Structure-Preserving Compressing Primitives: Vector Commitments and Accumulators

Stephan Krenn[1], Omid Mir[1], and Daniel Slamanig[2]

[1] AIT Austrian Institute of Technology
{omid.mir, stephan.krenn}@ait.ac.at
[2] Research Institute CODE, Universität der Bundeswehr München, Germany
daniel.slamanig@unibw.de

Abstract. Compressing primitives such as accumulators and vector commitments, allow to represent large data sets with some compact, ideally constant-sized value. Moreover, they support operations like proving membership or non-membership with minimal, ideally also constant-sized, storage and communication overhead. In recent years, these primitives have found numerous practical applications, with many constructions based on various hardness assumptions. So far, however, it has been elusive to construct these primitives in a strictly structure-preserving setting, i.e., in a bilinear group in a way that messages, commitments and openings are all elements of the two source groups. Interestingly, backed by existing impossibility results, not even conventional commitments with such constraints are known in this setting. However, in many practical applications it would be convenient or even required to be structure-preserving, e.g., to commit or accumulate group elements.

In this paper we investigate whether strictly structure-preserving compressing primitives can be realized. We close this gap by presenting the first strictly structure-preserving commitment that is shrinking (and in particular constant-size). We circumvent existing impossibility results by employing a more structured message space, i.e., a variant of the Diffie-Hellman message space. Our main results are constructions of structure-preserving vector commitments as well as structure-preserving accumulators. We first discuss generic constructions and then present concrete constructions under the Diffie-Hellman Exponent assumption. To demonstrate the usefulness of our constructions, we discuss various applications.

1 Introduction

Compressing primitives like accumulators, vector commitments or key-value commitments (key-value maps) are cryptographic techniques to efficiently represent and manage large datasets in a space-efficient form. They allow to represent large datasets with a compact, ideally constant-sized, value and support operations like proving membership or non-membership with minimal storage and communication overhead (ideally constant-size witnesses). These primitives have many interesting applications for blockchain privacy, e.g., Zcash[3], and scalability, e.g., use of Verkle trees[4], stateless clients for blockchains[5], data availability sampling [45] or batching [21]. Moreover, they are extensively used in privacy-preserving systems, such as anonymous authentication primitives, e.g., ring signatures [33] or revocation in anonymous credentials [8, 11, 30], or data outsourcing [16, 23, 66]. In this paper, we focus mainly on vector commitments [23] and accumulators [13, 17].

Vector Commitments. We recall that a vector commitment (VC) scheme enables a user to commit to a vector $\mathbf{m}$, with the ability to later open the commitment at specific positions without being able to cheat (position binding). Moreover, they might also support to update values committed at certain position. Crucially, both the commitment size and the size of the opening must remain succinct, ideally of constant size. In recent years we have seen significant advancements and growing interest in the development and applications of vector commitments. Beginning with the foundational Merkle tree [56], which leverages collision-resistant hash functions, the field has expanded to include a diverse array of algebraic constructions. These include schemes based on pairing-based assumptions [23, 41, 47, 48, 53, 54, 67] as well as those relying on assumptions over groups of unknown

[3] https://z.cash
[4] https://vitalik.eth.limo/general/2021/06/18/verkle.html
[5] https://ethresear.ch/t/the-stateless-client-concept/172

order, such as RSA groups or class groups [20, 23, 48] and post-quantum constructions based on lattices [10, 50, 64, 68]. For an extensive overview of schemes, see [62]. As we have already mentioned above, vector commitments have been widely utilized across various applications. Moreover, generalizations of vector commitments and in particular polynomial commitments [47] and functional commitments [53], have become foundational components in many recent constructions of succinct non-interactive arguments of knowledge (SNARKs).

Accumulators. We also recall that a (static) accumulator [13, 17] is another type of compressing primitive, which allows to represent a set of elements $X = \{x_1, \ldots, x_q\}$ in the form of a succinct accumulator value acc_X. For each element $x_i \in X$, a witness wit_{x_i} can be efficiently computed to certify its membership in the set, while ensuring that it is computationally infeasible to forge witnesses for non-members $y \notin X$ (collision resistance). Universal accumulators in addition provide the functionality of non-membership witnesses for values $y \notin X$ and dynamic accumulators supporting efficient additions and deletions from the accumulated set. It is worth remarking that Catalano and Fiore have shown that any vector commitment can be used to construct universal dynamic accumulators [23]. Unfortunately, it is unclear whether this approach can be immediately applied to the SP setting or effectively hide the index, a property often required in privacy-preserving applications. Like vector commitments, accumulators have found wide-ranging applications and actually evolved into a foundational component in various advanced cryptographic constructs. They are integral to revocation in anonymous credentials [8, 11, 30], membership revocation for group signatures [19] and the construction of ring signatures [32, 33, 49].

Over time, cryptographic accumulators have evolved through various instantiations. Initially based on the RSA assumption [13], early schemes were later expanded upon [19, 33]. Nguyen introduced pairing-based accumulators [60], sparking further advancements in this area [11, 12, 18, 29, 40]. More recent developments include lattice-based accumulators [46, 50, 63].

Structure-Preserving Compressing Primitives. While, as outlined above, numerous constructions exist for both vector commitments and accumulators, a significant challenge remains in developing such schemes that are structure-preserving (SP). We recall that a cryptographic scheme is called structure preserving [3] if all public inputs and outputs consist of elements of the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P})$ and functional correctness can be verified only by testing group membership, computing group operations, and evaluating pairing product equations (PPEs) [1] of the form $\prod_i \prod_j e(A_i, \hat{B}_j)^{c_{i,j}} = 1$, where $A_i \in \mathbb{G}_1$ and $\hat{B}_j \in \mathbb{G}_2$ are group elements and $c_{i,j} \in \mathbb{Z}_p$ are constants. This domain is very rich in its constructions and allows for a modular design of (advanced) cryptographic primitives, e.g., structure-preserving signatures (SPS) [3] (cf. [39] for a recent overview on the large body of works), threshold SPS [27, 59], blind signatures [3, 36], group signatures [3, 52], traceable signatures [2], homomorphic signatures [51] or delegatable anonymous credentials [28, 35, 58]. Such constructions are highly desirable because they maintain the algebraic structure of the committed values and proofs, ensuring compatibility with a broader range of cryptographic protocols and in particular the Groth-Sahai [44] and related proof systems. Moreover, especially for compressing primitives they are of concrete practical interest beyond modular protocol design, as we will discuss in this paper.

Unfortunately, there are known impossibility results of Abe et al. [6] establishing that strictly structure-preserving commitments to source-group elements cannot be smaller than the input message size and thus scale linearly with the number of group elements. In order to be compressing though, for conventional commitments, there are some approaches that relax the SP setting by allowing elements from the target group $\mathbb{G}_T$ in the commitment [5, 6] (see Sec.2.1). Another approach by Abe et al. [7] is to construct SP commitments that have a relaxed notion of binding, where the message space and the verification space differ (e.g., being $\mathbb{Z}_p$ and $\mathbb{G}_1$ respectively). Nevertheless, these are not strictly structure-preserving commitments and in particular do not allow to commit to commitments, a features that is interesting for practical applications. When it comes to accumulators, to the best of our knowledge, the approach that conceptually comes closest is that of determinantal accumulators [55]. While they are in spirit of SP primitives and their combination with Groth-Sahai proofs [44], the focus of determinantal primitives is on designing primitives that can be modularity combined with algebraic NIZKs in the vein of Couteau and Hartmann [25] and Couteau et al. [26]. Consequently, they are not structure-preserving. For vector commitments, there is an impossibility result for algebraic vector commitments in pairing-free groups [24], but this does not rule out the existence of structure-preserving ones. While [24] notes that the impossibility of strictly SP commitments [5, 6] rules out constructing succinct vector commitments in this structure-preserving setting, we are not aware of any work trying to bypass such an impossibility and thus this state of affairs is not satisfactory. This leads us to the following question:

Can we design (vector) commitments and accumulators that retains algebraic structure, i.e., being strictly structure-preserving, while being succinct?

Addressing this gap is crucial for advancing the field and our aim is to close it.

1.1 Our Results

Our contributions can be summarized as follows:

- We present the first construction of group-to-group commitments that are shrinking and strictly structure-preserving, i.e., messages, commitments and openings are all elements in the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$. We obtain this by bypassing known impossibility results due to Abe et al. [5,6] as we require a more structured message space and in particular a Diffie-Hellman (DH) message space [3, 34]. Such a message space has recently shown to be relevant for various applications [27,57].
- We present the first structure-preserving vector commitments (SPVC's). As a warm up, we show a (probably folklore) construction of weak-binding WSPVC from any EUF-CMA secure SPS scheme. However as our main result, we turn to SPVC's providing the conventional notion of position-binding and present a construction of SPVC under the Diffie-Hellman Exponent (q-DHE) assumption with a message space that is defined with respect to some global parameters (inspired by the recent work by Griffy et al. [42] and q-DHE VC schemes [41,54]).
- We present the first structure-preserving accumulators (SPAs). We first discuss how a weak-binding vector commitment WSPVC naturally yields an accumulator. Second, and more importantly, we introduce a generic construction that allows any SPVC to be applied in a black-box manner to build an SPA in a strong model. In this setting, accumulators can be adversarially generated, leading us to define a SPA based on q-DHE in the strong collision resistance model. Third, we extend q-DHE based SPA and construct a perfect randomizable SP accumulator starting for another and in particular DH-type message space. Here we aim to hide the index (or position), i.e., it should be possible to give out a witness and accumulated value such that they cannot be linked back to the accumulated value. This is accomplished through a randomization technique that ensures the randomized accumulated values do not reveal anything about the position of the original accumulated value and DH-type messages.

1.2 Technical Overview

Strictly Structure-Preserving Commitments: Firstly, we present shrinking and strictly structure-preserving group-to-group commitments, thereby bypassing aforementioned impossibility results by using a structured message space. Considere therefore an asymmetric Type-3 pairing group: $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGSetup}(1^\lambda)$.

We start from the structure-preserving (bilinear pairing-based) Pedersen commitment scheme proposed by Abe et al. [4], where the commitment public key consists of q group elements $(X_1, \ldots, X_q)$ from $\mathbb{G}_1$. To commit to $(\hat{M}_1, \ldots, \hat{M}_q) \in \mathbb{G}_2$, a random element $\hat{R} \in \mathbb{G}_2$ is selected, and the commitment takes the form:

$$C = e(H, \hat{R}) \cdot \prod_{i=1}^{q} e(X_i, \hat{M}_i).$$

This commitment scheme can be shown to be binding under the Double Pairing Problem (DBP) assumption.

Now, to *shrink* commitments into a single element in the source group $\mathbb{G}_1$, we leverage a variant of the Diffie-Hellman (DH) message space [3,4,34]. Specifically, we consider messages of the following form:

$$(\mathbf{M}, \hat{\mathbf{N}}) = (M_1, \ldots, M_q, \hat{N}_1, \ldots, \hat{N}_q),$$

such that there exist $m_i \in \mathbb{Z}_p$ for $1 \leq i \leq q$, satisfying:

$$M_i = {X_i}^{m_i}, \quad \hat{N}_i = \hat{P}^{m_i}, \quad \forall i \in [q].$$

In other words, the commitment public keys are embedded in $\mathbf{M}$, providing flexibility to generate commitments in $\mathbb{G}_1$ as:

$$\mathsf{com} := P^r \cdot \prod_{i=1}^{q} M_i \quad \text{and} \quad \mathsf{Open} := \hat{P}^r .$$

Meanwhile, $\hat{\mathbf{N}}$ is used for verification. This still allows us to prove security under the DBP assumption as the second group elements $\hat{\mathbf{N}}$ along with $\hat{R}$ can be used to break the assumption, similar to the security proof in [4]. Thus, a DH message space circumvents certain constraints by enabling commitments in one group while verification occurs in another. At the same time, the random group elements or commitment keys remain/encode in the same group as the committed messages.

Vector Commitments. Then we shift our attention to vector commitments. As a preliminary step, we present a weakly binding structure-preserving vector commitment (WSPVC), where commitments are generated honestly. We demonstrate that this can be realized through a generic construction based on any compact structure-preserving signature (SPS) scheme, effectively yielding a structure-preserving accumulator. We defer this construction to Appendix A. This construction directly yields a first SP accumulator, though only in a weak setting and in particular a setting where the accumulator has been computed honestly.

We then construct a SP vector commitment from the q-DHE assumption. Our construction is inspired by the q-DHE based mercurial commitment scheme from Libert and Yung [54], which however is not SP and just commits to scalars. To turn the underlying idea into a full-blown SPVC we use two essential ideas. We again rely on a type of DH message space, but this time the messages to be committed are generated with respect to a common reference string (CRS) i.e., instead of using $\mathbf{X}$ we use q-DHE parameters to build messages. Latter idea is inspired by the recent construction of strongly private mercurial signatures by Griffy et al. [42].

