G}}}] + \Pr\left[E_{\mathsf{IP}_{\mathbb{Z}_p}}\right] + \Pr[E_{\mathsf{IP}_{\mathsf{BIT}}}]$$

Indexes/Polynomials	$i \neq j$	$i = j$
$Q(X)$	$\theta \cdot \mathcal{L}_i(X) \cdot \mathcal{L}_j(X) \cdot z_{\mathbb{H}}^{-1}(X)$	$\theta \cdot (\mathcal{L}_i^2(X) - \mathcal{L}_i(X)) \cdot z_{\mathbb{H}}^{-1}(X)$
$R(X)$	0	$(\theta \cdot \mathcal{L}_i(X) - \theta/n) \cdot X^{-1}$
$R^*(X)$	0	$(\theta \cdot \mathcal{L}_i(X) - \theta/n)$
$\Delta(X)$	0	$\theta \cdot X^{n-1}$
$\mathbf{v}^*(X)$	$\mathcal{L}_j(X)$	$\mathcal{L}_i(X)$

Fig. 8. Polynomials to simulate $\mathsf{IP}_{\mathbb{G}}$ proofs.

We show one by one that all $\Pr[E_T]$'s are negligible. In this part of the proof we always have a reduction from FB-$(n-1,n)$-DLOG assumption. For all cases, we set $\mathbf{H} := (g, h)$ and $\hat{\mathbf{H}} := (\hat{g})$ and send it to the FB-$(n-1,n)$-DLOG challenger. When we get the g_i, h_i, and $\hat{g}_i$ values, we use them to set the crs and set the h_i values in the simulator state st_{S}. Note that if $h = 1_{\mathbb{G}}$, we do not need the h_i values and also FB-$(n-1,n)$-DLOG problem does not allow us to choose $h = 1_{\mathbb{G}}$. We could have a reduction from $(n-1,n)$-DLOG when $h = 1_{\mathbb{G}}$. Or, we can repeat all the argumentations below by simply arranging the equations for $\theta = 0$ and sending $\mathbf{H} := (g)$ and $\hat{\mathbf{H}} := (\hat{g})$ to the FB-$(n-1,n)$-DLOG challenger. Below, we consider the non-trivial case that $h \neq 1_{\mathbb{G}}$.

Extractor in the Event $E_{\mathsf{IP}_{\mathbb{Z}_p}}$. Our aim in this part of the proof is to show that if π is a verifying proof for $\hat{x}$, then we can either extract a vector $\mathbf{v}$ according to the policy's requirements or we can build an adversary that breaks FB-$(n-1,n)$-DLOG Assumption. Extractability of $\mathsf{IP}_{\mathbb{Z}_p}$ proofs rely on Lemma 2. Let $\hat{x} := \big((t_1, c_u), (t_2, c_v)\big)$. Commitments c_u and c_v encode the polynomials $\mathbf{u}(X)$ and $\mathbf{v}(X)$ of Equation 4 in the exponent. While the first pairing product equation in the proof verification checks that Equation 4 holds for some $Q(X)$ and $R(X)$ which were encoded in the exponents of g_Q and g_R, respectively, the second pairing product equation performs a degree check on the $R(X)$ value encoded in g_R's exponent. Namely, it checks that the degree of the encoded $R(x)$ is at most $n-2$. Below we show that these pairing product equations hold if either the encoded polynomials satisfy the aforementioned checks, or we can break the FB-$(n-1,n)$-DLOG Assumption.

