

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

12-2021

Concise mercurial subvector commitments: Definitions and constructions

Yannan LI

University of Wollongong

Willy SUSILO

Guomin YANG

Singapore Management University, gmyang@smu.edu.sg

Tran Viet Xuan PHUONG

Yong YU

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research



Part of the [Information Security Commons](#)

Citation

LI, Yannan; SUSILO, Willy; YANG, Guomin; PHUONG, Tran Viet Xuan; YU, Yong; and LIU, Dongxi. Concise mercurial subvector commitments: Definitions and constructions. (2021). *Information Security and Privacy: 26th Australasian Conference, Virtual Conference, December 1-3: Proceedings*. 13083, 353-371. Available at: https://ink.library.smu.edu.sg/sis_research/7406






This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Author

Yannan LI, Willy SUSILO, Guomin YANG, Tran Viet Xuan PHUONG, Yong YU, and Dongxi LIU



Concise Mercurial Subvector Commitments: Definitions and Constructions

Yannan Li¹ , Willy Susilo¹ , Guomin Yang¹ ,
Tran Viet Xuan Phuong¹ , Yong Yu² , and Dongxi Liu³

¹ Institute of Cybersecurity and Cryptology, School of Computing and Information
Technology, University of Wollongong, Wollongong, NSW 2522, Australia

{yannan,wsusilo,gyang,txuan}@uow.edu.au

² School of Cyberspace Security, Xi'an University of Posts and Telecommunications,
Xi'an 710121, China

³ Data61, CSIRO, Sydney, Australia
Dongxi.Liu@data61.csiro.au

Abstract. Vector commitment and its variants have attracted a lot of attention recently as they have been exposed to a wide range of applications in blockchain. Two special extensions of vector commitments, namely subvector commitments and mercurial commitments, have been proposed with attractive features that are desirable in many applications. Nevertheless, to the best of our knowledge, a single construction satisfying all those attractive features is still missing. In this work, we analyze those important properties and propose a new primitive called mercurial subvector commitments, which are efficiently updatable, mercurial hiding, position binding, and aggregatable. We formalize the system model and security model for such a primitive and present a concrete construction with security proofs to show that it satisfies all of the properties. Moreover, we also illustrate some applications of mercurial subvector commitments, including zero-knowledge sets and blockchain with account-based models.

Keywords: Vector commitments · Blockchain · Aggregation · Zero-knowledge sets

1 Introduction

Vector commitments (VC) allow a user to commit to a set of ordered messages, which can be opened at a specific position. Normally, a vector commitment is updatable, position binding and concise. Specifically, a VC is *updatable*, which means it enabling a committer to efficiently update the committed message at some positions after the committing phase. *Position binding* is a basic requirement generalized from the binding property of normal commitments [24], meaning one cannot find two different and valid messages in a position in a vector

commitment in polynomial time. The *concise* property requires the size of the commitment and the opening proof are independent of the number of the committed message. Thanks to the above desirable properties, vector commitments contribute to many applications such as accumulators, cloud storage and so forth [4, 11, 12]. Subsequently, a series of follow-up works have been proposed, such as polynomial commitments [19] and functional commitments [21]. One of the most noticeable work is mercurial commitments (MC), which is a special kind of VC. An MC has two ways to commit, namely hard commitment and soft commitment. Hard commitments are the same as normal VC while soft commitments do not have the binding property to the committed messages. A committer needs to choose a preferred way of commitment at the beginning of a commitment phase. There are two options in the opening phase as well, namely the hard opening and soft opening. Hard opening is only for hard commitments and can generate a proof to open the committed message at a specific position. Meanwhile, soft opening is for both hard commitments and soft commitments. Soft opening for hard commitment at a position cannot be different from the committed value, whereas soft opening for soft commitment enables the committer to open to a message in a position at his/her choice. Besides, a mercurial commitment is *mercurial hiding*, meaning that hard commitments are indistinguishable from soft ones. The special property of MC is promising to enable membership and non-membership proofs in a set without revealing any information of the set including its cardinality, i.e. zero-knowledge set (ZKS).

Subvector commitments (SVC) are another important extension of VC, which were initially presented for supporting stateless cryptocurrencies in blockchain [13]. SVC allows a user to open a vector commitment at a set of positions at the same time. Compared to VC, SVC has a stronger requirement that the opening proof to a subset of position is independent of not only the size of the committed messages, but also the size of the chosen subset. With the wide applications in blockchain, many research works on SVC have been proposed [15, 15, 17].

Related work. The concept of VC was proposed by Catalano and Fiore [10]. In that seminal work, they also presented two concrete constructions based on CDH and RSA assumptions, respectively. A similar notion to VC is polynomial commitment (PC), proposed in [19], which enables a user to commit to a polynomial so that a verifier can later be convinced to a claimed evaluation at a point. The size of a polynomial commitment is independent of the degree of the polynomial, so is the opening proof at a point. Besides, it supports batch verification due to the basic polynomial quotient theory, in which the size of the opening proof for multiple evaluations is the same as that for a single point. Libert et al. [21] generalized the concept to a functional commitment (FC) that can open to a function of the committed messages. Specifically, the commitment can be opened to (f, y) such that $y = f(\mathbf{m})$, where \mathbf{m} is the committed messages. They provided a construction on linear function f based on a composite order groups, where $y = \sum_{i=1}^n x_i m_i$. Chepurnoy et al. [13] presented a new algebraic vector commitment scheme based on multilinear polynomial and applied the new construction to EDRAW, a stateless verification for account-based cryptocurrencies

[2], where the miners do not necessarily store the current state, but validate the transactions by accessing the latest block to check the balance in the relevant accounts.

Mercurial commitments (MC), proposed by Chase et al. [11], are a special kind of vector commitment supporting hard and soft commitments as outlined above. Catalano et al. [9] presented a more efficient construction on trapdoor mercurial commitments (TMC) based on one-way function with weaker assumption. In their construction, the size of the soft opening is much shorter, while the size of the hard opening is still linear with the number of messages. Libert and Yung [22] proposed the concept of mercurial vector commitment (MVC) and devised a construction based on broadcast encryption [5] with compact proofs for both hard opening and soft opening.

The primitive of subvector commitment (SVC) was proposed by Lai and Malavolta [20]. They presented two concrete constructions [20] under variants of the root assumption and the CDH assumption based on [10]. The construction supports batch proof generation, meaning that it can generate proofs for multiple positions at one-time. However, the proofs can not be aggregated after generation. Boneh et al. [4] proposed an accumulator with batch verification, which can be used to design a VC with aggregatable proofs for both membership and non-membership. The primitive of aggregatable subvector commitment (aSVC) was proposed [15] to enable aggregation of multiple opening proofs into a single SVC proof and hence reduce the verification overhead. Campanelli et al. [8] proposed an incrementally aSVC (iaSVC), which can aggregate the opening for an unbounded number of times to further improve the efficiency. It also ensures fast generation of the opening by leveraging preprocessing. Gorbunov et al. [17] proposed Pointproofs (PP), which can aggregate proofs generated by multiple commitments by any entity non-interactively. Agrawal et al. [1] proposed a new VC named key-value commitment (KVC), whose committed messages are key-value maps. The setting can generally link to the blockchain-based cryptocurrency, where the key is the account address and the value is the account balance.