More precisely, each message is represented as a vector over $\mathbb{G}_1$ and consists of two main components: *i*) the main message component $\mathbf{M}$ and *ii*) a tag vector $\mathbf{T}$. Latter is derived from the CRS, and helps to encode the position of the message in the vector (i.e., the commitments and the witnesses). This structure is leveraged alongside mercurial commitments of Libert and Yung [41,54] to construct our SPVC. A mercurial commitment takes the form:

$$C = P^{\sum \alpha^i \cdot m_i},$$

where $(P^{\alpha^i})_{i\in[q]}$ is the CRS (i.e., q-DHE elements). A key challenge in the structure-preserving setting is the lack of direct access to discrete logarithms. To address this, we introduce a new messages structure such that we encode messages implicitly in a mercurial vector commitment using $M_{i,0} = P^{m_i} \in \mathbb{G}_1$ and $M_{i,1} = M_{i,0}^{\alpha^i} = B_i^{m_i} \in \mathbb{G}_1$, where the α^i are now available in encoded messages and $B_i = P^{\alpha^i}$ are the CRS elements. This allows us to build a vector commitment over $\mathbf{M}_{i,1}$ for all $i \in [q]$.

To compute an opening for a committed message at some index i, mercurial commitments employ a shifting technique to create a prof for a message m_i as:

$$P^{\sum_{j\neq i} m_j \alpha^{q+1-i+j}}.$$

This transformation ensures that the message for which we want to generate a proof is always positioned at $q+1$. Since we do not have $P^{\alpha^{q+1}}$, it is naturally the missing element in the proof. However, during verification, it can still be checked using $g_t^{m_i}$ and this element can actually be explicitly included in the proof π – indeed, the security follows from the fact that the adversary does not know α^{q+1}, so it is not able to prove a message at this position – while other messages are shifted to positions within the range $[q+1, 2q]$. The commitments are also shifted accordingly in the same way, i.e., $e(C, \hat{P}^{\alpha^{q-i+1}})$. Clearly, when m_i is known and one has access to q-DHE parameters, this proof can be computed efficiently.

In our case, since m_i is unknown, it is pre-encoded with α using our message space and q-DHE elements. This ensures that when messages are shifted (as required for mercurial VC proofs), the necessary elements remain available for proof generation. We assign these elements as tag components and as part of our messages e.g., $\mathbf{T} = (P^{\alpha^j \cdot m_i})$ for $i \in [q]$ and $j \in [i+1, q+i] \backslash \{i\}$ (note that the details of how j is defined will be clarified and further explained in the construction section), playing a crucial role in encoding the missing terms during proof construction. We note that verifying messages of the form $M = P^{\alpha^i \cdot m_i}$ requires a slight modification of the verification equations. Instead of checking

$$e(C, \hat{P}^{\alpha^{q-i+1}}) = e(\pi, \hat{P}) \cdot g_t^{m_i} \text{ or } e(\pi, \hat{P}) \cdot e(P^{\alpha}, \hat{P}^{\alpha^q})^{m_i},$$

we set it to

$$e(C, \hat{P}^{\alpha^{q-i+1}}) = e(\pi, \hat{P}) \cdot e(P^{\alpha^i \cdot m_i}, \hat{P}^{\alpha^{q+1-i}}).$$

This approach enables verification while still allowing the binding proof to be performed.

We can view this message structure as a generalized DH message space built on q-DHE parameters. Finally, we structure the messages based on the q-DHE assumption in such a way that verifying the correctness of the message structure enables the extraction of discrete logarithms in security proofs and ultimately reduces binding to the q-DHE assumption. The message verification has the form $e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{P})$ and $e(M_{i1}, \hat{P}) = e(M_{i0}, \hat{B}_i)$, where one uses the public elements $\hat{B}_{j-i}$ from the CRS.

q-DHE SP Accumulators (SPA). We then turn to the construction of SP accumulators with stronger guarantees as the ones obtained from WSPVC. While there are results due to Catalano and Fiore [23] which show how to construct accumulators from vector commitments in a generic way, this idea does not immediately apply to the structure-preserving setting.

Thus, we present a *compiler (or generic construction) from vector commitments to accumulators*, demonstrating how any SPVC can be used to construct an SPA. Although our generic construction is applicable to all types of vector commitments and can be extended to accumulators we focus here on SP setting. An interesting observation when building accumulators from VCs is that we achieve a somewhat stronger notion of collision resistance, where the accumulator is fully controlled by the adversary. In contrast, in standard collision resistance definitions, accumulators are typically generated by honest parties. Therefore, we introduce our generic constructions (i.e., the first structure-preserving accumulators based on the q-DHE assumption in the strong model) along with a strong collision resistance notion specifically tailored for black-box accumulator constructions derived from VCs.

Perfect Randomizable Accumulators: It is clear that our generic construction—and consequently, our q-DHE SP accumulators—utilizes SPVC, which inherently reveals the positions of elements within the set. This can potentially compromise sensitive information, for example, in privacy-preserving primitives such as ring signatures, where the accumulator is used to accumulate public keys, thereby exposing indices, linking users, and breaking anonymity. To enhance privacy in such applications, we aim for an approach where witnesses do not disclose any information about the elements. Thus, we modify the q-DHE accumulator construction and introduce this as a q-DHE type *Perfect Randomizable* accumulator.

The idea is to leverage the q-DHE SPVC but through the relations that are required to verify the correctness of message/tag pairs:

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

where $\hat{B}_{j-i}$ are public elements from the CRS, thereby revealing the positions of elements in the vector. To address this, we use a randomization technique to break this connection. Specifically we pick a random value $y \in \mathbb{Z}_p^*$ and modify the verification equations as follows:

$$e(M_{i1}, \hat{Y}) = e(T_{ij}^y, \hat{P}) \wedge e(M_{i1}, \hat{P}^y) = e(M_{i0}, \hat{B}_i^y),$$

where $\hat{B}_{j-i}^y$ is applied for all j, and $\mathbf{Y} = (\hat{P}^y, \hat{B}_i^y)_{i \in [q]}$ is included as part of the tag. Moreover, the verification of proof also checks the index as follows:

$$e\left(C, \hat{B}_{q+1-i}\right) = e\left(\pi_i, \hat{P}\right) \cdot e\left(M_{i1}, \hat{B}_{q+1-i}\right).$$

In a similar technique, we randomize the witness using a random $\rho \in \mathbb{Z}_p^*$ and include $(\hat{B}_{q+1-i})^{\rho}$ as part of the witness. Assume that $W_1 = \pi_i^{\rho}$ and $\hat{W}_2 = (\hat{B}_{q+1-i})^{\rho}$, then the verification equation takes the form:

$$e\left(C, \hat{W}_2\right) = e\left(W_1, \hat{P}\right) \cdot e\left(M_{i1}, \hat{W}_2\right).$$

Note that the verification prevents any leakage of the initial witnesses.

We show that the collision resistance of this construction can be reduced to the binding property of SPVC in a strong model, i.e., where the accumulators can be adversarially generated. This follows straightforwardly from SPVC, as commitments are generated by the adversary, and we use the randomizers (ρ, y) to de-anonymize the witnesses and return the proofs for commitments in the reduction.

Applications: Finally, in addition to the natural applications of VC and accumulators—such as anonymous authentication, ring signatures, revocation in anonymous credentials, and data outsourcing—we discuss their applications in the blockchain space in Sec. 6. Firstly, data availability sampling [9, 45] which is a technique to achieve blockchain scaleability by reducing the amount of data clients have to download. Roughly speaking clients with limited resources store a tuple $\mathsf{com} = (C_1, \ldots, C_q)$, where each C_i is a KZG commitment [47]. With an SPVC, such a vector of KZG commitments can be compressed into a compact commitment instead. Secondly, we Verkle Trees, which improve upon Merkle Trees by using vector commitments at the leaf nodes to enable smaller witness sizes and more efficient verification. But still it uses hash functions for all upper layers in the tree. Realizing algebraic Verkle Trees (AVTs) based on an SPVC enables seamless commitment to commitments without switching between cryptographic primitives like VCs and hash functions.

2 Preliminaries

Notations. The main security parameter will be denoted by λ. We use $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGSetup}(1^\lambda)$ to denote a bilinear group generator for asymmetric type-3 bilinear groups, where p is a prime of bit length λ. We use $[q]$ to denote the set $\{1, 2, \ldots, q\}$. When drawing multiple values from a set, we may omit notation for products of sets, e.g. $(x, y) \in \mathbb{Z}_p$ is the same as $(x, y) \in (\mathbb{Z}_p)^2$. For a map from the set Z to the set S, $m : Z \to S$, we will denote $m[i] \in S$ as the output of the map in S with input $i \in Z$. We use bold font to denote a vector (e.g. $\mathbf{V}$). For brevity, we will sometimes denote the elements in a vector as $\mathbf{V} = (V_i)_{i \in [q]} = (V_1, \ldots, V_q)$. Given a finite set S, we denote by $x \leftarrow S$ or $x \leftarrow\!\!\$\, S$ the sampling of an element uniformly at random from S. For an algorithm A, let $y \leftarrow A(x)$ be the process of running A on input x with access to uniformly random coins and assigning the result to y. With A^B we denote that A has oracle access to B. We assume all algorithms are polynomial-time (PPT) unless otherwise specified and public parameters are an implicit input to all algorithms in a scheme. We use $[a, b]$ to denote the range $\{a, a + 1, \ldots, b\}$. Additionally, $|N|$ represents the length of N. By the symbol $\approx$, we denote indistinguishability between two distributions $D_1 \approx D_2$.

Diffie-Hellman Message Space. Over an asymmetric bilinear group, a pair $(M, \hat{N}) \in \mathbb{G}_1 \times \mathbb{G}_2$ is called a Diffie-Hellman (DH) message $\mathcal{M}_{\mathsf{DH}}$ [4, 34] if there exists $m \in \mathbb{Z}_p$ s.t. $M = P^m$ and $\hat{N} = \hat{P}^m$. One can efficiently verify whether $(M, \hat{N}) \in \mathcal{M}_{\mathsf{DH}}$ by checking $e(M, \hat{P}) = e(P, \hat{N})$. More formally, the message space are elements of the subgroup of $\mathbb{G}_1 \times \mathbb{G}_2$ defined as the image of the map $\psi' : \mathbb{Z}_p \to \mathbb{G}_1 \times \mathbb{G}_2 : x \mapsto (P^x, \hat{P}^x)$. One can easily extend the message space to a vector of Diffie-Hellman pairs $(\mathbf{M}, \hat{\mathbf{N}}) = (M_1, \ldots, M_q, \hat{N}_1, \ldots, \hat{N}_q)$ s.t. for all $i \in [q]$, $(M_i, \hat{N}_i) = (P^{m_i}, \hat{P}^{m_i}) \in \mathcal{M}_{\mathsf{DH}}$ for $m_i \in \mathbb{Z}_p$.

We note that related notations exist, such as equivalence classes [37]. Given the vector $\mathbf{M}$, the message $\mathbf{M}$ represents an equivalence class of all scaled messages $\mathbf{M}^\mu$. This allows for randomizing the message vector by switching between $\mathbf{M}$ and $\mathbf{M}^\mu$ for any non-zero μ. This concept extends to Diffie-Hellman message spaces, including Indexed DH [27] and tag-based DH message spaces [57], where randomized DH message vectors retain their validity with respect to their tags. Although latter variants are not directly related to our primitives, it provides useful context.

2.1 Structure-Preserving Commitments

A structure-preserving commitment scheme was proposed by Abe et al. [4]. The public key consists of $q+1$ group elements $(H, U_1, \ldots, U_q)$ from $\mathbb{G}_1$. To commit to $(\hat{M}_1, \ldots, \hat{M}_q) \in \mathbb{G}_2$, a random element $\hat{R} \in \mathbb{G}_2$ is selected, and the commitment is computed as $C = e(H, \hat{R}) \prod_{i=1}^{q} e(U_i, \hat{M}_i)$. This commitment scheme is computationally binding under the DBP assumption. Moreover, it is both a length-reducing (constant-size) scheme and a homomorphic trapdoor commitment.

2.2 Assumptions

Definition 1 (The Diffie-Hellman Assumption (q-DHE) [18]). *Let $\mathbb{G}$ be a finite cyclic group of order p, P be a generator of $\mathbb{G}$. The q-DHE problem is, given a tuple of elements $(P, B_1, \ldots, B_q, B_{q+2}, \ldots, B_{2q})$ such that $B_i = P^{\alpha^i}$ for $i = 1, \ldots, q, q+2, \ldots, 2q$ and where $\alpha \leftarrow\$ \mathbb{Z}_p^*$, to compute the missing group element $B_{q+1} = P^{\alpha^{q+1}}$ in the sequence. The q-DHE assumption now states that no PPT adversary has more than negligible advantage in solving the q-DHE problem.*

As shown in [41, 54], q-DHE can be modified so as to work in asymmetric pairing configurations i.e., given $\{B_1 = P^{\alpha^1}, \ldots, B_q = P^{\alpha^q}, B_{q+2} = P^{\alpha^{q+2}}, \ldots, B_{2q} = P^{\alpha^{2q}}; \hat{B}_1 = \hat{P}^{\alpha^1}, \ldots, \hat{B}_q = \hat{P}^{\alpha^q}; e(P, \hat{P})^{\alpha^{q+1}}\}$ it is hard to find $B_{q+1} = P^{\alpha^{q+1}}$.

