Efficient Proactive Secret Sharing for Large Data via Concise Vector Commitments

Matthias Geihs(✉), Lucas Schabhüser, and Johannes Buchmann

Technische Universität Darmstadt, Darmstadt, Germany
mgeihs@cdc.tu-darmstadt.de

Abstract. Proactive secret sharing has been proposed by Herzberg, Jarecki, Krawczyk, and Yung (CRYPTO'95) and is a powerful tool for storing highly confidential data. However, their scheme is not designed for storing large data and communication and computation costs scale linearly with the data size. In this paper we propose a variant of their scheme that uses concise vector commitments. We show that our new scheme, when instantiated with a variant of the Pedersen commitment scheme (CRYPTO'92), reduces computation costs by up to 50% and broadcast communication costs by a factor of L, where L is the length of the commitment message vectors.

1 Introduction

Threshold secret sharing has been proposed independently by Blakley [5] and Shamir [18] and allows to store a piece of secret information x at a set of N shareholders such that any coalition of up to T shareholders obtains no information about the secret. Proactive secret sharing was later proposed by Herzberg, Jarecki, Krawczyk, and Yung [12] and additionally allows the shareholders to update their data shares such that after the update the new shares are independent of the old shares. This property ensures protection against a mobile adversary that gradually obtains data shares over time. Moreover, proactive secret sharing is robust against up to $T < \frac{N}{2}$ malicious shareholders which means that the data owner is guaranteed to retrieve the initially stored data even if up to T shareholders behave arbitrarily bad.

While proactive secret sharing is a powerful tool for storage of highly confidential data, the performance of existing schemes appears insufficient for large data items. For example, storing a data item of size 128 kB at $N = 3$ shareholders and using a threshold of $T = 1$, the scheme described in [12] requires the data owner to broadcast 2 MB of data and to compute more than $16 * 10^3$ modular exponentiations. Moreover, updating the data shares requires 15 MB of data broadcast and each shareholder must compute more than $40 * 10^3$ modular exponentiations. However, securely storing highly confidential data such as legal documents or medical records over long periods of time requires proactive secret

C. Carlet et al. (Eds.): C2SI 2019, LNCS 11445, pp. 171–194, 2019.
https://doi.org/10.1007/978-3-030-16458-4_11

sharing schemes that are capable of efficiently storing data of size several mega bytes or even giga bytes.

In this paper we present a proactive secret sharing scheme that requires significantly less computational resources and immensely less communication than the scheme described in [12] when used for data sizes D that do not fit into the native message space of that scheme, e.g., $D > 32$ B. Our scheme can be instantiated such that in the setting described above, data storage requires the data owner to broadcast only about 16 kB of data and compute only about $8 * 10^3$ modular exponentiations. Similarly, updating the shares requires only 120 kB of data broadcast and each shareholder must compute only about $20 * 10^3$ modular exponentiations.

We achieve these performance improvements by combining the techniques of [12] with concise vector commitments [10]. While [12] uses cryptographic commitments where a commitment is of the same size as the committed message, concise vector commitments allow for committing to a vector of messages with a commitment that is much smaller than the committed message vector. By using such vector commitment we are able to reduce the broadcast communication costs by a factor of L, where L is the length of the message vectors. Furthermore, we also save up 50% of the computation costs because computing L single commitments requires $2 * L$ modular exponentiations while computing a vector commitment for message vectors of length L requires only $L + 1$ modular exponentiations. We remark that we use the same network model assumptions as [12], i.e., we assume a synchronous authenticated network with broadcast.

1.1 Organization

Our paper is organized as follows. In Sect. 2 we introduce notation and define the notions of a vector commitment scheme and a proactive secret sharing scheme as we will use them in this paper. In contrast to [12] we give a more precise definition of a proactive secret sharing scheme and respective security properties, which we believe is a contribution in its own. Then, in Sect. 3 we present our new vector proactive secret sharing scheme and analyze its security. Finally, in Sect. 4 we show how to instantiate the proposed scheme with a concise vector commitment scheme and then evaluate the theoretical and practical performance of the proposed instantiation.

1.2 Related Work

Since the work of [12] several proactive secret sharing schemes with various properties have been proposed. [11,19] proposed proactive secret sharing schemes where the number N of shareholders and the threshold value T can be changed during a share update. [7,20] proposed proactive secret sharing schemes that work in asynchronous networks where no global clock is available. [17] proposed a scheme which has both properties. However, all of these schemes have high communication and computation costs when storing large data items.

