

Structure-Preserving Compressing Primitives, Revisited

Gennaro Avitabile¹, Dario Fiore², and Gonzalo Martinez de Sola^{2,3}

¹ University of Surrey, Guildford, UK
avitabilegenn@gmail.com

² IMDEA Software Institute, Madrid, Spain

{dario.fiore,gonzalo.martinezdesola}@imdea.org

³ Universidad Politecnica de Madrid, Madrid, Spain

Abstract. Structure-preserving (SP) cryptography enables the modular composition of pairing-based primitives through the definition of cryptographic algorithms whose public inputs and outputs are group elements and whose computations are generic group operations or pairings. Despite their potential usefulness, obtaining compressing primitives such as accumulators and vector commitments in the SP setting has remained a long-standing challenge, partly due to known impossibility results [Abe et al. Eurocrypt 2012]. Recently, [Krenn, Mir, and Slamanig PKC 2026] proposed the first SP vector commitments (SPVC) and accumulators (SPA) that circumvent these barriers via the adoption of structured message spaces that, albeit restricted, enable interesting applications such as constant-size pairing-based ring signatures.

In this work, we revisit the security of these recent constructions with a twofold contribution. First, we identify attacks against the recently proposed SPVC, SPA, and ring signature schemes. Second, we present new constructions of SPVCs and SPAs that preserve the efficiency and structure-preserving nature of the original designs while achieving the desired standard security notions of position-binding and collision-resistance respectively. Specifically, we construct a new SPVC with constant-size proofs and verification under the power discrete logarithm assumption in the algebraic group model. Building upon it, we introduce a new notion of SPA with extended message spaces and give a construction achieving collision resistance and constant-size membership proofs under the same assumption. Finally, we apply our techniques to obtain constant-size ring signatures in bilinear groups.

1 Introduction

Compressing cryptographic primitives enable representing large data structures using succinct, ideally constant-size, values. Even more interestingly, they allow generating short proofs attesting membership or correctness properties with respect to the underlying data.

In this work, we focus on two prominent examples of this paradigm: *accumulators* [BdM94, BP97] and *vector commitments* [LY10, CF13]. Accumulators

operate over sets: they allow compressing a set $X = \{x_1, \dots, x_q\}$ into a short value **Acc** while enabling succinct membership proofs for elements $x \in X$. Vector commitments (VCs, for brevity) operate over ordered data: they compress a vector $\mathbf{v} = (v_1, \dots, v_q)$ into a commitment **com**, and later allow proving that a value x corresponds to a given position, i.e., that $x = v_i$.

A fundamental feature of these primitives is that both commitments and proofs are succinct relative to the size of the underlying data structure, typically logarithmic or even constant in q . Succinctness makes accumulators and vector commitments useful building blocks in a wide variety of protocols, where they help minimize communication complexity. Examples of applications include zero-knowledge sets [MRK03, LY10, CF13], succinct arguments [Kil94, Mic94, BBF19, LM19], privacy-preserving cryptocurrencies [MGGR13, BCG⁺14], and stateless blockchains [CPZ18, BBF19].

Structure-Preserving Compressing Primitives. In many applications, an important challenge is ensuring that the different cryptographic primitives composing the protocol interact *efficiently and in a compatible way*. *Structure-preserving cryptography* [AFG⁺10, Abe15] has emerged as a powerful paradigm for obtaining modular and efficiently composable constructions in the pairing-based setting. Informally, a structure-preserving (SP) cryptographic scheme is one in which all public inputs and outputs are group elements, and all algorithms consist solely of generic group operations and pairing computations, without relying on non-algebraic operations such as hashing into bit strings or manipulating explicit group representations. SP cryptography is particularly attractive because such primitives compose naturally with one another. For instance, the outputs of one primitive can directly serve as inputs to another without requiring additional encodings or conversions between different domains. This algebraic compatibility is especially useful in protocols involving complex statements about cryptographic objects themselves, such as proving knowledge of signatures, commitments, or public keys in zero-knowledge [GS08] – a typical paradigm to construct anonymous credentials and other privacy-preserving primitives.

The realization of structure-preserving accumulators and vector commitments, namely schemes that can commit to or accumulate group elements, has remained elusive for many years. If such primitives exist, they would enable interesting applications, for example combining the short communication of succinct opening proofs with efficient zero-knowledge proofs integration [GS08].

The lack of SP accumulators and vector commitments is not a coincidence. Abe et al. [AHO12] showed an impossibility result ruling out compressing binding commitments where both the messages and the commitment belong to the same source group. In particular, this excludes constructions of SP accumulators and VCs, since both these primitives can be seen as compressing binding commitments of large messages, such as sets or vectors.

SP Accumulators and VCs for Structured Message Spaces. The recent work of Krenn, Mir, and Slamanig [KMS26] challenged this state of affairs and proposed the first structure-preserving constructions of vector commitments and accumulators that circumvent the known impossibility results. Their key insight

is to relax the structure-preserving requirement on the message space. Rather than supporting arbitrary group elements, their schemes operate over more structured messages satisfying specific algebraic relations with respect to a set of public parameters. Interestingly, they demonstrate that these structured messages remain expressive enough for nontrivial applications. Notably, they use their SP accumulator construction to obtain the first constant-size ring signatures based on bilinear groups, a result that was unknown before their work.

1.1 Our Work

In this paper, we revisit the SP vector commitment, SP accumulator, and ring signature schemes recently proposed in [KMS26]. Our results contribute on two fronts. First, we identify attacks showing that the original constructions do not fully achieve their claimed security guarantees. Second, we present revised constructions that preserve the main efficiency and structural properties of the original schemes while achieving the desired standard security guarantees.

More in detail, our results are as follows.

Structure-preserving Vector Commitments (SPVCs). The authors of [KMS26] propose two SPVC schemes for a structured message space $\mathcal{M}_q^{\text{pp}}$ that consists of vectors $(\text{msg}_1, \dots, \text{msg}_q)$ where the i -th element msg_i is as follows. It consists of a *message component* $(M_{i0}, M_{i1}) = (g_1^{m_i}, g_1^{\alpha^i m_i})$ and $q - 1$ *auxiliary tags* $T_{ij} = g_1^{\alpha^j m_i}$ for $j \in [i + 1, i + q]$ and $j \neq q + 1$. Here $g_1^{\alpha^j}$ for $j \in [1, 2q] \setminus \{q + 1\}$ are part of the scheme’s public parameters (aka CRS). The two SPVCs in [KMS26], that we call SPVC_0 and SPVC_1 , proceed identically except for a difference in the verification algorithm.

SPVC_0 ’s verification needs access to the full message tuple msg_i of length q , and validates its structure. The position binding of SPVC_0 is proven in the generic group model and crucially relies on the validation of the q -long message msg_i . Validating the structured message however is a major efficiency cost that increases the verifier’s work and can vanish the benefit of succinctness, if one has to transmit the auxiliary values to the verifier.

The scheme SPVC_1 addresses this efficiency challenge via a simpler verification that only works on input the (M_{i0}, M_{i1}) component of the message. This scheme is proven position binding under the q -DHE assumption in the AGM.

As our first result, we identify a vulnerability in the efficient verification equation of SPVC_1 , and show an efficient attack that breaks its position binding with probability 1. We give this result in Section 3.2.

Learning from our security analysis, our next contribution is to design a new SPVC scheme that is inspired to SPVC_1 from [KMS26] but introduces important changes that ensure its position binding, that we prove under the power discrete logarithm assumption [Lip12] in the AGM [FKL18]. Our SPVC scheme presents several advantages. First, it has constant-size proofs and verification time, as opposed to SPVC_0 ’s verification which needs $O(q)$ tags and takes $O(q)$ time to check their validity. Second, it is conceptually simpler: (i) it needs a shorter CRS including $\{g_1^{\alpha^j}, g_2^{\alpha^j}\}_{j=1}^q$ (i.e., without elements $g_1^{\alpha^{q+2}}, \dots, g_1^{\alpha^{2q}}$ and no need of a

missing $q + 1$ -th power), (ii) the message component of its structured message space is a single value $M_i = g_1^{m_i}$, independent of α . See Section 4 for all details.

Structure-preserving accumulators (SPAs). Krenn et al. [KMS26] introduce an SPA that is constructed upon their structure-preserving vector commitment SPVC_1 . The high level idea is to use the VC to commit to the set, encoded as a vector after fixing some ordering in the set. Building upon SPVC_1 , they define their SPA to work with sets made of group elements that follow the same structure as in the SPVC, namely a message component and auxiliary tags that depend on the elements of the CRS. The main challenge in using an SPVC to build an SPA is to avoid, in a membership proof for a set element x , including the position of x in the vector representation of the set. Such position is an integer and would make the membership proof non-structure-preserving. Alternatively, one could replace such integer with an element from the CRS. Done naively, this would imply an $O(q)$ work for the verifier and a difficulty to keep the position’s information private in a succinct zero-knowledge proof. Namely, privacy-preserving applications of such an SPA would not inherit succinct proofs.

The SPA of [KMS26] follows the latter approach aiming to solve these privacy and efficiency challenges. The authors prove their SPA to satisfy a specialized security notion, weaker than standard collision resistance, but potentially sufficient in applications like ring signatures. Their proof relies (in a non-black-box way) on the position-binding of SPVC_1 . While our first set of attacks already invalidates this assumption on SPVC_1 , we extend our analysis to confirm a vulnerability in this SPA, identifying two attacks against its security (see Sec. 3.4).

Our main contribution about SPAs is then to propose a new scheme that achieves the same efficiency of the one from [KMS26]—namely constant-size proofs—while ensuring the standard security notion of collision resistance, that we prove under the power discrete logarithm assumption in the AGM. Our SPA construction builds upon the same promising idea from [KMS26] that one can bypass existing impossibility results via the use of a structured message space. To achieve our result, we identify a limitation of existing definitions if one aims to capture SPA constructions where the verifier does not hold the (possibly long) auxiliary information of the element to verify (e.g., the $g^{x\alpha^j}$) – i.e., constructions that can realize the full succinctness potential of SPA. Therefore, we first propose a rigorous definition – called SPA with extended message space – that captures this kind of constructions in terms of algorithms’ syntax and security. Next, we give an SPA construction that satisfies our definition. To do so, we build upon the techniques of our new SPVC, showing how to obtain structure-preserving membership proofs that satisfy collision resistance.

Constant-size Ring Signatures. Finally, as our last contribution (Section 6) we show an application of SPAs with extended message space to build pairing-based ring signatures (RS) [RST01] whose signatures size is constant in the dimension of the ring. This is the same application proposed in [KMS26], which, given our attacks, did not admit secure instantiations. Furthermore, even assuming that the RS of [KMS26] were instantiated with a collision-resistant SPA, we show an attack revealing the identity of the signer. Therefore, we propose

a different modular construction using our new SPA, along with BLS signatures [BLS04] and Groth-Sahai proofs [GS08]. All components integrate nicely with each other and we are able to argue security in a straightforward manner via black-box reductions to our new definitions of SPAs with extended message space. Concretely, our signature is composed of 33 elements in $\mathbb{G}_1$ and 34 in $\mathbb{G}_2$, giving a size of roughly 4.8 kB when using the BLS12-381 curve.

Open Problems. As discussed earlier, our constructions rely on security proofs in the AGM. Obtaining SPVCs and SPAs with comparable efficiency under standard-model reductions to falsifiable assumptions remains an interesting open problem. More fundamentally, given the apparent necessity of structured message spaces together with the algebraic restrictions imposed by the SP setting, it is unclear whether such standard-model constructions are achievable at all, or whether an AGM/GGM-based security is inherently necessary. Another important direction is to identify further applications of our SP primitives that can benefit from their succinctness and work with their structured message spaces.

Note: After the completion of our work, we communicated our findings to the authors of [KMS26] and shared our manuscript with them. After this communication, the authors of [KMS26] revised the full version of their paper on ePrint, providing an alternative solution to overcome the issues we identified.

2 Preliminaries

Notation. We define $[n]$ as the range $\{1, \dots, n\}$ for an arbitrary $n \in \mathbb{N}$. We use bold font when referring to vectors $\mathbf{v} = (v_i)_{i \in [m]} = (v_1, \dots, v_m)$. When referring to a probabilistic polynomial-time algorithm $\mathcal{A}$, we abbreviate it as $\mathcal{A}$ PPT. By $r \leftarrow \$ S$, we denote that r is chosen uniformly at random from a set S . Furthermore, we write $\mathcal{A}(a_1, \dots, a_n; r)$ if we want to make the randomness r used by a probabilistic algorithm $\mathcal{A}(a_1, \dots, a_n)$ explicit. We write $\text{negl}(\lambda)$ to denote an arbitrary negligible function in the security parameter λ . That is, a function $f : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ such that for every positive polynomial $p(\cdot)$, there exists $N \in \mathbb{N}$ such that for all $\lambda > N$, it holds that $f(\lambda) < 1/p(\lambda)$.

We denote polynomials by uppercase letters P, Q and specify them by a list of their coefficients. If m is an integer, we denote by $\mathbb{Z}_p[X_1, \dots, X_m]$ the set of m -variate polynomials with coefficients in $\mathbb{Z}_p$, and by $\mathbb{Z}_p(X_1, \dots, X_m)$ the set of rational fractions in m variables with coefficients in $\mathbb{Z}_p$. For a rational fraction $R \in \mathbb{Z}_p(X_1, \dots, X_m)$, we denote its numerator by $\hat{R}$ and its denominator by $\check{R}$. That is, $\hat{R}, \check{R} \in \mathbb{Z}_p[X_1, \dots, X_m]$ are such that $R = \hat{R}/\check{R}$.

2.1 Bilinear Groups and Assumptions

Definition 1 (Bilinear Groups). A bilinear group consists of three cyclic groups $(\mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T)$ of prime order p , together with a bilinear map (or pairing) $e : \mathbb{G}_1 \times \hat{\mathbb{G}}_2 \rightarrow \mathbb{G}_T$ that satisfies the following properties:

- *Bilinearity*: For all $g_1 \in \mathbb{G}_1$, $\hat{g}_2 \in \hat{\mathbb{G}}_2$, and $a, b \in \mathbb{Z}_p$, $e(g_1^a, \hat{g}_2^b) = e(g_1, \hat{g}_2)^{ab}$.
- *Non-degeneracy*: $e(g_1, \hat{g}_2) \neq 1$ for some generators $g_1 \in \mathbb{G}_1$, $\hat{g}_2 \in \hat{\mathbb{G}}_2$.
- *Efficiency*: The map e can be computed efficiently.

We usually denote a bilinear group setup as $(p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$, where $g_1 \in \mathbb{G}_1$, $\hat{g}_2 \in \hat{\mathbb{G}}_2$ are the respective generators.

We distinguish between 3 types of bilinear-group generators. *Type-I* groups have $\mathbb{G}_1 = \hat{\mathbb{G}}_2$ and are called *symmetric* bilinear groups. *Type-II* and *Type-III* groups, which are referred to as *asymmetric* bilinear groups, have $\mathbb{G}_1 \neq \hat{\mathbb{G}}_2$. In *Type-II* there is an efficiently computable homomorphism $\psi : \hat{\mathbb{G}}_2 \rightarrow \mathbb{G}_1$, while in *Type-III* there are no efficiently computable homomorphisms in either direction. We use $\text{BG} = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2) \leftarrow \text{GroupGen}(1^\lambda)$ to denote a bilinear group generator for asymmetric *Type-III* bilinear groups, where p is a prime of λ bits.

Structure Preserving Cryptography. Structure-Preserving (SP) [AFG⁺10, Abe15] cryptography refers to a class of cryptographic schemes where public inputs and outputs (e.g., messages, public keys, proofs) are represented as elements of algebraic groups, and all operations (e.g., encryption, signing, verification) are limited to group operations and pairings. These schemes are defined over bilinear groups [BF01], which are expressive enough to realize powerful cryptographic primitives while preserving compatibility with algebraic proof systems, such as Groth-Sahai [GS08]. A cryptographic scheme in the bilinear group setting is called *structure-preserving* if all public inputs and outputs consist of elements of the source groups $\mathbb{G}_1$ and $\hat{\mathbb{G}}_2$ of a bilinear group $(p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$, and correctness can be verified only by testing group membership, computing group operations, and evaluating pairing product equations (PPEs) [GS08] of the form

$$\prod_i \prod_j e(A_i, \hat{B}_j)^{c_{i,j}} = 1,$$

where $A_i \in \mathbb{G}_1$, $\hat{B}_j \in \hat{\mathbb{G}}_2$ are group elements, and $c_{i,j} \in \mathbb{Z}_p$ are constants.

Algebraic Group Model (AGM). The security of our schemes is proven in the algebraic group model [FKL18] that lies between the standard model and the widely used generic group model [Sho97]. In the AGM, we consider only so-called algebraic adversaries. Such adversaries have direct access to group elements and, in particular, can use their bit representation, as in the standard model. However, they are assumed to output new group elements only by applying the group operation to received group elements. This requirement is formalized as follows. Suppose an adversary is given group elements $X_1, \dots, X_n \in \mathbb{G}$. Then, for every group element $Z \in \mathbb{G}$ that the adversary outputs, it must also output an algebraic explanation $\{z_i\}_{i \in [n]}$ such that $Z = \prod_{i=1}^n X_i^{z_i}$.

Assumptions. The security of our constructions is based on the (q_1, q_2) -Power Discrete Logarithm Assumption ((q_1, q_2) -PDL) [Lip12] that we recall below. Informally, this is an extension of the classical discrete logarithm assumption in

bilinear groups, in the case where the adversary receives powers of the secret discrete logarithm in the source groups.

Definition 2 ((q_1, q_2)-Power Discrete Logarithm Assumption). Let $\text{BG} = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$ be a bilinear group generated by $\text{GroupGen}(1^\lambda)$. For integers $q_1, q_2 \in \mathbb{Z}_p$, the (q_1, q_2) -PDL problem is: for a random $\alpha \leftarrow \mathbb{Z}_p$, given

$$(g_1, g_1^\alpha, g_1^{\alpha^2}, \dots, g_1^{\alpha^{q_1}}, \hat{g}_2, \hat{g}_2^\alpha, \hat{g}_2^{\alpha^2}, \dots, \hat{g}_2^{\alpha^{q_2}}), \quad (1)$$

compute α . The (q_1, q_2) -PDL assumption states that any PPT adversary has probability $\text{negl}(\lambda)$ in solving the (q_1, q_2) -PDL problem.

Finally, we recall the Diffie-Hellman Exponent assumption [BBG05, CKS09] that was used to argue the security of the SPVC and SPA in [KMS26].

Definition 3 (The Diffie-Hellman Exponent Assumption (q-DHE)). Let $\text{BG} = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$ be a bilinear group generated by $\text{GroupGen}(1^\lambda)$. The q -DHE problem is: for a random $\alpha \leftarrow \mathbb{Z}_p$, given a tuple of elements

$$\left(\text{BG}; B_1 = g_1^\alpha, \dots, B_q = g_1^{\alpha^q}, B_{q+2} = g_1^{\alpha^{q+2}}, \dots, B_{2q} = g_1^{\alpha^{2q}}; \begin{matrix} \hat{B}_1 = \hat{g}_2^\alpha, \dots, \hat{B}_q = \hat{g}_2^{\alpha^q} \end{matrix} \right), \quad (2)$$

compute the missing group element $B_{q+1} = g_1^{\alpha^{q+1}}$. The q -DHE assumption states that any PPT adversary has probability $\text{negl}(\lambda)$ in solving the q -DHE problem.

2.2 Vector Commitments

We recall the notion of *vector commitments* following the definitions in [CF13].

Definition 4 (Vector Commitment). A vector commitment is a tuple of efficient PPT algorithms $\text{VC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify})$ together with a message space $\mathcal{M}$:

Setup($1^\lambda, q$): Given the security parameter λ and the size q of the committed vector (with $q = \text{poly}(\lambda)$), the key generation outputs some public parameters pp (which implicitly define a message space $\mathcal{M}$).

Commit($\text{pp}, m_1, \dots, m_q$): On input a sequence of q messages $m_1, \dots, m_q \in \mathcal{M}$ and the public parameters pp , the committing algorithm outputs a commitment string com and an auxiliary information aux .

Open($\text{pp}, m, i, \text{aux}$): This algorithm is run by the committer to produce a proof π_i that m is the i -th committed message.