Definition 2 ((Double Pairing Problem (DBP)) [4]). *We say the double pairing problem holds relative to* BG *if for any probabilistic polynomial-time algorithm $\mathcal{A}$*

$$\Pr\left[\begin{array}{c}\mathsf{BG} \leftarrow \mathsf{BGSetup}(1^\lambda); h_z \leftarrow\$ \mathbb{G}_1^* \\ (\hat{Z}, \hat{R}) \leftarrow \mathcal{A}(\mathsf{BG}, h_z) : (\hat{Z}, \hat{R}) \in \mathbb{G}_2^* \text{ and } 1 = e(h_z, \hat{Z})e(P, \hat{R})\end{array}\right] \leq \epsilon(\lambda)$$

This assumption follows from the decisional Diffie-Hellman (DDH) assumption in $\mathbb{G}_1$. By swapping $\mathbb{G}_1$ and $\mathbb{G}_2$ (assuming DDH in $\mathbb{G}_2$), we obtain a dual assumption. Hence, if DDH holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$, DBP holds in both groups.

2.3 Vector Commitments

Definition 3 (Vector Commitment [23]). *A vector commitment is a tuple of algorithms defined as follows:*

$\mathsf{KeyGen}(1^\lambda, q)$: Given the security parameter λ and the size q of the committed vector, the key generation outputs some public parameters pp (which implicitly define the message space $\mathcal{M}$ and are input to all algorithms).

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

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

$\mathsf{Verify}(\mathsf{com}, m, i, \pi_i)$: The verification algorithm accepts (i.e., it outputs 1) if and only if π_i is a valid proof that com was created for a vector $(m_1, \ldots, m_q)$ such that $m = m_i$.

In addition, some applications require an update property (to update the commitment and the corresponding openings), which is defined using two additional (and optional) algorithms:

$\mathsf{Update}(\mathsf{com}, m, m', i)$: This algorithm is run by the committer who produced com and wants to update it by changing the i-th message to m'. It takes as input the old message m, the new message m', and the position i. The algorithm outputs a new commitment com' together with update information U. In particular, notice that in the case when some updates have occurred, the auxiliary information aux can include the update information produced by these updates.

$\mathsf{ProofUpdate}(\mathsf{com}, \pi_j, m', i, U)$: This algorithm is run by any user who holds a proof π_j for the message at position j with respect to com. It allows the user to compute an updated proof π'_j (and an updated commitment com'), such that π'_j will be valid with respect to com', where m' is the new message at position i. The value U contains the update information needed to compute these updated values.

VC should satisfy the property of position binding as follows:

Position Binding: This property ensures that a PPT adversary (with knowledge of pp) cannot produce two proofs for the same position in a fixed commitment com that open to different values. There are two flavors of this property:

- **Weak Position Binding:** Holds only for honestly generated commitments.
- **Position Binding:** Holds even for adversarially generated commitments.

Weak binding suffices for stateless validation (e.g., in Byzantine agreement on updates [62]) and can be easily achieved using accumulators. Position binding is essential in adversarial scenarios, like transparency logs, where commitments are generated by log servers.

Definition 4 (Position Binding [23]). *A* VC *satisfies position binding if,* $\forall i = 1, \ldots, q$*, and for every PPT adversary* $\mathcal{A}$*, the following probability (taken over all honestly generated* pp*) is at most negligible:*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{com}, m, i, \pi) = 1 \wedge \\ \mathsf{Verify}(\mathsf{com}, m', i, \pi') = 1 \wedge \\ m \neq m'\end{array}\middle| ((\mathsf{com}, \mathsf{aux}), m, m', i, \pi, \pi') \leftarrow \mathcal{A}(\mathsf{pp})\right] \leq \epsilon(\lambda)$$

If we relax the above definition to hold only for honestly generated commitments com, we obtain the weak position binding notion.

Definition 5 (Weak Position Binding [22, 41]). *A* VC *satisfies weak position binding if* $\forall i = 1, \ldots, q$ *and for every PPT adversary* $\mathcal{A}$*, the following probability (which is taken over all honestly generated parameters) is at most negligible:*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{com}, m, i, \pi) = 1 \wedge \mathsf{Verify}(\mathsf{com}, m', i, \pi') = 1 \wedge m \neq m' \; st: \\ (\mathsf{pp}) \leftarrow \mathsf{KeyGen}(1^\lambda), (\mathbf{m}, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pp}), \\ (\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{Commit}(\mathbf{m}), (i, m, \pi, m', \pi') \leftarrow \mathcal{A}(\mathsf{com}, \mathsf{state})\end{array}\right] \leq \epsilon(\lambda).$$

Indeed, our constructions will satisfy a slightly stronger variant of this notion, where the adversary is also given aux in the second phase.

Conciseness: This property requires that both the commitment and the opening for a position i are of constant size, independent of the vectors length.

Hiding. VC can also be required to be hiding, meaning that one should not be able to distinguish whether a commitment was created to $(m_1, \ldots, m_q)$ or to $(m'_1, \ldots, m'_q)$ (both chosen by the adversary), even after seeing some proofs for positions where $m_i = m'_i$. However, hiding is not a crucial property in the realization of VC and typically not considered.

2.4 Accumulators

We provide a formal a definition of static accumulators based on the definition given by Derler et al. in [31].

Definition 6 (Static Accumulator). *A static accumulator is a tuple of algorithms defined as follows:*

$\mathsf{Setup}(1^\lambda, q)$: Take 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. Return a key pair $(\mathsf{sk}_{acc}, \mathsf{pk}_{acc})$, where $\mathsf{sk}_{acc} = \emptyset$ if no trapdoor exists.

$\mathsf{Eval}((\mathsf{sk}_{acc}, \mathsf{pk}_{acc}), \mathcal{X})$: This (probabilistic) algorithm takes a key pair $(\mathsf{sk}_{acc}, \mathsf{pk}_{acc})$ and a set $\mathcal{X}$ to be accumulated and returns an accumulator $\mathsf{Acc}_{\mathcal{X}}$ together with some auxiliary information aux.

$\mathsf{WitCreate}((\mathsf{sk}_{acc}, \mathsf{pk}_{acc}), \mathsf{Acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$: This algorithm takes a key pair $(\mathsf{sk}_{acc}, \mathsf{pk}_{acc})$, an accumulator $\mathsf{Acc}_{\mathcal{X}}$, auxiliary information aux, and a value x_i. It returns $\perp$, if $x_i \notin \mathcal{X}$, and a witness wit_{x_i} for x_i otherwise.

$\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, wit_{x_i}, x_i)$: This algorithm takes a public key pk_{acc}, an accumulator $\mathsf{Acc}_{\mathcal{X}}$, a witness wit_{x_i}, and a value x_i. It returns *true* if wit_{x_i} is a witness for $x_i \in \mathcal{X}$ and *false* otherwise.

The above definition focuses on static accumulators. In contrast, dynamic accumulators enable the accumulated value and witnesses to be publicly updated whenever an element is added or removed from the set. Various properties, such as being trapdoorless, universal, or supporting subset queries, are associated with this primitive. For a comprehensive list of definitions, functionalities, and properties, we refer to [14, 31]. Here, we only consider the basic properties.

Security. One requires collision resistance which states that it should be computationally infeasible to find a witness for any non-accumulated value $x \notin \mathcal{X}$.

Definition 7 (Collision resistance [14, 31]). *An accumulator scheme is said to satisfy collision resistance if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}_{acc}, \mathsf{pk}_{acc}) \leftarrow \mathsf{Setup}(1^\lambda, q), (\mathsf{Acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}((\mathsf{sk}_{acc}, \mathsf{pk}_{acc}), \mathcal{X}), \\ (\mathcal{X}, wit_{x_i}, x_i) \leftarrow \mathcal{A}^{\mathcal{O}^W}(\mathsf{pk}_{acc}) : \mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, wit_{x_i}, x_i) = 1 \wedge x_i \notin \mathcal{X}\end{array}\right]$$

where $\mathcal{O}^{\mathrm{W}}$ represent an oracle for the algorithm $\mathsf{WitCreate}$. An adversary is allowed to query them an arbitrary number of times. Note that if $\mathcal{O}^{\mathrm{W}}$ is queried for an element $x \notin \mathcal{X}$, the oracle outputs a reject symbol $\perp$.

2.5 Structure-Preserving Signatures

Structure-preserving signatures (SPSs) [4] are signatures where public keys and the signatures are source group elements of a bilinear group, and verification will be done only using group-membership tests and pairing-product equations. We recall the structure-preserving signature scheme FHS [37], which also allows for the efficient and joint randomization of messages and signatures in public, where the message space consists of group-element vectors. In this paper, message randomization is not required; thus, we remove the randomization algorithms and the equivalence class relation concept and to sign messages $\mathbf{M}$ of length $\ell - 1$ one signs message $(\mathbf{M}, P)$ and in verificaton checks whether the ℓ'th element of the message equals P. For further details, see [37].

$\mathsf{BG}_{\mathcal{R}}(1^\lambda)$: This algorithm on input of a security parameter λ outputs a bilinear group BG.

$\mathsf{KeyGen}(\mathsf{BG}, \ell)$: This algorithm on input of a bilinear group BG and a vector length $\ell > 1$ outputs a key pair $(\mathsf{sk}, \mathsf{vk})$ as $\mathsf{sk} \leftarrow (x_i)_{i\in[\ell]}$, and the public key $\mathsf{vk} \leftarrow (\hat{X}_i)_{i\in[\ell]} = (\hat{P}^{x_i})_{i\in[\ell]}$.

$\mathsf{Sign}(\mathsf{sk}, \mathbf{M})$: This algorithm on input a $\mathbf{M} \in (\mathbb{G}_i^*)^{\ell-1}$and a secret key sk, sets $\mathbf{M}' = (\mathbf{M}, P)$ outputs a signature σ as $\sigma = (Z \leftarrow (\prod_{i\in[\ell]} {M'}_i^{x_i})^y, Y \leftarrow P^{\frac{1}{y}}, \hat{Y} \leftarrow \hat{P}^{\frac{1}{y}})$.

$\mathsf{Verify}(\mathbf{M}, \sigma, \mathsf{vk})$: This algorithm on input of a representative $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$, a signature σ and a public key vk outputs a bit $b \in \{0, 1\}$ if $\mathbf{M}_\ell = P$ and $\prod_{i\in[\ell]} e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{P}) = e(P, \hat{Y})$.

3 Shrinking SP Commitments for DH Messages

We begin by constructing *constant-size* and *strict* structure-preserving commitments for group elements using a variant of Diffie-Hellman (DH) messages. By *strict*, we refer to the definition provided in [6], which means that the messages, commitments, and openings are all confined to the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Our construction is notable for bypassing the impossibility result of Abe et al. [6], which establishes that strictly structure-preserving commitments to source-group elements cannot be smaller than the input message size and scale linearly with the number of group elements.

In contrast, in a relaxed structure-preserving setting, constant-size commitments to group elements can be achieved by allowing elements from the target group $\mathbb{G}_T$ in the commitment [5,6] (see Sec. 2.1). While including $\mathbb{G}_T$ elements is acceptable when only witness indistinguishability is required for accompanying Groth-Sahai (GS) proofs, it becomes problematic when zero-knowledge is necessary (see [6]). Therefore, ensuring that group-to-group commitments remain entirely within the source groups is crucial for achieving zero-knowledge, and it also allow one to commit to other commitments, latter being very interesting for applications. Another way of bypassing this impossibility is done by Abe et al. in [7], who construct SP commitments that are shrinking in a relaxed binding setting, e.g., where the message space for committing is $\mathbb{Z}_p$ and the one for verification is $\mathbb{G}_1$.

Our approach is inspired by circumventing impossibility results and lower bounds in SPS [38,39], achieved by switching from arbitrary group elements in the message space to a more structured message space and in particular a Diffie-Hellman (DH) message space [3,34]. This has also recently been used to construct the first threshold SPS in [27]. In doing so we obtain shrinking strictly structure-preserving (group-to-group) commitments. The so obtained SP commitment scheme in nature is similar to the γ-binding commitment scheme in [7], but is outputs not trapdoor and is strictly structure-preserving, i.e., messages, commitments, and openings are all elements of the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

We first define a new DH message space $\mathcal{M}_{\mathsf{DH}}$ and then present a construction for shrinking strictly structure-preserving (group-to-group) commitments based on this new DH message type.

New DH **Message Space** $\mathcal{M}_{\mathsf{DH}}$**.** We slightly adapt the DH message technique (cf. Sec. 2) to use a distinct base element in $\mathbb{G}_1$ for each index i:

Definition 8 (DH **Message Space** ($\mathcal{M}_{\mathsf{DH}}^{new}$)**).** *Let the public parameters be a vector* $\mathbf{X}$ *of random elements in* $\mathbb{G}_1$ *(e.g.,* $\mathbf{X} = (P^{x_i})_{i\in[q]}$*). We then define* $\mathcal{M}_{\mathsf{DH}}^{new}$ *as a* DH *message space, if the following property hold: For the message vector* $(\mathbf{M}, \hat{\mathbf{N}}) = (M_1, \dots, M_q, \hat{N}_1, \dots, \hat{N}_q)$ *there exist* $m_i \in \mathbb{Z}_p$ *(*$1 \le i \le q$*) s.t.* $M_i = X_i^{m_i}$ *and* $\hat{N}_i = \hat{P}^{m_i}$ *for all* i*.*

Note that membership in this message space can be efficiently checked by $e(M_i, \hat{P}) = e(X_i, \hat{N}_i)$. We can obtain the plain messages $\hat{P}^{m_i}$ in $\mathbb{G}_1$ as well by simply switching the vector $\mathbf{X}$ to reside in $\mathbb{G}_2$ and keeping the vector $\mathbf{M} = P^{m_i}$ in $\mathbb{G}_1$.