Degree Check on g_R. As we are in AGM, the adversary outputs the algebraic representations of g_R and g_{R^*} in the form of

$$g_R := \prod_{i=0}^{n-1} g_i^{\tilde{R}_i} \cdot h_i^{\bar{R}_i} \qquad g_{R^*} := \prod_{i=0}^{n-1} g_i^{\tilde{R}_i^*} \cdot h_i^{\bar{R}_i^*}$$

We define the polynomials $\tilde{R}(X) := \sum_{i=0}^{n} \tilde{R}_i \cdot X^i$ and $\bar{R}(X) := \sum_{i=0}^{n} \bar{R}_i \cdot X^i$. The polynomials $\tilde{R}^*(X)$ and $\bar{R}^*(X)$ are defined similarly. Note that the main polynomials we want to use for degree checks are $R(X) := \tilde{R}(X) + \theta \cdot \bar{R}(X)$ and $R^*(X) := \tilde{R}^*(X) + \theta \cdot \bar{R}^*(X)$. Finally, we define the vector of polynomials $\mathbf{f} := [X \cdot \tilde{R}(X) - \tilde{R}^*(X), X \cdot \bar{R}(X) - \bar{R}^*(X)]$. As π is a valid proof for $\hat{x}$, we

know that $e(g_R, \hat{g}_2) = e(g_{R^*}, \hat{g})$ holds, so $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$. Thus, either $\mathbf{f_H}(X)$ is zero polynomial, or $\mathbf{f}$ is a valid answer to the FB-$(n-1,n)$-DLOG challenge. The former implies that $R(X) \cdot X = R^*(X)$, so the degree of $R(X)$ is at most $n-2$. The latter is negligible by FB-$(n-1,n)$-DLOG Assumption.

Degree Check on c_v. The second degree check is performed on c_v using the pairing product equation with g_{v^*}. We define the polynomials from the algebraic representations vor g_{v^*} as $\mathbf{v}^*(X) := \tilde{\mathbf{v}}^*(X) + \theta \cdot \bar{\mathbf{v}}^*(X)$. We note that $\mathbf{deg}(\mathbf{v}^*, \mathbf{H}) \leq n-1$ due to the algebraic representation of g_{v^*}. We also define polynomial $\mathbf{v}(X)$ from c_v's algebraic representation which has degree $\leq n$. Now we build the vector of polynomials $\mathbf{f} := [\tilde{\mathbf{v}}^*(X) - \mathbf{v}(X), \bar{\mathbf{v}}^*(X)]$. As π is a valid proof for $\hat{x}$, we know that $e(g_{v^*}, \hat{g}) = e(g, c_v)$ holds, so $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$.Thus, either $\mathbf{f_H}(X)$ is zero polynomial, or $\mathbf{f}$ is a valid answer to the FB-$(n-1,n)$-DLOG challenge. The former implies that $\mathbf{v}^*(X) = \mathbf{v}(X)$, so the degree of $\mathbf{v}(X)$ is at most $n-1$. The latter is negligible by FB-$(n-1,n)$-DLOG Assumption.

Inner-Product Polynomial Relation Check. Now we can check that the main equation of inner-product relation, Equation 4, holds if the proof verification is successful and we can extract the vector $\mathbf{v}$ from the proof. According to the extraction policy, the adversary provides us a valid opening $\mathbf{u}$ for the commitment c_u as auxiliary information. Let $\mathbf{u}(X) \sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)$ be the polynomial representation of the vector $\mathbf{u}$. Let further $\mathbf{v}(X) := \sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)$ be the algebraic representation of the commitment c_v. Note that, we can compute $\mathbf{v}(X)$ according to the Lagrange basis polynomials as it has degree $\leq n-1$. Now we set the vector of polynomials $\mathbf{f} := [\mathbf{u}(X) \cdot \mathbf{v}(X) - \gamma/n - X \cdot \tilde{R}(X) - z_{\mathbb{H}}(X) \cdot \tilde{Q}(X), -X \cdot \bar{R}(X) - z_{\mathbb{H}}(X) \cdot \bar{Q}(X)]$. We know by the proof verification equation that $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$. If $\mathbf{f_H}(X) \neq 0$, then $\mathbf{f}$ is a valid answer to the FB-$(n-1,n)$-DLOG challenge which occurs only with a negligible probability. Otherwise, $\mathbf{f_H}(X) = 0$ and we have

$$(\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)) \cdot (\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)) = \gamma/n + X \cdot R(X) + Q(X) \cdot z_{\mathbb{H}}(X)$$

By Lemma 2, we conclude that $\mathbf{u} \cdot \mathbf{v} = \gamma$ where $\mathbf{v} := [v_1, ..., v_n]$ which is also a valid opening for c_v and we output $\hat{w}' := \mathbf{v}$.