We summarize all the existing works on VC and its variants mentioned above based on the properties they provide. The result is shown in Table 1. We can see from the table that none of the works mentioned above provides all the promising features. In this paper, we fill this gap by presenting a vector commitment enjoying the nice features including efficient update, aggregation, mercurial properties and privacy.

Contributions. In this paper, we propose a new primitive named mercurial subvector commitments, which enjoy the desirable features for both mercurial commitments and subvector commitments. To be more specific, the contributions of this paper are three-fold as follows.

- We put forward a new primitive of mercurial subvector commitments (MSVC), which support two ways to commit and open the messages and also enjoy efficient update and aggregation in the opening proofs.

Table 1. Summary of the existing vector commitments

Schemes	Updatable	Aggregatable	Mercurial	Hiding
VC [10]	✓	×	×	×
PC [19]	×	×	×	✓
FC [21]	×	×	×	✓
EDRAX [13]	✓	×	×	×
MC [11]	×	×	✓	✓
TMC [9]	×	×	✓	✓
MVC [22]	×	×	✓	✓
SVC [4]	✓	✓	×	✓
aSVC [15]	✓	✓	×	×
iaSVC [8]	✓	✓	×	×
PP [17]	✓	✓	×	✓
KVC [1]	✓	✓	×	×
Our scheme	✓	✓	✓	✓

- We formalize the system model and security models of MSVC and propose a concrete construction. We prove the security of the proposal.
- We present theoretical analysis of the proposal to show its practicality. We also show the possible applications of MSVC in blockchain-based cryptocurrencies with account-based model.

Organization. The rest of the paper is organized as follows. We provide some preliminaries used through the paper in Sect. 2. The system model and security model are illustrated in Sect. 3. We present a concrete construction in Sect. 4. Potential applications are shown in Sect. 5. We conclude the paper in Sect. 6.

2 Preliminaries

In this section, we introduce preliminaries that used throughout this paper, including bilinear maps, weak bilinear Diffie-Hellman exponent assumption and algebraic group model.

2.1 Bilinear Groups

Let G_1, G_2, G_T be multiplication groups of large prime order p . g_1 and g_2 are the generators of groups G_1 and G_2 , respectively. ψ is a computable isomorphism from group G_2 to G_1 ¹, which means $\psi(g_2) = g_1$. A non-degenerate bilinear pairing $e : G_1 \times G_2 \rightarrow G_T$ is denoted, which satisfies $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, for random $a, b \in \mathbb{Z}_p$.

¹ This setting is only used in the security proof rather than the proposed scheme.

2.2 Diffie-Hellman Exponent Assumption

The Diffie-Hellman exponent (l -DHE) problem [7] defined in $\mathbb{G} = (G_1, G_2, G_T, p, e)$ is as follows. On input

$$\left(\left(g_1^\alpha, g_1^{\alpha^2}, \dots, g_1^{\alpha^l} \right), \left(g_1^{\alpha^{l+2}}, \dots, g_1^{\alpha^{2l}} \right), \left(g_2^\alpha, g_2^{\alpha^2}, \dots, g_2^{\alpha^l} \right) \right)$$

for a random $\alpha \in Z_p$, output $g_1^{\alpha^{l+1}}$. The l -Diffie-Hellman exponent assumption in \mathbb{G} says that no efficient algorithm can solve the aforementioned problem in \mathbb{G} with non-negligible probability.

Following [7, 22], the problem is not easier than bilinear Diffie-Hellman exponent (BDHE) problem defined in [3, 5], in which with the same input and an additional $h \in G$, it outputs $e(g, h)^{l+1}$.

2.3 Algebraic Group Model

The algebraic group model (AGM) [16] is a model lying in between the standard model and generic group model (GGM). It is proposed to overcome the limitation of GGM that GGM does not cover group-specific algorithms to use the representation of a group. GGM model proves security in reduction, which is same as the standard model. In the AGM model, algebraic adversaries are considered, that is allowed to compute the elements in the target group and can use the representation in binary. To be more specific, suppose $\mathbf{L} = (\mathbf{L}_0, \dots, \mathbf{L}_m) \in G$ be a list of group elements in group G . An algebraic adversary can output a vector $\vec{z} = (z_1, \dots, z_m) \in Z_p$ such that $\mathbf{Z} = \prod_i \mathbf{L}_i^{z_i}$.

3 System Model and Security Model

A mercurial subvector commitment comprises the following algorithms.

Setup(λ, N) \rightarrow (*param*). This is a probabilistic algorithm run by a trusted party. On input a security parameter λ and the length of the messages N , it generates the public parameter *param*.

HCommit($\mathbf{m}, \textit{param}$) \rightarrow (C, \textit{aux}). This is a probabilistic algorithm run by the committer. On input a group of messages \mathbf{m} and the public parameter *param*, it generates a hard commitment C and some auxiliary information *aux*.

HProve($i, \mathbf{m}[-i], \textit{aux}$) \rightarrow (π_i). This is a deterministic algorithm run by the committer. On input the position i , message $\mathbf{m}[-i]^2$ and the auxiliary information *aux*, it outputs a proof π_i to prove that m_i is committed in the hard commitment.

HVerify(C, i, m_i, π_i) \rightarrow (0/1). This is a deterministic algorithm run by the verifiers. On input the commitment C , the position i and message m_i and proof π_i , it outputs 0 or 1 to indicate whether π_i is a valid proof.

² This is the message group without the i -th message.

$\text{HUpdate}(C, S, \mathbf{m}[S], \mathbf{m}'[S], aux) \rightarrow (C', aux')$. This is a deterministic algorithm run by the committer. On input the commitment C , the set of positions S , the original messages $\mathbf{m}[S]$, the updated messages $\mathbf{m}[S']$ and the auxiliary information aux , it outputs the new hard commitment C' and corresponding auxiliary information aux' .

$\text{SCommit}(param) \rightarrow (C, aux)$. This is a probabilistic algorithm run by the committer. On input the public parameter $param$, it generates a soft commitment C , which is not bound to any specific messages, and auxiliary information aux .