More recently, Baron, Defrawy, Lampkins, and Ostrovsky in [1,2] proposed proactive secret sharing schemes with optimal amortized communication complexity. However, while their schemes enjoy optimal communication costs asymptotically, they do not work well with a small number of shareholders (e.g., $N = 3$) as they require $T < \frac{N}{8}$ and enabling $T < \frac{N}{2}$ requires expensive party virtualization techniques. This approach uses *packed secret sharing* where a set of messages is batched together. The authors propose a batch size of $N - 3T$ which is obviously infeasible for small parameters like $N = 3, T = 1$. They also make use of *double sharings*. For l messages this would require every shareholder to send at least $2l$ shares to every other shareholder. Compared to this our approach based on generalized Pedersen commitments, requires $l + 1$ shares and t commitments to be broadcasted per shareholder. For suitably large l and small t this leads to significantly less bandwidth consumption.

2 Preliminaries

2.1 Notation

We use the convention that $\mathbb{N} = \{1, 2, \ldots\}$ and define $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For $(a, b) \in \mathbb{Z}^2$, $a \leq b$, we define $[a, b] = \{x \in \mathbb{Z} : a \leq x \leq b\}$. For $n \in \mathbb{N}$, we define $[n] = [1, n]$ and $\mathbb{Z}_n = [0, n - 1]$. By MODINV we denote an algorithm that on input $(a, m) \in \mathbb{N}_0^2$ outputs the smallest $b \in \mathbb{N}$ such that $(a * b) \bmod m = 1$, or $\perp$ if such b does not exist. For a finite cyclic group $\mathbb{G}$ associated with operator $\circ$, we denote by $\mathsf{GEN}(\mathbb{G})$ the set of generators of $\mathbb{G}$. Furthermore, we denote by EXP an exponentiation algorithm that on input $(a, b) \in \mathbb{G} \times \mathbb{N}$ outputs a^b such that $a^1 = a$ and $a^{i+1} = a^i \circ a$. For a finite set S, we denote by $U(S)$ the uniform distribution over S. For $\tau \in \mathbb{N}$, we denote by $\mathsf{ProbAlgo}(\tau)$ the set of probabilistic algorithms that for any input halt after at most τ steps. By $\Im(\mathcal{A})$ we denote the image of algorithm $\mathcal{A}$.

2.2 Network Model

A probabilistic protocol P defines an input-output behavior for a set of communicating parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. We write $P\langle\mathcal{P}_1(x_1) \to y_1, \ldots, \mathcal{P}_n(x_n) \to y_n\rangle$ to denote an execution of protocol P, where party $\mathcal{P}_i$ gets input x_i and outputs y_i. Here we assume that each party has a direct communication channel with each other party. In addition, we assume that there exists a broadcast channel with the property that if a party $\mathcal{P}_i$ receives a broadcast message m from party $\mathcal{P}_j$, then it is guaranteed that all other parties $\mathcal{P}_k$ receive the same broadcast message m from $\mathcal{P}_j$. When we write that during a protocol execution $P\langle\{\mathcal{P}_i(x_1) \to y_1\}_{i\in[N]}\rangle$ an adversary $\mathcal{A}$ controls $T \in [0, N]$ parties, we mean that there exists $I \subset [T]$ such that for $i \in I$, the input-output behavior and communication behavior of party $\mathcal{P}_i$ is controlled by $\mathcal{A}$. A majority of the protocol participants can, however, decide to reboot corrupted parties, in which case the adversary loses control over them, their state is cleared, and they return to their

specified behavior. We remark that our protocols require the usage of private authenticated channels, which means that messages are always delivered to the correct communication partner, that their content and order cannot be modified, and that no information about the message content can be obtained by tapping the channel.

2.3 Discrete Logarithm Problem

We state the fixed generator discrete logarithm problem [16].

Definition 1 (Discrete logarithm problem). *Let $\mathbb{G}$ be a finite cyclic group, $g \in \mathsf{GEN}(\mathbb{G})$, and $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function. We say $\mathsf{DLOG}(\mathbb{G}, g)$ is ϵ-hard if for all τ, for all $\mathcal{A} \in \mathsf{ProbAlgo}(\tau)$,*

$$\Pr\left[\begin{matrix}\mathsf{EXP}(g,x) = y : \\ U(\mathbb{G}) \to y, \mathcal{A}(y) \to x\end{matrix}\right] \leq \epsilon(\tau).$$

2.4 Vector Commitments

We define vector commitment schemes as we will use them in this paper. We remark that our vector commitment schemes do not support selective opening as opposed to those proposed in [8].