Construction. We introduce our group-to-group commitments, where the messages, commitments are all restricted to the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Despite this, the construction remains concise and compact, achieving a compactness property.

Scheme 1 (Shrinking Strictly Structure-Preserving Commitment) *Our commitment scheme is composed of the following PPT algorithms:*

$\mathsf{KeyGen}(1^\lambda)$: Run $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGSetup}(1^\lambda)$. For $i \in [q]$, choose $X_i \leftarrow\!\!\$\ \mathbb{G}_1$. Note that $\mathbf{X}$ does not need to have structure, and therefore, we also can use a random oracle to generate these elements.
Output the commitment key $\mathsf{pk} := (\mathsf{BG}, X_1, \dots, X_q)$.

$\mathsf{Commit}(\mathsf{pk}, msg)$: Parse msg as $((M_1, \hat{N}_1), \dots, (M_q, \hat{N}_q)) \in \mathcal{M}_{\mathsf{DH}}^{new}$ (check if they are generated correctly by $e(M_i, \hat{P}) = e(X_i, \hat{N}_i)$ for $i \in [q]$). Choose $r \leftarrow\!\!\$\ \mathbb{Z}_p$, and compute

$$\mathsf{com} := P^r \cdot \prod_{i=1}^{q} M_i \ \wedge\ \mathsf{Open} := (\hat{R} = \hat{P}^r).$$

$\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, msg, \mathsf{Open})$: Parse msg as $((M_1, \hat{N}_1), \dots, (M_q, \hat{N}_q)) \in \mathcal{M}_{\mathsf{DH}}^{new}$, and Open as $\hat{R}$. Check for well-formedness of the messages by $e(M_i, \hat{P}) = e(X_i, \hat{N}_i)$ for $i \in [q]$, and abort otherwise. Output 1 if and only if:

$$e(\mathsf{com}, \hat{P}) = e(P, \hat{R}) \prod_{i=1}^{q} e(X_i, \hat{N}_i)\,.$$

Theorem 1. *Scheme 1 is perfectly hiding. Furthermore, it is binding under the DBP assumption.*

Proof. **Hiding.** This property is obvious as P^r is a uniformly random element.

Binding. For an adversary $\mathcal{A}$ breaking the binding property of the commitment scheme, consider the following adversary $\mathcal{B}$: On input a DBP challenge (BG, h_z), $\mathcal{B}$ sets $X_i := h_z^{x_i} \cdot P^{\xi_i}$ for random $(x_i, \xi_i) \leftarrow\!\!\$\ \mathbb{Z}_p$ for $i \in [q]$. $\mathcal{B}$ aborts if $X_i = 1$ for any i, but this happens only with negligible probability. $\mathcal{B}$ then runs $\mathcal{A}$ on $\mathsf{pk} = (\mathsf{BG}, X_1, \dots, X_q)$. If $\mathcal{A}$ returns a commitment com together with two different valid messages and openings $(\hat{N}_1, \dots, \hat{N}_q, \hat{R})$ and $(\hat{N}'_1, \dots, \hat{N}'_q, \hat{R}')$, then $\mathcal{B}$ computes

$$\hat{Z}^* := \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{x_i}, \quad \hat{R}^* := \frac{\hat{R}}{\hat{R}'} \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{\xi_i}.$$

As it holds that $e(\mathsf{com}, \hat{P}) = e(P, \hat{R}) \prod_{i=1}^{q} e(X_i, \hat{N}_i) = e(P, \hat{R}') \prod_{i=1}^{q} e(X_i, \hat{N}'_i)$, and by the structure of X_i, it follows that

$$\begin{aligned} 1 &= e\left(P, \frac{\hat{R}}{\hat{R}'}\right) \cdot \prod_{i=1}^{q} e\left(h_z^{x_i} \cdot P^{\xi_i}, \frac{\hat{N}_i}{\hat{N}'_i}\right) \\ &= e\left(P, \frac{\hat{R}}{\hat{R}'}\right) \cdot e\left(P, \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{\xi_i}\right) \cdot e\left(h_z, \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{x_i}\right) \\ &= e\left(P, \frac{\hat{R}}{\hat{R}'} \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{\xi_i}\right) \cdot \left(h_z, \prod_{i=1}^{q} \left(\frac{\hat{N}_i}{\hat{N}'_i}\right)^{x_i}\right) = e\left(P, \hat{R}^*\right) \cdot e\left(h_z, \hat{Z}^*\right). \end{aligned}$$

We first verify that the simulated inputs to $\mathcal{A}$ are correctly distributed. In KeyGen, each X_i distributes uniformly over $\mathbb{G}_1$. Thus, the simulated parameters are statistically close to the real ones.

Moreover, note that since a valid output from $\mathcal{A}$ satisfies $msg \neq msg'$, there exists an index $i^* \in [q]$ such that $\hat{N}_{i^*} \neq \hat{N}'_{i^*}$. The view of $\mathcal{A}$ is independent of x_{i^*} as x_{i^*} is information theoretically hidden in X_{i^*}. Thus $\hat{Z}^*$ follows the distribution of $\left(\frac{\hat{N}_{i^*}}{\hat{N}'_{i^*}}\right)^{x_i^*}$. Since $\frac{\hat{N}_{i^*}}{\hat{N}'_{i^*}} \neq 1$ and x_{i^*} is uniform over $\mathbb{Z}_p^*$, we conclude that $\hat{Z}^* \neq 1$ with overwhelming probability.

Thus $\hat{Z}^*, \hat{R}^* \in \mathbb{G}_2^*$ is a solution to the given DBP instance, and $\mathcal{B}$ succeeds with roughly the same probability as $\mathcal{A}$. □

4 Structure Preserving Vector Commitments

We next introduce our structure preserving vector commitment (SPVC) scheme over a message space $\mathbf{M} \in \mathbb{G}_1$ (or $\mathbb{G}_2$) within the standard position-binding model based on the q-DHE assumption, referred to as q-DHE SPVC. Before diving into the construction, we define a structured message space built on the q-DHE parameters, which can be viewed as a CRS. This design of a message structure on top of a CRS is inspired by [42] (although their CRS and the way it is used differ from ours). Indeed, this message space enables the construction of the SPVC based on the q-DHE VC schemes [41, 54] in the strong model, i.e., transforming these constructions for $m \in \mathbb{Z}_p$ into a structure-preserving commitment. Furthermore, it allows us to extract the discrete logarithm of the messages in the proof (see Lemma 1), reducing the construction to q-DHE.

Message Structure. 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 $\mathcal{M}^{\mathsf{pp},q}$ are determined by the q-DHE parameters (see Def. 1). The message space $\mathcal{M}^{\mathsf{pp},q}$ is defined according to these parameters.

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

Note that pp does not include $B_{q+1} = P^{\alpha^{q+1}}$, meaning that no index $j = q+1$ exist. We use bases in $\mathbb{G}_2$ meaning $\hat{\mathbf{B}} = \{\hat{B}_1, \ldots, \hat{B}_q\}$ to verify whether a vector is in the message space. Specifically, given a message msg, we can check whether the pairing satisfies the relationship with respect to i and j as defined in (1):

$$e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{P}) \wedge e(M_{i1}, \hat{P}) = e(M_{i0}, \hat{B}_i) \quad (2)$$

Example: To clarify this process, let's consider an example with $q = 3$.

$$\begin{aligned} msg_1 &= ((M_{10}, M_{11}), (T_{12}, T_{13})) = (\underbrace{(P^{m_1}, P^{\alpha^1 \cdot m_1})}_{\text{main part}}, \underbrace{(B_2^{m_1}, B_3^{m_1})}_{\text{tag}}), \\ msg_2 &= ((M_{20}, M_{21}), (T_{23}, T_{25})) = ((P^{m_2}, P^{\alpha^2 \cdot m_2}), (B_3^{m_2}, B_5^{m_2})), \text{ and} \\ msg_3 &= ((M_{30}, M_{31}), (T_{35}, T_{36})) = ((P^{m_3}, P^{\alpha^3 \cdot m_3}), (B_5^{m_3}, B_6^{m_3}))\,. \end{aligned}$$

Note that j cannot be $q+1$ (which is 4 in this example). Considering that $T_{ij} = B_j^{m_i} = P^{m_i \cdot \alpha^j}$, we get:

$$\begin{aligned} msg_1 &= ((P^{m_1}, P^{\alpha^1 \cdot m_1}), (P^{\alpha^2 \cdot m_1}, P^{\alpha^3 \cdot m_1})), \\ msg_2 &= ((P^{m_2}, P^{\alpha^2 \cdot m_2}), (P^{\alpha^3 \cdot m_2}, P^{\alpha^5 \cdot m_2})), \text{ and} \\ msg_3 &= ((P^{m_3}, P^{\alpha^3 \cdot m_3}), (P^{\alpha^5 \cdot m_3}, P^{\alpha^6 \cdot m_3}))\,. \end{aligned}$$

One can verify each message as follows: For msg_1, (1) corresponds to the following three equations:

$$i = 1, j = 2 : e(M_{11}, \hat{B}_{2-1}) = e(T_{12}, \hat{P}) \Rightarrow e(P^{m_1 \cdot \alpha^1}, \hat{P}^{\alpha^1}) = e(P^{m_2 \cdot \alpha^2}, \hat{P})$$

$$i = 1, j = 3 : e(M_{11}, \hat{B}_{3-1}) = e(T_{13}, \hat{P}) \Rightarrow e(P^{m_1 \cdot \alpha^1}, \hat{P}^{\alpha^2}) = e(P^{m_1 \cdot \alpha^3}, \hat{P})$$

$$e(M_{11}, \hat{P}) = e(M_{10}, \hat{B}_1) \Rightarrow e(P^{m_1 \cdot \alpha^1}, \hat{P}) = e(P^{m_1}, \hat{P}^{\alpha^1})$$

Similarly, for msg_2 we obtain:

$$i = 2, j = 3 : e(M_{21}, \hat{B}_{3-2}) = e(T_{23}, \hat{P}) \Rightarrow e(P^{m_2 \cdot \alpha^2}, \hat{P}^{\alpha^1}) = e(P^{m_2 \cdot \alpha^3}, \hat{P})$$

$$i = 2, j = 5 : e(M_{21}, \hat{B}_{5-2}) = e(T_{25}, \hat{P}) \Rightarrow e(P^{m_2 \cdot \alpha^2}, \hat{P}^{\alpha^3}) = e(P^{m_2 \cdot \alpha^5}, \hat{P})$$

$$e(M_{21}, \hat{P}) = e(M_{20}, \hat{B}_2) \Rightarrow e(P^{m_2 \cdot \alpha^2}, \hat{P}) = e(P^{m_2}, \hat{P}^{\alpha^2})$$

This process holds also the same for msg_3.

Construction. Based on the q-DHE assumption, we now present a SPVC in the standard model which ensures that the commitment can be generated even in the presence of malicious behavior. In this model, tags are used to generate the proof/witness, while the main messages are used for verification and commitment.

In the definition, we introduce an auxiliary subroutine (VerifyMsg) to verify the correctness of messages with respect to the scheme parameters. This is only to ease presentation and avoid duplication in the presentation.

Scheme 2 (q-DHE SPVC) *Our q-DHE SPVC scheme is defined as follows:*

KeyGen$(1^\lambda, q)$: Given the security parameter λ and the size q of the committed vector, the key generation picks $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$ and outputs some public parameters:

$$\mathsf{pp} = \begin{pmatrix} \mathsf{BG}, B_1 = P^{\alpha^1}, \ldots, B_q = P^{\alpha^q}, B_{q+2} = P^{\alpha^{q+2}}, \ldots, B_{2q} = P^{\alpha^{2q}}; \\ \hat{B}_1 = \hat{P}^{\alpha^1}, \ldots, \hat{B}_q = \hat{P}^{\alpha^q}; g_t^{\alpha^{q+1}} \end{pmatrix}$$

Commit$(msg_1, \ldots, msg_q)$: On input a vector of q messages $(msg_1, \ldots, msg_q) \in \mathcal{M}^{\mathsf{pp},q}$, check if messages are correctly generated via $1 \leftarrow \mathsf{VerifyMsg}(\mathsf{pp}, msg_i)$ for all $i \in [q]$. Pick $r \leftarrow\!\!\$\ \mathbb{Z}$ and output a commitment com and auxiliary information aux:

$$\mathsf{com} = \left(C = P^r \cdot \prod_{i \in [q]} M_{i1} \right) \wedge \mathsf{aux} = r$$

Open(msg, i, aux): This algorithm is run by the committer to produce a proof π_i that msg is the i-th committed message:

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

Verify$(\mathsf{com}, msg, i, \pi_i)$: The verification algorithm accepts (i.e., it outputs 1) if and only if the messages are well-formed (i.e., $1 \leftarrow \mathsf{VerifyMsg}(\mathsf{pp}, msg_i)$ for all $i \in [q]$), and also π_i is a valid proof that com was created for msg_i:

$$e\left(C, \hat{B}_{q+1-i}\right) = e\left(\pi_i, \hat{P}\right) \cdot e\left(M_{i1}, \hat{B}_{q+1-i}\right)$$

VerifyMsg(pp, msg): Takes a message $msg_i = (\mathbf{M} = (M_{i0}, M_{i1}), \mathbf{T} = (T_{ij}))$ and verifies whether it is well-formed with respect to the pp. It returns 1 if the following equation holds, and 0 otherwise.

$$e\left(M_{i1}, \hat{B}_{j-i}\right) = e\left(T_{ij}, \hat{P}\right) \wedge e\left(M_{i1}, \hat{P}\right) = e\left(M_{i0}, \hat{B}_i\right)$$

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

Our commitment scheme also allows for updates (cf. Sec. 2.3). Moreover, since all elements of the scheme are group elements, we can consistently randomize commitments and messages. To emphasize this randomization property, we define a new algorithm, Rand, specifically for this purpose.