$\text{SProve}(i, m_i, \mathbb{F}, aux) \rightarrow (\pi_i)$. On input the position i and message m_i , it outputs a proof π_i to m_i at position i in the commitment. $\mathbb{F} \in \{\mathbb{H}, \mathbb{S}\}$ states the auxiliary information aux corresponds to a hard commitment or a soft commitment. If $\mathbb{F} = \mathbb{H}$ and m_i is not the originally committed value, the algorithm aborts and outputs \perp .

$\text{SVerify}(C, i, m_i, \pi_i) \rightarrow (0/1)$. This is a deterministic algorithm run by verifiers. On input the commitment C , the position i , the message m_i and proof π_i , it outputs 0 or 1 to indicate whether π_i is a valid proof.

$\text{Aggregate}(\mathbb{F}, C, S, \mathbf{m}[S], \{\pi_i : i \in S\}) \rightarrow (\hat{H})$. This is a probabilistic algorithm run by the committer. On input a flag \mathbb{F} to indicate whether this is an aggregation for soft commitment or hard commitment, the commitment C , the position set S , the messages $\mathbf{m}[S]$ and the proofs $\{\pi_i, i \in S\}$, it outputs a proof \hat{H} as an aggregated proof.

$\text{AggreVerify}(\mathbb{F}, C, S, \mathbf{m}[S], \hat{H}) \rightarrow (0/1)$. This is a deterministic algorithm run by verifier. On input the flag \mathbb{F} to indicate this is the verification for soft aggregation or hard aggregation, the commitment C , the position set S , the messages $\mathbf{m}[S]$ and the proof \hat{H} , it outputs 0 or 1 to indicate whether \hat{H} is a valid aggregated proof.

Correctness. The correctness of an MSVC applies in several cases. Specifically, for all λ, N , an ordered group of messages \mathbf{m} and a set $S \in [N]$, $(param) \leftarrow \text{Setup}(\lambda, N)$, the following conditions must hold with an overwhelming probability.

- For a hard commitment $(C, aux) \leftarrow \text{HCommit}(\mathbf{m}, param)$, a hard opening $(\pi_i[\mathbb{H}]) \leftarrow \text{HProve}(i, \mathbf{m}[-i], aux)$ and a soft opening for hard commitment $(\pi_i[\mathbb{S}]) \leftarrow \text{SProve}(m_i, i, \mathbb{H}, aux)$, we have

$$\text{HVerify}(C, i, m_i, \pi_i[\mathbb{H}]) = 1, \quad \text{SVerify}(C, i, m_i, \pi_i[\mathbb{S}]) = 1.$$

- For a soft commitment $(C, aux) \leftarrow \text{HCommit}(param)$, a soft opening for soft commitment $(\pi_i) \leftarrow \text{SProve}(m_i, i, \mathbb{S}, aux)$, we have

$$\text{SVerify}(C, i, m_i, \pi_i) = 1.$$

- For an aggregate proof $(\hat{H}) \leftarrow \text{Aggregate}(\mathbb{F}, C, S, \mathbf{m}[S], \{\pi_i : i \in S\})$, we have

$$\text{AggreVerify}(\mathbb{F}, C, S, \mathbf{m}[S], \hat{H}) = 1,$$

where $C = \text{HCommit}(\mathbf{m}, param)$, if $\mathbb{F} = \mathbb{H}$ and $C = \text{SCommit}(param)$, if $\mathbb{F} = \mathbb{S}$.

- For an updated messages \mathbf{m}' , such that $\mathbf{m}[N \setminus S] = \mathbf{m}'[N \setminus S]$, we have

$$\text{HUpdate}(C, S, \mathbf{m}[S], \mathbf{m}'[S]) = \text{HCommit}(\mathbf{m}', param),$$

where $C = \text{HCommit}(\mathbf{m}, param)$.

An MSVC needs to satisfy Mercurial binding and Mercurial hiding, which are defined as follows.

Mercurial Binding [22]: Compared to normal commitments, the mercurial binding applies for both hard commitments and soft commitments to the messages for some positions. For hard commitments, no adversary can generate an MSVC C such that it can be opened into two different messages in a specific position. For soft commitments, no adversary can generate an MSVC C such that it can be opened to a single value but partially decommitted (teased) to another value.

Specifically, given the system parameters $param$, it is computationally infeasible to output a commitment C and the pairs $(\mathbf{m}^0[S^0], \pi^0)$ and $(\mathbf{m}^1[S^1], \pi^1)$ that satisfy the following equations, where S_0 and S_1 denote the subsets of $[1, N]$.

$$\begin{aligned} \text{AggrVerify}(\mathbb{H}, C, S^0, \mathbf{m}^0[S^0], \Pi^0) &= 1, \text{AggrVerify}(\mathbb{H}, C, S^1, \mathbf{m}^1[S^1], \Pi^1) = 1, \\ \text{AggrVerify}(\mathbb{H}, C, S^0, \mathbf{m}^0[S^0], \Pi^0) &= 1, \text{AggrVerify}(\mathbb{S}, C, S^1, \mathbf{m}^1[S^1], \Pi^1) = 1, \\ \mathbf{m}^0[S^0 \cap S^1] &\neq \mathbf{m}^1[S^0 \cap S^1] \end{aligned}$$

If the size of the set is one, it implies the following equation holds with negligible probability.

$$\begin{aligned} \text{HVerify}(C, i, m_i, \pi_i) &= 1, \text{HVerify}(C, i, m'_i, \pi'_i) = 1, m_i \neq m'_i \\ \text{HVerify}(C, i, m_i, \pi_i) &= 1, \text{SVerify}(C, i, m'_i, \pi'_i) = 1, m_i \neq m'_i \end{aligned}$$

Mercurial Hiding [9]: We now define the mercurial hiding, which has the following requirements. (1) No probabilistic polynomial time (PPT) adversary can learn whether C is a soft commitment or hard commitment. (2) For hard commitments, no PPT adversary can learn the committed values \mathbf{m} ; (3) For a soft commitment, it cannot be teased to any value before partially de-committed.

We define the simulation-based statistical security to depict the mercurial hiding property above, in which there exists a simulator executing the algorithms $\text{Setup}^*(\lambda, N)$, $\text{Commit}^*(param, tk)$, $\text{HProve}^*(i, \mathbf{m}, aux)$ and $\text{SProve}^*(i, m_i, aux)$ defined as follows.

- $\text{Setup}^*(\lambda, N) \rightarrow (param, tk)$. On input a security parameter λ and the size of a group N , it outputs the system parameter $param$ and a trapdoor key tk .
- $\text{Commit}^*(param, tk) \rightarrow (C, aux)$. This is a randomized algorithm that takes as input the system parameter $param$ and a trapdoor tk . It outputs a fake commitment C and auxiliary information aux . However, C doesn't bind to any group of messages.