Verify($\text{pp}, \text{com}, m, i, \pi_i$): The verification algorithm accepts (i.e. it outputs 1) only if π is a valid proof that com was created to a sequence $m_1, \dots, m_q \in \mathcal{M}$ such that $m = m_i$.

Definition 5 (Correctness). A vector commitment is correct, if for any $\lambda \in \mathbb{N}$, any parameter q , any honestly generated parameters $\text{pp} \leftarrow \text{Setup}(1^\lambda, q)$, any vector $(m_1, \dots, m_q) \in \mathcal{M}^q$, and any element $i \in [q]$, it holds that:

$$\Pr \left[\text{Verify}(\text{pp}, \text{com}, m_i, i, \pi_i) = 1 \mid \begin{matrix} (\text{com}, \text{aux}) \leftarrow \text{Commit}(\text{pp}, m_1, \dots, m_q); \\ \pi_i \leftarrow \text{Open}(\text{pp}, m, i, \text{aux}) \end{matrix} \right] = 1$$

Vector commitments are required to meet a security notion named *position binding*. Informally, this means that it should be infeasible for any polynomially bounded adversary, given knowledge of pp , to produce a commitment com together with two distinct valid openings for the same position i . More formally:

Definition 6 (Position Binding). *A vector commitment satisfies position binding if, for any integer $q \geq 1$, and for every PPT adversary $\mathcal{A}$, the following probability is at most negligible in λ :*

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{pp}, \text{com}, m, i, \pi) = 1 \wedge \\ \text{Verify}(\text{pp}, \text{com}, m', i, \pi') = 1 \wedge \\ m \neq m' \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, q) \\ (\text{com}, m, m', i, \pi, \pi') \leftarrow \mathcal{A}(\text{pp}) \end{array} \right] \leq \text{negl}(\lambda)$$

2.3 Accumulators

We recall the notion of *accumulators*. Precisely, we focus on *static accumulators* following the definitions from [DHS15].

Definition 7 (Static Accumulator). *A static accumulator is a tuple of efficient PPT algorithms $\text{Acc} = (\text{Setup}, \text{Eval}, \text{WitCreate}, \text{Verify})$ together with a message space $\mathcal{M}$:*

- $\text{Setup}(1^\lambda, q)$: Takes a security parameter λ and a parameter q . If $q \neq \infty$, then q is an upper bound on the number of elements to be accumulated. Returns public parameters pp .
- $\text{Eval}(\text{pp}, \mathcal{X})$: This (probabilistic) algorithm takes public parameters pp and a set $\mathcal{X} \subseteq \mathcal{M}$ to be accumulated and returns an accumulator $\text{Acc}_{\mathcal{X}}$ together with some auxiliary information aux .
- $\text{WitCreate}(\text{pp}, \text{Acc}_{\mathcal{X}}, x, \text{aux})$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, auxiliary information aux , and a value $x \in \mathcal{M}$. It returns $\perp$, if $x \notin \mathcal{X}$, and a witness wit_x for x otherwise.
- $\text{Verify}(\text{pp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x)$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, a witness wit_x , and a value $x \in \mathcal{M}$. It returns true if wit_x is a witness for $x \in \mathcal{X}$ and false otherwise.

Definition 8 (Correctness of a Static Accumulator). *We say that a static accumulator is correct, if for any $\lambda \in \mathbb{N}$, any parameter q , any set $\mathcal{X} \subseteq \mathcal{M}$, and any element $x \in \mathcal{X}$, it holds that:*

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{pp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x) = 1 \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, q); \\ (\text{Acc}_{\mathcal{X}}, \text{aux}) \leftarrow \text{Eval}(\text{pp}, \mathcal{X}); \\ \text{wit}_x \leftarrow \text{WitCreate}(\text{pp}, \text{Acc}_{\mathcal{X}}, \text{aux}, x) \end{array} \right] = 1$$

In terms of security an accumulator must satisfy **collision resistance** [BdM94] which states that it should be computationally infeasible to find a witness for any non-accumulated value.

Definition 9 (Collision Resistance). *An accumulator scheme satisfies collision resistance if for all PPT adversaries $\mathcal{A}$ and security parameter λ , it holds*

$$\Pr \left[\begin{array}{l} (\text{Acc}_{\mathcal{X}}, \text{aux}) = \text{Eval}(\text{pp}, \mathcal{X}; r) \quad \wedge \\ \text{Verify}(\text{pp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x) = 1 \quad \wedge \\ x \notin \mathcal{X} \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, q); \\ (\mathcal{X}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x, r) \leftarrow \mathcal{A}(\text{pp}) \end{array} \right] \leq \text{negl}(\lambda)$$

Note that this notion essentially models security for accumulators $\text{Acc}_{\mathcal{X}}$ that validly accumulate a set $\mathcal{X}$. Here the use of aux allows the adversary to fully choose the accumulator $\text{Acc}_{\mathcal{X}}$ though it should make it recomputable, as noted earlier in the correctness definition (aux can be empty for deterministic Eval).

We only focus on static accumulators, since our application to ring signatures does not require a dynamic setting. In contrast, dynamic accumulators can publicly update the accumulated value and witness whenever an element is added to the set $\mathcal{X}$. There are many properties associated to this primitives, but we are only going to consider the basic properties that we have mentioned. For more definitions, results and properties we refer to [DHS15].

3 Security Analysis of the SPVC and SPA in [KMS26]

3.1 The SPVC construction of [KMS26]

We start by recalling the SPVC scheme from [KMS26]. The foundation of this scheme is the definition of a message space that, instead of including (vectors of) arbitrary group elements, includes messages satisfying a specific structure w.r.t. q -DHE parameters, which form the public parameters of the SPVC. This approach is what allows the authors to circumvent the known impossibility results.

More in detail, given bilinear groups $\text{BG} = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2) \leftarrow \text{GroupGen}(1^\lambda)$ for a security parameter $\lambda \in \mathbb{N}$, and given an integer q denoting the length of the vectors to be committed, the $\text{Setup}(1^\lambda, q)$ algorithm samples $\alpha \leftarrow \mathbb{Z}_p$ and produces

$$\text{pp} = \left(\begin{array}{l} \text{BG}; B_1 = g_1^\alpha, \dots, B_q = g_1^{\alpha^q}, B_{q+2} = g_1^{\alpha^{q+2}}, \dots, B_{2q} = g_1^{\alpha^{2q}}; \\ \hat{B}_1 = \hat{g}_2^\alpha, \dots, \hat{B}_q = \hat{g}_2^{\alpha^q} \end{array} \right) \quad (3)$$

Note that pp does not include $B_{q+1} = g_1^{\alpha^{q+1}}$, a crucial fact for the security of the scheme, since computing it constitutes a break of the q -DHE assumption.

Then, given pp , [KMS26] defines the set $\mathcal{M}_q^{\text{pp}}$ of admissible vectors to be committed as follows (cited almost verbatim).

Message Structure [KMS26]. Each message consists of vectors with $q + 1$ elements over $\mathbb{G}_1$, where each vector includes a main message component $\mathbf{M}$ of length 2, as well as a tag vector $\mathbf{T}$ with $q - 1$ elements. The structure of the vector and the definition of the message space $\mathcal{M}_q^{\text{pp}}$ are determined by the q -DHE parameters pp (see (3)). The message space $\mathcal{M}_q^{\text{pp}}$ is defined according to

these parameters:

$$\mathcal{M}_q^{\text{pp}} = \left\{ ((\mathbf{M}_1, \mathbf{T}_1), \dots, (\mathbf{M}_q, \mathbf{T}_q)) \mid \exists \mathbf{m} = (m_1, \dots, m_q) \in \mathbb{Z}_p^q \text{ s.t. } \forall i \in [q], \right. \\ \left. \text{msg}_i = (M_{i0} = g_1^{m_i}, M_{i1} = B_i^{m_i} = g_1^{\alpha^i \cdot m_i}, T_{ij} = B_j^{m_i} = g_1^{\alpha^j \cdot m_i}) \right\} \\ \text{where } j \in [i+1, q+i] \setminus \{q+1\} \quad (4)$$

In $\mathcal{M}_q^{\text{pp}}$, while messages and tags are in the group $\mathbb{G}_1$, the set of bases in $\mathbb{G}_2$, meaning $\hat{B} = \{\hat{B}_1, \dots, \hat{B}_q\}$, can be used to verify whether a vector is in the message space. Specifically, [KMS26] defines a routine **VerifyMsg** that checks the following pairing equations to ensure the relationships defined above:

$$\forall j \in [i+1, i+q] \setminus \{q+1\} : e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{g}_2) \wedge e(M_{i1}, \hat{g}_2) = e(M_{i0}, \hat{B}_i)$$

SPVC schemes. The authors of [KMS26] propose two SPVC schemes for committing to vectors in $\mathcal{M}_q^{\text{pp}}$. For ease of reference, we call these schemes SPVC_0 and SPVC_1 . The schemes proceed identically except for the verification algorithms. SPVC_1 is their main scheme and assumes a verifier who only receives the message component $\mathbf{M}_i = (M_{i0}, M_{i1})$. SPVC_0 instead assumes a verifier who receives the full message $\text{msg}_i = (\mathbf{M}_i, \mathbf{T}_i)$, including its vector of tags $\mathbf{T}_i$, and additionally checks if $\text{VerifyMsg}(\text{pp}, \text{msg}_i, i) = 1$. Clearly, SPVC_0 is less efficient since it requires the verifier to do at least $O(q)$ work and to receive $O(q)$ -long information. In [KMS26], SPVC_0 is mainly introduced as an alternative strategy for proving binding, relying in the GGM (see [KMS26, Remark 2] and [KMS, Appendix C.1]).

Scheme 1 (q-DHE SPVCs [KMS26]) *Consider the following algorithms.*

Setup($1^\lambda, q$): *Given the security parameter $\lambda \in \mathbb{N}$ and the size q of the committed vector, pick $\alpha \leftarrow \mathbb{Z}_p$ and output public parameters pp as in equation (2).*

Commit($\text{pp}, \text{msg}_1, \dots, \text{msg}_q$): *On input a vector of q messages $(\text{msg}_1, \dots, \text{msg}_q) \in \mathcal{M}_q^{\text{pp}}$, such that each $\text{msg}_i = (\mathbf{M} = (M_{i0}, M_{i1}), \mathbf{T}_i)$, check correctness of the messages if $1 = \text{VerifyMsg}(\text{pp}, \text{msg}_i, i)$ for all $i \in [q]$. Pick $r \leftarrow \mathbb{Z}_p$ and output:*

$$\text{com} = \left(g_1^r \cdot \prod_{i \in [q]} M_{i1} \right) \quad \wedge \quad \text{aux} = (r, \mathbf{T})$$

Open($\text{pp}, \text{msg}, i, \text{aux}$): *This algorithm is run by the committer to produce a proof π_i that msg is the i -th committed message:*

$$\pi_i = B_{q+1-i}^r \cdot \prod_{j \in [q] \setminus \{i\}} T_{j, q+1-i+j}$$

Verify₀($\text{pp}, \text{com}, \text{msg}, i, \pi_i$): *This verification algorithm accepts (i.e. it outputs 1) only if msg is a valid message for index i (i.e. $1 \leftarrow \text{VerifyMsg}(\text{pp}, \text{msg}, i)$) and π_i is a valid proof that com was created for $\text{msg} = (M_{i0}, M_{i1}, \mathbf{T} = (T_{ij}))$:*

$$e(\text{com}, \hat{B}_{q+1-i}) = e(\pi_i, \hat{g}_2) \cdot e(M_{i1}, \hat{B}_{q+1-i})$$

$\text{Verify}_1(\text{pp}, \text{com}, \mathbf{M}, i, \pi_i)$: This verification algorithm accepts (i.e. it outputs 1) only if π_i is a valid proof that com was created for $\mathbf{M} = (M_{i0}, M_{i1})$:

$$e(\text{com}, \hat{B}_{q+1-i}) = e(\pi_i, \hat{g}_2) \cdot e(M_{i1}, \hat{B}_{q+1-i}) \quad (5)$$

$\text{VerifyMsg}(\text{pp}, \text{msg}_i, i)$: Takes $\text{msg}_i = (\mathbf{M} = (M_{i0}, M_{i1}), \mathbf{T} = (T_{ij}))$ and an index $i \in [q]$. Verifies whether it is well-formed message with respect to pp . It returns 1 if the following equations hold, and 0 otherwise.

$$e(M_{i0}, \hat{B}_i) = e(M_{i1}, \hat{g}_2) \quad \wedge \quad e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{g}_2)$$

where $j \in [i+1, q+i] \setminus \{q+1\}$.

We define the SPVC schemes $\text{SPVC}_0 = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}_0)$ and $\text{SPVC}_1 = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}_1)$.

In [KMS26], the authors claim the position binding of SPVC_0 in the generic group model and the position binding of the more efficient SPVC_1 under the q -DHE assumption in the algebraic group model.

The main idea of the GGM-based proof for SPVC_0 is that any adversary who produces a message msg that passes the $\text{VerifyMsg}(\text{pp}, \text{msg}, i)$ procedure must know the discrete logarithm $m_i = \text{dlog}_{B_i} M_{i1} = \text{dlog}_{B_j} T_{ij}$. In the GGM one can therefore extract these values and then do a reduction to q -DHE in a way similar to the original, non-structure-preserving, VC of [LY10].

3.2 Security analysis of the scheme SPVC_1

For the scheme SPVC_1 , the authors emphasize that the tag components are not required during verification but are only used during proof generation and included as auxiliary data. This feature is particularly advantageous in vector commitment and accumulator applications, as it enables efficient verification. The authors thus provide a security proof of the position binding of SPVC_1 , under the q -DHE assumption in the AGM (cf. [KMS26, Theorem 2]).

Our first result is an attack, described below, against the position binding notion on which this vector commitment is intended to rely.

Attack 1 Let SPVC_1 be the SPVC scheme presented in Scheme 1, where the verifier checks the equation (5). There is an efficient adversary that breaks the position binding property of SPVC_1 with probability 1.

Proof. We can build an adversary $\mathcal{A}$ that efficiently provides an honest commitment to a vector, and at the same time is able to open it to two distinct messages at the same position.

$\mathcal{A}(\text{pp})$ chooses an arbitrary vector $(m_1, \dots, m_q) \in \mathbb{Z}_p^q$, computes honestly the vector $(\text{msg}_i)_{i \in [q]} \in \mathcal{M}_q^{\text{pp}}$ according to (4), and commits to it by computing

$$\text{com} = g_1^r \prod_{i \in [q]} M_{i1}.$$

Next $\mathcal{A}$ chooses an arbitrary value $m \in \mathbb{Z}_p$ such that $m \neq m_i$ for every $i \in [q]$. Then, for any position $i \in [q]$, $\mathcal{A}$ can first compute an honest opening $\pi_i \leftarrow \text{Open}(\text{pp}, \text{msg}_i, i, \text{aux})$, and then it can generate a valid opening for the message $\mathbf{M}^* = (M_0^* = g_1^m, M_1^* = \prod_{i \in [q]} M_{i1})$, by setting $\pi_i^* = B_{q+1-i}^r$. $\mathcal{A}$ returns $(\text{com}, \mathbf{M}_i, \mathbf{M}^*, \pi_i, \pi_i^*)$, where $\mathbf{M}_i = (M_{i0}, M_{i1})$ from msg_i .

Clearly, $\mathbf{M}^* \neq \mathbf{M}_i$ by construction, and $\text{Verify}_1(\text{pp}, \text{com}, \mathbf{M}_i, i, \pi_i) = 1$ by correctness. Also, $\text{Verify}_1(\text{pp}, \text{com}, \mathbf{M}^*, i, \pi_i^*) = 1$ accepts since

$$\underbrace{e(g_1^r \prod_{i \in [q]} M_{i1}, \hat{B}_{q+1-i})}_{e(\text{com}, \hat{B}_{q+1-i})} = \underbrace{e(B_{q+1-i}^r, \hat{g}_2)}_{e(\pi_i^*, \hat{g}_2)} \cdot \underbrace{e(\prod_{i \in [q]} M_{i1}, \hat{B}_{q+1-i})}_{e(M_{i1}^*, \hat{B}_{q+1-i})}.$$

Namely, equation (5) is satisfied. $\square$

Remark 1 (On the security proof of [KMS26, Theorem 2]). Given the attack described above, it can be useful to observe where the security proof presented in [KMS26, Theorem 2] is inconsistent. Let $\mathbf{m}^*$ and $\mathbf{m}$ be the algebraic representations of messages M_{i1}^* and M_{i1} , according to Attack 1. The polynomial representations of these elements are of the form $P_{\mathbf{m}}(X) = \sum_{i \in [2q] \setminus \{q+1\}} m_i X^i$. In the AGM binding proof for the SPVC in [KMS26] (SPVC₁ in our notation), it is assumed $m_i - m_i^* \neq 0$, where m_i and m_i^* correspond to the coefficient of X^i in the polynomial representations of the opened messages M_{i1} and M_{i1}^* , respectively. This condition is crucial for the reduction to break the q-DHE assumption, since it needs to compute $1/(m_i - m_i^*)$. In Attack 1, the representations however satisfy $m_i = m_i^*$, contradicting their assumption.

The attack shows that if the verifier does not check any structure over the opened message, the scheme cannot ensure binding guarantees. Based on this observations, one may see that Attack 1 can be immediately avoided by adding an additional equation that checks the supposed relation between the components of a message $\mathbf{M} = (M_{i0}, M_{i1})$, namely if one replaces the check of equation (5) by the following check:

$$e(C, \hat{B}_{q+1-i}) = e(\pi, \hat{g}_2) e(M_{i1}, \hat{B}_{q+1-i}) \quad \wedge \quad e(M_{i0}, \hat{B}_i) = e(M_{i1}, \hat{g}_2). \quad (6)$$

Unfortunately, this addition is not enough to make the scheme position binding.

Attack 2 Let SPVC'₁ be the SPVC scheme presented in Scheme 1, modified so that Verify_1 checks the equations (6). There is an efficient adversary that breaks the position binding property of SPVC'₁ with probability 1.

Proof. We can build an adversary $\mathcal{A}$ that efficiently provides an honest commitment to a vector, and at the same time is able to open it to two distinct messages at the same position.

$\mathcal{A}(\text{pp})$ chooses an index $k \in [q]$ such that $2k < q + 1$ as well as two values $m_1, m_2 \in \mathbb{Z}_p$ with $m_1 \neq m_2$. The adversary then sets

$$\mathbf{M}_k = (g_1^{m_1+m_2}, B_{2k}^{m_1+m_2}) \quad \wedge \quad \text{com} = B_{2k}^{m_1+m_2},$$

Note that by this choice we are implicitly committing to a vector $(\text{msg}_j)_{j \in [q]} \in \mathcal{M}_q^{\text{pp}}$ where $\text{msg}_i = (1_{G_1}, \dots, 1_{G_1}) \forall i \neq 2k$.

Next, $\mathcal{A}$ generates an opening for the message $\mathbf{M}^* = (B_k^{m_1}, B_{2k}^{m_1})$ at position k by setting $\pi^* = B_{q+1+k}^{m_2}$. Analogously, it generates an opening for the message $\mathbf{M}^{**} = (B_k^{m_2}, B_{2k}^{m_2})$ at the same position k by setting $\pi^{**} = B_{q+1+k}^{m_1}$.

By construction, $m_1 \neq m_2$ and thus the two messages are different, $\mathbf{M}^* \neq \mathbf{M}^{**}$. Yet both openings are accepted by Verify_1 : for $(\mathbf{M}^*, \pi^*)$ we have

$$\begin{aligned} \underbrace{e(B_{2k}^{m_1+m_2}, \hat{B}_{q+1-k})}_{e(\text{com}, \hat{B}_{q+1-k})} &= \underbrace{e(B_{q+1+k}^{m_2}, \hat{g}_2)}_{e(\pi^*, \hat{g}_2)} \cdot \underbrace{e(B_{2k}^{m_1}, \hat{B}_{q+1-k})}_{e(M_1^*, \hat{B}_{q+1-k})} \\ &\wedge \underbrace{e(B_k^{m_1}, \hat{B}_k)}_{M_0^*} = \underbrace{e(B_{2k}^{m_1}, \hat{g}_2)}_{M_1^*} \end{aligned}$$

i.e., equation (6) is verified, and analogously it holds for $(\mathbf{M}^{**}, \pi^{**})$. $\square$

We emphasize that our Attack 2 can be avoided by checking correctness of the additional tags over the messages, as it is done in SPVC_0 and argued in the GGM in [KMS26]. Unfortunately, adding the additional tags to verification yields an $O(q)$ cost for the verifier and it can be highly problematic in applications since it likely introduces an $O(q)$ cost to communicate this information to the verifier, defeating the succinctness motivation.