$\mathsf{Update}(\mathsf{com}, msg, msg', i)$: This algorithm is run by the committer who produced com and wants to update it by changing the i-th message to $msg'_i = (\mathbf{M}', \mathbf{T}')$. Check if the new message is created correctly via $\mathsf{VerifyMsg}$ then outputs a new commitment com' as: $C' = (C/M_{i1}) \cdot M'_{i1}$.

$\mathsf{Rand}(\mathsf{com}, \mathsf{aux}, \pi_i, msg_i) \to (\mathsf{com}', \mathsf{aux}', \pi'_i, msg'_i)$: Randomize the commitment and proof for a randomized message msg'_i as: Pick a random $\mu \leftarrow\$ \mathbb{Z}_p$ and compute:

$$\mathsf{com}' = C' = C^{\mu} \wedge \pi'_i = \pi_i^{\mu},$$

which is valid for the randomized message $msg'_i = (\mathbf{M}^{\mu}, \mathbf{T}^{\mu})$. Update aux with μ, i.e., set $\mathsf{aux}' = \mu \cdot \mathsf{aux}$.

Correctness: If commitments are properly generated, then proofs will always satisfy the verification, which can be seen as follows:

$$\begin{aligned}
e(C, \hat{B}_{q+1-i}) &= e\left(P^r \cdot \prod_{j\in[q]} M_j, \hat{B}_{q+1-i}\right) \\
&= e\left(P^r \cdot P^{\sum_{j\in[q]} \alpha^j \cdot m_j}, \hat{P}^{\alpha^{q+1-i}}\right) \\
&= e\left(P^{r\cdot\alpha^{q+1-i}} \cdot (P^{\sum_{j\in[q]} \alpha^j \cdot m_j})^{\alpha^{q+1-i}}, \hat{P}\right) \\
&= e\left(B^r_{q+1-i}, \hat{P}\right) \cdot e\left((P^{\sum_{j\in[q]} \alpha^j \cdot m_j})^{\alpha^{q+1-i}}, \hat{P}\right) \\
&= e\left(B^r_{q+1-i}, \hat{P}\right) \cdot e\left(P^{\sum_{j\in[q]} m_j \cdot \alpha^{q+1-i+j}}, \hat{P}\right) \\
&= e\left(B^r_{q+1-i}, \hat{P}\right) \cdot e\left(P^{\sum_{j\in[q]\setminus\{i\}} m_j \cdot \alpha^{q+1-i+j}}, \hat{P}\right) \cdot e\left(P^{m_i \alpha^{q+1}}, \hat{P}\right) \\
&= e\left(B^r_{q+1-i} \cdot \prod_{j\in[q]\setminus\{i\}} P^{m_j\cdot\alpha^{q+1-i+j}}, \hat{P}\right) \cdot e\left(P^{\alpha^i \cdot m_i}, \hat{P}^{\alpha^{q+1-i}}\right) \\
&= e\left((B^r_{q+1-i} \cdot \prod_{j\in[q]\setminus\{i\}} T_{j,q+1-i+j}), \hat{P}\right) \cdot e\left(M_{i1}, \hat{B}_{q+1-i}\right) \\
&= e\left(\pi_i, P\right) \cdot e\left(M_{i1}, \hat{B}_{q+1-i}\right)
\end{aligned}$$

Our main technical result is to prove that our scheme satisfies binding in the generic group model (GGM) [65] for asymmetric (type-3) bilinear groups, for which there are no efficiently computable homomorphism between P and $\hat{P}$. In this model, the adversary is only given handles of group elements, which are uniform random strings. To perform group operations, it uses an oracle to which it can submit handles and receives back the handle of the sum, inversion, etc., of the group elements for which it submitted handles.

Before proving the binding property of our scheme, we further need the following auxiliary lemma:

Lemma 1 (Extraction of Discrete Logarithms from Valid Messages). *Let* pp *be a public parameter. If the messages satisfy the following conditions for all* $i \in [q]$*:*

$$e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{P}) \quad \textit{and} \quad e(M_{i1}, \hat{P}) = e(M_{i0}, \hat{B}_i),$$

then the adversary is able to extract the discrete logarithms $\{m_i\}_{i\in[q]}$ *such that:*

$$m_i = \mathsf{dlog}_{B_i}(M_{i1}) = \mathsf{dlog}_{B_j}(T_{ij}) \quad \textit{for } 1 \le i \le q,$$

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

Proof. For a fixed i, consider the values $M_{i0}, M_{i1}, \{T_{ij}\}_{j=i+1, j\neq q+1}^{i+q}$ output by an adversary. With P and $\{B_u\}_{u=1,u\neq q+1}^{2q}$ being the values specified in pp, these values must now have the form:

$$M_{i0} = P^{b_0} \cdot \prod_{\substack{k=1\\k\neq q+1}}^{2q} B_k^{b_k}\,, \quad M_{i1} = P^{c_0} \cdot \prod_{\substack{k=1\\k\neq q+1}}^{2q} B_k^{c_k}\,, \text{ and } \quad T_{ij} = P^{a_{j0}} \cdot \prod_{\substack{k=1\\k\neq q+1}}^{2q} B_k^{a_{jk}}\,,$$

where all b_u, c_u, a_{ju} are known to the adversary. For the remainder of this proof, all sums and products are over $k = 1, \ldots, 2q$ with $k \neq q+1$, which will be omitted for notational convenience.

Using the structure of the B_j as defined in in Eq. 1 and taking the discrete logarithm in P we obtain:

$$m_{i0} = b_0 + \sum b_k \alpha^k \tag{3}$$

$$m_{i1} = c_0 + \sum c_k \alpha^k \tag{4}$$

$$t_{ij} = a_{j0} + \sum a_{jk}\alpha^k \quad \forall j = i+1, \ldots, i+q,\ j\neq q+1\,. \tag{5}$$

Furthermore, by the verification equations $e(M_{i1}, B_{j-i}) = e(T_{ij}, P)$ and $e(M_{i1}, P) = e(M_{i0}, B_i)$ we obtain by a similar argument that:

$$m_{i1}\alpha^{j-i} = t_{ij} \qquad \forall j = i+1, \ldots, i+q,\ j \neq q+1 \tag{6}$$

$$m_{i1} = m_{i0}\alpha^i\,. \tag{7}$$

$\mathbf{c_0, \ldots, c_{i-1} = 0}$: By combining Equations (3), (4) and (7) we obtain:

$$c_0 + \sum c_k\alpha^k = b_0\alpha^i + \sum b_k \alpha^{k+i}\,.$$

Given that the lowest degree on the right hand side is i, we directly obtain that $c_0 = \cdots = c_{i-1} = 0$.

$\mathbf{c_{i+1}, \ldots, c_q = 0}$: By combining Equations (4) to (6) we obtain for all j:

$$c_0 + \sum c_k \alpha^{k+j-i} = a_{j0} + \sum a_{jk}\alpha^k\,. \tag{8}$$

As α^{q+1} does not occur on the right hand side, the term $c_{q+1-j+i}\alpha^{q+1}$ on the left hand side must be 0 for all $j = i+1, \ldots, i+q$ satisfying $j \neq q+1$. Thus, in particular for $j = i+1, \ldots, q$, it follows that $c_{i+1} = \cdots = c_q = 0$.

$\mathbf{c_{q+2}, \ldots, c_{q+i-1}, c_{q+i+1}, \ldots, c_{2q} = 0}$: In (8), the highest degree on the right hand side equals $2q$. Thus, the coefficients of α^{2q+1} on the right hand side equals 0, i.e., $c_{2q+1-j+i} = 0$ for all $j = i+1, \ldots, i+q$ satisfying $j \neq q+1$. This immediately yields $c_{q+2} = \cdots = c_{2q} = 0$ except for c_{q+i} corresponding to $j = q+1$.

$\mathbf{c_{q+i} = 0}$: In the case that $i = 1$, there is nothing to prove as there is no c_{q+1} in (4). For $i > 1$, consider the term $c_{q+i}\alpha^{q+j}$ in (8) for $j = q+2$. As α^{2q+2} does not exist on the right hand side, it directly follows that $c_{q+i} = 0$.

Combining the above observations we obtain that $c_k = 0$ for all $k \neq i$. Rewriting (8) now yields for all j that:

$$c_i\alpha^j = a_{j0} + \sum a_{jk}\alpha^k\,.$$

Comparing coefficients gives us that $a_{jk} = 0$ for all $k \neq j$ and $a_{jj} = c_i$, such that $t_{ij} = c_i\alpha^j$.

Overall, this implies that $M_{i1} = P^{c_0} \cdot \prod B_k^{c_k} = B_i^{c_i}$ and $T_{ij} = P^{a_{j0}} \cdot \prod B_k^{a_{jk}} = B_j^{c_i}$, or equivalently $m_i := c_i = \mathsf{dlog}_{B_i} M_i = \mathsf{dlog}_{B_j} T_{ij}$ for all $j = i+1, \ldots, i+q$ with $j \neq q+1$. Thus, the adversary must be able to extract the discrete log of the message, and thus by induction, must always know the discrete logs of messages during the game. □

Theorem 2. *Scheme 2 is binding in the Generic Group Model (GGM) assuming the q-DHE assumption.*

Proof. Let $\mathcal{A}$ come up with a commitment (C, aux), an index $i \in \{1, \dots, q\}$, an valid openings π_i and π'_i to distinct messages $(\mathbf{M}, \mathbf{T}), (\mathbf{M}', \mathbf{T}')$. By Lemma 1, we can now extract the corresponding $m_i \neq m'_i$.

From the verification equations, it follows that:

$$e(\pi_i, \hat{P}) \cdot e(P^{\alpha^i}, \hat{P}^{\alpha^{q+1-i}})^{m_i} = e(\pi'_i, \hat{P}) \cdot e(P^{\alpha^i}, \hat{P}^{\alpha^{q+1-i}})^{m'_i}$$

Simply rewriting yields $e(\pi_i/\pi'_i, \hat{P}) = e(P^{\alpha^i}, \hat{P}^{\alpha^{q+1-i}})^{m'_i - m_i}$, or equivalently $e((\pi_i/\pi'_i)^{1/(m'_i - m_i)}, \hat{P}) = e(P^{\alpha^i}, \hat{P}^{\alpha^{q+1-i}})$.

Now, since $m_i \neq m'_i$, the latter relation implies that $P^{\alpha^{q+1}} = (\pi_i/\pi'_i)^{1/(m'_i - m_i)}$ is revealed by the collision, which contradicts the q-DHE assumption. □

Remark 1. We note that in the context of vector commitments (VC), the commitments do not need to be hiding, which makes the inclusion of randomness r seem unnecessary. Moreover, the randomness r is not required for the binding proof. Nevertheless, we retain it here as it might be useful for other applications in the future or for achieving properties like hiding, which are not directly required in our current setting.

Reducing Trust in CRS. The bilinear pairing-based construction typically requires either public parameters generated in a trusted setup, which are linear in the number of elements added to the Acc, or a trusted party with a trapdoor to compute it. Trust in parameter generation can be reduced or removed using MPC protocols, such as [15]. Alternatively, Groth et al. [43] proposed updatable reference strings, which allow any party to update them securely, as demonstrated in Ethereum's 'powers of tau' ceremony [61].

5 Structure-Preserving Accumulator

In this section, we introduce the concept of a structure-preserving accumulator (SPA) and demonstrate how our vector commitment can be adapted into an accumulator, resulting in the first structure-preserving accumulator of this kind.

SPA **from (signature-based) Weakly Binding VC (Def. 4).** A weakly binding vector commitment can naturally be expressed as an accumulator, as both schemes are essentially equivalent when the accumulator and the vector commitment are honestly generated. Consequently, the witness becomes a signature for the element M_i, which is bound to specific public parameters

5.1 Blackbox Accumulators from SPVC

SPA **from q-DHE VC**: Catalano and Fiore [23] proposed a black-box construction of accumulators based on vector commitments. Their approach involves creating a succinct commitment C to a vector $\mathbf{X} = (x_1, \dots, x_q)$ through a vector commitment. The commitment C ensures that it is computationally infeasible to open any position i to a value x'_i different from the original x_i. In their construction, the accumulation domain is represented by the set $D = \{1, \dots, q\}$, and the accumulator is modeled as a commitment to a binary vector of length t. Each bit i indicates whether the element $i \in D$ is included in the accumulator. Membership or non-membership of an element is verified by revealing the corresponding position i of the commitment as either 1 or 0. However, it is not clear how to apply this result to the SP setting as our SPVC is not suitable for committing to messages that are mapped to integers i (i.e., messages index).