- $\text{HProve}^*(i, \mathbf{m}, aux) \rightarrow (\pi_i)$. This is the equivocal hard opening (hard equivocation) algorithm. Given (C, aux) generated by $\text{Commit}^*(param, tk)$, it outputs a fake proof of hard decommitment π_i on position i for C .
- $\text{SProve}^*(i, m_i, aux) \rightarrow (\pi_i)$. This is the equivocal soft opening (soft equivocation) algorithm. Given (C, aux) generated by $\text{Commit}^*(param, tk)$, it outputs a fake proof of soft decommitment π_i on position i for C .

The basic idea of the simulation-based game is that the commitments and the proofs generated by the real protocol and the simulator are statistically indistinguishable, even given the commitments, in which the committed messages are chosen by the adversary. From the algorithms listed above, we can tell that there is no message in a fake commitment and the proofs only involve the message to be committed, thus the fake commitment and the fake proofs do not leak any information of the other committed values. The formal process of the security game is defined as follows.

Equivocation Game. The simulator \mathcal{C} executes $\text{Setup}^*(\lambda, N)$ to get the security parameters $param$ and a trapdoor tk . In the game, \mathcal{A} interacts with \mathcal{C} to distinguish whether it is the real-world setting or the ideal one. To be more specific, the setting of the game is determined by \mathcal{C} by flipping a coin $b \in \{0, 1\}$. If $b = 0$, the game is with the real commitment and opening proof. Otherwise, it is with a fake commitment. \mathcal{A} is allowed to make queries to \mathcal{C} . At the end of the game, \mathcal{A} needs to guess the bit b . The detailed setting is shown as follows.

- **HHEquivocation.** \mathcal{A} chooses a message tuple (m_1, \dots, m_N) . \mathcal{C} flips a coin b . If $b = 0$, \mathcal{C} computes $(C, aux) \leftarrow \text{HCommit}(\mathbf{m}, param)$ and if $b = 1$, $(C, aux) \leftarrow \text{Commit}^*(param, tk)$. Then \mathcal{A} is provided C by the challenger \mathcal{C} . \mathcal{A} is allowed to make queries on his choices of $S \in [N]$. If $b = 0$, \mathcal{C} returns $\pi_i \leftarrow \text{HProve}(i, \mathbf{m}[-i], aux)$. Otherwise if $b = 1$, \mathcal{C} replies with $\pi_i \leftarrow \text{HProve}^*(i, \mathbf{m}, aux)$.
- **HSEquivocation.** \mathcal{A} chooses a message tuple (m_1, \dots, m_N) . \mathcal{C} flips a coin b . If $b = 0$, \mathcal{C} computes $(C, aux) \leftarrow \text{HCommit}(\mathbf{m}, param)$ and if $b = 1$, $(C, aux) \leftarrow \text{Commit}^*(param, tk)$. Then \mathcal{A} is provided C by the challenger \mathcal{C} . \mathcal{A} is allowed to make queries on his choices of $S \in [N]$. If $b = 0$, \mathcal{C} returns $\pi_i \leftarrow \text{SProve}(i, m_i, \mathbb{H}, aux)$. Otherwise if $b = 1$, \mathcal{C} replies with $\pi_i \leftarrow \text{SProve}^*(i, m_i, aux)$.
- **SSEquivocation.** \mathcal{C} flips a coin b . If $b = 0$, \mathcal{C} computes $(C, aux) \leftarrow \text{SCommit}(param)$ and if $b = 1$, $(C, aux) \leftarrow \text{Commit}^*(param, tk)$. Then \mathcal{A} is provided C by the challenger \mathcal{C} . \mathcal{A} is allowed to make queries on his choices of $S \in [N]$. If $b = 0$, \mathcal{C} returns $\pi_i \leftarrow \text{SProve}(i, m_i, \mathbb{S}, aux)$. Otherwise if $b = 1$, \mathcal{C} replies with $\pi_i \leftarrow \text{SProve}^*(i, m_i, aux)$.

4 Proposed MSVC

In this section, we put forward the detailed construction of an MSVC. Then we show the correctness of the proposal. In the construction, we borrow the

idea from both mercurial commitments [22] and subvector commitments [17] to achieve all the desirable properties, which advances the performance of traditional applications in each field. For mercurial property, we follow the techniques in [22] to lose the binding in the verification. The involvement of C is the key point achieving soft commitment and opening. Privacy is fulfilled via the commitment forms similar to Pedersen commitments. For the aggregation, we include the idea in [17] to add a new scalar for each individual proof. The detailed construction is shown as follows.

Setup(λ, N) \rightarrow (*param*). On input a security parameter λ , it generates $\mathbb{G} = (p, G_1, G_2, G_T, e)$, where G_1, G_2, G_T are cyclic multiplicative groups of prime order p . g_1 and g_2 are the generators in the corresponding groups. H is a cryptographic hash function mapping from $\{0, 1\}^*$ to Z_p . $e : G_1 \times G_2 \rightarrow G_T$ denotes a bilinear mapping. Select a random $\alpha \leftarrow Z_p^*$ and compute $\mathbf{a} = (\alpha_1, \dots, \alpha_N)$, where $\alpha_i = \alpha^i$. Compute $g_1^{\mathbf{a}} = \{g_1^{\alpha_1}, \dots, g_1^{\alpha_N}\}$, $g_2^{\mathbf{a}} = \{g_2^{\alpha_1}, \dots, g_2^{\alpha_N}\}$ and $g_1^{\mathbf{a}[-1] \cdot \alpha^N} = \{g_1^{\alpha^{N+2}}, \dots, g_1^{\alpha^{2N}}\}$. The public parameters are generated as *param* = $(\mathbb{G}, H, g_1^{\mathbf{a}}, g_2^{\mathbf{a}}, g_1^{\mathbf{a}[-1] \cdot \alpha^N})$. α is discarded after initializing the system.

HCommit($\mathbf{m}, \textit{param}$) \rightarrow $((C, V), \textit{aux})$. On input a set of messages \mathbf{m} , it generates the commitment pair as follows.

$$V = g_1^\gamma \cdot \prod_{j=1}^N g_1^{m_j \alpha_j}, \quad C = g_1^\theta,$$

where γ and θ are randomness and m_j is the message of the j -th position in \mathbf{m} . The commitment is (C, V) and the auxiliary information is $(m_1, \dots, m_N, \gamma, \theta)$.

HProve($i, \mathbf{m}[-i], \textit{aux}$) \rightarrow (π_i) . On input the messages and the auxiliary information, generate the proof on position i as follows.

$$W_i = \left(g_2^\gamma \cdot \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j} \right)^{\alpha^{N+1-i}/\theta}$$

Finally, the proof is generated as $\pi_i = (W_i, \theta)$.