As a result, we conclude that

$$\Pr\left[E_{\mathsf{IP}_{\mathbb{Z}_p}}\right] \leq 3 \cdot \mathsf{negl}_{\text{FB-DLOG}}(\lambda)$$

Extractor in the Event $E_{\mathsf{IP}_{\mathbb{G}}}$. We will rely on a similar argumentation to the $E_{\mathsf{IP}_{\mathbb{Z}_p}}$'s to show that $E_{\mathsf{IP}_{\mathbb{G}}}$ is negligible. $\mathsf{IP}_{\mathbb{G}}$ proof relies on the Lemma 2 just like the $\mathsf{IP}_{\mathbb{Z}_p}$ proofs. The first difference is that we need to check the validity of the inner-product result in the exponent. μ/n is a constant term in the inner-product polynomial relation of Lemma 2. In $\mathsf{IP}_{\mathbb{Z}_p}$ proof, this is checked trivially as we run the pairing product equation by performing the exponentiation g^{μ} explicitly. However, this check can not be performed in $\mathsf{IP}_{\mathbb{G}}$ proof verification as the verifier

does not know the exponent value. We solve this issue by adding another pairing product equation to perform a degree check on γ.

Another change compared to the $\mathsf{IP}_{\mathbb{Z}_p}$ proof extraction is because of the different auxiliary information that the extractor gets. For $\mathsf{IP}_{\mathbb{Z}_p}$ proofs, the extractor directly gets an opening of the commitment c_u. However, for $\mathsf{IP}_{\mathbb{G}}$ proofs, the extractor gets n other proofs on the same commitment c_u.

Before starting the proof, we parse some values that the extractor gets to express our notation clearly. The extractor gets the valid proofs π_j's for the statements $\hat{x}_j$'s.

$$\pi_j := (g_{Q,j}, g_{R,j}, g_{R^*,j}, \delta_j, \hat{g}_{v^*,j}) \qquad \hat{x}_j := \big(U_j, (t_1, c_u), (t_2, c_{\mathbf{e}_j})\big)$$

where $c_{\mathbf{e}_j} = \hat{g}^{\mathcal{L}_j(\tau)}$. Furthermore, we build the polynomials $Q(X)$, $R(X)$, $Q_j(X)$, and $R_j(X)$ are algebraic representations of g_Q, g_R, $g_{Q,j}$, and $g_{R,j}$, respectively to use them in the rest of the proof.

Degree Checks. We write down all degree checks that can be done on the values we have here. All $R_j(X)$ polynomials and $R(X)$ polynomial has degree $\leq n-2$. $\mathbf{v}(X)$ has degree at most $n-1$. Finally, the algebraic representations of all U_j values and γ are in the form $U_j := g^{\tilde{u}'_j} \cdot h^{\bar{u}'_j}$ and $\gamma := g^{\tilde{\mu}} \cdot h^{\bar{\mu}}$, respectively.

Inner-Product Polynomial Relation Check for γ. Using degree checks, Lemma 2, and FB-$(n-1,n)$-DLOG Assumption, we are going to show that the extractor fails for $\mathsf{IP}_{\mathbb{G}}$ proofs only with a negligible probability.