This attack also illustrates that if we do not provide additional information related to the tags of the messages in order to validate correctness over the structured message space, the pairing equations presented in (6), are not sufficient to provide binding guarantees over the vector commitment.

3.3 The SPA Construction of [KMS26]

We now turn our attention to the SP accumulator proposed in [KMS26].

SP Accumulators from SP Vector Commitments. Catalano and Fiore [CF13] proposed a black-box construction of accumulators for sets in a polynomial-size universe, based on vector commitments. Unfortunately, their compiler cannot be directly adapted to the structure-preserving setting, as pointed out in [KMS26]. This is due to the index-based structure of the compiler, which is inherently incompatible with a structure-preserving message space.

The authors of [KMS26] propose another generic compiler showing that any SPVC can be used to construct a SPA. The construction works as follows: **Setup** samples the SPVC public parameters pp and q pairwise distinct group elements $(U_1, \dots, U_q)$, setting $\text{pp}_{\text{Acc}} = (\text{pp}, (U_1, \dots, U_q))$. To accumulate a set $\mathcal{X}$, **Eval** defines a mapping of the elements in $\mathcal{X}$ to indices in $[q]$, padding $\mathcal{X}$ with random elements if $|\mathcal{X}| \leq q$, and obtains Acc by running **Commit**. A witness for x is computed by **WitCreate** as $\text{wit}_x = (U_{i_x}, \pi_x)$, where i_x is the index corresponding to x and π_x is the output of **Open**. Finally, the verification process retrieves the index corresponding to U_{i_x} , and outputs the result of $\text{Verify}(\text{pp}, \text{Acc}_{\mathcal{X}}, x, \pi_x, i_x)$.

This generic construction achieves a somewhat different security notion called *bounded soundness* (see [KMS26, Definition 4]), where the accumulator is fully controlled by the adversary, who is considered to win if it is able to generate an accumulator and openings to more than q distinct values. If the adversary can output $q + 1$ values and valid witnesses for a fixed accumulator value Acc , then at least two witnesses $(\text{wit}_x, \text{wit}_{x'})$ for $x \neq x'$ must contain the same index $i_x = i_{x'}$. Therefore, the corresponding proofs π_x and $\pi_{x'}$, are valid openings for the same position, immediately breaking the position binding of the SPVC.

Given our results of Section 3.1, this compiler can only be instantiated with SPVC_0 , since SPVC_1 does not achieve sufficient binding guarantees. The resulting SPA achieves $O(q)$ opening size and $O(q)$ verification time. Furthermore, a verifier algorithm implemented in a structure-preserving manner should determine which element of the crs corresponds to the U_{i_x} in wit_x , which would require $O(q)$ work (even when instantiated with SPVC_1). This becomes a challenging issue in privacy-preserving applications where one may want to prove verification in zero-knowledge to hide the index i_x , while maintaining succinctness.

The q -DHE SPA of [KMS26] To overcome the challenges mentioned above, Kren et al. [KMS26] introduce a new SPA, based on q -DHE. Similarly to the generic scheme, the main idea behind their construction is to realize an accumulator via a VC; this requires choosing some ordering of the set elements, placing them in a vector and committing to this vector. The straightforward way to prove membership in this construction would require to reveal to the verifier the index of the element and then provide the corresponding VC opening proof. Even if one starts from an SPVC, this approach fails in maintaining the structure preserving property since the membership proof would now include an integer.

The authors of [KMS26] define their SPA for sets represented as vectors over the vector message space $\mathcal{M}_q^{\text{pp}}$, and use the techniques of their vector commitment SPVC_1 (see Scheme 1). To avoid revealing the position in the membership proof, they introduce an additional proof element $\hat{W}_2 = \hat{B}_{q+1-i}$, which is plugged into a slight modification of the SPVC_1 verification equation (5) as follows:

$$e(\text{com}, \hat{W}_2) = e(\pi_i, \hat{g}_2) \cdot e(M_{i1}, \hat{W}_2),$$

without explicitly revealing the index i . This accumulator achieves constant size openings and verification time. Moreover, to support usage in privacy-enhancing environments such as ring signatures, proofs are randomized so that openings do not leak information about the index.

Below we recall the accumulator following the same notation as in [KMS26].

Scheme 2 (q-DHE SP Accumulator [KMS26]) *The scheme is as follows:*

Setup($1^\lambda, q$): Given the security parameter $\lambda \in \mathbb{N}$ and a parameter q , picks $\alpha \leftarrow \mathbb{Z}_p$ and computes some public parameters:

$$\text{pp} = \left(\text{BG}; B_1 = g_1^\alpha, \dots, B_q = g_1^{\alpha^q}, B_{q+2} = g_1^{\alpha^{q+2}}, \dots, B_{2q} = g_1^{\alpha^{2q}}; \begin{matrix} \hat{B}_1 = \hat{g}_2^\alpha, \dots, \hat{B}_q = \hat{g}_2^{\alpha^q} \end{matrix} \right) \quad (7)$$

Finally, sets $\text{pk}_{\text{acc}} = \text{pp}$ and $\text{sk}_{\text{acc}} = \perp$.

$\text{Eval}(\text{pk}_{acc}, \mathcal{X})$: Given public parameters pp and set $\mathcal{X}$ it returns an accumulator $\text{Acc}_{\mathcal{X}}$ together with auxiliary information aux as follows:

- Check $\forall i$ if $1 \leftarrow \text{VerifyMsg}(\text{pp}, \text{msg}_i, i)$ for all $\text{msg}_i = (M_{i0}, M_{i1}, \mathbf{T}_i) \in \mathcal{X}$.
- For a random $r \leftarrow \mathbb{Z}_p$, compute:

$$\text{Acc}_{\mathcal{X}} = g_1^r \cdot \prod_{i \in [q]} M_{i1} \quad \wedge \quad \text{aux} = (r, \mathbf{T})$$

$\text{WitCreate}(\text{pk}_{acc}, \text{Acc}_{\mathcal{X}}, \text{msg}_i, \text{aux})$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, auxiliary information aux , and a value msg_i . It returns $\perp$, if $\text{msg}_i \notin \mathcal{X}$ or $0 \leftarrow \text{VerifyMsg}(\text{pp}, \text{msg}_i)$, otherwise, computes a witness $\text{wit}_{\text{msg}_i}$ for msg_i : Pick $\rho \leftarrow \mathbb{Z}_p^*$, and compute:

$$W_1 = \left(B_{q+1-i}^r \cdot \prod_{j \in [q] \setminus \{i\}} T_{j, q+1-i+j} \right)^\rho \quad \wedge \quad \hat{W}_2 = \left(\hat{B}_{q+1-i}^\rho \right) = \left(\hat{g}_2^{\alpha^{q+1-i}} \right)^\rho \quad (8)$$

$\text{Verify}(\text{pk}_{acc}, \text{Acc}_{\mathcal{X}}, \text{wit}_{\text{msg}_i}, \mathbf{M})$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, a witness $\text{wit}_{\text{msg}_i}$, and a value $\mathbf{M} = (M_{i0}, M_{i1})$. It returns true (i.e., it outputs 1) if $\text{wit}_{\text{msg}_i}$ is a valid witness for $\mathbf{M} \in \mathcal{X}$ and false (i.e., it outputs 0) otherwise:

$$e(\text{Acc}_{\mathcal{X}}, \hat{W}_2) = e(W_1, \hat{g}_2) \cdot e(M_{i1}, \hat{W}_2) \quad (9)$$

3.4 Security Analysis of the q -DHE SPA from [KMS26]

To prove security of the accumulator, in [KMS26] they prove the following theorem (almost verbatim as Theorem 3 in [KMS26]).

Theorem 1 (q-DHE SPA Bounded Soundness ([KMS26])). *If SPVC_1 is position-binding, then the q -DHE Accumulator (Scheme 2) satisfies for all PPT adversaries $\mathcal{A}$ the following probability is negligible for every $\mathcal{X} = \{(\mathbf{M}_i, \mathbf{T}_i)\}_{i=1}^q \in \mathcal{M}_q^{\text{pp}}$:*

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, q), ((r, \rho), (\mathbf{M}', \text{wit}'), \text{Acc}) \leftarrow \mathcal{A}(\text{pp}, \mathcal{X}) : \\ \text{Verify}(\text{pp}, \text{Acc}, \text{wit}', \mathbf{M}') = 1 \wedge \forall i \in [q] : M'_1 \neq M_{i1} \wedge \\ \text{Acc} = g_1^r \cdot \prod_{i \in [q]} M_{i1} \text{ and } \exists n' \in [q] : \hat{W}'_2 = \hat{B}_{n'}^\rho \end{array} \right],$$

where $\mathbf{M}' = (M'_0, M'_1)$ and $\text{wit}' = (W'_1, \hat{W}'_2)$.

On the security notion of Theorem 1. The security notion claimed by this theorem is weaker than the standard collision resistance notion of accumulators (see Definition 9) in two main ways. First, it considers adversaries that are non-adaptive in the choice of the set $\mathcal{X}$ since this is given. Second, the adversary must reveal the randomness ρ used in the computation of the witness $\hat{W}_2$. This additional requirement allows the challenger to check that $\hat{W}_2$ is correctly

formed, i.e., that $\hat{W}_2 = \hat{B}_{n'}^\rho$, for some $n' \in [q]$. This requirement follows from the fact that, if $\hat{W}_2$ is not guaranteed to be the randomization of one of the CRS elements, then known attacks against Scheme 2 become possible [dS25].

Also, note that, in order to avoid verifying the correctness of $\hat{W}_2$, one might consider sending it in the clear and requiring the verifier to check that it is indeed an element of the CRS. However, this choice would defeat the goal of achieving constant verification time and would moreover be incompatible with the privacy-enhancing constructions envisioned by the authors.

Insecurity of the SPA. Theorem 1 relies on assuming the position binding of SPVC_1 . However, as we have seen in Section 3.2, this assumption does not hold. The invalidity of this assumption (i.e., an attack against SPVC_1) might not immediately yield an explicit attack against the SPA. This is because an adversary against SPVC_1 may use the ability of being adaptive in the choice of the vector whereas the SPA attacker against Theorem 1 receives the set as an input and may not know the discrete logarithm of those group elements. As a concrete example, the adversary of Attack 2 is indeed adaptive in the choice of the vector and would not yield an adversary meeting the notion of Theorem 1.

On the other hand, one can check that the adversary of Attack 1 actually works by choosing *any* vector of group elements in the structured message space, and it does not need to know the discrete logarithms of these elements. It is not hard to see that this adversary can therefore be adapted to work as an adversary against Theorem 1. For completeness, in Appendix A, we present this adaption of Attack 1 and a more direct attack against the SPA.

These attacks can be avoided in a similar way as Attack 1 on SPVC_1 is avoided, i.e., by adding the verification equation (6). Yet, after this change it would remain to prove security under the notion in Theorem 1, which cannot rely on the position binding of SPVC_1 's variant in which equations (6) are checked during verification, since this variant is also not binding due to Attack 2.

An option to make this SPA secure might be to include in the membership proof all the tags related to the element and to require verification of the full message, as in SPVC_0 . However, this approach would undermine the main goal of achieving $O(1)$ -size openings and, consequently, $O(1)$ verification time.

4 Our Structure-Preserving Vector Commitment

In this section, we present our new SPVC scheme, dubbed SPVC_2 . Inspired to [KMS26], SPVC_2 works with a structured message space parametrized by its public parameters pp . More in detail, in our scheme Setup generates

$$\text{pp} = (\text{BG}; B_1 = g_1^\alpha, \dots, B_q = g_1^{\alpha^q}; \hat{B}_1 = \hat{g}_2^\alpha, \dots, \hat{B}_q = \hat{g}_2^{\alpha^q}) \quad (10)$$

that, in comparison to those of [KMS26], do not include the elements $g_1^{\alpha^{q+2}}, \dots, g_1^{\alpha^{2q}}$.

Given these pp , our structured message space is defined as follows:

$$\mathcal{V}_q^{\text{pp}} = \left\{ ((M_1, \mathbf{T}_1), \dots, (M_q, \mathbf{T}_q)) \mid \exists \mathbf{m} = (m_1, \dots, m_q) \in \mathbb{Z}_p^q \text{ s.t. } \begin{array}{l} \forall i \in [q] : \text{msg}_i = (M_i = g_1^{m_i}, (T_{ij} = B_j^{m_i})_{j \in [q]}) \end{array} \right\} \quad (11)$$

Namely, the i -th entry msg_i of a vector in $\mathcal{V}_q^{\text{pp}}$ consists of a message component $M_i \in \mathbb{G}_1$ and a vector of q tags T_{ij} . The consistency of msg_i can be publicly checked with a routine $\text{VerifyMsg}_2(\text{pp}, \text{msg}_i, i)$ verifying the pairing equations:

$$\forall j \in [q] : e(M_i, \hat{B}_j) = e(T_{ij}, \hat{g}_2)$$

We now present the scheme in full detail and then provide intuition about its ideas and the differences with Schemes 1 from [KMS26].

Scheme 3 (Position Binding SPVC) *The structure-preserving vector commitment SPVC₂ = (Setup₂, Commit₂, Open₂, Verify₂) is defined by the following algorithms:*

Setup₂(1^λ, q): *Given the security parameter $\lambda \in \mathbb{N}$ and the length q of the vectors, pick $\alpha \leftarrow \mathbb{Z}_p$ and output public parameters pp as in equation (10).*
Commit₂(pp, msg₁, ..., msg_q): *On input a vector of q messages $(\text{msg}_1, \dots, \text{msg}_q) \in \mathcal{V}_q^{\text{pp}}$, such that each $\text{msg}_i = (M_i, \mathbf{T}_i)$, output:*

$$\text{com} = \prod_{i \in [q]} T_{ii} \quad \wedge \quad \text{aux} = (\mathbf{T})$$

Open₂(pp, msg, i, aux): *This algorithm is run by the committer to produce a proof $\pi_i = (\pi_{i1}, \pi_{i2}, \pi_{i3}, \pi_{i4})$ that msg is the i -th committed message:*

$$\begin{aligned} \pi_{i1} &= \prod_{j \in [i-1]} T_{j, q+1-i+j} \quad \wedge \quad \pi_{i2} = \prod_{j \in [i+1, q]} T_{j, 1-i+j} \\ \wedge \quad \pi_{i3} &= \prod_{j \in [i+1, q]} T_{j, j-i-1} \text{ where } T_{j,0} = M_j \quad \wedge \quad \pi_{i4} = T_{iq} \end{aligned}$$

Verify₂(pp, com, M, i, π_i): *The verification algorithm accepts (i.e. it outputs 1) only if π_i is a valid proof that com was created for M:*

$$e(\text{com}, \hat{B}_{q+1-i}) = e(\pi_{i1}, \hat{g}_2) \cdot e(\pi_{i2}, \hat{B}_q) \cdot e(\pi_{i4}, \hat{B}_1) \quad (12)$$

$$\wedge \quad e(M, \hat{B}_q) = e(\pi_{i4}, \hat{g}_2) \quad (13)$$

$$\wedge \quad e(\pi_{i3}, \hat{B}_2) = e(\pi_{i2}, \hat{g}_2) \quad (14)$$

We prove the correctness of the scheme in Appendix B.

The commit algorithm of our scheme proceeds essentially in the same way as in Scheme 1, namely $\text{com} = \prod_{i \in [q]} T_{ii} = \prod_{i \in [q]} g_1^{\alpha^i m_i}$. The major difference is in the opening algorithm that, in comparison to Scheme 1, cannot use elements that encode powers of α beyond q .

The first change we introduce in the opening is that the ‘message’ element plugged in the verification equation (12) involving com is $\pi_{i4} = T_{iq} = B_q^m$; this is paired with $\hat{B}_1$, but notice that the result $e(B_q^m, \hat{B}_1)$ is the same as in the analogous pairing $e(B_i^m, \hat{B}_{q+1-i})$ of Scheme 1. To ensure that π_{i4} is well formed, i.e., it has only B_q as basis, we add the verification equation (13) that imposes

a lower bound of q on the degree of the α -power encoded in π_{i4} , and since any element in $\mathbb{G}_1$ is upper bounded by degree $q + 1$, π_{i4} has only B_q as basis.

The next main change is that in our scheme we “broke up” the opening proof in two elements π_{i1} and π_{i2} such that $\pi_{i1} \cdot (\pi_{i2})^{\alpha^q}$ is the same as the opening proof of Scheme 1. Giving only π_{i1}, π_{i2} as opening might however lead to attacks since $e(\pi_{i1}, \hat{g}_2) \cdot e(\pi_{i2}, \hat{B}_q)$ might encode terms of the form $g_T^{\alpha^{q+1}}$ which could cancel/replace the term $e(B_1^m, \hat{B}_q)$. Since the **pp** only includes $g_1^{\alpha^j}$ with $j \leq q$, such attack may only be raised by π_{i2} . We prevent this by adding a lower-bound degree check on π_{i2} that ensures it encodes a polynomial of degree ≥ 2 in α . This is achieved via π_{i3} and the last verification equation.

Theorem 2. *Scheme 3 is position binding under the (q, q) -PDL assumption, in the algebraic group model.*

Proof. Let $\mathcal{A}$ be an algebraic adversary that breaks position binding of Scheme 3 with non-negligible probability. We construct a reduction with oracle access to $\mathcal{A}$, which yields an algebraic adversary $\mathcal{B}$ against the (q, q) -PDL assumption.

Adversary $\mathcal{B}$: On input an instance of the (q, q) -PDL assumption, $\mathcal{B}$ simulates the vector commitment binding game for $\mathcal{A}$ as follows. First, it defines the public parameters **pp** from it (q, q) -PDL input.

Next, $\mathcal{B}$ runs $\mathcal{A}(\text{pp})$ which outputs a commitment **com**, an index $i \in [q]$, two distinct message tuples M and M' such that $M \neq M'$, together with corresponding openings $(\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$ and $(\pi'_1, \pi'_2, \pi'_3, \pi'_4, \pi'_5)$ that pass the verification algorithm.

Since $\mathcal{A}$ is algebraic, it additionally outputs the algebraic explanations of all the group elements. Note that all outputted elements by $\mathcal{A}$ are in $\mathbb{G}_1$. Given the algebraic representation **a** of an element $H \in \mathbb{G}_1$, the polynomial representation of such element would be

$$P_h(X) = a_h + \sum_{i \in [q]} a_{h,i} X^i \quad (15)$$

Let us observe that if, for instance, $M = 1_{\mathbb{G}_1}$, then $\text{dlog}_{g_1} M = 0$ and $P_m(X) \equiv 0$. On the other hand, since $M' \neq M$, it follows that $P_{m'}(X) \not\equiv 0$. Hence, at least one of the polynomials $P_m(X)$ and $P_{m'}(X)$ is non-zero.

Without loss of generality, let us assume that $P_m(X)$ is non-zero. From the verification equation (13) it follows that $P_m(\alpha)\alpha^q = P_{\pi_4}(\alpha)$. Let us define the polynomial

$$Q(X) = P_m(X)X^q - P_{\pi_4}(X).$$

Then there are two possible cases: either $Q(X) \not\equiv 0$ or $Q(X) \equiv 0$. $\mathcal{B}$ proceeds differently depending on each case:

Case $Q(X) \not\equiv 0$: $\mathcal{B}$ factorizes $Q(X)$ and computes its roots. Since $Q(\alpha) = 0$, it follows that α is a root of Q , and the reduction therefore recovers α , thereby breaking the power discrete logarithm assumption.