HVerify($((C, V), i, m_i, \pi_i)$) \rightarrow $(0/1)$. On input the commitment, the messages, check whether the following equation holds to validate an opening proof.

$$e\left(V, g_2^{\alpha^{N+1-i}}\right) = e(C, W_i) \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} \tag{1}$$

If it holds, it outputs 1 to indicate it is a valid proof. Otherwise it outputs 0.

HUpdate($C, V, S, \mathbf{m}[S], \mathbf{m}'[S], \textit{aux}$) \rightarrow $((C', V'), \textit{aux}')$. On input the commitment, parse the messages $\mathbf{m}[S]$ and $\mathbf{m}'[S]$, and compute the following equation to update the messages in position set S in the commitment.

$$V' = V \cdot \prod_{i \in S} g_1^{(m'_i - m_i) \alpha^i}, \quad C' = C$$

The updated commitment is (C', V') and the updated auxiliary information is $(\mathbf{m}', \gamma, \theta)$, where $\mathbf{m}[S]$ is replaced by $\mathbf{m}'[S]$ in position set S .

$SCommit(param) \rightarrow (C, aux)$. On input the security parameter $param$, it chooses random θ and γ , and generates the commitment pair as follows.

$$V = (g_1^\alpha)^\gamma, \quad C = (g_1^\alpha)^\theta$$

Output (C, V) as the soft commitment pair and the auxiliary information is $aux = (\theta, \gamma)$.

$SProve(i, m_i, \mathbb{F}, aux) \rightarrow (\pi_i)$. If $\mathbb{F} = \mathbb{H}$, parse the auxiliary information as $aux = (m_1, \dots, m_N)$. The algorithm outputs \perp if m_i is not the same message as that in aux . Otherwise, the algorithm computes

$$W_i = \left(g_2^\gamma \cdot \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j} \right)^{\alpha^{N+1-i}/\theta}.$$

If $\mathbb{F} = \mathbb{S}$, parse the auxiliary information and generate the proof as follows.

$$W_i = \left(g_2^{\alpha^{N-i}\gamma} g_2^{\alpha^N(-m_i)} \right)^{1/\theta}$$

Finally, the proof is generated as $\pi_i = W_i$. The auxiliary information is based on the flag bit.

$SVerify((C, V), i, m_i, \pi_i) \rightarrow (0/1)$. On input the message, the commitment and the proof, check if the Eq. (1) in $HVerify((C, V), i, m_i, \pi_i)$ holds. If it satisfies, it outputs 1. Otherwise, output 0.

$Aggregate(\mathbb{F}, (C, V), S, \mathbf{m}[S], \{\pi_i : i \in S\}) \rightarrow (\hat{H})$. On input the flag bit as \mathbb{H} or \mathbb{S} , aggregate the proofs in position set S , compute

$$\hat{W} = \prod_{i \in S} W_i^{t_i}$$

where $t_i = H(i, (C, V), S, m_i[S])$, W_i is either a proof for hard opening or soft opening based on \mathbb{F} . The aggregated proof is $\hat{\pi} = (\theta, \hat{W})$.

$AggrVerify(\mathbb{F}, (C, V), S, \mathbf{m}[S], \hat{H}) \rightarrow (0/1)$. To validate an aggregated proof with flag \mathbb{H} or \mathbb{S} , check the following equation.

$$e \left(V, g_2^{\sum_{i \in S} \alpha^{N+1-i} t_i} \right) = e \left(C, \hat{W} \right) \cdot e \left(g_1, g_2 \right)^{\alpha^{N+1} \sum_{i \in S} m_i t_i}$$

Output 1 to indicate this is a valid proof. Otherwise, output 0.

4.1 Correctness

We show the correctness of the proposal from the following aspects.

Correctness of Hard Opening. In hard commitments, given $V = g_1^\gamma \cdot \prod_{j=1}^N g_1^{m_j \alpha_j}$, $C = g_1^\theta$, and $W_i = \left(g_2^\gamma \cdot \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j} \right)^{\alpha^{N+1-i}/\theta}$, we have

$$\begin{aligned} \frac{e\left(V, g_2^{\alpha^{N+1-i}}\right)}{e(C, W_i)} &= \frac{e\left(g_1^\gamma \cdot \prod_{j=1}^N g_1^{m_j \alpha_j}, g_2^{\alpha^{N+1-i}}\right)}{e\left(g_1^\theta, \left(g_2^\gamma \cdot \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j}\right)^{\alpha^{N+1-i}/\theta}\right)} \\ &= \frac{e\left(g_1^\gamma, \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j}\right) \cdot e\left(g_1^\gamma, g_2^{\alpha^{N+1-i}}\right)}{e\left(g_1, g_2^{\gamma \alpha^{N+1-i}}\right) \cdot e\left(g_1, \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j}\right)} \\ &= e(g_1, g_2)^{\alpha^{N+1} m_i} \end{aligned}$$

Thus the correctness of hard opening holds.

Correctness of Soft Opening. In soft opening, given $V = g_1^\gamma$, $C = (g_1^\alpha)^\theta$ and $W_i = \left(g_2^{\alpha^{N+1-i} \gamma - \alpha^{N+1} m_i} \right)^{1/\theta}$, we have

$$\begin{aligned} e(C, W_i) \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} &= e\left((g_1^\alpha)^\theta, \left(g_2^{\alpha^{N+1-i} \gamma - \alpha^{N+1} m_i}\right)^{1/\theta}\right) \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} \\ &= e\left(g_1, g_2^{\alpha^{N+1-i} \gamma - \alpha^{N+1} m_i}\right) \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} \\ &= e\left(g_1, g_2^{\alpha^{N+1-i} \gamma}\right) \cdot e(g_1, g_2)^{-\alpha^{N+1} m_i} \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} \\ &= e\left(V, g_2^{\alpha^{N+1-i}}\right) \end{aligned}$$

Thus the correctness of soft opening holds.

Correctness of Aggregation. The aggregation works for both hard opening and soft opening witnesses. From the previous analysis, we have

$$e(C, W_i) \cdot e(g_1, g_2)^{\alpha^{N+1} m_i} = e\left(V, g_2^{\alpha^{N+1-i}}\right)$$

for both hard opening and soft opening. For each position i in a set S , the above equation holds. Multiplying these equations for $i \in S$, we will get

$$e\left(V, g_2^{\sum_{i \in S} \alpha^{N+1-i} t_i}\right) = e(C, \hat{W}) \cdot e(g_1, g_2)^{\alpha^{N+1} \sum_{i \in S} m_i t_i}.$$

Thus the correctness of soft opening holds.