Compiler From Vector Commitments to Accumulators. We show in the following how any SPVC can be used to construct a SPA. Actually, our generic construction achieves a somewhat stronger notion of collision resistance, where the accumulator is fully controlled by the adversary. The adversary is now considered to win if it is able to generate an accumulator and openings to more (i.e., at least $q+1$) distinct values than the upper bound of the accumulator allows for:

Definition 9 (Strong collision resistance). *An accumulator scheme is said to satisfy strong collision resistance if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}_{acc}, \mathsf{pk}_{acc}) \leftarrow \mathsf{Setup}(1^\lambda, q), (opt, \{(x_k, wit_{x_k})\}_{k=1}^{q+1}, \mathsf{Acc}) \leftarrow \mathcal{A}(\mathsf{pk}_{acc}) : \\ \forall k \in [q+1] : \mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}, wit_{x_k}, x_k) = 1 \land \forall m \neq n \in [q+1] : x_m \neq x_n\end{array}\right]$$

Note that this definition is parametrized by the additional output *opt* requested from the adversary $\mathcal{A}$. Depending on how much information the adversary needs to provide, the model can be either stronger or weaker. For our black-box accumulators from SPVC in the following, we set $opt = \perp$, meaning that $\mathcal{A}$ does not need to output anything here, which implies a stronger model.[6] In the next section, we will request the adversary to output some randomness.

Generic Construction: Indeed, the construction is very straightforward:

- $\mathsf{Setup}(1^\lambda, q)$ first samples pp as the public parameters of the SPVC. Furthermore, it samples pairwise distinct group elements $(U_1, \ldots, U_q)$.[7] The algorithm sets $\mathsf{pk}_{acc} \leftarrow (\mathsf{pp}, (U_1, \ldots, U_q))$, as well as sk_{acc} is set to $\perp$.
- To compute the accumulator, Eval first pads $\mathcal{X}$ with random elements in the case that $|\mathcal{X}| \leq q$. It then defines a mapping of the elements to the index set $\{1, \ldots, q\}$, which is stored as aux. The actual accumulator value Acc is then obtained by simply running the Commit algorithm.
- To create a witness for a value x, WitCreate first uses aux to re-identify the position i_x corresponding to x, and then runs the Open algorithm on the corresponding inputs. It outputs $wit_x = (U_{i_x}, \pi_x)$.
- The Verify now works canonically by outputting the result of the SPVC's Verify algorithm.

The strong collision resistance of this construction can now be seen by a simple observation. If the adversary can output $q+1$ values and valid witnesses (corresponding to messages and openings of a commitment) for a fixed accumulator value Acc (corresponding to a commitment value), at least two witnesses $wit_x, wit_{x'}$ for $x \neq x'$ must contain the same index $i_x = i_{x'}$. Therefore, the corresponding $\pi_x, \pi_{x'}$ are valid openings for the same position, immediately breaking the position binding property of the SPVC.

5.2 (Perfect Randomizable) Accumulator based on the q-DHE VC

The above construction reveals the position of an element in the underlying vector commitment, which unfortunately turns out to be insufficient for some privacy-enhancing primitives such as ring signatures. In this case, revealing the index of public key which was used for signing would immediately violate anonymity. Therefore, we are interested in an approach where the witness does not reveal sensitive information (beyond the message), not even the index, which however turns out to be non-trivial.

In the following, we now introduce such a (perfectly randomizable) q-DHE Structure-Preserving Accumulator (q-DHE SPA) in which the witness does not disclose any information beyond the message itself – not even its index. By making minor adjustments to the vector commitment scheme outlined in Scheme 2, we achieve that the message is no longer tied to any specific position or element in pp.

To achieve this, we randomize the elements that reveal the positions. Specifically, the message msg_i and the associated tags can disclose the index through the relations $e(M_{i1}, \hat{B}_{j-i}) = e(T_{ij}, \hat{P})$ and $e(M_{i1}, \hat{P}) = e(M_{i0}, \hat{B}_i)$ (i.e., the message/tag verification needs public elements $\hat{B}_{j-i}$). To address this, we use a random value $y \in \mathbb{Z}_p$ and replace the verification equation with the following: $e(M_{i1}, \hat{Y}) = e(T_{ij}^y, \hat{P})$ and $e(M_{i1}, \hat{P}^y) = e(M_{i0}, \hat{B}_i^y)$, where $\hat{B}_{j-i}^y$ is applied for all j, and $\mathbf{\hat{Y}} = (\hat{P}^y, \hat{B}_i^y)_{i \in [q]}$ is included as part of the tag. In the subsequent step, the verification equation $e(C, \hat{B}_{q+1-i}) = e(\pi_i, \hat{P}) \cdot e(M_{i1}, \hat{B}_{q+1-i})$ is initially designed to confirm the positions of the messages with respect to $\hat{B}_{q+1-i}$. We randomize this verification by picking a new random $\rho \in \mathbb{Z}_p$, modifying the equation to $e(C, \hat{W}_2) = e(\pi_i, \hat{P}) \cdot e(M_{i1}, \hat{W}_2)$, where $\hat{W}_2 = (\hat{B}_{q+1-i})^\rho$ is part of the witness. By using the bilinear pairing property, we can appropriately randomize π_i with ρ, thereby ensuring that the verification remains valid as intended.

We handle new and updated tags/messages values, and the randomization of witness (or proofs in SPVC) and commitments by defining two additional algorithms, UpTagMsg and Rand, which generate tags for messages in the set and randomize the witness/accumulators, respectively. Since we now

[6] Informally, e.g., setting *opt* to contain aux as well as the random coins used to generate the accumulator can force the adversary to act semi-honestly. Alternatively, setting *opt* to contain only information for which knowledge is typically proven in higher-level applications also offers additional flexibility in the design of accumulator schemes.

[7] E.g., one can set $U_i = P^i$, it is only used for binding messages to specific public elements.

require some additional tag elements and their verification, we slightly adjust Def. 6 and Def. 7 to reflect these properties.

Definition 10 (Extended Static Accumulator). *A Extended static accumulator has the following additional algorithms in addition to Static Accumulator Def. 6:*

$\mathsf{UpTagMsg}(\mathsf{pk}_{acc}, \mathcal{X}, y)$: Given a public key pk_{acc} and a set $\mathcal{X}$. For each $msg_i \in \mathcal{X}$, where $msg_i = (\mathbf{M}_i, \mathbf{T}_i) \in \mathcal{M}^{\mathsf{pp},q}$, first check if the messages are correctly generated using $\mathsf{SPVC.VerifyMsg}$ then output the updated messages/tags $msg = (\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}})$ for all msg in $\mathcal{X}$ using a random $y \in \mathbb{Z}_p^*$.

$\mathsf{VerifyMsgTag}(\mathsf{pp}, msg)$: Takes a message and tag $msg = (\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}})$, verifies whether it is well-formed with respect to the pp.

$\mathsf{Rand}(\mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i, (\mu, \beta, \gamma))$: On input an accumulator $\mathsf{Acc}_{\mathcal{X}}$, witness wit_{msg_i}, message msg_i and randomness (μ, β, γ), compute a randomized accumulator and witness $(\mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i})$ for a randomized message msg'_i with $(\beta, \mu, \gamma) \in \mathbb{Z}_p$ and output $(\mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i}, msg'_i)$.

We present the complete construction as follows:

Scheme 3 (q-DHE Accumulator) *Let* SPVC *be the vector commitment in Scheme 2, our q-DHE randomizable* SPA *is defined as follows:*

$\mathsf{Setup}(1^\lambda, q)$: Given a security parameter λ and a parameter q, run $\mathsf{pp} \leftarrow \mathsf{SPVC.KeyGen}(1^\lambda, q)$ and set $\mathsf{pk}_{acc} = \mathsf{pp}$ and $\mathsf{sk}_{acc} = \perp$.

$\mathsf{UpTagMsg}(\mathsf{pk}_{acc}, \mathcal{X}, y)$: Given a public key pk_{acc} and a set $\mathcal{X}$, do the following:
- for each $msg_i \in \mathcal{X}$, where $msg_i = (\mathbf{M}_i, \mathbf{T}_i) \in \mathcal{M}^{\mathsf{pp},q}$, first check if for all i it holds that: $1 \leftarrow \mathsf{SPVC.VerifyMsg}(\mathsf{pp}, msg_i)$ (i.e., the messages are correctly generated).
- for each $msg_i \in \mathcal{X}$ check that $M_{i0} \neq 1$ (i.e., exclude $m = 0$).
- If it holds, update $\mathbf{T} = \mathbf{T}^y$.
- Compute $\hat{\mathbf{Y}} = (\hat{Y}_0 = \hat{P}^y, \hat{Y}_i = \hat{B}_i^y)_{i \in [q]}$.
- Output the updated messages/tags $(\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}})$ for all msg in $\mathcal{X}$. We assume msg contains the tags $\hat{\mathbf{Y}}$.

$\mathsf{VerifyMsgTag}(\mathsf{pp}, msg)$: Takes a message $msg = (\mathbf{M}, \mathbf{T})$ and tag $\hat{\mathbf{Y}}$, verifies whether it is well-formed with respect to the pp. Furthermore, check that $M_{i0} \neq 1$. It returns 1 if the following equation holds, and 0 otherwise.

$$e(M_{i1}, \hat{Y}_{j-i}) = e(T_{ij}, \hat{P}) \wedge e(M_{i1}, \hat{Y}_0) = e(M_{i0}, \hat{Y}_i)$$

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

$\mathsf{Eval}(\mathsf{pk}_{acc}, \mathcal{X})$: Given a public key pk_{acc} and set $\mathcal{X}$ it returns an accumulator $\mathrm{acc}_{\mathcal{X}}$ together with the aux as follows:
- Sample $y \in \mathbb{Z}_p^*$, and run $(\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}}) \leftarrow \mathsf{UpTagMsg}(\mathsf{pk}_{acc}, \mathcal{X}, y)$.
- Check for all if $1 \leftarrow \mathsf{VerifyMsgTag}(\mathsf{pp}, msg_i)$ for all $msg_i = (\mathbf{M} = (M_{i0}, M_{i1}), \mathbf{T}, \hat{\mathbf{Y}}) \in \mathcal{X}$.
- For a random $r \in \mathbb{Z}_p$ set, compute:

$$\mathsf{Acc}_{\mathcal{X}} = C = \left(P^r \cdot \prod_{i \in [q]} M_{i1} \right) \wedge \ \mathsf{aux} = (r, y)$$

Note that $\mathsf{Acc}_{\mathcal{X}}$ is similar to run $\mathsf{SPVC.Commit}(msg_1, \ldots, msg_q)$, where $msg_i = (\mathbf{M}, \mathbf{T})$.

$\mathsf{WitCreate}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, \mathsf{aux}, msg_i)$: This algorithm takes a key pair pk_{acc}, an accumulator $\mathsf{Acc}_{\mathcal{X}}$, auxiliary information aux, and a value msg_i. It returns $\perp$, if $msg_i \notin \mathcal{X}$ or $0 \leftarrow \mathsf{VerifyMsgTag}(\mathsf{pp}, msg_i)$, otherwise, compute a witness wit_{msg_i} for msg_i: Pick $\rho \leftarrow\!\!\$\ \mathbb{Z}_p^*$, and compute wit_{msg_i}

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

$\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i)$: This algorithm takes a public key pk_{acc}, an accumulator $\mathsf{Acc}_{\mathcal{X}}$, a witness wit_{msg_i}, and a value $msg_i = (\mathbf{M} = (M_{i0}, M_{i1}), \mathbf{T}, \hat{\mathbf{Y}})$. It returns *true* (i.e., it outputs 1) if the message and tag are created correctly $1 \leftarrow \mathsf{VerifyMsgTag}(\mathsf{pp}, msg_i)$ and check if wit_{msg_i} is a valid witness for $msg_i \in \mathcal{X}$ and *false* (i.e., it outputs 0) otherwise as:

$$e(C, \hat{W}_2) = e(W_1, \hat{P}) \cdot e(M_{i1}, \hat{W}_2)$$

$\mathsf{Rand}(\mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i, (\mu, \beta, \gamma)) \rightarrow (\mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i}, msg'_i)$: On input an accumulator $\mathsf{Acc}_{\mathcal{X}}$, witness wit_{msg_i}, message msg_i and randomness (μ, β, γ), compute a randomized accumulator and witness for a randomized message msg'_i with $(\beta, \mu, \gamma) \in \mathbb{Z}_p$ as:

$$\mathsf{Acc}'_{\mathcal{X}} = \mathsf{Acc}^{\mu}_{\mathcal{X}} \wedge wit'_{msg_i} = (W_1^{\mu\gamma}, \hat{W}_2^{\gamma}) \wedge msg'_i = (\mathbf{M}^{\mu}, \mathbf{T}^{\mu\beta}, \hat{\mathbf{Y}}^{\beta}).$$

This is valid accumulator-witness pair for the randomized message. Finally, aux is updated with (μ, β, γ).