By AGM, we have a polynomial representation of c_u, $\mathbf{u}(X) := \tilde{\mathbf{u}}(X) + \theta \cdot \bar{\mathbf{u}}(X)$. As these polynomials have degree $n-1$, we can write them as $\tilde{\mathbf{u}}(X) := \sum_{i=1}^{n} \tilde{u}_i \cdot \mathcal{L}_i(X)$ and $\bar{\mathbf{u}}(X) := \sum_{i=1}^{n} \bar{u}_i \cdot \mathcal{L}_i(X)$. Let further $\mathbf{v}(X) := \sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)$ be the algebraic representation of the commitment c_v. Note that, we can compute $\mathbf{v}(X)$ according to the Lagrange basis polynomials as it has a degree $\leq n-1$. Now we set the vector of polynomials

$$\begin{aligned}\mathbf{f} := [&\tilde{\mathbf{u}}(X) \cdot \mathbf{v}(X) - \tilde{\mu}/n - X \cdot \tilde{R}(X) - z_{\mathbb{H}}(X) \cdot \tilde{Q}(X),\\ &\bar{\mathbf{u}}(X) \cdot \mathbf{v}(X) - \bar{\mu}/n - X \cdot \bar{R}(X) - z_{\mathbb{H}}(X) \cdot \bar{Q}(X)]\end{aligned}$$

We know by the proof verification equation that $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$. If $\mathbf{f}_{\mathbf{H}}(X) \neq 0$, then $\mathbf{f}$ is a valid answer to the FB-$(n-1,n)$-DLOG challenge which occurs only with a negligible probability. Otherwise, $\mathbf{f}_{\mathbf{H}}(X) = 0$, and we have

$$(\sum_{i=1}^{n} (\tilde{u}_i + \theta \cdot \bar{u}_i) \cdot \mathcal{L}_i(X)) \cdot (\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(X)) = (\tilde{\mu} + \theta\bar{\mu})/n + X \cdot R(X) + Q(X) \cdot z_{\mathbb{H}}(X)$$

By Lemma 2, we get $\sum_{i=1}^{n} (\tilde{u}_i + \theta \cdot \bar{u}_i) \cdot v_i = \tilde{\mu} + \theta\bar{\mu}$.

Inner-Product Polynomial Relation Check for U_j's. Similar to γ, we show that the inner-product polynomial relation holds for all U_j's.

As $\mathsf{IP}_{\mathbb{G}}$ proofs on U_j's are performed for the commitments c_u and $c_{\mathbf{e}_j}$'s, we know polynomial representations of $c_{\mathbf{e}_j}$'s, $e_j(X) := \mathcal{L}_j(X)$. We set the vector of polynomials

$$\begin{aligned}\mathbf{f}^{(j)} := [&\tilde{\mathbf{u}}(X) \cdot \mathcal{L}_j(X) - \tilde{u}'_j/n - X \cdot \tilde{R}_j(X) - z_{\mathbb{H}}(X) \cdot \tilde{Q}_j(X),\\ &\bar{\mathbf{u}}(X) \cdot \mathcal{L}_j(X) - \bar{u}'_j/n - X \cdot \bar{R}_j(X) - z_{\mathbb{H}}(X) \cdot \bar{Q}_j(X)]\end{aligned}$$

We know by the proof verification equation that $\mathbf{H}^{\mathbf{f}^{(\mathbf{j})}(\tau)} = 1_{\mathbb{G}}$. If $\mathbf{f}^{(j)}_{\mathbf{H}}(X) \neq 0$, then $\mathbf{f}^{(j)}$ is a valid answer to the FB-$(n-1,n)$-DLOG challenge which occurs only with a negligible probability. Otherwise, $\mathbf{f}^{(j)}_{\mathbf{H}}(X) = 0$, and we have

$$(\sum_{i=1}^{n}(\tilde{u}_i + \theta \cdot \bar{u}_i) \cdot \mathcal{L}_i(X)) \cdot \mathcal{L}_j(X) = (\tilde{u}'_j + \theta \cdot \bar{u}'_j)/n + X \cdot R_j(X) + Q_j(X) \cdot z_{\mathbb{H}}(X)$$

By Lemma 2, we get $\tilde{u}_j + \theta \cdot \bar{u}_j = \tilde{u}'_j + \theta \cdot \bar{u}'_j$.