Correctness of Update. For update of a commitment for some position set S , given $V = g_1^\gamma \cdot \prod_{i=1}^N g_1^{m_i \alpha_i}$ and from the definition of a commitment, we have

$$\begin{aligned} V \cdot \prod_{i \in S} g_1^{(m'_i - m_i) \alpha_i} &= g_1^\gamma \cdot \prod_{i=1}^N g_1^{m_i \alpha_i} \cdot \prod_{i \in S} g_1^{(m'_i - m_i) \alpha_i} \\ &= g_1^\gamma \cdot \prod_{i=1, i \notin S}^N g_1^{m_i \alpha_i} \cdot \prod_{i \in S} g_1^{(m'_i - m_i + m_i) \alpha_i} \\ &= g_1^\gamma \cdot \prod_{i=1, i \notin S}^N g_1^{m_i \alpha_i} \cdot \prod_{i \in S} g_1^{m'_i \alpha_i} = V' \end{aligned}$$

Thus the correctness of update holds.

4.2 Mercurial Binding

Our construction satisfies mercurial binding in the AFM and ROM model if l -wBDHE assumption holds.

Theorem 1. *If there is an adversary \mathcal{A} who breaks the mercurial binding of the proposed scheme, then we can construct another algorithm \mathcal{B} to solve l -wBDHE problem with overwhelming probability.*

Proof. The proof is conducted with a game between a challenger \mathcal{C} and an algebraic adversary \mathcal{A} .

Setup. \mathcal{C} sets up the system by generating $\mathbb{G} = (p, G_1, G_2, G_T, e)$. On input the instance $g_1^{\mathbf{a}} = \{g_1^{\alpha^1}, \dots, g_1^{\alpha^N}\}$, $g_2^{\mathbf{a}} = \{g_2^{\alpha^1}, \dots, g_2^{\alpha^N}\}$ and $g_1^{\mathbf{a}[-1] \cdot \alpha^N} = \{g_1^{\alpha^{N+2}}, \dots, g_1^{\alpha^{2N}}\}$ as the system parameters $param$, \mathcal{C} forwards the system parameters $param$ to \mathcal{A} . Since \mathcal{A} is algebraic, it can output \mathbf{z} and γ such that

$$V = g_1^\gamma g_1^{\mathbf{z}^\top \mathbf{a}}, C = g_1^\theta.$$

Hash query. \mathcal{A} is allowed to make q_H hash queries on its choices. \mathcal{C} maintains a hash table with entry $H(i, C, V, S, m[S])$, and chooses t_i uniformly at random as the response. Repeated queries will get the same response. We note that for $\mathbf{z}[S] \neq \mathbf{m}[S]$, we have

$$\Pr[\mathbf{z}[S] \not\equiv_p \mathbf{m}[S] \text{ and } \mathbf{z}[S]t \equiv_p \mathbf{m}[S]t] = 1/p.$$

We call this an H -lucky query. If this happens, the proofs aborts. The probability for \mathcal{A} to make an H -lucky query is at most q_H/p .

Output. \mathcal{A} outputs

$$(C, V), \{S^b, \mathbf{m}^b[S^b], \hat{W}^b\}_{b=0,1},$$

where $V = g_1^\gamma g_1^{z^\top a}$ and $C = g_1^\theta$.

Now we show how to work out $g_1^{\alpha^{N+1}}$. Since $\mathbf{m}^0[S^0 \cap S^1] \neq \mathbf{m}^1[S^0 \cap S^1]$, then we have either $\mathbf{m}^0[S^0] \neq z[S^0]$ or $\mathbf{m}^1[S^1] \neq z[S^1]$. Define $(S^*, \mathbf{m}^*, \hat{W}^*)$ such that

$$\mathbf{m}^*[S^*] \neq z[S^*] \text{ and } \text{AggrVerify}(\mathbb{F}, (C, V), S^*, \mathbf{m}^*[S^*], \hat{W}^*) = 1.$$

Thus we have

$$e(V, g_2^{\sum_{i \in S^*} \alpha^{N+1-i} t_i}) = e(C, \hat{W}^*) \cdot e(g_1, g_2)^{\alpha^{N+1} \mathbf{m}^*[S^*]^\top t}.$$

As we may recall,

$$e\left(V, g_2^{\sum_{i \in S^*} \alpha^{N+1-i} t_i}\right) = e(C, \hat{W}) \cdot e(g_1, g_2)^{\alpha^{N+1} z[S^*]^\top t}.$$

With these two equations, we have

$$e(C, \hat{W}^*) \cdot e(g_1, g_2)^{\alpha^{N+1} \mathbf{m}^*[S^*]^\top t} = e(C, \hat{W}) \cdot e(g_1, g_2)^{\alpha^{N+1} z[S^*]^\top t}.$$

We can obtain

$$g_2^{\alpha^{N+1} \cdot (z[S^*] - \mathbf{m}^*[S^*])^\top t} = (\hat{W}^* / \hat{W})^\theta.$$

Recall that $z[S^*] \neq \mathbf{m}^*[S^*]$ and that there is no H -lucky queries, thus, $z[S^*] - \mathbf{m}^*[S^*]^\top t \not\equiv_p 0$. We can easily get its inverse modulo p and get

$$g_2^{\alpha^{N+1}} = \left[(\hat{W}^* / \hat{W})^\theta \right]^{(z[S^*] - \mathbf{m}^*[S^*])^\top t}.$$

With $g_1 = \psi(g_2)$ and the above equation, we can easily work out $g_1^{\alpha^{N+1}}$.

4.3 Mercurial Hiding

It is shown in [9] that mercurial hiding is implied in the equivocation games. Thus in this subsection, we prove the security of our proposal under HH Equivocation, HS Equivocation and SS Equivocation.

Theorem 2. *Our construction satisfies HH Equivocation, HS Equivocation and SS Equivocation.*

Proof. \mathcal{C} setups the system and obtains the system parameter $param = (\mathbb{G}, H, g_1^a, g_2^a, g_1^{a[-1] \cdot \alpha^N})$ as in the real setup algorithm. α is set as the trapdoor tk .

\mathcal{C} flips a coin $b \in \{0, 1\}$. If $b = 1$, \mathcal{C} calls $\text{Commit}^*(param, tk)$ to generate a fake commitment as $(C, V) = (g_1^\theta, g_1^\gamma)$. To answer a query for the decommitment

proof on position i , \mathcal{C} runs $\text{HProve}^*(i, \mathbf{m}, aux)$ and gets the corresponding hard equivocation on position i to m_i as

$$\pi_i = \left(\theta, W_i = \left(g_2^{\alpha^{N+1-i}\gamma} g_2^{-\alpha^{N+1}m_i} \right)^{1/\theta} \right).$$

If $b = 0$, for a group of messages \mathbf{m} , \mathcal{C} calls $\text{HCommit}(\mathbf{m}, param)$ to get the commitment as

$$(C, V) = \left(g_1^\theta, g_1^{\tilde{\gamma}} \cdot \prod_{j=1}^N g_1^{m_j \alpha_j} \right)$$

for some randomly chosen $\tilde{\gamma}$. Then to answer the query by \mathcal{A} , \mathcal{C} generates the corresponding hard opening as

$$\tilde{\pi}_i = \left(\theta, \tilde{W}_i = \left(g_2^{\tilde{\gamma}} \cdot \prod_{j=1, j \neq i}^N g_2^{m_j \alpha_j} \right)^{\alpha^{N+1-i}/\theta} \right).$$

It is easy to find that the fake commitment and hard equivocations have the same distribution as the hard commitments and the hard openings for any random $param, i, \mathbf{m}$.