Definition 2 (Vector commitment scheme). *A vector commitment scheme is a tuple $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, where $L \in \mathbb{N}$, $\mathcal{P}$, $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{D}$ are sets, Setup and Commit are probabilistic algorithms, and Open is a deterministic algorithm, with the following properties.*

$\mathsf{Setup} : \emptyset \to \mathcal{P}$. *This algorithm gets no input and outputs parameters $\rho \in \mathcal{P}$.*

$\mathsf{Commit} : \mathcal{P} \times \mathcal{M}^L \to \mathcal{C} \times \mathcal{D}$. *This algorithm gets as input parameters $\rho \in \mathcal{P}$ and message $m \in \mathcal{M}^L$, and outputs a commitment $c \in \mathcal{C}$ and a decommitment $d \in \mathcal{D}$.*

$\mathsf{Open} : \mathcal{P} \times \mathcal{M}^L \times \mathcal{C} \times \mathcal{D} \to \{0,1\}$. *This algorithm gets as input parameters $\rho \in \mathcal{P}$, message $m \in \mathcal{M}^L$, commitment $c \in \mathcal{C}$, and decommitment $d \in \mathcal{D}$, and outputs $b \in \{0,1\}$.*

Correct Functionality. *We say VC is correct if for all $m \in \mathcal{M}^L$,*

$$\Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 : \\ \mathsf{Setup}() \to \rho, \mathsf{Commit}(\rho, m) \to (c, d)\end{matrix}\right] = 1 \ .$$

Binding Security. *Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function. We say VC is ϵ-binding if for all $\tau \in \mathbb{N}$, $\mathcal{A} \in \mathsf{ProbAlgo}(\tau)$,*

$$\Pr\left[\begin{matrix}b = 1 \wedge b' = 1 \wedge m \neq m' : \\ \mathsf{Setup}() \to \rho, \mathcal{A}(\rho) \to (c, m, d, m', d'), \\ \mathsf{Open}(\rho, m, c, d) \to b, \mathsf{Open}(\rho, m, c, d') \to b'\end{matrix}\right] \leq \epsilon(\tau) \ .$$

Hiding Security. *We say* VC *is perfectly hiding if for all* $\rho \in \mathcal{P}$, $(m_1, m_2) \in \mathcal{M}^{L\times 2}$, $c \in \mathcal{C}$,

$$\Pr\left[\begin{matrix} c = c' : \\ \mathsf{Commit}(\rho, m_1) \to (c', d') \end{matrix}\right] = \Pr\left[\begin{matrix} c = c' : \\ \mathsf{Commit}(\rho, m_2) \to (c', d') \end{matrix}\right] .$$

Homomorphic Operation. *For* $\rho \in \mathcal{P}$, *define* $\mathtt{COMS}(\rho) = \{(m, c, d) \in \mathcal{M}^L \times \mathcal{C} \times \mathcal{D} : \mathsf{Open}(\rho, m, c, d) = 1\}$. *We say* VC *is homomorphic if there exist binary operations* $+$, $*$, *and* $\circ$ *such that for all* $\rho \in \mathcal{P}$, $(m_1, c_1, d_1) \in \mathtt{COMS}(\rho)$, *and* $(m_2, c_2, d_2) \in \mathtt{COMS}(\rho)$,

$$\mathsf{Open}(\rho, m_1 + m_2, c_1 * c_2, d_1 \circ d_2) = 1 .$$

2.5 Proactive Secret Sharing

We give a definition of proactive secret sharing which will be useful for analyzing the security of the scheme proposed later in this work. We remark that while other authors only sketch syntax and security definitions for proactive secret sharing (e.g., [12]), our definition captures many subtleties of these schemes (e.g., it states exactly when the adversary gains control over parties and when it loses control which is a delicate subject [14]). Such a more precise definition is a valuable contribution in its own.

Informal Description. We first give an overview of the formal definition and then present the precise definition later in Definition 3. A proactive secret sharing scheme consists of a set of protocols that are run between a dealer $\mathfrak{D}$ and a set of shareholders $\mathfrak{S}_1, \ldots, \mathfrak{S}_N$. The goal of the dealer is to store some secret information at the shareholders in a way that none of the shareholders obtains information about the secret. The information can only be reconstructed if a sufficient number of shares are combined together. Protocol Setup is used for initializing the parties. Protocol Share is used for distributing the secret information to the shareholders in terms of secret shares. Protocol $\mathsf{Reshare}$ refreshes the secret shares such that the new shares have no correlation with the old shares. Protocol $\mathsf{Reconstruct}$ retrieves the shares, asserts their validity, and reconstructs the secret information.

We require several properties of a proactive secret sharing scheme. Correct functionality guarantees that if the scheme is run by honest parties, the original information will be restored. Secrecy guarantees that a coalition of curious shareholders up to a threshold number cannot learn any information about the secret. Robustness guarantees that the scheme tolerates up to a threshold number of shareholders that act maliciously and do not follow the protocol.