Finally, we are able to argue that the vector $\mathbf{v} := [v_1, \ldots, v_n]$ is a valid opening to the extractability policy. First, $\mathbf{v}$ is a valid opening to the commitment c_v. Secondly, By the inner-product polynomial relation check on γ, we know that $\gamma = g^{\sum_{i=1}^{n} v_i \cdot \tilde{u}_i} \cdot h^{\sum_{i=1}^{n} v_i \cdot \bar{u}_i} = \prod_{i=1}^{n}(g^{\tilde{u}_i} \cdot h^{\bar{u}_i})^{v_i}$. By the inner-product polynomial relation checks on U_j values, we get $\prod_{i=1}^{n}(g^{\tilde{u}_i} \cdot h^{\bar{u}_i})^{v_i} = \prod_{i=1}^{n}(g^{\tilde{u}'_i} \cdot h^{\bar{u}'_i})^{v_i} = \prod_{i=1}^{n}(U_i)^{v_i}$, so $\mathbf{U}^{\mathbf{v}} = \gamma$.

$$\Pr[E_{\mathsf{IP}_{\mathbb{G}}}] \leq (2n+5) \cdot \mathsf{negl}_{\text{FB-DLOG}}(\lambda)$$

Extractor in the Event E_{BIT}. The extractor for the relation BIT will rely on the Hadamard-product polynomial relation from Lemma 1.

Equality Check on g_u. The pairing product equation $e(g, c_u) = e(g_u, \hat{g})$ checks whether c_u and g_u encodes the same polynomial or not. As we are in AGM, we have the following representations

$$g_{u^*} := \prod_{i=0}^{n-1} g_i^{\tilde{\kappa}_i} \cdot h_i^{\bar{\kappa}_i} \qquad\qquad c_u := \prod_{i=0}^{n} \hat{g}_i^{\iota_i}$$

Let $\tilde{\kappa}(X) := \sum_{i=0}^{n-1} \tilde{\kappa}_i \cdot X^i$, $\bar{\kappa}(X) := \sum_{i=0}^{n-1} \tilde{\kappa}_i \cdot X^i$, and $\iota(X) := \sum_{i=0}^{n} \iota_i \cdot X^i$. Finally, let the vector of polynomials $\mathbf{f} := [\tilde{\kappa}(X) - \iota(X), \bar{\kappa}(X)]$. As $e(g, c_u) = e(g_{u^*}, \hat{g})$, $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$. If $\mathbf{f}_{\mathbf{H}}(X) = 0$, then we get $\tilde{\kappa}(X) + \theta\bar{\kappa}(X) = \iota(X)$. Otherwise, $\mathbf{f}$ is a valid answer to the FB-$(n-1,n)$-DLOG problem. Note that this equality also serves as a degree check as the algebraic representations in $\mathbb{G}$ have degree $\leq n-1$.

Hadamard-Product Polynomial Relation Check. Finally, we are going to how to extract the opening $\mathbf{u}$ by relying on Lemma 1 and FB-(n,n)-DLOG Assumption. Let the vector of polynomials $\mathbf{f} := [\iota(X) \cdot (1 - \tilde{\kappa}(X)) - z_{\mathbb{H}}(X) \cdot \tilde{Q}(X), -\iota(X) \cdot \bar{\kappa}(X) -$

$z_{\mathbb{H}}(X) \cdot \bar{Q}(X)]$. By the proof verification algorithm, $\mathbf{H}^{\mathbf{f}(\tau)} = 1_{\mathbb{G}}$. If $\mathbf{f}_{\mathbf{H}}(X) \neq 0$, then $\mathbf{f}$ is a valid answer to the FB-$(n-1, n)$-DLOG problem. Otherwise, we have

$$(1 - (\tilde{\kappa}(X) + \theta \cdot \bar{\kappa}(X))) \cdot \iota(X) = z_{\mathbb{H}}(X) \cdot Q(X)$$