The HSEquivocation follows the same arguments as above.

For SSEquivocation, if $b = 1$, \mathcal{C} calls $\text{Commit}^*(param, tk)$ to generate a fake commitment as $(C, V) = (g^\theta, g^\gamma)$. If $b = 0$, the soft commitment is set as $(C, V) = (g^{\theta'}, g^{\gamma'})$ for some random θ' and γ' . It is easy to tell the soft commitment and the fake commitment has the same distribution since $\theta' = \theta/\alpha$ and $\gamma' = \gamma/\alpha$. The equivocal soft opening

$$W_i = \left(g_2^{\alpha^{N+1-i}\gamma} g_2^{\alpha^{N+1}(-m_i)} \right)^{1/\theta}$$

also has the same distribution since it can be written as

$$W_i = \left(g_2^{\alpha^{N-i}\gamma'} g_2^{\alpha^N(-m_i)} \right)^{1/\theta'}.$$

With the theorem shown above and the claim in [9] (Sect. 2.3), our construction satisfies mercurial hiding.

4.4 Performance Analysis

In this subsection, we present the theoretical analysis of the algorithms. The detailed result is shown in Table 2. n is the number of the maximum messages in the commitments and l is the size of the subset S . We only count the expensive group operations and ignore the cheap ones such as hash and the operation in Z_p ; *mult* is the multiplication and *exp* is short for the exponentiation. The footnote of the operations represents those in the corresponding groups. The

Table 2. Theoretical analysis of the proposal

Algorithm	HCommit	HProve	HVerify
Time	$(n+2)exp_1+n \text{ mult}_1$	$(n+1)exp_2+(n-1)mult_2$	$2 \text{ pair}+1mult_T+1exp_T$
size	$2 G_1 $	$1 G_2 $	–
Algorithm	SCommit	SProve (\mathbb{F})	SVerify
Time	2 exp_1	$3 \text{ exp}_2+1 \text{ mult}_2$	$2 \text{ pair}+1mult_T+1exp_T$
size	$2 G_1 $	$1 G_2 $	–
Algorithm	HUpdate	Aggregate	AggrVerify
Time	$lexp_1+lmult_1$	$(l-1) \text{ mult}_1$	$2 \text{ pair}+1mult_T+1exp_T+1exp_2$
Size	$2 G_1 $	$1 G_2 $	–

cost of generating a hard commitment is linear with the size of the messages. However, this is a one-time phase. After generating the commitment, one can update the messages with HUpdate, which is only linear with the number of the updated positions. In all the verification algorithm, $e(g_1, g_2)^{\alpha^{N+1}}$ is known in the *param*, thus this part does not require any pairing operations. For the storage size, we can observe that the commitments are only two elements and the proofs for whichever kind of opening are only one element in the corresponding groups.

5 Applications of Mercurial Subvector Commitments

In this section, we show applications of the proposed mercurial subvector commitments.

5.1 Zero-Knowledge Elementary Database with Batch Verification

Zero-knowledge sets, proposed by Micali et al. [25], enable an entity to commit to a set S confidentially and later prove whether a random element x is in the set or not without leaking any information of the set. Zero-knowledge elementary database (ZK-EDB) is a follow-up work where data are stored in the form of key-value pairs. If the queried key x is in the database, the value v corresponding to x is responded, where $v = D(x)$. Otherwise, \perp is responded. It is shown that the commitments with mercurial properties can be leveraged to build ZKS and ZK-EDB [11, 22]. In this section, following the framework in [11, 22], we show the proposed MSVC can be used to construct ZK-EDB, so as ZKS.

Normally, there are two phases in the process, the committing phase and proving phase. In the *committing phase*, an N -ray commitment tree is built in the following way. The leaf nodes of the tree are the values in the queried set with the keys as the indices and the root of this tree is the commitment for the set. In order to reduce the size and enhance the efficiency, we can do as follows. Firstly, the subtree is pruned if the keys in all the leaf nodes are not in the database.

After that, only the subtrees with at least one leaf node in the database is kept. To build the tree, for a leaf node, if the embedded $D(x) \neq \perp$, it contains a hard commitment of the hash of $D(x)$ as the message. Otherwise, if x is not in the database, it contains a soft commitment of empty message. The remaining nodes of the tree have commitments to the hash of all the children nodes. The commitment in the root node is the final commitment to the set. In the *proving phase*, to prove x is in the database with $D(x) = v$, the prover generates a proof of the hard opening, from the specific position where $D(x)$ is embedded, to the root. At each level in the tree, the proof for the commitment is with respect to the position in the commitment. While for a key x that is not in the database, the prover firstly patches up the missing subtree (which is pruned before) and generates a tease opening for the soft commitments.

The advantage to using the proposed MSVC in ZK-EDB is that it guarantees privacy and leaks no information of the database even the size of the set, at the same time, it enhances the efficiency due to the aggregation of the opening, which supports batch verification for the opening proofs of the commitments in the structure.

5.2 Mercurial Subvector Commitments in Blockchain

Blockchain [27] is a public ledger that records transactions in the system. In blockchain-based cryptocurrencies, there are two models in general, namely UTXO (unspent transaction output) model by Bitcoin [27] and account-based model by ethereum [14, 29]. In UTXO model, a list of all the unspent accounts are maintained. A transaction includes input accounts and output accounts, and a confirmed transaction results in the input accounts removed from the UTXO list and output accounts added to the UTXO list with some specific amount. Regarding privacy, e.g. hiding the amount in the account, Miers et al. [26] and Sasson et al. [28] presented frameworks and solutions to well-address the issue in UTXO model. In comparison, the account-based model is similar to the financial system in the real world. When a user spends some money in the system, the corresponding amount of coins is deducted from the balance and when receiving money, the amount is added to the balance of the account. However, in the account-based model, it remains an open problem to build either stateful verification or stateless verification [13]. Several existing works considered this problem. Guan et al. [18] proposed a privacy-preserving account-based blockchain based on heavy zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) [6]. Ma et al. [23] enhanced the privacy of the transactions in account-based model based on non-interactive zero-knowledge proof and homomorphic encryption. In [17], Gorbunov et al. briefly mentioned their construction is promising to be extended to enjoy hiding property, but there are no formal construction or security proofs. In this section, we introduce a model to apply Mercurial subvector commitments in Blockchain. We provide two frameworks, for public blockchains and consortium blockchains, respectively, which are shown as follows. Compared to the existing protocols [13, 17], we protect the privacy of the accounts in the commitments and the proofs.