Note that one can also randomize only the witness without randomizing the messages or the accumulator, i.e., by computing $W_1^{\mu}, \hat{W}_2^{\mu}$ as randomized witnesses. Moreover, the verification does not require knowledge of message position (index), or public value that binds messages to specific positions.

Collision resistance: We take the strong collision resistance definition (cf. Def. 9), parametrized with $opt = (r, y, \rho)$. That is, we request the adversary to also output the randomness used to create accumulators and the set $\mathcal{X}$, i.e., to derive the messages $msg = (\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}})$ as well as witness. The challenger then checks whether the messages in the set are valid using $\mathsf{VerifyMsgTag}$.

In higher-level protocols, this is equivalent to proving knowledge of (r, y, ρ) which in the formal analysis allows a reduction to extract them. This is a meaningful assumption in many applications. For instance, for ring signatures, one would need to prove that the key that was used for signing corresponds to one of the public keys in the ring (i.e., before re-randomization).

Theorem 3. *If the q-DHE Vector Commitment in Scheme 2 is position-binding, then the q-DHE Accumulator in Scheme 3 is strongly collision resistant according to Def. 9 with $opt = (r, y, \rho)$ as explained above.*

Proof. Let $\mathcal{A}$ be an adversary against the collision resistance property of Scheme 3. We show how to build an equally efficient adversary $\mathcal{B}$ against the position-binding property of the vector commitment. $\mathcal{B}$ receives as input the parameters pp, and sets $\mathsf{pk} = \mathsf{pp}$ and $\mathsf{sk} = \perp$ and send pk to $\mathcal{A}$. At some point, $\mathcal{A}$ hands to $\mathcal{B}$ a tuple $(opt = (\mathsf{aux} = (r, y), \rho), \{(msg_k, wit_{msg_k})\}_{k=1}^{q+1}, \mathsf{Acc})$. Notice that, in order to break collision-resistance it must hold hat all messages are distinct and also:

$$\forall k \in [q+1] : \mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}, wit_{msg_k}, msg_k) = 1$$

Also all msg in $\mathcal{X}$ should be correctly generated as mentioned in $\mathsf{UpTagMsg}$: $msg_i = (\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}}))$ for all $msg_i \in \mathcal{X}$ using $\mathsf{VerifyMsgTag}$.

We note that both the accumulator C and π are the same as vector commitment and opening. With the help of (y, ρ), now the reduction $\mathcal{B}$ can derandomize $(\hat{W}'_2)^{1/\rho \cdot y}$, and this is equal to $\hat{P}^{\alpha^{q+1-i}}$ (from this, we can determine the required index i for msg'). Similarly, it derandomizes the tag as well, $\mathbf{T} = \mathbf{T}'^{1/y}$, to be able to check and pass $\mathsf{VerifyMsg}$ in the SPVC i.e., $(M'_{i1}, B_{j-i}) = (T_{ij}, \hat{P})$. It is clear that $(W'_1)^{1/\rho \cdot y}$ and $msg'_i = (\mathbf{M}', \mathbf{T}')$ are now valid proofs for our vector commitment. Moreover, $\mathcal{B}$ can compute locally $\pi_i = wit_i$ for a valid msg and index i, as all tags $\mathbf{T}$ for all messages in $\mathcal{X}$ are available. This means the tuple $((\mathrm{acc}_{\mathcal{X}} = C, \mathsf{aux} = r), msg, msg', i, \pi_i, \pi'_i = (W'_1)^{1/\rho \cdot y})$ will contradict the position-binding property of the underlying vector commitment. □

Perfect Randomization We define a privacy property here inspired by signature adaptation in SPSEQ signatures [37]. We want to guarantee that fresh and randomized accumulators, messages and witnesses are indistinguishable, which is important to guarantee the unlinkability of witness presentation. As fresh and randomized instances look identical and do not leak any information, the messages are not associated with particular values or public parameters, thus making it infeasible to identify their location in the set.

Definition 11 (Perfect Randomization of Acc). *An accumulator scheme provides perfect randomization if for all λ, and for all $\mathsf{pk}_{acc} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and for honestly generated $(\mathsf{pk}_{acc}, msg_i, \mathsf{Acc}_{\mathcal{X}}, \mathsf{aux}, wit_{msg_i})$, the following holds:*
If $\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i) = 1$, then $(\mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i}, msg'_i) \leftarrow \mathsf{Rand}(\mathsf{Acc}, wit_{msg_i}, msg_i)$ satisfies that $\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i}, msg'_i) = 1$ and the distributions satisfy $(\mathsf{Acc}'_{\mathcal{X}}, wit'_{msg_i}, msg'_i) \approx (\mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i)$.

Theorem 4. *The q-DHE Accumulator in Scheme 3 is perfectly randomizable.*

Proof. Let $(\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}}) \in (\mathbb{G}_1^*)^2 \times (\mathbb{G}_1^*)^q \times (\mathbb{G}_2^*)^q$, $\mathsf{pk}_{acc} \in (\mathbb{G}_1^*)^{2q-1} \times (\mathbb{G}_2^*)^q$ and $\alpha \in \mathbb{Z}_p^*$. An accumulator, witness and messages/tags $(msg_i, \mathsf{Acc}_{\mathcal{X}}, wit_{msg_i})$ satisfying $\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}_{\mathcal{X}}, wit_{msg_i}, msg_i) = 1$ are of the form:

$$\mathsf{Acc} = \left(P^r \cdot \prod_{i \in [q]} M_{i1} \right),$$

$$wit = \left(W_1 = (B^r_{q+1-i} \cdot \prod_{j \in [q] \setminus \{i\}} T_{j,q+1-i+j})^\rho, \hat{W}_2 = (\hat{P}^{\alpha^{q+1-i}})^{\rho \cdot y} \right)$$

$$\textit{and} \quad msg_i = (\mathbf{M}, \mathbf{T}, \hat{\mathbf{Y}})$$

On the other hand, for randomness $(\mu, \beta, \gamma) \in \mathbb{Z}_p^*$, the randomization algorithm $\mathsf{Rand}(\mathsf{Acc}, wit_{msg_i}, msg_i, (\mu, \beta, \gamma))$ outputs:

$$\mathsf{Acc}' = \left(P^{r\mu} \cdot \prod_{i \in [q]} M_{i1}^\mu \right),$$

$$wit' = \left(W'_1 = (B^{r\mu \cdot \gamma}_{q+1-i} \cdot \prod_{j \in [q] \setminus \{i\}} T^{\mu\gamma}_{j,q+1-i+j})^\rho, \hat{W}'_2 = (\hat{P}^{\alpha^{q+1-i}})^{\rho \cdot y \cdot \gamma} \right)$$

$$\textit{and} \quad msg'_i = (\mathbf{M}^\mu, \mathbf{T}'^{\mu\beta}, \hat{\mathbf{Y}}^\beta)$$

which are uniformly random elements conditioned on $\mathsf{Verify}(\mathsf{pk}_{acc}, \mathsf{Acc}^\mu_{\mathcal{X}}, wit'_{msg_i}, (\mathbf{M}^\mu, \mathbf{T}'^{\mu\beta}, \hat{\mathbf{Y}}^\beta)) = 1$. Indeed, each element is perfectly randomized with fresh randomness, so it is clear that Rand and Eval are identically distributed for all $(\mathsf{Acc}', wit'_{msg_i}, msg'_i)$. □

6 Applications

Compressing primitives such as accumulators and vector commitments are highly versatile tools and can be used in many applications, as mentioned in the introduction. However, in this section, we describe how our SPVC and SPA primitives can lead to new and interesting applications, in addition to common ones.

6.1 Succinct Data Availability Sampling

Data availability sampling addresses a major blockchain challenge—scalability [9, 45]. This approach allows light clients to verify the availability and integrity of block data using multi-dimensional Reed-Solomon codes within an erasure coding strategy. Ethereum has shifted from fraud proofs, as highlighted in [9], to validity proofs based on polynomial commitments, leveraging their homomorphic properties. As a result, Ethereum aims to integrate this mechanism into its sharding protocol[8]. In this setting, each blockchain validator, similar to light clients, only needs to store the commitment instead of the full dataset and proof. However, within this multi-dimensional structure, clients with limited resources must store a tuple $\mathsf{com} = (C_1, \ldots, C_q)$, where each C_i is a KZG commitment [47] to a Reed-Solomon code (tensor code) [45]. We think this application can benefit from our SPVC schemes

[8] https://notes.ethereum.org/@vbuterin/protodankshardingfaq

by reducing the commitment size (and, consequently, the communication size) from 256 elements per block to just one, allowing client storage to decrease from 256 KZG commitments per shard to a single commitment. This is achieved by treating the commitment com of a block of data as a single SPVC commitment.

In our approach, one can create a commitment to these KZG commitments for a more succinct representation. Using our weakly binding SPVC, this can be achieved without significant modifications to the KZG scheme or the SPVC itself. Also for q-DHE PSVC, we can easily extend the public parameters of KZG with those received from the q-DHE PSVC to compute KZG commitments in different bases. For example, assume we have $(B_1, B_2, \hat{B}_1, \hat{B}_2)$ alongside $(P^{x^i})_{i\in[q]}$, where x is the KZG trapdoor. We need to compute $(B_1^{x^i})_{i\in[q]}$ and $(B_2^{x^i})_{i\in[q]}$ to derive a KZG commitment with respect to our SPVC message space.

6.2 Algebraic Verkle Trees

In a stateless blockchain client model, nodes do not store the entire state but rely on "witness"—compact proofs that verify the necessary state for transaction validation. Verkle Trees[9], which improve upon Merkle Trees by using VCs at the leaf nodes, enable smaller witness sizes and more efficient verification.

From a theoretical point of view, a key challenge has been the inability to commit to commitments within the same algebraic structure due to the lack of structure-preserving VCs. Algebraic Verkle Trees (AVTs) with the WSPVC resolve this issue, enabling seamless commitment to commitments without switching between cryptographic primitives like hash functions. As already mentioned, WSPVC is sufficient for stateless validation as commitments are always honestly produced through (Byzantine) agreement on a sequence of updates. This can potentially simplify verification and reduce witness sizes. AVTs also expand the Verkle Tree in both depth and width while maintaining constant-size proofs by incrementally committing to VC commitments. Moreover, we can efficiently prove knowledge of a message without needing to prove the pre-image of a hash in a SNARK.

We believe that the use of SPVC in AVTs can pave the way for future research, particularly in optimizing scalability by integrating structure-preserving VCs into frameworks, which could lead to more efficient techniques in stateless blockchain.

7 Conclusion and Future Work

In this paper, we show that strictly structure-preserving compressing primitives can be realized. We present the first strictly structure-preserving commitment that is shrinking, and in particular, constant-size. By employing a more structured message space—specifically, a variant of the DH message space—we circumvent existing impossibility results. As our main contribution, we construct structure-preserving vector commitments (SPVC) and accumulators (SPA). We begin by discussing generic constructions and then provide concrete implementations under the Diffie-Hellman Exponent assumption.

We view this work as the initial step toward building structure-preserving compressing primitives, and the first stepping stone for further study in this direction. While our scheme gives good insight, building a fully-featured, unrestricted solution remains an open problem for future research. One interesting direction for future work is designing a vector commitment scheme that utilizes a more natural message space—such as messages as simple as P^m —instead of relying on a CRS. Apart from that, it would be interesting to explore methods of compressing and shrinking every message msg, currently of size q, to constant or sublinear size as future work. Moreover, further exploration of applications for our schemes appears to be a promising research direction.

Acknowledgments

Omid Mir and Stephan Krenn were supported by the European Union's Horizon Europe project SUNRISE (grant agreement no. 101073821), PREPARED, a project funded by the Austrian security research programme KIRAS of the Federal Ministry of Finance (BMF), and the CHIST-ERA project

[9] Vitalik Buterin, Guillaume Ballet, Dankrad Feist. Verkle tree EIP, 2021.

REMINDER through the Austrian Science Fund (FWF) project number I 6650-N. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the funding agencies. Neither European Union nor the granting authorities can be held responsible for them. We want to thank anonymous reviewers for their helpful comments.