Case $Q(X) \equiv 0$: since $P_m(X) \not\equiv 0$ and $P_m(X)X^q = P_{\pi_4}(X)$, we have that $P_{\pi_4}(X) \not\equiv 0$ and X^q divides $P_{\pi_4}(X)$. The degree of $P_{\pi_4}(X)$ is at most q , and given the representation of $P_{\pi_4}(X)$ shown in (15), we can conclude that

$$P_{\pi_4}(X) = a_{\pi_4,q}X^q.$$

Therefore, the reduction $\mathcal{B}$ has the discrete logarithm $a_{\pi_4,q}$ of M from $\mathcal{A}$'s algebraic explanations. By the same argument, the reduction also extracts the discrete logarithm $a_{\pi'_4,q}$ of M' (in the special case where $M' = 1_{\mathbb{G}_1}$, we have $a_{\pi'_4,q} = 0$).

Now, from verification equation (14) we have that $P_{\pi_3}(\alpha)\alpha^2 = P_{\pi_2}(\alpha)$. One more time, we define a polynomial

$$L(X) = P_{\pi_3}(X)X^2 - P_{\pi_2}(X)$$

Again, either $L(X) \not\equiv 0$ or $L(X) \equiv 0$, and we define $\mathcal{B}$'s behaviour depending on which case we are in:

Case $Q(X) \equiv 0 \wedge L(X) \not\equiv 0$: since $L(\alpha) = 0$, the reduction factorizes $L(X)$ to recover α , thereby breaking the power discrete logarithm assumption.

Case $Q(X) \equiv 0 \wedge L(X) \equiv 0$: since $P_{\pi_3}(X)X^2 = P_{\pi_2}(X)$, we have that X^2 divides $P_{\pi_2}(X)$, concluding that its degree is ≥ 2 , i.e.,

$$P_{\pi_2}(X) = \sum_{j=2}^q a_{\pi_2,j}X^j.$$

By the same argument applied over π'_2 and π'_3 we can conclude that we can factor X^2 from $P_{\pi'_2}(X)$, and this polynomial has an analogous representation as the one shown above for $P_{\pi_2}(X)$.

Finally, from the remaining verification equations we get

$$\begin{aligned} e(\text{com}, \hat{B}_{q+1-i}) &= e(\pi_1, \hat{g}_2) \cdot e(\pi_2, \hat{B}_q) \cdot e(\pi_4, \hat{B}_1) \quad \wedge \\ e(\text{com}, \hat{B}_{q+1-i}) &= e(\pi'_1, \hat{g}_2) \cdot e(\pi'_2, \hat{B}_q) \cdot e(\pi'_4, \hat{B}_1) \end{aligned}$$

which imply that

$$\begin{aligned} e(\text{com}, \hat{B}_{q+1-i}) &= e(\pi_1, \hat{g}_2) \cdot e(\pi_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{\pi_4,q}} \quad \wedge \\ e(\text{com}, \hat{B}_{q+1-i}) &= e(\pi'_1, \hat{g}_2) \cdot e(\pi'_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{\pi'_4,q}} \end{aligned}$$

and thus

$$e(\pi_1, \hat{g}_2) \cdot e(\pi_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{\pi_4,q}} = e(\pi'_1, \hat{g}_2) \cdot e(\pi'_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{\pi'_4,q}}$$

that by simply rewriting yields

$$e(\pi_1\pi'^{-1}_1, \hat{g}_2) \cdot e(\pi_2\pi'^{-1}_2, \hat{B}_q) = e(B_q, \hat{B}_1)^{a_{\pi_4,q} - a_{\pi'_4,q}} \quad (16)$$

Let us define the polynomial

$$R^*(X) = (P_{\pi_1}(X) - P_{\pi'_1}(X)) + (P_{\pi_2}(X) - P_{\pi'_2}(X))X^q - (a_{\pi_4,q} - a_{\pi'_4,q})X^{q+1}$$

By the fact that we can factor X^2 from $P_{\pi_2}(X)$ and $P_{\pi'_2}(X)$ (since X^2 divides both polynomials), there exists a polynomial $P_{\pi_2}^*(X)$ such that

$$P_{\pi_2}(X) - P_{\pi'_2}(X) = X^2 P_{\pi_2}^*(X)$$

Then, we can argue that the term $P_{\pi_2}^*(X)X^{q+2}$ does not contain the term X^{q+1} . Now, since $P_{\pi_1}(X) - P_{\pi'_1}(X)$ does also not contain the term X^{q+1} , and $a_{\pi_4,q} \neq a_{\pi'_4,q}$, we have that $R^*(X) \not\equiv 0$. Also, by equation (16), $R^*(\alpha) = 0$. Therefore, $\mathcal{B}$ can factor $R^*(X)$ to recover α and break the power discrete logarithm assumption. $\square$

5 Our Structure-Preserving Accumulator

In this section, we propose our solution of a structure-preserving accumulator that achieves the practicality of q -DHE structure-preserving-like accumulators, while ensuring security based on the standard notion of collision resistance.

We start from the promising idea in [KMS26] that one can bypass existing impossibility results via the use of an extended message space. In our view, the issues arising in the q -DHE accumulator (Scheme 2) come from the fact that the construction does not aim for a standard collision resistance notion and that the notion proposed in the statement of Theorem 1 does not fully capture the use of their extended message space. In particular, the tricky part is to model the fact that the verifier (and thus the challenger in the security game) only sees the message component $\mathbf{M}'$ of the false set element, and not all its related tags. This also implies that the set membership check in the winning condition should be possible by using only this component.

5.1 Defining Structure-Preserving Accumulators for Extended Sets

To solve the issues mentioned above, our first contribution is to propose a definition of SPAs that properly supports extended message spaces like the one of [KMS26].

This space, which we denote by $\mathcal{S}_q^{\text{pp}}$, captures the sets of messages that are well-formed with respect to the given public parameters. To facilitate formal reasoning about correctness and soundness, we further introduce an auxiliary algorithm `VerifySet` that efficiently and publicly verifies the validity of sets in $\mathcal{S}_q^{\text{pp}}$, or alternatively we define $\mathcal{S}_q^{\text{pp}}$ as the set of sets that are accepted by `VerifySet`. By restricting attention to well-formed inputs in this manner, we obtain stronger correctness and soundness guarantees for the accumulator.

Definition 10 (SP Accumulator for Extended Sets (xSPA)). *A static structure-preserving accumulator for extended sets is a tuple of structure-preserving (see Section 2.1) algorithms (`Setup`, `VerifySet`, `Eval`, `WitCreate`, `Verify`) defined as follows. In particular, the algorithm `VerifySet` defines the extended set space $\mathcal{S}_q^{\text{pp}}$.*

- $\text{Setup}(1^\lambda, q)$: Takes a security parameter λ and an integer q . If $q \neq \infty$, then q is an upper bound on the cardinality of the sets to be accumulated. Returns prover's public parameters pp and verifier's public parameters vp , which include a definition of the set $\mathcal{S}$.
- $\text{VerifySet}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}})$: given a set $\mathcal{X} \subseteq \mathcal{S}$ and auxiliary information $\mathcal{X}_{\text{aux}}$, is a deterministic polynomial-time algorithm that outputs 1 if $\mathcal{X}$ is a well-formed set of messages with respect to pp , and 0 otherwise. This algorithm implicitly defines a message space, which we refer to as the accumulator's extended set space:

$$\mathcal{S}_q^{\text{pp}} = \{ (\mathcal{X}, \mathcal{X}_{\text{aux}}) \mid \text{VerifySet}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}) = 1 \}. \quad (17)$$

- $\text{Eval}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}})$: This (probabilistic) algorithm takes public parameters pp , a set $\mathcal{X} \subseteq \mathcal{S}$ and auxiliary set $\mathcal{X}_{\text{aux}}$. Returns an accumulator $\text{Acc}_{\mathcal{X}}$ together with some auxiliary information aux .
- $\text{WitCreate}(\text{pp}, \text{Acc}_{\mathcal{X}}, x, \text{aux})$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, auxiliary information aux , and a value $x \in \mathcal{S}$. It returns $\perp$, if $x \notin \mathcal{X}$, and a witness wit_x for x otherwise.
- $\text{Verify}(\text{vp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x)$: This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, a witness wit_x , and a value $x \in \mathcal{S}$. It returns true if wit_x is a witness for $x \in \mathcal{X}$ and false otherwise.

Note that we require VerifySet to take as input the prover's public parameters pp , rather than the verifier's public parameters vp . This choice is convenient for our construction; however, we believe that, depending on the specific design, either option could be adopted.

Elements of $\mathcal{S}_q^{\text{pp}}$ can now be viewed as tuples consisting of two main components. The first component is a set $\mathcal{X} \subseteq \mathcal{S}$ in the set space of the static accumulator, while the second component contains an auxiliary set $\mathcal{X}_{\text{aux}}$ that allows one to verify the correctness of $\mathcal{X}$ with respect to pp (and, when relevant, the parameter q). This way, we can model accumulated messages as elements of the set $\mathcal{S}$, yielding a clearer interpretation of the meaning of membership $x \in \mathcal{X}$ (resp. non-membership $x \notin \mathcal{X}$) for every element $x \in \mathcal{S}$.

Definition 11 (Correctness of xSPA). We say that an xSPA is correct, if for any $\lambda \in \mathbb{N}$, any bound parameter q , any pair $(\mathcal{X}, \mathcal{X}_{\text{aux}})$ in the extended accumulators message space $\mathcal{S}_q^{\text{pp}}$, and any element $x \in \mathcal{X}$, it holds that:

$$\Pr \left[\text{Verify}(\text{vp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x) = 1 \mid \begin{array}{l} (\text{pp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, q); \\ (\text{Acc}_{\mathcal{X}}, \text{aux}) \leftarrow \text{Eval}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}); \\ \text{wit}_x \leftarrow \text{WitCreate}(\text{pp}, \text{Acc}_{\mathcal{X}}, x, \text{aux}) \end{array} \right] = 1$$

Definition 12 (Collision Resistance of xSPA). An xSPA satisfies collision resistance if for all PPT adversaries $\mathcal{A}$ and security parameter λ , it holds that

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{vp}, \text{Acc}_{\mathcal{X}}, \text{wit}_x, x) = 1 \\ \wedge \text{VerifySet}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}) = 1 \\ \wedge (\text{Acc}_{\mathcal{X}}, \cdot) = \text{Eval}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}; r) \\ \wedge x \in \mathcal{S} \wedge x \notin \mathcal{X} \end{array} \mid \begin{array}{l} (\text{pp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, q); \\ (\text{Acc}_{\mathcal{X}}, r, \mathcal{X}, \mathcal{X}_{\text{aux}}, \\ \text{wit}_x, x) \leftarrow \mathcal{A}(\text{pp}, \text{vp}) \end{array} \right] \leq \text{negl}(\lambda)$$

5.2 Our xSPA Construction

Following the definitions of static accumulators for extended sets, we adapt the structured message space of our SPVC presented in equation (10), in order to fit the definitions of static accumulators for extended sets.

We start by defining the `VerifySet` routine, which implicitly defines $\mathcal{S}_q^{\text{pp}}$. This algorithm takes as arguments the provers public parameters pp , a set $\mathcal{X} \subseteq \mathcal{S} = \mathbb{G}_1$ such that $n = |\mathcal{X}| \leq q$, and an auxiliary set $\mathcal{X}_{\text{aux}} \subseteq \mathbb{G}_1^q$ such that $|\mathcal{X}_{\text{aux}}| = n$.

In our presentation, we assume total ordering over sets, induced by an ordering function $\text{ord}(\cdot)$. This function takes as input a finite subset of $\mathcal{S}$ and outputs an ordered sequence containing all its elements, for example, according to lexicographical order. We assume that $\text{ord}(\cdot)$ is fixed and included as part of the public parameters pp .

For ease of presentation, we assume that both $\mathcal{X}$ and $\mathcal{X}_{\text{aux}}$ are represented as ordered vectors and assume a coordinate-wise correspondence between these two vectors. Namely, the element at position i in $\text{ord}(\mathcal{X})$ is assumed to be associated with the element at position i in the vector representation of $\mathcal{X}_{\text{aux}}$, after applying the same permutation induced $\text{ord}(\mathcal{X})$ over $\mathcal{X}$. This allows us to map each $x_i \in \text{ord}(\mathcal{X})$ to a corresponding auxiliary information which we will denote by $\tau_i \in \mathcal{X}_{\text{aux}}$. Notice that the `VerifySet` procedure ensures the validity of the relation between each $x_i \in \text{ord}(\mathcal{X})$ and $\tau_i \in \mathcal{X}_{\text{aux}}$.

`VerifySet`($\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}$) : Given public parameters pp , a set $\mathcal{X} \subseteq \mathcal{S} = \mathbb{G}_1$ such that $n = |\mathcal{X}| \leq q$, and auxiliary set information $\mathcal{X}_{\text{aux}} \subseteq \mathbb{G}_1^q$, it must hold that $|\mathcal{X}_{\text{aux}}| = n$, otherwise, `VerifySet` outputs 0.

Finally, the algorithm returns 1 if for every $i \in [n]$ the following equations hold, and 0 otherwise. Setting $\text{ord}(\mathcal{X}) = [x_1, \dots, x_n]$ checks

$$e(x_i, \hat{B}_j) = e(\tau_{ij}, \hat{g}_2)$$

for every $j \in [q]$, where $\tau_i = (\tau_{ij})_{j \in [q]} \in \mathcal{X}_{\text{aux}}$ is the corresponding auxiliary information associated to x_i .

Description of our scheme. Our xSPA construction is deeply inspired by our SPVC scheme of Section 4, and is constructed along the same lines. The main concern in this setting is the fact that we cannot reveal the position of the opened message in the form of an integer, due to structure-preserving constraints. Also, one might think that since the only element depending on the index during verification is $\hat{B}_{q+1-i}$, we can add it as part of the proof. As we have mentioned in Section 3.4, including $\hat{B}_{q+1-i}$ in the proof has two issues. If one does not prove it is a CRS element in $\mathbb{G}_2$, potential attacks become possible as in Scheme 2 and pointed out in [dS25]. Leaving to the verifier the check that such proof element is $\hat{B}_{q+1-i}$ for some i raises a challenge as this should be done via a structure-preserving algorithm – essentially another problem of proving set membership in an SP manner, which may lead to $O(q)$ -complexity in proof size and verification time, when proven in zero-knowledge.

We solve this issue by signing all the $\hat{B}_i$ elements of $\mathbb{G}_2$ and adding these signatures to the public parameters. We leverage an optimal structure-preserving signature presented in [Gro15]. Intuitively, in **Setup** we generate a public and secret key pair, which is used to sign all elements $\hat{B}_i$. The secret key is then discarded to prevent the generation of signatures on new messages, while the public key is retained and made available for signature verification.

Now, a prover can provide $\hat{B}_{q+1-i}$ and the corresponding signature as part of the membership proof. A valid signature on an element $\hat{H}$ implies that $\hat{H} \in \{\hat{B}_1, \dots, \hat{B}_q\}$ without the need of knowing the integer i such that $\hat{H} = \hat{B}_{q+1-i}$. Also, since the verification is structure-preserving, we can integrate it in the accumulator's verification maintaining this structural property. Furthermore, openings sizes and verification time remain constant, and the verifiers public parameters are constant-size too.

Scheme 4 (Collision Resistant xSPA) *Our xSPA scheme is defined by the following PPT algorithms in addition to the **VerifySet** routine described above:*

Setup($1^\lambda, q$): *Given the security parameter $\lambda \in \mathbb{N}$ and the size q of the committed vector, picks $\alpha \leftarrow \mathbb{Z}_p$ and computes:*

$$\text{pp}_q = (\text{BG}; B_1 = g_1^\alpha, \dots, B_q = g_1^{\alpha^q}; \hat{B}_1 = \hat{g}_2^\alpha, \dots, \hat{B}_q = \hat{g}_2^{\alpha^q})$$

Next, samples $y, v \leftarrow \mathbb{Z}_p$, and computes $\Sigma_y \leftarrow \hat{g}_2^y$ and $\Sigma_v \leftarrow g_1^v$. Additionally, for every $j \in [q]$ samples $z_j \leftarrow \mathbb{Z}_p$ and computes:

$$\Sigma_j = (\sigma_{rj}, \hat{\sigma}_{sj}, \hat{\sigma}_{tj}) \leftarrow (g_1^{\frac{1}{z_j}}, \hat{g}_2^{z_j(y+v)}, \hat{g}_2^{z_j(\alpha^j + vy)})$$

Finally, outputs $\text{pp} = (\text{pp}_q, \Sigma_y, \Sigma_v, (\Sigma_j)_{j \in [q]}, \text{ord}(\cdot))$ as the provers public parameters; and $\text{vp} = (\text{BG}, \hat{B}_1, \hat{B}_2, \hat{B}_q, \Sigma_y, \Sigma_v)$ as the verifiers public parameters.

Eval($\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}}$): *On input a set of messages $\mathcal{X} \subseteq \mathcal{S} = \mathbb{G}_1 \setminus \{1\}$, and a set of auxiliary information $\mathcal{X}_{\text{aux}} \subseteq \mathbb{G}_1^q$, checks correctness of the set via $1 \leftarrow \text{VerifySet}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}})$. Parse $[x_1, \dots, x_n] \leftarrow \text{ord}(\mathcal{X})$ and $(x_i, \tau_i)_{i \in [n]}$, where $\tau_i \in \mathcal{X}_{\text{aux}}$ is the associated auxiliary information of x_i . Outputs:*

$$\text{Acc}_{\mathcal{X}} = \prod_{i \in [n]} \tau_{ii} \quad \wedge \quad \text{aux} = (\mathcal{X}, \mathcal{X}_{\text{aux}})$$

WitCreate($\text{pp}, \text{Acc}_{\mathcal{X}}, x, \text{aux}$): *This algorithm takes public parameters pp , an accumulator $\text{Acc}_{\mathcal{X}}$, auxiliary information aux , a value x , and proceeds as follows. Parse $\text{aux} = (r, \mathcal{X}, \mathcal{X}_{\text{aux}})$, $[x_1, \dots, x_n] \leftarrow \text{ord}(\mathcal{X})$ and $(x_i, \tau_i)_{i \in [n]}$, where $\tau_i \in \mathcal{X}_{\text{aux}}$ is the associated auxiliary information of x_i . Return $\perp$ if $x \notin \mathcal{X}$. Otherwise, if $x \in \mathcal{X}$, assume $x = x_i$ and accordingly select auxiliary infor-*

ation $\tau_i = (\tau_{ij})_{j \in [q]} \in \mathcal{X}_{\text{aux}}$ and compute:

$$\begin{aligned} W_1 &= \prod_{j \in [i-1]} \tau_{j, q+1-i+j} \quad \wedge \quad W_2 = \prod_{j \in [i+1, n]} \tau_{j, 1-i+j} \\ \wedge \quad W_3 &= \prod_{j \in [i+1, n]} \tau_{j, j-i-1} \quad \text{where } \tau_{j, 0} = x_j \\ \wedge \quad W_4 &= \tau_{iq} \quad \wedge \quad W_5 = \Sigma_{q+1-i} \quad \wedge \quad \hat{W}_6 = \hat{B}_{q+1-i} \end{aligned}$$

Outputs $\text{wit}_x = (W_1, W_2, W_3, W_4, W_5, \hat{W}_6)$ as a witness for x .

Verify(vp, $\text{Acc}_{\mathcal{X}}, x, \text{wit}$): The verification algorithm accepts (i.e. it outputs 1) only if $\text{wit} = (W_1, W_2, W_3, W_4, W_5, \hat{W}_6)$ is a valid proof that $\text{Acc}_{\mathcal{X}}$ was created for a set including x . First, checks that $x \neq 1_{G_1}$; otherwise, output $\perp$. Then, parses $W_5 = (\sigma_r, \hat{\sigma}_s, \hat{\sigma}_t)$, and then check the following pairing product equations

$$e(\sigma_r, \hat{\sigma}_s) = e(\Sigma_v, \hat{g}_2) \cdot e(g_1, \Sigma_y) \quad (18)$$

$$\wedge \quad e(\sigma_r, \hat{\sigma}_t) = e(g_1, \hat{W}_6) \cdot e(\Sigma_v, \Sigma_y) \quad (19)$$

$$\wedge \quad e(x, \hat{B}_q) = e(W_4, \hat{g}_2) \quad (20)$$

$$\wedge \quad e(W_3, \hat{B}_2) = e(W_2, \hat{g}_2) \quad (21)$$

$$\wedge \quad e(\text{Acc}_{\mathcal{X}}, \hat{W}_6) = e(W_1, \hat{g}_2) \cdot e(W_2, \hat{B}_q) \cdot e(W_4, \hat{B}_1) \quad (22)$$

The correctness of our xSPA scheme intuitively follows from that of our SPVC of Section 4 and the validity of the signatures (equations (18)–(19)). A full correctness proof appears in Appendix C.