By the degree check on c_u, $\iota(X)$ has degree at most $n-1$, so we can compute a vector encoded in $\iota(X) := \sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)$. Furthermore, by relying on the equality check and $1 = \sum_{i=1}^{n} \mathcal{L}_i(X)$, we get

$$(\sum_{i=1}^{n}(1 - u_i) \cdot \mathcal{L}_i(X)) \cdot (\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(X)) = z_{\mathbb{H}}(X) \cdot Q(X)$$

which shows $\mathbf{u} \circ (\mathbf{1} - \mathbf{u}) = \mathbf{0}$ by Lemma 1. This Hadamard-product equality shows that $\mathbf{u}$ is a bit vector which is a valid opening to c_u and the extractor outputs $\mathbf{u}$.

$$\Pr[E_{\mathsf{BIT}}] \leq 2 \cdot \mathsf{negl}_{\text{FB-DLOG}}(\lambda)$$

Finally, we conclude our proof by

$$\Pr\Big[\mathsf{Exp}^{f\text{-}\Phi\text{-}SE}_{\Pi,\mathcal{A},\mathcal{E}}(\lambda) = \mathsf{true}\Big] \leq (2n + 10) \cdot \mathsf{negl}_{\text{FB-DLOG}}(\lambda)$$

□

G.2 Simulation-Sound Binding

Now we show that Construction 5 is f-Φ_h-simulation-sound-binding. An important difference is that the f-Φ_h-SBND property can be proven without relying on AGM.

Theorem 10. *Construction 5 is f-Φ_h-SBND for for the function f and the policy $\Phi_h := (\Phi_{sim}, \Phi_{ext}, \Phi_{bnd})$ in Figure 4 for all $h \in \mathbb{G}$ if FB-$(n-1, n)$-DLOG assumption in Definition 8 holds.*

Proof. As the simulation-extractability and the simulation-sound-binding games are identical in the query phase, we can answer the simulator queries identically to Game_1 of the simulation extractability proof. thus, we omit to explain the simulation part and show that if an adversary can break the binding property according to the policy Φ_h, we can break the FB-(n_1, n_2)-DLOG Assumption.

We set $\mathbf{H} := (g, h)$ and $\hat{\mathbf{H}} := (\hat{g})$ and send it to the FB-$(n-1, n)$-DLOG challenger. When we get the g_i, h_i, and $\hat{g}_i$ values, we use them to set the crs and set the h_i values in the simulator state st_{S} as in the proof of simulation-extractability. Now let $\mathbf{u}$ and $\mathbf{v}$ be two distinct vectors and C be a type t_2 commitment such that $\mathsf{VfCom}_{ck}(t_2, C, \mathbf{u})$ and $\mathsf{VfCom}_{ck}(t_2, C, \mathbf{v})$. By the verification equation,

$$C = \hat{g}^{\sum_{i=1}^{n} u_i \cdot \mathcal{L}_i(\tau)} \qquad\qquad C = \hat{g}^{\sum_{i=1}^{n} v_i \cdot \mathcal{L}_i(\tau)}$$

We set polynomial $f(X) := \sum_{i=1}^{n}(u_i - v_i) \cdot \mathcal{L}_i(X)$. As $\mathbf{u}$ and $\mathbf{v}$ are distinct vectors, we know that there is a non-zero $u_i - v_i$, so $f(X)$ is a non-zero polynomial. However, by the verification equations above, we know that $g^{f(\tau)} = C/C = 1_{\mathbb{G}}$, so $f(\tau) = 0$. Thus, τ is a root of the polynomial $f(X)$ which can be computed in a polynomial time. □