In public blockchain, we consider the account-based model cryptocurrencies, such as Ethereum [29]. Without losing generality, let's assume that each user in this network has multiple accounts, and the user can generate a vector commitment (C, aux) by calling $\text{HCommit}(\mathbf{m}, param)$ and publish the commitment on blockchain. In the commitments, the messages \mathbf{m} are the balance of the accounts and the account addresses can be the indices in the commitments. To publish transaction T , the user computes a proof $\pi_i = \text{HProve}(i, \mathbf{m}[-i], aux)$, proving the spending account is one of the committed messages in the commitment. If more accounts are involved in a transaction, the user can aggregate the individual proofs $\{\pi_i\}(i \in S)$ into a single one by calling $\text{Aggregate}(\mathbb{H}, (C, V), S, \mathbf{m}[S], \{\pi_i : i \in S\})$ for efficient verification, where S is the involved accounts. The (aggregated) proof is included as part of the transaction, which can be validated with $\text{HVerify}(C, i, m_i, \pi_i)$ or $\text{AggreVerify}(\mathbb{F}, C, S, \mathbf{m}[S], \hat{T})$. Upon the transaction being proved and logged on the blockchain, the user updates the balance in the corresponding positions in the commitment C with $\text{HUpdate}(C, V, S, \mathbf{m}[S], \mathbf{m}'[S], aux)$. For consortium blockchain, we assume there are many organizations forming a consortium blockchain. For example, the blockchain is for a financial union, which is composed of many banks. Each bank has its own users and possesses their accounts. The bank sets the largest number of users and generates a commitment with the current users' accounts. For the vacant positions, the messages are set to be zero on the exponent. When a new user is registered, the corresponding position is updated with the information of the new user. There is a block generator that manages the system and produces new blocks by validating the transaction and logging valid blocks on the chain. In this case, when a transaction is conducted, the bank coordinator generates a transaction and computes a proof on behalf of the user. The block generator can validate the transaction and produce the block.

The advantage to using the proposed MSVC in blockchain with the account-based model is that it not only reduces the space size to store a large number of accounts, but also it guarantees the privacy of the accounts. The commitments leak no information of the accounts and the proofs only involve the related accounts and get zero knowledge of the rest of the accounts for external observers.

6 Conclusion

Vector commitments are effective tools to reduce the storage space and to enhance the efficiency, and hence, they have a great many potential applications. In this paper, we proposed a new primitive of MSVC with a concrete construction based on [22] with all the desired properties a vector commitment is supposed to enjoy. We formalize the system model and security model and prove the security under the proposed model. We provided possible applications with MSVC in ZKS and stateless blockchain. The future work includes generating vector commitments with hiding properties and aggregation across commitments.

Acknowledgement. Y. Li is partially supported by the UOW RevITALise grant (RITA). This work is also supported in part by the National Natural Science Foundation of China (61872229, U19B2021), the Blockchain Core Technology Strategic Research Program of the Ministry of Education of China (2020KJ010301), Key Research and Development Program of Shaanxi (2020ZDLGY09-06, 2021ZDLGY06-04).

References

1. Agrawal, S., Raghuraman, S.: KVAC: key-value commitments for blockchains and beyond. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 839–869. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_28
2. Marcella, A.: Blockchain technology and decentralized governance: Is the state still necessary? Available at SSRN 2709713 (2015)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_26
4. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 561–586. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_20
5. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_16
6. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 64–77. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_5
7. 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. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_27
8. 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. LNCS, vol. 12492, pp. 3–35. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_1
9. Catalano, D., Dodis, Y., Visconti, I.: Mercurial commitments: minimal assumptions and efficient constructions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 120–144. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_7
10. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_5
11. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. *J. Cryptol.* **26**(2), 251–279 (2013)
12. Chen, X., Li, J., Huang, X., Ma, J., Lou, W.: New publicly verifiable databases with efficient updates. *IEEE Trans. Dependable Secur. Comput.* **12**(5), 546–556 (2014)

13. Chepurnoy, A., Papamanthou, C., Zhang, Y.: Edrax: a cryptocurrency with stateless transaction validation. *IACR Cryptol. ePrint Arch.* **2018**, 968 (2018)
14. Dannen, C.: *Introducing Ethereum and Solidity*, vol. 1. Springer, Heidelberg (2017)
15. 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 2020. LNCS*, vol. 12238, pp. 45–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57990-6_3
16. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) *CRYPTO 2018. LNCS*, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
17. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: aggregating proofs for multiple vector commitments. *IACR Cryptol. ePrint Arch.* **2020**, 419 (2020)
18. Guan, Z., Wan, Z., Yang, Y., Zhou, Y., Huang, B.: Blockmaze: an efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Trans. Dependable Secur. Comput.* (2020). <https://doi.org/10.1109/TDSC.2020.3025129>. <https://ieeexplore.ieee.org/abstract/document/9200775>
19. 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. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
20. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) *CRYPTO 2019. LNCS*, vol. 11692, pp. 530–560. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_19
21. Libert, B., Ramanna, S., Yung, M.: Functional commitment schemes: from polynomial commitments to pairing-based accumulators from simple assumptions. In: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)* (2016)
22. 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. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_30
23. Ma, S., Deng, Y., He, D., Zhang, J., Xie, X.: An efficient nizk scheme for privacy-preserving transactions over account-model blockchain. *IEEE Trans. Dependable Secur. Comput.* **18**(2), 641–651 (2020)
24. Metere, R., Dong, C.: Automated cryptographic analysis of the pedersen commitment scheme. In: Rak, J., Bay, J., Kottenko, I., Popyack, L., Skormin, V., Szczypiorski, K. (eds.) *MMM-ACNS 2017. LNCS*, vol. 10446, pp. 275–287. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65127-9_22
25. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 80–91. IEEE (2003)
26. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: *2013 IEEE Symposium on Security and Privacy*, pp. 397–411. IEEE (2013)
27. Nakamoto, S.: *Bitcoin: A peer-to-peer electronic cash system*. Technical report, Manubot (2019)
28. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: *2014 IEEE Symposium on Security and Privacy*, pp. 459–474. IEEE (2014)
29. Wood, G.: *Ethereum: a secure decentralised generalised transaction ledger*. *Ethereum Proj. Yellow Pap.* **151**(2014), 1–32 (2014)