5.3 Collision Resistance of our xSPA scheme

The security of our xSPA is stated in the following theorem.

Theorem 3. *Scheme 4 is collision resistant under the $(2q, q+1)$ -PDL assumption, in the algebraic group model.*

We prove the above theorem via the following strategy. The main part introduces an assumption over bilinear groups to prove that our xSPA is collision resistant under this assumption in the AGM (see Definition 14 and Theorem 4 below). Then, we show that our assumption is an instance of the flexible-targets Uber assumption defined in [BFL20]. Finally, we obtain the statement of Theorem 3 by combining the above results with an application of [BFL20, Theorem 6.1], which shows that this family of assumptions is implied by the (q_1, q_2) -PDL assumption for an appropriate choice of (q_1, q_2) that, as argued later, in our case is $(2q, q+1)$.

Informally, the Uber assumption is parametrized by a tuple of rational fractions $(\mathbf{R}, \mathbf{S}, \mathbf{F}, R^*, S^*, F^*) \in \mathbb{Z}_p(\mathbf{X})^{r+s+f+3}$ over a vector of secret scalars $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}_p^m$. Given the group elements

$$\left(g_1^{R_i(\mathbf{x})} \right)_{i \in [r]}, \quad \left(\hat{g}_2^{S_j(\mathbf{x})} \right)_{j \in [s]}, \quad \left(g_T^{F_k(\mathbf{x})} \right)_{k \in [f]},$$

the assumption states that it is hard to compute the target elements $g_1^{R^*(\alpha)}$, $\hat{g}_2^{S^*(\alpha)}$, or $g_T^{F^*(\alpha)}$, as long as the tuple is *non-trivial*. Intuitively, this means that at least one of the target elements computed by the adversary cannot be derived from the input elements by linear combinations or pairing operations. For a formal definition we refer to Definition 3.3 in [BFL20]. This notion captures the hardness of computing any group element that lies outside the span of the provided inputs, and generalizes many standard assumptions in pairing-based cryptography, such as CDH, DDH, and q -DHE, within a single unified framework.

In our setting, the target elements and their corresponding polynomials are adaptively chosen by the adversary, giving rise to what is known as the *flexible-target Uber assumption for rational fractions*.

Definition 13 (($\mathbf{R}, \mathbf{S}, \mathbf{F}$)-Uber assumption for flexible targets and rational fractions in the AGM [BFL20]). Let $\text{BG} = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$ be a bilinear group generated by $\text{GroupGen}(1^\lambda)$, $g_T \leftarrow e(g_1, \hat{g}_2)$, and $(\mathbf{R}, \mathbf{S}, \mathbf{F}) \in \mathbb{Z}_p(\mathbf{X})^{r+s+f}$.

We say that the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -flexible-target über assumption holds for BG in the algebraic group model if no algebraic adversary $\mathcal{A}_{\text{alg}}$ can win the following game with non-negligible probability. First, sample $\mathbf{x} = (x_1, \dots, x_m) \leftarrow \$ \mathbb{Z}_p^m$; if $\check{R}_i(\mathbf{x}) \equiv_p 0$, or $\check{S}_j(\mathbf{x}) \equiv_p 0$, or $\check{F}_k(\mathbf{x}) \equiv_p 0$ for some i, j, k (where $R_i = \hat{R}_i / \check{R}_i$ for $\hat{R}_i, \check{R}_i \in \mathbb{Z}_p[\mathbf{X}]$, and analogously for S_j and F_k), the game aborts. The adversary is then given

$$\mathbf{C} := \left(g_1^{R_i(\mathbf{x})} \right)_{i \in [r]}, \quad \mathbf{D} := \left(\hat{g}_2^{S_j(\mathbf{x})} \right)_{j \in [s]}, \quad \mathbf{E} := \left(g_T^{F_k(\mathbf{x})} \right)_{k \in [f]}.$$

The adversary then outputs a tuple $(C^*, D^*, E^*) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T$ and $(R^*, S^*, F^*) \in \mathbb{Z}_p(\mathbf{X})^3$. We say that $\mathcal{A}_{\text{alg}}$ wins if

$$(C^*, D^*, E^*) = \left(g_1^{R^*(\mathbf{x})}, \hat{g}_2^{S^*(\mathbf{x})}, g_T^{F^*(\mathbf{x})} \right)$$

and $(\mathbf{R}, \mathbf{S}, \mathbf{F}, R^*, S^*, F^*)$ is non-trivial.

Note that above, non-triviality of the tuple (R^*, S^*, F^*) holds even if just one of its components is non-trivial. In particular, in Type III bilinear groups, R^* is non-trivial if it is linearly independent of $\mathbf{R}$.

To prove the security of our xSPA scheme we define a special case of the flexible-target uber assumption of Definition 14, in which the tuple of polynomials $(\mathbf{R}, \mathbf{S}, \mathbf{F}) \in \mathbb{Z}_p(A, V, Y, Z_1, \dots, Z_q)^{(2q+1)+(3q+1)}$ match the distribution of the xSPA public parameters.

Definition 14. Consider the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -flexible-target über assumption of Definition 13 where, for $\mathbf{X} = (A, V, Y, Z_1, \dots, Z_q)$, the polynomials are:

$$\mathbf{R} = ((R_{\alpha_i}(\mathbf{X}) = A^i)_{i \in [q]}, R_v(\mathbf{X}) = V, (R_{r_i}(\mathbf{X}) = 1/Z_i)_{i \in [q]}) \quad (23)$$

$$\mathbf{S} = \left((S_{\alpha_i}(\mathbf{X}) = A^i)_{i \in [q]}, S_y(\mathbf{X}) = Y, (S_{s_i}(\mathbf{X}) = Z_i(Y + V))_{i \in [q]}, \right. \\ \left. (S_{t_i}(\mathbf{X}) = Z_i(A^i + VY))_{i \in [q]} \right) \quad (24)$$

$$\mathbf{F} = \emptyset \quad (25)$$

and for $\mathbf{x} = (\alpha, v, y, z_1, \dots, z_q) \leftarrow \mathbb{Z}_p^{q+3}$ the corresponding group elements are

$$\mathbf{C} = \left((B_i = g_1^{R_{\alpha_i}(\mathbf{x})})_{i \in [q]}, B_v = g_1^{R_v(\mathbf{x})}, (\sigma_{r_i} = g_1^{R_{r_i}(\mathbf{x})})_{i \in [q]} \right) \quad (26)$$

$$\mathbf{D} = \left((\hat{B}_i = \hat{g}_2^{S_{\alpha_i}(\mathbf{x})})_{i \in [q]}, \hat{B}_y = \hat{g}_2^{S_y(\mathbf{x})}, (\sigma_{s_i} = \hat{g}_2^{S_{s_i}(\mathbf{x})})_{i \in [q]}, \right. \\ \left. (\sigma_{t_i} = \hat{g}_2^{S_{t_i}(\mathbf{x})})_{i \in [q]} \right) \quad (27)$$

$$\mathbf{E} = \emptyset \quad (28)$$

We prove the security of our xSPA scheme in the following theorem.

Theorem 4. Scheme 4 is collision resistant under the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -flexible target über assumption of Definition 14, in the algebraic group model.

Brief description of the proof strategy. Intuitively, the security of Scheme 4 relies on the position-binding guarantees of our SPVC given by Theorem 2, and the unforgeability of the structure-preserving signature scheme from [Gro15]. The main challenge lies in identifying the appropriate framework under which the security of both constructions can be established together in the AGM, and it is precisely in this context that the über assumption plays a central role. The proof follows a similar structure to that of Theorem 2, where the reduction leverages the algebraic nature of the adversary to extract non-trivial polynomials, which can be factorized to find a discrete logarithm and produce an attack against the power discrete logarithm assumption. In the context of Scheme 4, however, the number of variables increases significantly when adding signed elements to the CRS, and we rely on the über assumption to unify and consolidate this reduction to the power discrete logarithm assumption.

Proof. Let $\mathcal{A}$ be an algebraic adversary that breaks collision resistance of Scheme 4 with non-negligible probability. We construct a reduction with oracle access to $\mathcal{A}$, which yields an algebraic adversary $\mathcal{B}$ against the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -flexible-target über assumption.

Adversary $\mathcal{B}(\mathbf{C}, \mathbf{D}, \mathbf{E})$: First of all, $\mathcal{B}$ uses $(\mathbf{C}, \mathbf{D}, \mathbf{E})$ to compute the accumulator public parameters (pp, vp) , which is straightforward given the correspondence shown in (26) and (27).

Next, $\mathcal{B}$ runs $\mathcal{A}(\text{pp}, \text{vp})$ which outputs a tuple $(\text{Acc}_{\mathcal{X}}^*, \mathcal{X}^*, \mathcal{X}_{\text{aux}}^*, x^*, \text{wit}^*)$. Additionally, $\mathcal{A}$ outputs the corresponding algebraic explanations of each group element it returns. Note that these algebraic relations define linear combinations

over the rational fractions $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ that parametrize the über assumption. In particular, these are Laurent polynomials, since the only denominators appearing in these fractions are of the form Z_i^{-1} . Given the algebraic representation $\mathbf{a}$ of an element $H \in \mathbb{G}_1$, the polynomial representation of such element would be

$$P_h(\mathbf{X}) = a_h + a_{h,v}V + \sum_{i \in [q]} a_{h,\alpha_i} A^i + \sum_{j \in [q]} a_{h,r_j} 1/Z_j \quad (29)$$

Analogously, for an algebraic explanation $\mathbf{b}$ of an element $H \in \mathbb{G}_2$, the polynomial representation of such element is

$$\begin{aligned} \hat{Q}_h(\mathbf{X}) = & b_h + b_{h,y}Y + \sum_{i \in [q]} b_{h,\alpha_i} A^i + \sum_{j \in [q]} b_{h,s_j} Z_j(Y + V) \\ & + \sum_{k \in [q]} b_{h,t_k} Z_k(A^i + VY) \end{aligned} \quad (30)$$

Let us assume that $\mathcal{A}$ wins the collision resistance game for xSPA, such that $1 = \text{VerifySet}(\text{pp}, \mathcal{X}^*, \mathcal{X}_{\text{aux}}^*), 1 = \text{Verify}(\text{vp}, \text{Acc}_{\mathcal{X}}^*, x^*, \text{wit}^*), x^* \notin \mathcal{X}$ and $x^* \neq 1$. By parsing $\text{wit}^* = (W_1, W_2, W_3, W_4, W_5, W_6)$ and $W_5 = (\sigma_r, \hat{\sigma}_s, \hat{\sigma}_t)$, the reduction defines from each of the verification equations of the xSPA scheme Verify routine (see (18) - (22)) the associated polynomials

$$\begin{aligned} T_1(\mathbf{X}) &:= P_{\sigma_r}(\mathbf{X}) \hat{Q}_{\sigma_s}(\mathbf{X}) - (V + Y) \\ T_2(\mathbf{X}) &:= P_{\sigma_r}(\mathbf{X}) \hat{Q}_{\sigma_t}(\mathbf{X}) - (\hat{Q}_{w_6}(\mathbf{X}) + VY) \\ T_3(\mathbf{X}) &:= P_{x^*}(\mathbf{X}) A^q - P_{w_4}(\mathbf{X}) \\ T_4(\mathbf{X}) &:= P_{w_3}(\mathbf{X}) A^2 - P_{w_2}(\mathbf{X}) \\ T_5(\mathbf{X}) &:= P_{\text{Acc}_{\mathcal{X}}}(\mathbf{X}) \hat{Q}_{w_6}(\mathbf{X}) - (P_{w_1}(\mathbf{X}) + P_{w_2}(\mathbf{X}) A^q + P_{w_4}(\mathbf{X}) A) \end{aligned}$$

Then, for each $i \in [5]$, there are two possible cases: $T_i(\mathbf{X}) \neq 0$ or $T_i(\mathbf{X}) \equiv 0$. $\mathcal{B}$ proceeds differently depending on each case:

Case $T_i(\mathbf{X}) \neq 0$ (for any $i \in [5]$): $\mathcal{B}$ sets $C^* = 1_{\mathbb{G}_1}$, and the polynomial

$$R^*(\mathbf{X}) = T_i(\mathbf{X}) \cdot (V \cdot Y).$$

Note that since $T_i(\mathbf{X}) \neq 0$, then $R^*(\mathbf{X}) \neq 0$. Furthermore, R^* is linearly independent from $\mathbf{R}$ since it contains the factor $(V \cdot Y)$, which does not appear in any linear combination of $\mathbf{R}$. Moreover, since the verification equation associated with $T_i(\mathbf{X})$ in xSPA.Verify is satisfied, we have that $T_i(\mathbf{x}) = 0$, which implies that $R^*(\mathbf{x}) = 0$, and therefore $C^* = 1_{\mathbb{G}_1} = g_1^{R^*(\mathbf{x})}$.

$\mathcal{B}$ outputs (C^*, R^*) and the corresponding algebraic explanation $\mathbf{c}^* = \mathbf{0}$, and this output, under the assumption that $\mathcal{A}$ wins the collision resistance game, breaks the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -f-rüber assumption.

Case $T_i(\mathbf{X}) \equiv 0$ for every $i \in [5]$: Let us show how in this case $\mathcal{B}$ can still break the über assumption. We start by presenting several auxiliary claims concerning the $T_i(\mathbf{X})$ polynomials. The proofs of these claims are deferred to Appendix D.

Claim 1 If $T_1(\mathbf{X}) \equiv 0$, there exists $l \in [q]$ such that $P_{\sigma_r}(\mathbf{X}) = a_{\sigma_r, r_l} 1/Z_l$ with $a_{\sigma_r, r_l} \neq 0$. Thereby, σ_r is in the output space of [Gro15] signature (i.e. $\sigma_r = g_1^{a_{\sigma_r, r_l} 1/Z_l}$).

Claim 2 If $T_2(\mathbf{X}) \equiv 0$ and $P_{\sigma_r}(\mathbf{X}) = a_{\sigma_r, r_l} 1/Z_l \neq 0$ for some $l \in [q]$, then $\hat{Q}_{w_6}(\mathbf{X}) = A^l$. Thereby, $\hat{W}_6$ is a public element of the form $\hat{B}_l$.

Claim 3 If $T_3(\mathbf{X}) \equiv 0$ and $x^* \neq 1_{\mathbb{G}_1}$, then $P_{w_4}(\mathbf{X}) = a_{w_4, \alpha_q} A^q$ and $P_{x^*}(\mathbf{X}) = a_{w_4, \alpha_q}$. Thereby, $\mathcal{B}$ recovers the discrete logarithm a_{w_4, α_q} of x^* from $\mathcal{A}$'s algebraic explanations.

Claim 4 If $T_4(\mathbf{X}) \equiv 0$ then A^2 divides $P_{w_2}(\mathbf{X})$. Thereby, W_2 is of the form $\prod_{i \in [2, q]} B_i^{a_{w_2, \alpha_i}}$.

By Claim 1 and 2, there exists some $l \in [q]$ such that $\hat{Q}_{w_6}(\mathbf{X}) = A^l$. Then $\mathcal{B}$ can define $f = q + 1 - l$ such that

$$\hat{Q}_{w_6}(\mathbf{X}) = A^{q+1-f}$$

and $f \in [q]$. By setting $n = |\mathcal{X}_{\text{aux}}|$, then either $f \leq n$ or $n < f \leq q$. $\mathcal{B}$ proceeds differently depending on each case:

Subcase $f \leq n$: $\mathcal{B}$ selects x_f^* from $[x_1^*, \dots, x_n^*] \leftarrow \text{ord}(\mathcal{X})$, and the corresponding auxiliary information $\tau_f^* \in \mathcal{X}_{\text{aux}}$. Then computes a witness $\text{wit}_f^* \leftarrow \text{WitCreate}(\text{pp}, \text{Acc}_{\mathcal{X}}^*, x_f^*, \text{aux})$ for x_f^* . Since $1 \leftarrow \text{VerifySet}(\text{pp}, \mathcal{X}^*, \mathcal{X}_{\text{aux}}^*)$, correctness of wit_f^* holds.

Subcase $n < f \leq q$: $\mathcal{B}$ sets $x_f^* = 1_{\mathbb{G}_1}$, and the corresponding auxiliary information $\tau_f^* = (1_{\mathbb{G}_1})_{i \in [q]}$. $\mathcal{B}$ can compute a witness wit_f^* by following the description of the WitCreate algorithm of Scheme 4 for $x_f^* = 1_{\mathbb{G}_1}$. Note that the correctness of wit_f^* follows by direct inspection of the algorithms.

Now, since $\mathcal{B}$ knows the algebraic explanations of x_f^* and τ_f^* , it can efficiently compute the polynomial representations of the elements involved in $\text{wit}_f^* = (W_{1f}, W_{2f}, W_{3f}, W_{4f}, W_{5f}, \hat{W}_{6f})$. By defining polynomials of the form $T'_i(\mathbf{X})$ for each equation involved in the verification of the accumulator, corresponding to the element x_f^* and witness wit_f^* , $\mathcal{B}$ can apply analogous arguments and claims to the tuple (x_f^*, wit_f^*) . With the exception of the case in which $x_f^* = 1_{\mathbb{G}_1}$, where Claim 3 does not hold, but $\mathcal{B}$ knows the discrete logarithm of x_f^* , namely 0.

If until now, $\mathcal{B}$ has not stopped, it proceeds as follows: By Claim 3, $\mathcal{B}$ has knowledge of x^* and x_f^* discrete logarithms, a_{w_4, α_q} and a_{w_{4f}, α_q} respectively. Also, note that since $x^* \neq 1_{\mathbb{G}_1}$ and $x^* \notin \mathcal{X}$, we have that $a_{w_4, \alpha_q} \neq a_{w_{4f}, \alpha_q}$. From verification equation (22) we have the following:

$$\begin{aligned} e(\text{Acc}_{\mathcal{X}}^*, \hat{W}_6) &= e(W_1, \hat{g}_2) \cdot e(W_2, \hat{B}_q) \cdot e(W_4, \hat{B}_1) \quad \wedge \\ e(\text{Acc}_{\mathcal{X}}^*, \hat{W}_{6f}) &= e(W_{1f}, \hat{g}_2) \cdot e(W_{2f}, \hat{B}_q) \cdot e(W_{4f}, \hat{B}_1) \end{aligned}$$

note that $\hat{W}_6 = \hat{B}_{q+1-f}$ by Claim 2, and by correctness $\hat{W}_{6f} = \hat{B}_{q+1-f}$. Then

$$\begin{aligned} e(\text{Acc}_{\mathcal{X}}^*, \hat{B}_{q+1-f}) &= e(W_1, \hat{g}_2) \cdot e(W_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{w_4, \alpha_q}} \wedge \\ e(\text{Acc}_{\mathcal{X}}^*, \hat{B}_{q+1-f}) &= e(W_{1f}, \hat{g}_2) \cdot e(W_{2f}, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{w_{4f}, \alpha_q}} \end{aligned}$$

and thus

$$e(W_1, \hat{g}_2) \cdot e(W_2, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{w_4, \alpha_q}} = e(W_{1f}, \hat{g}_2) \cdot e(W_{2f}, \hat{B}_q) \cdot e(B_q, \hat{B}_1)^{a_{w_{4f}, \alpha_q}}$$

that by simply rewriting yields

$$e(W_1 W_{1f}^{-1}, \hat{g}_2) \cdot e(W_2 W_{2f}^{-1}, \hat{B}_q) = e(B_q, \hat{B}_1)^{a_{w_4, \alpha_q} - a_{w_{4f}, \alpha_q}} \quad (31)$$

Let us define the polynomial

$$R^*(\mathbf{X}) = (P_{w_1}(\mathbf{X}) - P_{w_{1f}}(\mathbf{X})) + (P_{w_2}(\mathbf{X}) - P_{w_{2f}}(\mathbf{X}))A^q - (a_{w_4, \alpha_q} - a_{w_{4f}, \alpha_q})A^{q+1}$$

By Claim 4 we can factor X^2 from $P_{w_2}(\mathbf{X})$ and $P_{w_{2f}}(\mathbf{X})$, which means that there exists a polynomial $P_{w_2}^*(\mathbf{X})$ such that

$$P_{w_2}(\mathbf{X}) - P_{w_{2f}}(\mathbf{X}) = A^2 P_{w_2}^*(\mathbf{X})$$

Then, we can argue that the term $P_{w_2}^*(\mathbf{X})A^{q+2}$ does not contain the term A^{q+1} . Now, since $P_{w_1}(\mathbf{X}) - P_{w_{1f}}(\mathbf{X})$ does also not contain the term A^{q+1} , and $a_{w_4, \alpha_q} \neq a_{w_{4f}, \alpha_q}$, we have that $R^*(\mathbf{X}) \neq 0$. Also, by equation (31), $R^*(\mathbf{x}) = 0$.