References

1. Abe, M.: Structure-preserving cryptography. Invited Talk at Asiacrypt 2015, https://www.iacr.org/archive/asiacrypt2015/94520356/94520356.pdf
2. Abe, M., Chow, S.S.M., Haralambiev, K., Ohkubo, M.: Double-trapdoor anonymous tags for traceable signatures. In: Lopez, J., Tsudik, G. (eds.) ACNS 11. LNCS, vol. 6715, pp. 183–200 (Jun 2011). https://doi.org/10.1007/978-3-642-21554-4_11
3. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236 (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_12
4. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. Journal of Cryptology **29**(2), 363–421 (Apr 2016). https://doi.org/10.1007/s00145-014-9196-7
5. Abe, M., Groth, J., Kohlweiss, M., Ohkubo, M., Tibouchi, M.: Efficient fully structure-preserving signatures and shrinking commitments. Journal of Cryptology **32**(3), 973–1025 (Jul 2019). https://doi.org/10.1007/s00145-018-9300-5
6. Abe, M., Haralambiev, K., Ohkubo, M.: Group to group commitments do not shrink. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 301–317 (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_19
7. Abe, M., Kohlweiss, M., Ohkubo, M., Tibouchi, M.: Fully structure-preserving signatures and shrinking commitments. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 35–65 (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_2
8. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440 (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_26
9. Al-Bassam, M., Sonnino, A., Buterin, V.: Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. arXiv preprint arXiv:1809.09044 (2018)
10. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132 (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_4
11. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308 (Apr 2009). https://doi.org/10.1007/978-3-642-00862-7_20
12. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact e-cash from bounded accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195 (Feb 2007). https://doi.org/10.1007/11967668_12
13. Bari, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 480–494 (May 1997). https://doi.org/10.1007/3-540-69053-0_33
14. Barthoulot, A., Blazy, O., Canard, S.: Cryptographic accumulators: new definitions, enhanced security, and delegatable proofs. In: International Conference on Cryptology in Africa. pp. 94–119. Springer (2024)
15. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy. pp. 287–304. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.25
16. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131 (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_7
17. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 274–285 (May 1994). https://doi.org/10.1007/3-540-48285-7_24
18. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500 (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_27
19. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76 (Aug 2002). https://doi.org/10.1007/3-540-45708-9_5

20. Campanelli, M., Fiore, D., Greco, N., Kolonelos, D., Nizzardo, L.: Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 3–35 (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_1
21. Campanelli, M., Fiore, D., Han, S., Kim, J., Kolonelos, D., Oh, H.: Succinct zero-knowledge batch proofs for set accumulators. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 455–469. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560677
22. Campanelli, M., Nitulescu, A., Ràfols, C., Zacharakis, A., Zapico, A.: Linear-map vector commitments and their practical applications. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 189–219 (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_7
23. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72 (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
24. Catalano, D., Fiore, D., Gennaro, R., Giunta, E.: On the impossibility of algebraic vector commitments in pairing-free groups. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, vol. 13748, pp. 274–299 (Nov 2022). https://doi.org/10.1007/978-3-031-22365-5_10
25. Couteau, G., Hartmann, D.: Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 768–798 (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1_27
26. Couteau, G., Lipmaa, H., Parisella, R., Ødegaard, A.T.: Efficient NIZKs for algebraic sets. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 128–158 (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_5
27. Crites, E.C., Kohlweiss, M., Preneel, B., Sedaghat, M., Slamanig, D.: Threshold structure-preserving signatures. In: ASIACRYPT 2023, Part II. pp. 348–382. LNCS (Dec 2023). https://doi.org/10.1007/978-981-99-8724-5_11
28. Crites, E.C., Lysyanskaya, A.: Delegatable anonymous credentials from mercurial signatures. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 535–555 (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_27
29. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538 (2008), https://eprint.iacr.org/2008/538
30. Derler, D., Hanser, C., Slamanig, D.: A new approach to efficient revocable attribute-based anonymous credentials. In: Groth, J. (ed.) Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9496, pp. 57–74. Springer (2015). https://doi.org/10.1007/978-3-319-27239-9_4, https://doi.org/10.1007/978-3-319-27239-9_4
31. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 127–144 (Apr 2015). https://doi.org/10.1007/978-3-319-16715-2_7
32. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10786, pp. 419–440. Springer (2018). https://doi.org/10.1007/978-3-319-79063-3_20, https://doi.org/10.1007/978-3-319-79063-3_20
33. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626 (May 2004). https://doi.org/10.1007/978-3-540-24676-3_36
34. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), https://eprint.iacr.org/2009/320
35. Fuchsbauer, G.: Commuting signatures and verifiable encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 224–245 (May 2011). https://doi.org/10.1007/978-3-642-20465-4_14
36. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 233–253 (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_12
37. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. Journal of Cryptology **32**(2), 498–546 (Apr 2019). https://doi.org/10.1007/s00145-018-9281-4
38. Ghadafi, E.: More efficient structure-preserving signatures - or: Bypassing the type-iii lower bounds. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10493, pp. 43–61. Springer (2017). https://doi.org/10.1007/978-3-319-66399-9_3, https://doi.org/10.1007/978-3-319-66399-9_3

39. Ghadafi, E.: Further lower bounds for structure-preserving signatures in asymmetric bilinear groups. In: Buchmann, J., Nitaj, A., eddine Rachidi, T. (eds.) AFRICACRYPT 19. LNCS, vol. 11627, pp. 409–428 (Jul 2019). https://doi.org/10.1007/978-3-030-23696-0_21
40. Ghosh, E., Ohrimenko, O., Papadopoulos, D., Tamassia, R., Triandopoulos, N.: Zero-knowledge accumulators and set algebra. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 67–100 (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_3
41. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: Aggregating proofs for multiple vector commitments. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 2007–2023. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417244
42. Griffy, S., Lysyanskaya, A., Mir, O., Kempner, O.P., Slamanig, D.: Delegatable anonymous credentials from mercurial signatures with stronger privacy. Cryptology ePrint Archive, Paper 2024/1216, to appear at ASICACRYPT'24 (2024), https://eprint.iacr.org/2024/1216
43. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728 (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
44. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432 (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
45. Hall-Andersen, M., Simkin, M., Wagner, B.: Foundations of data availability sampling. Cryptology ePrint Archive (2023)
46. Jhanwar, M.P., Safavi-Naini, R.: Compact accumulator using lattices. Cryptology ePrint Archive, Report 2014/1015 (2014), https://eprint.iacr.org/2014/1015
47. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194 (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11
48. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 530–560 (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_19
49. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 1–31 (May 2016). https://doi.org/10.1007/978-3-662-49896-5_1
50. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. Journal of Cryptology **36**(3), 23 (Jul 2023). https://doi.org/10.1007/s00145-023-09470-6
51. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307 (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_17
52. Libert, B., Peters, T., Yung, M.: Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 296–316 (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_15
53. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.30
54. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517 (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_30
55. Lipmaa, H., Parisella, R.: Set (non-)membership NIZKs from determinantal accumulators. pp. 352–374. LNCS (2023). https://doi.org/10.1007/978-3-031-44469-2_18
56. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 369–378 (Aug 1988). https://doi.org/10.1007/3-540-48184-2_32
57. Mir, O., Bauer, B., Griffy, S., Lysyanskaya, A., Slamanig, D.: Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In: ACM CCS 2023. pp. 30–44. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623203
58. Mir, O., Slamanig, D., Bauer, B., Mayrhofer, R.: Practical delegatable anonymous credentials from equivalence class signatures. In: The 23rd Privacy Enhancing Technologies Symposium. pp. 488–513. de Gruyter (2023)
59. Mitrokotsa, A., Mukherjee, S., Sedaghat, M., Slamanig, D., Tomy, J.: Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In: PKC 2024, Part I. pp. 163–195. LNCS (May 2024). https://doi.org/10.1007/978-3-031-57718-5_6
60. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292 (Feb 2005). https://doi.org/10.1007/978-3-540-30574-3_19

61. Nikolaenko, V., Ragsdale, S., Bonneau, J., Boneh, D.: Powers-of-tau to the people: Decentralizing setup ceremonies. pp. 105–134. LNCS (Jun 2024). https://doi.org/10.1007/978-3-031-54776-8_5
62. Nitulescu, A.: Sok: Vector commitments (2021), https://www.di.ens.fr/~nitulesc/files/vc-sok.pdf
63. Papamanthou, C., Shi, E., Tamassia, R., Yi, K.: Streaming authenticated data structures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 353–370 (May 2013). https://doi.org/10.1007/978-3-642-38348-9_22
64. Peikert, C., Pepin, Z., Sharp, C.: Vector and functional commitments from lattices. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 480–511 (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_16
65. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266 (May 1997). https://doi.org/10.1007/3-540-69053-0_18
66. Slamanig, D.: Dynamic accumulator based discretionary access control for outsourced storage with unlinkable access - (short paper). In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 215–222 (Feb / Mar 2012). https://doi.org/10.1007/978-3-642-32946-3_16
67. Tomescu, A., Abraham, I., Buterin, V., Drake, J., Feist, D., Khovratovich, D.: Aggregatable subvector commitments for stateless cryptocurrencies. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 45–64 (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_3
68. Wee, H., Wu, D.J.: Succinct vector, polynomial, and functional commitments from lattices. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 385–416 (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_13

A Weakly Binding Vector Commitments

We present a simple compiler in which commitments are derived from signature public keys, with auxiliary data including the corresponding secret keys and messages, while the opening/proof is simply a signature. To bind a position to a message, we introduce public information $(U_1, \ldots, U_q) \in \mathcal{M}$ into the public parameters, where random elements U_i are signed together with M_i as indices to the messages. This approach is compatible with any compact SPS. We propose an instantiation of WSPVC using the FHS signature [37], noting that message randomization is not required in this context. This message randomization can be prevented by fixing the first element of the message vector to a predetermined element U, which needs to be verified during the verification. We note that WSPVC only applies in applications where the committer (or the party generating accumulators) is honest, such as in algebraic verkle trees, where commitments are always honestly produced through (Byzantine) agreement on a sequence of updates (see Sec 6.2 for more details).

In Sec. 4, we enhance our security model to present a structure-preserving vector commitment with standard binding by incorporating message tags (identifiers). This ensures that messages will not verify unless they are computed using a compatible setup with a Common Reference String (CRS).

We present our weakly binding vector commitment WSPVC, which, as mentioned above, can be constructed from a structure-preserving (SP) signature, assuming the committer is honest and the commitments are generated correctly.

Scheme 4 (Weakly Binding VC) *A* WSPVC *is a tuple of the following algorithms:*

$\mathsf{KeyGen}(1^\lambda, q)$: Given the security parameter λ and the size q of the committed vector, the key generation run $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGSetup}(1^\lambda)$, for $i = 1, \ldots, q$, choose $U_i \leftarrow\!\!\$\ \mathbb{G}_1$ and outputs public parameters $\mathsf{pp} = (\mathsf{BG}, U_1, \ldots, U_q)$.

$\mathsf{Commit}(M_1, \ldots, M_q)$: On input a vector of q messages as $(M_1, \ldots, M_q) \in \mathcal{M} \in \mathbb{G}_1^q$ and the public parameters pp, output a commitment com and auxiliary information aux as follows: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SPS.KeyGen}(1^\lambda, 2)$ and then set

$$\mathsf{com} = \mathsf{pk} \quad \wedge \quad \mathsf{aux} = (\sigma_1, \ldots, \sigma_q)$$

Where σ_i is a SPS signature on (M_i, U_i). For random $y \leftarrow\!\!\$\ \mathbb{Z}_p$, and using the FHS SPS as example:

$$\mathsf{pk} = (\hat{X}_1 = \hat{P}^{x_1}, \hat{X}_2 = \hat{P}^{x_2}) \wedge \sigma_i = \Big(Z = (U_i^{x_1} \cdot M_i^{x_2})^{1/y}, Y = P^y, \hat{Y} = \hat{P}^y\Big)$$

$\mathsf{Open}(M, i, \mathsf{aux})$: This algorithm is run by the committer to produce a proof π_i that M is the i-th committed message. Pick the related signature $\sigma_i \leftarrow \mathsf{aux}$ and output a proof $\pi_i = \sigma_i$ for M_i.

$\mathsf{Verify}(\mathsf{com}, M, i, \pi_i)$: The verification algorithm accepts (i.e., it outputs 1) only if π_i is a valid proof that com was created for M_i by verifying $\mathsf{SPS.Verify}(\mathsf{pk} = \mathsf{com}, M_i, \sigma_i = \pi_i) = 1$. For the FHS signature scheme, this is defined as follows, parse the proof $\pi_i = (Z, Y, \hat{Y})$, the commitment $\mathsf{com} = (X_1, X_2)$ and check:

$$\mathsf{SPS.Verify} : e(Z, \hat{Y}) = e(M_i, \hat{X}_2)e(U_i, \hat{X}_1) \wedge e(Y, \hat{P}) = e(P, \hat{Y})$$

$\mathsf{Rand}(\mathsf{com}, \pi_i)$: Randomize a proof for a message M_i as: Pick a random $\mu \leftarrow\$ \mathbb{Z}_p$:

$$\pi_i' = (\sigma_i' = (Z^{1/\mu}.Y^{\mu}, \hat{Y}^{\mu}))$$

Theorem 5. *If the SPS is unforgeable, then the* WSPVC *in Scheme 4 satisfies weak binding.*

Proof (Sketch). The proof of binding is straightforward. If the signature is unforgeable, then the commitment is position binding; specifically, a new proof requires a new signature on a different message, which would violate the unforgeability of our signature scheme. □

Remark 2. Note that since the commitment serves as a public key and is independent of the message, updating the commitments for a new message can be achieved simply by signing the new message. Moreover, the message space for the commitment can be adapted by selecting a suitable SPS. Specifically, we can configure the SPVC to handle unilateral messages, where messages are drawn solely from either $\mathbb{G}_1$ or $\mathbb{G}_2$, or bilateral messages, which allow for a mix of elements from both $\mathbb{G}_1$ and $\mathbb{G}_2$.