Furthermore, R^* is linearly independent from $\mathbf{R}$ since it contains the term A^{q+1} , which does not appear in any linear combination of $\mathbf{R}$. By setting $C^* = 1_{\mathbb{G}_1} = g_1^{R^*(\mathbf{x})}$, $\mathcal{B}$ breaks the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -f-rüber assumption by outputting (C^*, R^*) and the corresponding algebraic explanation $\mathbf{c}^* = \mathbf{0}$. $\square$

To give more details on how the result of Theorem 6.1 in [BFL20] can be used to complete our proof, here we provide an informal statement of it. We specialize it to Type III bilinear groups, which is our case of interest, and omit the details of the probabilistic bounds and time loss. For further details and a more thorough presentation, we refer the reader to [BFL20]. We first introduce some notation. Let $d_{\mathbf{R}} := \max\{\deg R_i\}$ denote the maximum degree of the rational fractions in $\mathbf{R}$, and let $\check{d}_{\mathbf{R}} = \check{d}_{(\hat{R}_1/\check{R}_1, \dots, \hat{R}_r/\check{R}_r)}$ denote the degree of the least common multiple of the polynomials $(\check{R}_1, \dots, \check{R}_r)$. Notice that the bounds on q_1 and q_2 in the theorem below explain why in our Theorem 3 we need $(2q, q+1)$ -PDL.⁴

Theorem 5 (Dlog implies flexible-target Uber for rational fractions in the AGM). *Let $(p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, e, g_1, \hat{g}_2)$ be a Type III bilinear group and let $\{\mathbf{R}, \mathbf{S}, \mathbf{F}\} \in (\mathbb{Z}_p(X_1, \dots, X_m))^{r+s+f}$ be a tuple of rational fractions.*

Let q_1 and q_2 be such that $q_1 \geq \check{d}_{\mathbf{R}|\mathbf{F}} + d_{\mathbf{R}}$, $q_2 \geq \check{d}_{\mathbf{S}} + d_{\mathbf{S}}$, and $q_1 + q_2 \geq \check{d}_{\mathbf{R}|\mathbf{F}} + \check{d}_{\mathbf{S}} + d_{\mathbf{F}}$. If (q_1, q_2) -PDL is (ε, t) -secure, then the $(\mathbf{R}, \mathbf{S}, \mathbf{F})$ -flexible-target über assumption is (ε', t') -secure in the AGM with $\varepsilon \approx \varepsilon'$ and $t \approx t'$.

⁴ This requires $q \geq 2$, which is a minimally interesting setting for an accumulator.

6 Constant-Size Ring Signatures

A ring signature scheme allows a signer to generate a signature on behalf of a publicly known group of potential signers called ring [RST01], the identity of the signer is hidden among all the possible signers in the ring (See App. E for formal definitions). Ring signatures are crucial in many applications including anonymous authentication [Nao02], privacy-protecting cryptocurrencies [TMSS22], whistleblowing [RST01], and e-voting [TW05, RAVR21].

To showcase the usefulness of structure-preserving accumulators, the authors of [KMS26] attempt to use their accumulator to construct the first constant-size ring signature in a *pure* bilinear group setting. This emphasizes that the construction only uses the groups algebraically and no general purpose tools such as SNARKs for NP are required. Constant-size constructions based on SNARKs are straightforward, but come with several inconveniences. Notably they would require to prove group operations encoded as an arithmetic circuit leading to signing time overheads and more complicated deployments due to implementations that must be tailored to specific cryptographic groups.

Insecurity of the [KMS26] ring signatures. Unfortunately, our attacks on collision resistance inherently make the ring signature constructed in [KMS26] not unforgeable. Furthermore, the anonymity of their construction hinges on an extra property required for their accumulator. They require that it is possible to re-randomize an accumulator, a witness, and a message, so that applying such randomization to two different accumulated elements produces indistinguishable results. As an additional contribution we identify a flaw in this re-randomization procedure that leads to distinguish, via simple pairing checks, the original message involved in the membership proof. In the ring signature application, this turns into identifying the signer, i.e., breaking anonymity. In their approach, public keys are statically anchored to a position i and thus a public key is of the form $\mathbf{pk}_i = (M_{i0}, M_{i1})$ and $\mathbf{pk}_i^{\text{aux}} = (T_{ij})_{j \in [i+1, i+q] \setminus \{q+1\}}$, where $\mathbf{pk}_i^{\text{aux}}$ is given to allow users to accumulate $\mathbf{pk}_i$ in position i (see the message space of Scheme 2). The re-randomized public key, which is then included in the ring signature, is computed as $\mathbf{pk}' = (M'_0 = M_{i0}^\mu, M'_1 = M_{i1}^\mu)$ with $\mu \leftarrow \mathbb{Z}_p$. Thus, we can test for every $j \in [q]$ the following pairing

$$e(M'_0, \hat{B}_j) = e(M'_1, \hat{g}_2)$$

and since $\mathbf{pk}'$ is a randomization of public key $\mathbf{pk}_i$, when $j = i$ we have that

$$e(M_{i0}^\mu, \hat{B}_i) = e(g_1^{m_i \mu}, \hat{g}_2^{\alpha^i}) = e(g_1^{m_i \alpha^i \mu}, \hat{g}_2) = e(B_i^{m_i \mu}, \hat{g}_2) = e(M_{i1}^\mu, \hat{g}_2)$$

thereby revealing the index i and thus the signer's key $\mathbf{pk}_i$.

Towards a ring signature based on our xSPA. Our xSPA cannot be plugged in the ring signature of [KMS26] because it is not re-randomizable. Since it does not seem straightforward to make our xSPA re-randomizable, in this section we propose a different template to construct a ring signature from SPA.

The idea of using accumulators to construct ring signatures is not new [MOY21, DKNS04], and it relies in representing the ring succinctly using an accumulator to the public keys and keeping the signer’s identity secret using zero-knowledge. However, the lack of structure preserving accumulators made instantiating this template in bilinear groups very inefficient. For example [MOY21] relies on an RSA accumulator [BdM94]. However, using an RSA accumulator has several disadvantages: 1) the accumulator’s message space consists of prime numbers and thus it does not nicely support public keys of signature schemes in the discrete-log setting; 2) RSA parameters are considerably larger than ones of elliptic curves for comparable security levels; 3) RSA accumulators do not combine nicely with NIZK over bilinear groups, creating yet another source of inefficiency. Using a structure-preserving accumulator allows overcoming all these issues, as it can be efficiently integrated with zero-knowledge proofs over bilinear groups.

High-level overview of our construction. We follow a very similar template to the one of [MOY21]. The idea of our construction is the following. Analogous to what was done in [KMS26], we define a variant of the BLS signature scheme [BLS04] called BLS_{acc} that has a public key space compatible with our xSPA accumulator. In particular, the public key $\text{pk} = g^{\text{sk}}$ is in $\mathbb{G}_1$ and it is augmented with q auxiliary elements of the form $(B_j^{\text{sk}})_{j \in [1, q]}$ allowing its accumulation with our SPA. To sign a message, the signer uses the BLS_{acc} scheme and then proves, via a NIZK, the knowledge of a BLS signature $\hat{\sigma} \in \mathbb{G}_2$ s.t. it verifies under one of the public keys belonging to the ring. It may seem that such a proof is linear in the size of the ring, but here is where the accumulator comes into play. Since the ring is public, the verifier of the signature can re-compute the accumulator to the ring. The signer will just have to prove that it knows an accumulator witness certifying membership of the signing public key. If such membership witness is of constant size and so is the accumulator’s verification equation (as in our xSPA), the resulting signature will be of constant size. An apparent obstacle is that the verification of xSPA needs to assure that the claimed element is not $1_{\mathbb{G}_1}$, and inequality checks are notoriously difficult to prove in zero knowledge. However, we bypass this issue by shifting such check to the ring signature verifier: the signer publishes a re-randomization pk' of pk and proves its correctness; the verifier will just need to check that this re-randomized $\text{pk}' \neq 1_{\mathbb{G}_1}$. Notably, pk' blinds pk and thus reveals no information about the signer’s identity. For lack of space, the detailed construction and the security proofs are given in App. G.

We instantiated our construction using the Groth-Sahai proof system [GS08]: a signature includes 33 elements in $\mathbb{G}_1$ and 34 in $\mathbb{G}_2$. This amounts to roughly 4.8 kB using the BLS12-381 curve. We give a detailed analysis in App. H.

Acknowledgements. We would like to thank Stephan Krenn, Omid Mir, and Daniel Slamanig for providing feedback on our analysis of their work [KMS26]. Gennaro Avitabile did part of the work while affiliated with the IMDEA Software Institute. This work is part of the PICOCRYPT project that has received funding from the European Research Council (ERC) under the European

Union’s Horizon 2020 research and innovation programme (Grant agreement No. 101001283), and is partially funded by the grants PID2022-142290OB-I00 and CEX2024-001471-M, funded by MCIN/AEI/10.13039/501100011033/ FEDER, UE.

References

- Abe15. Masayuki Abe. Structure-preserving cryptography. Invited talk at Asiacrypt 2015, 2015.
- AFG⁺10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010.
- AHO12. Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Group to group commitments do not shrink. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 301–317. Springer, Berlin, Heidelberg, April 2012.
- BBF19. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Cham, August 2019.
- BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Berlin, Heidelberg, May 2005.
- BCG⁺14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.
- BdM94. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT’93*, volume 765 of *LNCS*, pages 274–285. Springer, Berlin, Heidelberg, May 1994.
- BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001.
- BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Cham, August 2020.
- BKM09. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, January 2009.
- BLS04. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
- BP97. Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT’97*, volume 1233 of *LNCS*, pages 480–494. Springer, Berlin, Heidelberg, May 1997.

- CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Berlin, Heidelberg, February / March 2013.
- CKS09. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, Berlin, Heidelberg, March 2009.
- CPZ18. Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang. Edrax: A cryptocurrency with stateless transaction validation. Cryptology ePrint Archive, Report 2018/968, 2018.
- DHS15. David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 127–144. Springer, Cham, April 2015.
- DKNS04. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, Berlin, Heidelberg, May 2004.
- dS25. Gonzalo Martínez de Sola. Structure preserving accumulators with applications to short threshold ring signatures. Master thesis, 2025.
- EG13. Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. Cryptology ePrint Archive, Report 2013/662, 2013.
- EG14. Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Berlin, Heidelberg, March 2014.
- FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.
- Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Berlin, Heidelberg, December 2006.
- Gro15. Jens Groth. Efficient fully structure-preserving signatures for large messages. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 239–259. Springer, Berlin, Heidelberg, November / December 2015.
- GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008.
- Kil94. Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th FOCS*, pages 466–477. IEEE Computer Society Press, November 1994.
- KMS. Stephan Krenn, Omid Mir, and Daniel Slamanig. Structure-preserving compressing primitives: Vector commitments and accumulators and applications. Cryptology ePrint Archive, Paper 2024/1619. <https://eprint.iacr.org/archive/2024/1619/20260227:091354>.
- KMS26. Stephan Krenn, Omid Mir, and Daniel Slamanig. Structure-preserving compressing primitives: Vector commitments and accumulators and appli-

- cations. In Shi Bai and Edoardo Persichetti, editors, *PKC 2026, Part II*, volume 16552 of *LNCS*, pages 389–423. Springer, Cham, May 2026.
- Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012.
- LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Cham, August 2019.
- LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Berlin, Heidelberg, February 2010.
- MGGR13. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.
- Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.
- MOY21. Alexander Munch-Hansen, Claudio Orlandi, and Sophia Yakubov. Stronger notions and a more efficient construction of threshold ring signatures. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*, Lecture Notes in Computer Science, pages 363–381. Springer, 2021.
- MRK03. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th FOCS*, pages 80–91. IEEE Computer Society Press, October 2003.
- Nao02. Moni Naor. Deniable ring authentication. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 481–498. Springer, Berlin, Heidelberg, August 2002.
- RAVR21. Antonio Russo, Antonio Fernández Anta, María Isabel González Vasco, and Simon Pietro Romano. Chirotonia: A Scalable and Secure e-Voting Framework based on Blockchains and Linkable Ring Signatures. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 417–424, 2021.
- RST01. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Berlin, Heidelberg, December 2001.
- Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT’97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.
- TMSS22. Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, Fritz Schmid, and Dominique Schröder. Verifiable timed linkable ring signatures for scalable payments for monero. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part II*, volume 13555 of *LNCS*, pages 467–486. Springer, Cham, September 2022.
- TW05. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Ex-*

A Attack Against the Collision Resistance of the q-DHE SPA of Section 3.4

The following attack adapts Attack 1 against the q-DHE accumulator from Scheme 2 and the collision resistance notion of Theorem 1.

Attack 3 *Let Acc be the q-DHE SPA scheme presented in Scheme 2. There is an efficient adversary that breaks the collision resistance notion proposed in Theorem 1 with overwhelming probability.*

Proof. We can build an adversary $\mathcal{A}$ that with overwhelming probability efficiently provides an honest commitment to a set, and is able to open a non-accumulated message.

Given public parameters pp and any set $\mathcal{X} = \{(\mathbf{M}_i, \mathbf{T}_i)\}_{i=1}^q \in \mathcal{M}_q^{\text{pp}}$, adversary $\mathcal{A}(\text{pp}, \mathcal{X})$ honestly computes an accumulator to set $\mathcal{X}$ by computing

$$\text{Acc}_{\mathcal{X}} = g_1^r \prod_{i \in [q]} M_{i1}.$$

Next $\mathcal{A}$ chooses an arbitrary value $m \in \mathbb{Z}_p$ such that $m \neq m_i$ for every $i \in [q]$. Then, for any position $i \in [q]$, $\mathcal{A}$ generates an opening for message $\mathbf{M}^* = (g_1^m, \prod_{i \in [q]} M_{i1})$, by setting $\rho = 1$, $W_{i1}^* = B_{q+1-i}^r$ and $\hat{W}_{i2}^* = B_{q+1-i}$. Finally, $\mathcal{A}$ returns $((r, \rho), (\mathbf{M}^*, (W_{i1}^*, \hat{W}_{i2}^*)), \text{Acc}_{\mathcal{X}})$.

Note that if $\prod_{i \in [q]} M_{i1} \neq M_{j1}$ for every $j \in [q]$, then $M_1^* \neq M_{j1}$ for every $j \in [q]$. And thus, $\text{Acc.Verify}(\text{pp}, \text{Acc}_{\mathcal{X}}, \mathbf{M}^*, (W_{i1}^*, \hat{W}_{i2}^*)) = 1$ accepts since

$$\underbrace{e(g_1^r \prod_{i \in [q]} M_{i1}, \hat{B}_{q+1-i})}_{e(\text{com}, \hat{W}_{i2}^*)} = \underbrace{e(B_{q+1-i}^r, \hat{g}_2)}_{e(W_{i1}^*, \hat{g}_2)} \cdot \underbrace{e(\prod_{i \in [q]} M_{i1}, \hat{B}_{q+1-i})}_{e(M_{i1}^*, \hat{W}_{i2}^*)}.$$

□

Notice that Attack 3 succeeds whenever $\prod_{i \in [q]} M_{i1} \neq M_{j1}$ for every $j \in [q]$, it is easy to see that there are a fair amount of sets that satisfy this property. We then can conclude that there exists at least a PPT adversary and a set $\mathcal{X} \in \mathcal{M}_q^{\text{pp}}$ that refutes Theorem 1.

We additionally give an attack that exploits the hiding property of the q-DHE accumulator:

Attack 4 *Let Acc be the 1-DHE SPA scheme presented in Scheme 2. There is an efficient adversary that breaks the collision resistance notion proposed in Theorem 1 with probability 1.*

Proof. We can build an adversary $\mathcal{A}$ that efficiently provides an honest commitment to a set, and at the same time is able to open a non-accumulated message.

Given public parameters $\mathbf{pp}$ and any set $\mathcal{X} = \{(\mathbf{M}_i, \mathbf{T}_i)\}_{i=1}^q \in \mathcal{M}_q^{\mathbf{pp}}$, adversary $\mathcal{A}(\mathbf{pp}, \mathcal{X})$ chooses an index $i \in [q]$ and $r \in \mathbb{Z}_p$, such that,

$$g_1^r M_{i1} \neq M_{j1} \quad \forall j \in [q].$$

Then honestly commits to the set $\mathcal{X}$ by setting

$$\text{Acc}_{\mathcal{X}} = g_1^r \prod_{i \in [q]} M_{i1}.$$

Finally, opens the collision $((\rho, r), (W_1, \hat{W}_2), \mathbf{M}^*)$, by setting

$$\mathbf{M}^* = (g_1, g_1^r M_{i1}) \quad \wedge \quad W_1 = \prod_{j \in [q] \setminus \{i\}} T_{j, q+1-i+j} \quad \wedge \quad W_2 = B_{q+1-i} \quad \wedge \quad \rho = 1$$

such that,

$$\underbrace{e(g_1^r \prod_{i \in [q]} M_{i1}, \hat{B}_{q+1-i})}_{e(\text{Acc}_{\mathcal{X}}, \hat{W}_2)} = e(\underbrace{\prod_{j \in [q] \setminus \{i\}} T_{j, q+1-i+j}}_{e(W_1, \hat{g}_2)}, \underbrace{e(g_1^r M_{i1}, \hat{B}_{q+1-i})}_{e(M_1^*, \hat{W}_2)}).$$

Since $M_1^* = g_1^r M_{i1} \neq M_{j1} \quad \forall j \in [q]$, the attack succeeds. $\square$

These attacks can be avoided in a similar way as Attack 1 on SPVC_1 is avoided, by just adding the verification equation shown in (6). Yet, after this change it would remain to prove security under the notion in Theorem 1, which cannot rely on assuming the position binding of SPVC_1 's variant in which equations (6) are checked during verification, since this variant is also not binding due to Attack 2).

An option to make the SPA secure would be to include in the membership proof all the tags related to the element and to require verification of the full message, as in SPVC_0 . However, this approach would undermine the main goal of achieving $O(1)$ -size openings and, consequently, $O(1)$ verification time.

B Correctness of Our SPVC

Here we prove that Scheme 3 is correct. Let $\mathbf{msg} = (\mathbf{msg}_j)_{j \in [q]} \in \mathcal{V}_q^{\mathbf{pp}}$ (where $\mathcal{V}_q^{\mathbf{pp}}$ is as defined in 11) and $(\pi_{i1}, \pi_{i2}, \pi_{i3}, \pi_{i4}, \pi_{i5}) \leftarrow \text{Open}_2(\mathbf{pp}, \mathbf{msg}_i, i, \mathbf{aux})$ be an honest opening. We analyze correctness of all the pairing equations that involve

Verify₂:

$$\begin{aligned}
e(\text{com}, \hat{B}_{q+1-i}) &= e\left(\prod_{j \in [q]} g_1^{m_j \alpha^j}, \hat{g}_2^{\alpha^{q+1-i}}\right) \\
&= e\left(\prod_{j \in [q]} g_1^{(m_j \alpha^j) \alpha^{q+1-i}}, \hat{g}_2\right) \\
&= e\left(\prod_{j \in [q]} g_1^{m_j \alpha^{q+1-i+j}}, \hat{g}_2\right) \\
&= e\left(\prod_{j \in [q] \setminus \{i\}} g_1^{m_j \alpha^{q+1-i+j}}, \hat{g}_2\right) \cdot e(g_1^{m_i \alpha^{q+1}}, \hat{g}_2) \\
&= e\left(\prod_{j \in [i-1]} g_1^{m_j \alpha^{q+1-i+j}}, \hat{g}_2\right) \cdot e\left(\prod_{j \in [i+1, q]} g_1^{m_j \alpha^{q+1-i+j}}, \hat{g}_2\right) \cdot e(g_1^{m_i \alpha^{q+1}}, \hat{g}_2) \\
&= e\left(\prod_{j \in [i-1]} g_1^{m_j \alpha^{q+1-i+j}}, \hat{g}_2\right) \cdot e\left(\prod_{j \in [i+1, q]} g_1^{m_j \alpha^{1-i+j}}, \hat{g}_2^{\alpha^q}\right) \cdot e(g_1^{m_i \alpha^q}, \hat{g}_2^\alpha) \\
&= e\left(\prod_{j \in [i-1]} T_{j, q+1-i+j}, \hat{g}_2\right) \cdot e\left(\prod_{j \in [i+1, q]} T_{j, 1-i+j}, \hat{B}_q\right) \cdot e(T_{iq}, \hat{B}_1) \\
&= e(\pi_{i1}, \hat{g}_2) \cdot e(\pi_{i2}, \hat{B}_q) \cdot e(\pi_{i4}, \hat{B}_1)
\end{aligned}$$

we also have that

$$\begin{aligned}
e(M_{i0}, \hat{B}_q) &= e(g_1^{m_i}, \hat{g}_2^{\alpha^q}) \\
&= e(g_1^{m_i \alpha^q}, \hat{g}_2) \\
&= e(T_{iq}, \hat{g}_2) \\
&= e(\pi_{i4}, \hat{g}_2),
\end{aligned}$$

and finally

$$\begin{aligned}
e(\pi_{i3}, \hat{B}_2) &= e\left(\prod_{j \in [i+1, q]} T_{j, j-i-1}, \hat{g}_2^{\alpha^2}\right) \\
&= e\left(\prod_{j \in [i+1, q]} g_1^{(m_j \alpha^{j-i-1}) \alpha^2}, \hat{g}_2\right) \\
&= e\left(\prod_{j \in [i+1, q]} g_1^{m_j \alpha^{1-i+j}}, \hat{g}_2\right) \\
&= e\left(\prod_{j \in [i+1, q]} T_{j, 1-i+j}, \hat{g}_2\right) \\
&= e(\pi_{i2}, \hat{g}_2)
\end{aligned}$$

C Correctness of Our xSPA

We now prove that Scheme 4 is correct. Correctness of our xSPA is closely related to the set verification routine presented in Section 5.2. Let $(\mathcal{X}, \mathcal{X}_{\text{aux}}) \in \mathcal{S}_q^{\text{pp}}$,

such that $|\mathcal{X}| = n \leq q$. Since $1 \leftarrow \text{VerifySet}(\text{pp}, \mathcal{X}, \mathcal{X}_{\text{aux}})$, by setting $\text{ord}(\mathcal{X}) = [x_1, \dots, x_n]$ we have that for any $i \in [q]$,

$$e(x_i, \hat{B}_j) = e(\tau_{ij}, \hat{g}_2)$$

for every $j \in [q]$, where $\tau_i = (\tau_{ij})_{j \in [q]} \in \mathcal{X}_{\text{aux}}$ is the corresponding auxiliary information associated to x_i . Since there exists $m_i \in \mathbb{Z}_p$ such that $x_i = g_1^{m_i}$ then

$$m_i = d\log_{g_1} x_i = d\log_{B_j} \tau_{ij} \quad (32)$$

for every $i \in [q]$ and every $j \in [q]$. And thus we can rewrite the message-tag pairs in the following way

$$(x_i, \tau_i) = (g_1^{m_i}, (B_i^{m_j})_{j \in [q]})$$

for every message $x_i \in \text{ord}(\mathcal{X})$ and the corresponding auxiliary information $\tau_i \in \mathcal{X}_{\text{aux}}$.

Given this fact, correctness of our xSPA scheme follows from the correctness of our SPVC scheme (see Appendix B), and the correctness of the structure preserving signature [Gro15] used to sign the $\hat{B}_i$ elements from the crs.

D Missing Proofs of the Supporting Claims for the Proof of Theorem 4

We now provide formal proofs for each of the claims made in the proof of Theorem 4. Throughout this section, we assume familiarity with the proof of Theorem 4, as we adopt the same notation and conventions introduced therein.

Claim 1. If $T_1(\mathbf{X}) \equiv 0$, there exists $l \in [q]$ such that $P_{\sigma_r}(\mathbf{X}) = a_{\sigma_r, r_l} 1/Z_l$ with $a_{\sigma_r, r_l} \neq 0$.

Proof. Since $T_1(\mathbf{X}) \equiv 0$, we have that

$$P_{\sigma_r}(\mathbf{X}) \hat{Q}_{\sigma_s}(\mathbf{X}) = (V + Y), \quad (33)$$

which expanding both side of the equation, yields

$$\begin{aligned} & \left(a_{\sigma_r} + a_{\sigma_r, v} V + \sum_{i \in [q]} a_{\sigma_r, \alpha_i} A^i \right) \\ & + \sum_{j \in [q]} a_{\sigma_r, r_j} 1/Z_j \\ & \cdot \left(b_{\sigma_s} + b_{\sigma_s, y} Y + \sum_{i \in [q]} b_{\sigma_s, \alpha_i} A^i + \sum_{j \in [q]} b_{\sigma_s, s_j} Z_j (Y + V) \right) \\ & + \sum_{k \in [q]} b_{\sigma_s, t_k} Z_k (A^i + VY) \end{aligned} = V + Y.$$

For this equation to hold, all the coefficients of corresponding variables and powers must match.

We start by noting that $P_{\sigma_r}(\mathbf{X}) \not\equiv 0 \wedge \hat{Q}_{\sigma_s}(\mathbf{X}) \not\equiv 0$ since otherwise we cannot match the other side of equation (33). Please observe that negative powers, i.e. $1/Z_j$, only appear on the left hand side of the equation.

For sake of contradiction, let us start by assuming that $a_{\sigma_r, r_j} = 0$ for every $j \in [q]$. Then in order not to have terms involving variables Z_j 's we must have

$$\sum_{j \in [q]} b_{\sigma_s, s_j} Z_j (Y + V) + \sum_{k \in [q]} b_{\sigma_s, t_k} Z_k (A^i + VY) = 0.$$

The term Y in the right side of equation (33) give us $a_{\sigma_r} b_{\sigma_s, y} = 1$ and the term V gives us that $a_{\sigma_r, v} b_{\sigma_s} = 1$. This means that $a_{\sigma_r} \neq 0 \wedge b_{\sigma_s} \neq 0$, and therefore we reach a contradiction since the constant term should be $a_{\sigma_r} b_{\sigma_s} = 0$. We then conclude that there must exist at least some $l \in [q]$ such that $a_{\sigma_r, r_l} \neq 0$.

Fixing such $l \in [q]$, we now have the term $a_{\sigma_r, r_l} b_{\sigma_s} 1/Z_l = 0$, which lead us to $b_{\sigma_s} = 0$. The term $b_{\sigma_s, y} a_{\sigma_r, r_l} Y/Z_l = 0$ leads us to $b_{\sigma_s, y} = 0$.

Similarly, the coefficients of the terms $Z_k VY/Z_l$ satisfy $a_{\sigma_r, r_l} b_{\sigma_s, t_k} = 0$ for every $k \neq l$, and therefore $b_{\sigma_s, t_k} = 0$ for $k \neq l$. Also, in the case $k = l$, looking at the resulting term $a_{\sigma_r, r_l} b_{\sigma_s, t_l} VY$, we can also conclude that $b_{\sigma_s, t_l} = 0$.

Now, for every $i \in [q]$, the equality $a_{\sigma_r, r_l} b_{\sigma_s, \alpha_i} A^i/Z_l = 0$ leads us to $b_{\sigma_s, \alpha_i} = 0$.

The terms $a_{\sigma_r, r_l} b_{\sigma_s, s_j} (V + Y) Z_j/Z_l$ gives us $b_{\sigma_s, s_j} = 0$ for every $j \neq l$. In order to match the right side of equation (33), we see that $b_{\sigma_s, s_l} = 1/a_{\sigma_r, r_l} \neq 0$, showing us that

$$\hat{Q}_{\sigma_s}(\mathbf{X}) = \frac{1}{a_{\sigma_r, r_l}} Z_l (V + Y)$$

Now, let us analyze the structure of $P_{\sigma_r}(\mathbf{X})$. The term $a_{\sigma_r, v} b_{\sigma_s, s_l} Z_l V^2 = 0$ gives us $a_{\sigma_r, v} = 0$. The term $a_{\sigma_r, u} b_{\sigma_s, s_l} U Z_l Y = 0$ leads us to $a_{\sigma_r, u} = 0$. The term $a_{\sigma_r, \alpha_i} b_{\sigma_s, s_l} A^i Z_l V = 0$ leads us to $a_{\sigma_r, \alpha_i} = 0$, for every $i \in [q]$. And for every $i \neq l$ the terms $a_{\sigma_r, r_i} b_{\sigma_s, s_l} Z_l (V + Y)/Z_i = 0$ leads us to $a_{\sigma_r, r_i} = 0$. Therefore, our analysis shows that

$$P_{\sigma_r}(\mathbf{X}) = a_{\sigma_r, r_l} 1/Z_l$$

Finally, $a_{\sigma_r, r_l} \neq 0$ since $P_{\sigma_r}(\mathbf{X}) \not\equiv 0$. □

Claim 2. If $T_2(\mathbf{X}) \equiv 0$ and $P_{\sigma_r}(\mathbf{X}) = a_{\sigma_r, r_l} 1/Z_l \neq 0$ for some $l \in [q]$, then

$$\hat{Q}_{w_6}(\mathbf{X}) = A^l$$

Proof. Since $T_2(\mathbf{X}) \equiv 0$, we have that

$$P_{\sigma_r}(\mathbf{X}) \hat{Q}_{\sigma_t}(\mathbf{X}) = \hat{Q}_{w_6}(\mathbf{X}) + V \cdot Y \tag{34}$$

which by expanding the left hand side yields

$$\begin{aligned}
& a_{\sigma_r, r_l} 1/Z_l \left(b_{\sigma_t} + b_{\sigma_t, y} Y + \sum_{i \in [q]} b_{\sigma_t, \alpha_i} A^i + \sum_{j \in [q]} b_{\sigma_t, s_j} Z_j (Y + V) \right) \\
& + \sum_{k \in [q]} b_{\sigma_t, t_k} Z_k (A^i + VY) \\
& = \hat{Q}_{w_6}(\mathbf{X}) + V \cdot Y
\end{aligned}$$

First of all, note that since $\hat{Q}_{w_6}$ is a polynomial representing an element in $\mathbb{G}_2$, the right side of the equation does not contain negative powers, i.e. $1/Z_j$.

The terms $a_{\sigma_r, r_l} b_{\sigma_t} 1/Z_l = 0$ and $a_{\sigma_r, r_l} b_{\sigma_t, y} Y/Z_l = 0$, show us that $b_{\sigma_t} = 0$ and $b_{\sigma_t, y} = 0$, since $a_{\sigma_r, r_l} \neq 0$. The terms $a_{\sigma_r, r_l} b_{\sigma_t, \alpha_i} A^i/Z_l$ cannot be canceled by any other product, meaning that $a_{\sigma_r, r_l} b_{\sigma_t, \alpha_i} = 0$, and in consequence $b_{\sigma_t, \alpha_i} = 0$ for every $i \in [q]$.

We can easily see that $a_{\sigma_r, r_l} b_{\sigma_t, s_j} Z_j V/Z_l = 0$ for $j \neq l$, which means that $b_{\sigma_t, s_j} = 0$ for $j \neq l$. Note that the special case of the term $a_{\sigma_r, r_l} b_{\sigma_t, s_l} V$ also leads us to $b_{\sigma_t, s_l} = 0$, since in the right hand side of equation (34) there is no term just containing variable V , because $\hat{Q}_{w_6}$ is a polynomial representing an element in $\mathbb{G}_2$ (see (30)).

Finally, for the terms of the form $a_{\sigma_r, r_l} b_{\sigma_t, t_k} Z_k U A^k/Z_l$ with $k \neq l$, there are no matching elements in the right side of equation (34), and thus $b_{\sigma_t, t_k} = 0$ for every $k \neq l$. Then, our analysis shows that

$$\hat{Q}_{\sigma_t}(\mathbf{X}) = a_{\sigma_r, r_l} b_{\sigma_t, t_l} (A^l + VY)$$

Now, since polynomial representations of elements in $\mathbb{G}_2$ do not contain terms of the form VY , inspecting equation (34) implies that $a_{\sigma_r, r_l} b_{\sigma_t, t_l} = 1$, and consequently,

$$A^l + VY = \hat{Q}_{w_6}(\mathbf{X}) + VY$$

which by simple observation lead us to $\hat{Q}_{w_6}(\mathbf{X}) = A^l$. $\square$

Claim 3. If $T_3(\mathbf{X}) \equiv 0$ and $x^* \neq 1_{\mathbb{G}_1}$, then $P_{w_4}(\mathbf{X}) = a_{w_4, \alpha_q} A^q$ and $P_{x^*}(\mathbf{X}) = a_{w_4, \alpha_q}$. Thereby, $\mathcal{B}$ recovers the discrete logarithm a_{w_4, α_q} of x^* from $\mathcal{A}$'s algebraic explanations.

Proof. First of all, note that $x^* \neq 1_{\mathbb{G}_1}$ and then $P_{x^*}(\mathbf{X}) \neq 0$.

Now, since $T_3(\mathbf{X}) \equiv 0$, we have that $P_{x^*}(\mathbf{X}) A^q = P_{w_4}(\mathbf{X})$, which means that $P_{w_4}(\mathbf{X}) \neq 0$ and A^q divides $P_{w_4}(\mathbf{X})$. Considering the polynomial representation of elements in $\mathbb{G}_1$ given in (29), the degree of $P_{w_4}(\mathbf{X})$ is at most q , and we have that

$$P_{w_4}(\mathbf{X}) = a_{w_4, \alpha_q} A^q$$

Finally, since $P_{x^*}(\mathbf{X}) \neq 0$ and $P_{x^*}(\mathbf{X}) A^q = P_{w_4}(\mathbf{X})$, we can say that $P_{x^*}(\mathbf{X}) = a_{w_4, \alpha_q} \neq 0$ and $\mathcal{B}$ extracts the discrete logarithm a_{w_4, α_q} of x^* . $\square$

Claim 4. If $T_4(\mathbf{X}) \equiv 0$ then A^2 divides $P_{w_2}(\mathbf{X})$.

Proof. Since $T_4(\mathbf{X}) \equiv 0$, we have that $P_{w_3}(\mathbf{X})A^2 = P_{w_2}(\mathbf{X})$, which means that A^2 divides P_{w_2} . Then, given the polynomial representation of elements in $\mathbb{G}_1$ as defined in (29), P_{w_2} is of the form

$$P_{w_2}(\mathbf{X}) = \sum_{j \in [2, q]} a_{w_2, \alpha_j} A^j$$

□

E Ring Signatures

A ring signature scheme is a digital signature scheme that allows a user to sign a message while hiding their identity within a set of public keys called ring. It is defined as follows.

Definition 15 (Ring Signature). *A ring signature (RS) scheme is a tuple of PPT algorithms (Setup, KeyGen, Sign, Verify) defined as follows*

Setup($1^\lambda, 1^q$) : *On input the security parameter 1^λ and a bound on the number of users q , outputs the public parameters $\mathbf{pp}$.*

KeyGen($\mathbf{pp}$) : *On input the public parameters, returns a user key pair $(\mathbf{pk}, \mathbf{sk})$.*

Sign($\mathbf{pp}, \mathbf{pk}_i, \mathbf{sk}_i, m, \mathcal{R}$) : *On input the public parameters $\mathbf{pp}$, a public and secret key $\mathbf{pk}_i, \mathbf{sk}_i$, a message m , and a ring $\mathcal{R} = (\mathbf{pk}_j)_{j \in [n]}$ of $n \leq q$ public keys such that $\mathbf{pk}_i \in \mathcal{R}$, outputs a signature σ .*

Verify($\mathbf{pp}, m, \sigma, \mathcal{R}$) : *On input the public parameters $\mathbf{pp}$, a message m , a signature σ , and a ring $\mathcal{R}$, outputs 1 to accept and 0 to reject.*

A ring signature must satisfy completeness, unforgeability, and anonymity. Completeness says that for any security parameter λ , any upper bound q , any $\{(\mathbf{sk}_i, \mathbf{pk}_i)\}_{i \in [n]}$ generated via **KeyGen**, any $s \in [n]$, and any message m , we have that $\text{Verify}(\mathbf{pp}, m, \text{Sign}(\mathbf{pp}, \mathbf{pk}_s, \mathbf{sk}_s, m, \mathcal{R}), \mathcal{R}) = 1$, where $\mathbf{pp} \leftarrow \text{Setup}(\lambda, 1^q)$, $\mathcal{R} = (\mathbf{pk}_1, \dots, \mathbf{pk}_n)$, and $n \leq q$. We adapt the definitions of unforgeability and anonymity from [BKM09]. We redefine the equality of public keys in case such keys are tuples. In particular given $\mathbf{pk}_i = (\mathbf{pk}_0^i, \dots, \mathbf{pk}_n^i)$ and $\mathbf{pk}_j = (\mathbf{pk}_0^j, \dots, \mathbf{pk}_n^j)$, then $\mathbf{pk}_i = \mathbf{pk}_j$ iff $\mathbf{pk}_0^i = \mathbf{pk}_0^j$. Looking ahead, this minor adjustment will be useful when reducing to the security of our accumulator.

Definition 16 (Unforgeability). *An RS scheme provides unforgeability if for all PPT adversaries $\mathcal{A}$ it hold that:*

$$\Pr \left[\begin{array}{l} \mathbf{pp} \leftarrow \text{Setup}(1^\lambda, q), \\ \{(\mathbf{pk}_i, \mathbf{sk}_i)\}_{i \in [q]} \leftarrow \text{KeyGen}(\mathbf{pp}), \\ \mathcal{O} \leftarrow \{\text{Sig}(\cdot, \cdot, \cdot), \text{Key}(\cdot)\}, \\ (m^*, \sigma^*, \mathcal{R}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\mathbf{pk}_i\}_{i \in [q]}) \end{array} \middle| \begin{array}{l} \text{Verify}(m^*, \sigma^*, \mathcal{R}^*) = 1 \wedge \\ (\cdot, m^*, \mathcal{R}^*) \notin \mathcal{Q}_{\text{Sig}} \wedge \\ \mathcal{R}^* \subseteq \{\mathbf{pk}_i\}_{i \in [q] \setminus \mathcal{Q}_{\text{Key}}} \end{array} \right] \leq \text{negl}(\lambda),$$

where $\text{Sig}(i, m, \mathcal{R}) := \text{Sign}(\mathbf{sk}_i, m, \mathcal{R})$, Sig returns $\perp$ if $\mathbf{pk}_i \notin \mathcal{R}$ or $i \notin [q]$, and $\mathcal{Q}_{\text{Sig}}$ records the queries to Sig . Furthermore, $\text{Key}(i)$ returns $\mathbf{sk}_i$ and $\mathcal{Q}_{\text{Key}}$ records the queries to Key .

Definition 17 (Anonymity). An RS scheme provides anonymity if for all PPT adversaries $\mathcal{A}$ it holds that:

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, q), \ b \leftarrow \$_\{0,1\}, \\ \{(\text{pk}_i, \text{sk}_i)\}_{i \in [q]} \leftarrow \text{KeyGen}(\text{pp}), \\ (m, (j_0, j_1), \mathcal{R}, \text{st}) \leftarrow \mathcal{A}(\{(\text{pk}_i, \text{sk}_i)\}_{i \in [q]}), \\ \sigma \leftarrow \text{Sign}(\text{sk}_{j_b}, m, \mathcal{R}), \\ b' \leftarrow \mathcal{A}(\text{st}, \sigma) \end{array} \middle| \begin{array}{l} b = b' \wedge \\ \{\text{pk}_{j_0}, \text{pk}_{j_1}\} \subseteq \mathcal{R} \wedge \\ \mathcal{R} \subseteq \{\text{pk}_i\}_{i \in [q]} \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda).$$

F Non-Interactive Zero-Knowledge Proof Systems

A non-interactive zero-knowledge proof system (NIZK) allows a prover to convince a verifier that a given statement is true by sending a single proof message. We define them as follows using the definitions of [Gro06].

Definition 18. A non-interactive zero-knowledge proof system (NIZK) is a set of PPT algorithms $\text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify})$. It is defined w.r.t. a poly-time relation R and the corresponding NP-language L_R . The algorithm works as follows:

- $\text{Setup}(\lambda)$: On input the security parameter λ , outputs public parameters pp .
- $\text{Prove}(\text{pp}, x, w)$: On input the public parameters pp , the statement x , and the witness w , outputs a proof π .
- $\text{Verify}(\text{pp}, x, \pi)$: On input the public parameters pp , the statement x , and a proof π , outputs 1 to accept and 0 to reject.

We require completeness, zero knowledge, and extractability. Completeness says that for all $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ and $(x, w) \in R$, we have $V(\text{pp}, x, \pi) = 1$ for all proofs $\pi \leftarrow \text{Prove}(\text{pp}, x, w)$. The other two properties are defined below.

Definition 19 (Zero Knowledge). There exists a PPT simulator algorithm $S = (S_1, S_2)$ such that $S_1(1^\lambda)$ outputs (pp_S, τ_S) , and $S_2(\text{pp}_S, \tau_S, x)$ outputs a value π_S such that for all $(x, w) \in R$ and PPT adversaries $\mathcal{A}$, the following two interactions are indistinguishable:

1. In the first, we compute $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ and give $\mathcal{A}$ pp and oracle access to $P(\text{pp}, \cdot, \cdot)$ (where P will output $\perp$ on input (x, w) such that $(x, w) \notin R$, and $\text{Prove}(\text{pp}, x, w)$ otherwise).
2. In the second, we compute (pp_S, τ_S) and give $\mathcal{A}$ pp_S and oracle access to $S(\text{pp}_S, \tau_S, \cdot, \cdot)$, where, on input (x, w) , S outputs $S_2(\text{pp}_S, \tau_S, x)$ if $(x, w) \in R$ and $\perp$ otherwise.

Definition 20 (Extractability). There exists a polynomial-time extractor algorithm $E = (E_1, E_2)$ such that $E_1(1^\lambda)$ outputs (pp_E, τ_E) , and $E_2(\text{pp}_E, \tau_E, x, \pi)$ outputs a value w such that

1. A pp_E output by $E_1(1^\lambda)$ is indistinguishable from pp output by $\text{Setup}(1^\lambda)$.

2. For all PPT $\mathcal{A}$, the probability that $\mathcal{A}(\text{pp}_E, \tau_E)$ (where $(\text{pp}_E, \tau_E) \leftarrow E_1(1^\lambda)$) outputs (x, π) such that $V(\text{pp}, x, \pi) = 1$ and $(x, E_2(\text{pp}_E, \tau_E, x, \pi)) \notin R = 0$, is negligible.

$$\Pr \left[\begin{array}{l} V(\sigma, x, \pi) = 1 \\ (x, w) \notin R \end{array} \wedge \begin{array}{l} (\text{pp}_E, \tau_E) \leftarrow E_1(1^\lambda); \\ (x, \pi) \leftarrow \mathcal{A}(\text{pp}_E); \\ w \leftarrow E_2(\text{pp}_E, \tau_E, x, \pi) \end{array} \right] \leq \text{negl}(\lambda).$$

G Our Constant-size Ring Signature

We follow a very similar template to the one of [MOY21]. The idea of our construction is the following. Analogous to what done in [KMS26], we define a variant of the BLS signature scheme [BLS04] called BLS_{acc} that has a public key space compatible with our xSPA accumulator.

Scheme 5 BLS_{acc} is defined with respect to our xSPA as follows:

Setup $(\lambda, 1^q)$: Get $(\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}) \leftarrow \text{xSPA.Setup}(\lambda, 1^q)$ and choose an hash function $H\{0, 1\}^* \rightarrow \mathbb{G}_2$. Output $\text{pp} = (\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H)$.
KeyGen (pp) : Sample $\text{sk} \leftarrow \mathbb{Z}_p$. Define $\text{pk}_0 = g_1^{\text{sk}}$ and $\text{pk}_{\text{aux}} = (B_j^{\text{sk}})_{j \in [q]}$, where the B_j are taken from pp_{Acc} . Return $\text{pk} = (\text{pk}_0, \text{pk}_{\text{aux}})$ and sk .
Sign (sk, m) : Compute $\hat{h} \leftarrow H(m)$, $\hat{\sigma} \leftarrow \hat{h}^{\text{sk}}$ and return $\hat{\sigma}$.
Verify $(\text{pk}, m, \hat{\sigma})$: Parse $\text{pk} = (\text{pk}_0, \text{pk}_{\text{aux}})$, compute $\hat{h} \leftarrow H(m)$ and check that $e(g_1, \hat{\sigma}) = e(\text{pk}, \hat{h})$. Return 1 if the check is successful and 0 otherwise.

It is trivial to show that if regular BLS is EUF-CMA then so is BLS_{acc} ; we refer to [KMS26] for a very similar proof.

In our ring signature, to sign a message, the signer uses the BLS_{acc} scheme defined above. She then proves, via a NIZK, the knowledge of $\hat{\sigma}$ s.t. it verifies under one of the public keys belonging to the ring.

It may seem that such a proof is linear in the size of the ring, but here is where the accumulator comes into play. Since the ring is public, the verifier of the signature can re-compute the accumulator to the ring. The signer will just have to prove that it knows an accumulator witness certifying membership of the signing public key. If such membership witness is of constant size and so is the accumulator's verification equation (as in our xSPA), the resulting signature will be of constant size.

We now give a detailed description of our ring signature which uses a NIZK for the following relation R

$$R = \left\{ \begin{array}{l} x = (\text{vp}_{\text{Acc}}, \text{Acc}, \hat{h}, \text{pk}'), \\ w = (\text{pk}, \hat{\sigma}, \text{wit}_{\text{pk}}, r) \end{array} \middle| \begin{array}{l} e(g, \hat{\sigma}) = e(\text{pk}, \hat{h}), \\ \text{xSPA.Verify}'(\text{vp}_{\text{Acc}}, \text{Acc}, \text{wit}_{\text{pk}}, \text{pk}) = 1, \\ \text{pk}' = \text{pk}^r \end{array} \right\}$$

where $\text{xSPA.Verify}'$ coincides xSPA.Verify except it does not check that the message, in this case pk , is $\neq 1_{\mathbb{G}_1}$. This check is shifted to the ring signature verifier

by letting the signer publish a re-randomization pk' of pk and proving its correctness. The verifier will just need to check that this re-randomized $\text{pk}' \neq 1_{\mathbb{G}_1}$. Notably pk' blinds pk and thus reveals no information about the signer's identity. More in details, the construction works as follows:

Scheme 6 (Our constant-size ring signature) *Our scheme is built upon our xSPA, the accumulator-compatible BLS_{acc} , and a NIZK for the relation $\mathcal{R}$ described above. The scheme works as follows:*

Setup($1^\lambda, 1^q$) : $(\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H) \leftarrow \text{BLS}_{\text{acc}}.\text{Setup}(1^\lambda, 1^q)$, $\text{pp}_{\text{NIZK}} \leftarrow \text{NIZK}.\text{Setup}(1^\lambda)$.
Return $\text{pp} = (\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H, \text{pp}_{\text{NIZK}})$.
KeyGen(pp) : *Return* $\text{BLS}_{\text{acc}}.\text{KeyGen}(\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H)$.
Sign($\text{pp}, \text{pk}_i, \text{sk}_i, m, \mathcal{R}$) : *Parse* $\mathcal{R}$ as two vectors $\mathcal{X} = (\text{pk}_0^1, \dots, \text{pk}_0^n)$ and $\mathcal{X}_{\text{aux}} = (\text{pk}_{\text{aux}}^1, \dots, \text{pk}_{\text{aux}}^n)$. *Parse* $\text{pp} = (\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H, \text{pp}_{\text{NIZK}})$ and accumulate all the public keys via $(\text{Acc}, \text{aux}) \leftarrow \text{Eval}(\text{pp}_{\text{Acc}}, \mathcal{X}, \mathcal{X}_{\text{aux}})$. *Parse* $\text{pk}_i = (\text{pk}_0^i, \text{pk}_{\text{aux}}^i)$, and generate the membership witness $\text{wit}_{\text{pk}_i} \leftarrow \text{WitCreate}(\text{pp}, \text{Acc}, \text{pk}_0^i, \text{aux})$. Finally, compute $\hat{\sigma} \leftarrow \text{BLS}_{\text{acc}}.\text{Sign}(\text{sk}_i, m)$, $\pi \leftarrow \text{NIZK}.\text{Prove}(\text{pp}_{\text{NIZK}}, x = (\text{vp}, \text{Acc}, \hat{h}, \text{pk}'), w = (\text{pk}_i, \hat{\sigma}, \text{wit}_{\text{pk}_i}))$, where $\text{pk}' \leftarrow \text{pk}^r$ for a random $r \in \mathbb{Z}_p$. *Return* $\sigma = (\pi, \text{pk}')$.
Verify($\text{pp}, m, \sigma, \mathcal{R}$) : *Parse* $\sigma = (\pi, \text{pk}')$. If $\text{pk}' = 1_{\mathbb{G}_1}$ return 0. Otherwise, *parse* $\mathcal{R}$ as two vectors $\mathcal{X} = (\text{pk}_0^1, \dots, \text{pk}_0^n)$ and $\mathcal{X}_{\text{aux}} = (\text{pk}_{\text{aux}}^1, \dots, \text{pk}_{\text{aux}}^n)$. *Parse* $\text{pp} = (\text{pp}_{\text{Acc}}, \text{vp}_{\text{Acc}}, H, \text{pp}_{\text{NIZK}})$ and accumulate all the public keys via $(\text{Acc}, \text{aux}) \leftarrow \text{Eval}(\text{pp}_{\text{Acc}}, \mathcal{X}, \mathcal{X}_{\text{aux}})$. Compute $\hat{h} \leftarrow H(m)$ and return $\text{NIZK}.\text{Verify}(\text{pp}_{\text{NIZK}}, x = (\text{vp}_{\text{Acc}}, \text{Acc}, \hat{h}, \text{pk}'), \pi)$.

Correctness follows by inspection. We now prove the unforgeability and anonymity properties.

Theorem 6. *If NIZK is extractable (Def. 20), BLS_{acc} is EUF-CMA, and xSPA is collision resistant (Def. 12), then Scheme 6 is unforgeable (Def. 16).*

Proof. The proof goes through the following hybrids:

$\mathcal{H}_0$: It is the ring signature unforgeability game.

$\mathcal{H}_1$: It is identical to $\mathcal{H}_0$ with the difference that the public parameters of NIZK are generated in extraction mode (i.e. using $\mathcal{E}_1$ instead of $\text{NIZK}.\text{Setup}$).

$\mathcal{H}_1$ is clearly indistinguishable from $\mathcal{H}_0$ due to the extractability of NIZK.

Next, assuming $\mathcal{A}$ breaks the unforgeability of the ring signature in $\mathcal{H}_1$, we build a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the EUF-CMA of BLS_{acc} . $\mathcal{B}$ selects a random index $j \in [q]$; it then generates all the other keys normally while it gets pk_j from the challenger of the EUF-CMA of BLS_{acc} . $\mathcal{B}$ handles the queries from $\mathcal{A}$ in the following way:

- $\text{Sig}(i, m, \mathcal{R})$: it is handled exactly as in $\mathcal{H}_1$, except that if $i = j$, $\mathcal{B}$ queries the challenger of the BLS_{acc} EUF-CMA game to get back $\hat{\sigma}$ that is then used to produce the proof π .
- $\text{Key}(i)$: it is handled exactly as in $\mathcal{H}_1$ except that $\mathcal{B}$ aborts if $i = j$.

When $\mathcal{A}$ outputs its forgery $(m^*, \sigma^* = (\pi, \text{pk}'), \mathcal{R}^*)$, it will be such that $\mathcal{B}$ never queried m^* to the EUF-CMA oracle, $\mathcal{R}^*$ will include exclusively non-corrupted keys, and the verification procedure is successful. Notice that with probability at least $1/q$ it happens that $\text{pk}_i \in \mathcal{R}^*$ and $\mathcal{B}$ does not abort.

$\mathcal{B}$ then runs the extractor E_2 of NIZK over π to get $w = (\text{pk}, \hat{\sigma}, \text{wit}_{\text{pk}}, r)$ and we can distinguish the following cases:

1. $((\text{vp}_{\text{Acc}}, \text{Acc}, \hat{h}, \text{pk}'), (\text{pk}, \hat{\sigma}, \text{wit}_{\text{pk}}, r)) \notin \mathcal{R}$. This case happens only with negligible probability due to the extractability of NIZK.
2. $\text{xSPA.Verify}'(\text{vp}_{\text{Acc}}, \text{Acc}, \text{wit}_{\text{pk}}, \text{pk}) = 1$ but $\text{pk} \notin \mathcal{R}$. However, if this happens with non-negligible probability, $\mathcal{A}$ can be used to break collision resistance of xSPA. Indeed, all the keys are honestly generated and thus VerifySet will return 1, also the successful verification of the forgery guarantees that Acc is correctly computed and that pk is not $1_{\mathbb{G}_1}$ since $\text{pk}' = \text{pk}^r$ is not $1_{\mathbb{G}_1}$. Thus xSPA.Verify will return 1 as $\text{xSPA.Verify}'$ returns 1 and pk is not the neutral element. Therefore, $(\text{Acc}, \mathcal{X}, \mathcal{X}_{\text{aux}}, \text{wit}_{\text{pk}}, \text{pk})$ is a valid collision.
3. If none of the previous cases occurs, $\hat{\sigma}$ is a valid BLS signature for pk and we know that $\text{pk}_i = \text{pk}$ with probability at least $1/q$. Thus, with probability at least $1/q$, $(m^*, \hat{\sigma})$ is a valid forgery for the EUF-CMA game of BLS_{acc} .

Remark 2 (On the security loss of Theorem 6). The adversary's advantage is bounded by the sum of the following advantages: NIZK Setup-Indistinguishability, NIZK Extractability, xSPA Soundness, and q times BLS-EUF-CMA. Here, q (which corresponds to the maximum ring size) arises from the need to guess the forgery-related key within the ring.

Theorem 7. *If NIZK is zero knowledge (Def. 19) then Scheme 6 is anonymous (Def. 17).*

Proof. The proof goes through the following hybrids:

- $\mathcal{H}_0$: It is the anonymity game with $b = 0$.
- $\mathcal{H}_1$: It is identical to $\mathcal{H}_0$ with the difference that pp_{NIZK} is sampled in simulation mode (i.e., using S_1 instead of NIZK.Setup).
- $\mathcal{H}_2$: It is identical to $\mathcal{H}_1$ except that the NIZK proof included in σ is generated with the simulator (i.e., using S_2 instead of NIZK.Prove).
- $\mathcal{H}_3$: It is identical to $\mathcal{H}_2$ except that pk' in σ is created as randomization of pk_{j_1} instead of being a randomization of pk_{j_0} .
- $\mathcal{H}_4$: It is identical to $\mathcal{H}_3$ except that the NIZK proof in σ is generated with the prover instead of the simulator and the correct accumulator witness $\text{wit}_{\text{pk}_{j_1}}$, as well as a signature $\hat{\sigma}$ generated with sk_{j_1} , are used to generate the proof.
- $\mathcal{H}_5$: It is identical to $\mathcal{H}_4$ except that pp_{NIZK} is sampled using the regular setup algorithm.

The proof is concluded observing that $\mathcal{H}_5$ is the anonymity game with $b = 1$, and that $\mathcal{H}_0 \approx_c \mathcal{H}_1 \approx_c \mathcal{H}_2 \equiv \mathcal{H}_3 \approx_c \mathcal{H}_4 \approx_c \mathcal{H}_5$. Where $\mathcal{H}_0 \approx_c \mathcal{H}_1 \approx_c \mathcal{H}_2$ and $\mathcal{H}_3 \approx_c \mathcal{H}_4 \approx_c \mathcal{H}_5$ trivially follow from the zero knowledge of NIZK.

Remark 3 (On the security loss of Theorem 7). The adversary's advantage is bounded by twice the sum of the NIZK Setup-Indistinguishability and NIZK Zero-Knowledge advantages, where the factor of two arises from applying these arguments twice in the proof.

Remark 4 (On structure-preserving ring signatures). In our construction we opted to use BLS as underlying signature for simplicity and efficiency reasons. However, we believe that the blueprint of our construction can be easily adapted to work with other signature schemes featuring a public key in $\mathbb{G}_1$. In particular, an interesting problem would be to instantiate it with a structure-preserving signature; this construction would yield a *structure-preserving* ring signature scheme, again with signatures of size constant in the ring. We leave this extension for future work.

H Instantiating our Ring Signature

We can instantiate the NIZK using the Groth-Sahai (GS) proof system [GS08, EG14]. GS is a commit-and-prove system to prove satisfiability of equations over bilinear groups, where all the secret variables (witness) are first committed and some proof elements are created for each of such equations.

In our case, the witness is composed by the xSPA accumulator's witness $\text{wit}_{\text{pk}} = (W_1, W_2, W_3, W_4, W_5, \hat{W}_6)$, where $W_5 = (\sigma_r, \hat{\sigma}_s, \hat{\sigma}_t)$, the signer's public key pk , the BLS signature $\hat{\sigma}$ and a scalar r . This accounts to a total of 7 elements of $\mathbb{G}_1$, 4 elements of $\mathbb{G}_2$, and one scalar. To commit to such elements, 14 elements of $\mathbb{G}_1$ and 10 elements of $\mathbb{G}_2$ are required (2 elements per commitment and the scalar needs to be committed in $\mathbb{G}_2$). The equations to prove are the following (where all the secret variables are colored in red):

$$\begin{aligned}
& e(\sigma_r, \hat{\sigma}_s) = e(B_v, \hat{g}_2) \cdot e(g_1, \hat{B}_y) \\
& \wedge \quad e(\sigma_r, \hat{\sigma}_t) = e(g_1, \hat{W}_6) \cdot e(B_v, \hat{B}_y) \\
& \quad \wedge \quad e(\text{pk}, \hat{B}_q) = e(W_4, \hat{g}_2) \\
& \quad \wedge \quad e(W_3, \hat{B}_2) = e(W_2, \hat{g}_2) \\
& \wedge \quad e(\text{Acc}, \hat{W}_6) = e(W_1, \hat{g}_2) \cdot e(W_2, \hat{B}_q) \cdot e(W_4, \hat{B}_1) \\
& \quad e(g, \hat{\sigma}) = e(\text{pk}, \hat{h}) \quad \wedge \quad \text{pk}' = \text{pk}^r
\end{aligned}$$

We computed how many proof elements are necessary using the tables in Fig. 6 and 9 of [EG13]. The third and fourth equation only involve secrets in $\mathbb{G}_1$ and thus are of type PConst_H requiring 2 elements in $\mathbb{G}_2$ each. The first, second, fifth, and sixth equation have secret variables in both groups and involve public elements other than the generators so they are PPE requiring 4 group elements in both groups each. Finally, the eighth equation is of type ME_G requiring 2 elements in $\mathbb{G}_1$ and 4 elements in $\mathbb{G}_2$. Accounting for all the commitments, all the proof elements, and pk' , the total signature size is 33 elements in $\mathbb{G}_1$ and 34 elements in $\mathbb{G}_2$, which amount to roughly 4.8 kB when using the BLS12-381 curve.

Some technical details. A technical detail is that GS is zero-knowledge only for equations that do not pair two public elements (both different from the generators) with each other. The second equation does not satisfy this requirement. However, GS is still witness-indistinguishable for such equations, and the only equation we really need to simulate is the last one about the re-randomization of the public key. Thus, we can still make the anonymity proof go through by just simulating the proof for such equation in $\mathcal{H}_2$ and $\mathcal{H}_3$ and relying on witness indistinguishability in $\mathcal{H}_4$. A final detail is that the GS extractor cannot extract scalars (but only a one-way function of them), however we only need to rely on soundness for the last equation and we can still extract all the group elements needed to break the collision resistance of the accumulator from the proof of the other equations.