The definitions of Secrecy and Robustness are given in terms of games played by an adversary that can corrupt a threshold number of parties and tries to either learn information or destroy the secret information (Figs. 1 and 2). For the secrecy game (Fig. 1), the adversary can choose to learn the secrets of a

given set of shareholders I after each round (e.g., sharing or resharing), where the freshly corrupted set of shareholders I' and the previously corrupted set I combined must be of size at most the threshold T. The goal of the adversary is to learn something about the secret information m in terms of a function value $F(m)$ for any function F. The secrecy definition requires that $F(m)$ can be computed equally successful by a simulator B which does not see any of the additional secret information that the adversary may obtain by corrupting certain shareholders. This definition of secrecy follows the ideas of Goldwasser and Micali for defining semantic security [9]. Similarly, for the robustness game (Fig. 2), the adversary can choose to act on behalf of a given set of shareholders during the protocol runs of Share, Reshare, or Reconstruct, but the number of new and old corrupted shareholders must never exceed T. The robustness definition requires that the reconstructed value after the interference of the adversary still corresponds to the value that has been initially stored.

Formal Definition. In the definition we use the following notation. We usually denote the dealer by $\mathfrak{D}$ and shareholder i by $\mathfrak{S}_i$. We write $\mathsf{Share}\langle\rho, m\rangle \to S$ as an abbreviation for $\mathsf{Share}\langle\mathfrak{D}(\rho, m), \{\mathfrak{S}_i(\rho) \to s_i\}_{i\in[N]}\rangle$, $S \leftarrow (s_1, \ldots, s_N)$. For $S = (s_1, \ldots, s_N)$, we write $\mathsf{Reshare}\langle\rho, S\rangle \to S'$ for $\mathsf{Reshare}\langle\{\mathfrak{S}_i(\rho, s_i) \to s'_i\}_{i\in[N]}\rangle$, $S' \leftarrow (s'_1, \ldots, s'_N)$. The game notation that we use follows the notation described in [3,4]. At the start of any game G the special algorithm **Initialize** is executed and its output is handed to the adversary. Afterwards the adversary can call the algorithms specified in the game and obtains the corresponding outputs. The game ends when the adversary calls the special algorithm **Finalize**. The output of the game is defined as the output of that algorithm.

Definition 3 (Proactive secret sharing scheme). *A proactive secret sharing scheme is a tuple* $\mathsf{PSS} = (N, T, \mathcal{P}, \mathcal{M}, \mathcal{S}, \mathsf{Setup}, \mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct})$, *where* $(N, T) \in \mathbb{N} \times \mathbb{N}_0$, $N > 1$, $T < \frac{N}{2}$, $\mathcal{P}$, $\mathcal{M}$, *and* $\mathcal{S}$ *are sets,* Setup *is a probabilistic algorithm, and* Share, Reshare, *and* Reconstruct *are probabilistic protocols with the following properties:*

$\mathsf{Setup} : \emptyset \to \mathcal{P}$. *This algorithm gets no input and outputs parameters* $\rho \in \mathcal{P}$.

$\mathsf{Share}\langle\mathfrak{D} : \mathcal{P} \times \mathcal{M} \to \emptyset, \{\mathfrak{S}_i : \mathcal{P} \to \mathcal{S}\}_{i\in[N]}\rangle$. *The dealer* $\mathfrak{D}$ *gets as input parameters* $\rho \in \mathcal{P}$, *and message* $m \in \mathcal{M}$. *For* $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *get as input parameters* $p \in \mathcal{P}$, *and outputs a secret share* $s_i \in \mathcal{S}$.

$\mathsf{Reshare}\langle\{\mathfrak{S}_i : \mathcal{P} \times \mathcal{S} \to \mathcal{S}\}_{i\in[N]}\rangle$. *For* $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *gets as input parameters* $\rho \in \mathcal{P}$ *and secret share* $s_i \in \mathcal{S}$, *and outputs a secret share* $s'_i \in \mathcal{S}$.

$\mathsf{Reconstruct}\langle\mathfrak{R} : \mathcal{P} \to \mathcal{M} \cup \{\perp\}, \{\mathfrak{S}_i : \mathcal{P} \times \mathcal{S} \to \emptyset\}_{i\in[N]}\rangle$. *The receiver* $\mathfrak{R}$ *gets as input parameters* $\rho \in \mathcal{P}$. *For* $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *gets as input parameters* $\rho \in \mathcal{P}$ *and secret share* $s_i \in \mathcal{S}$. *The receiver* $\mathfrak{R}$ *outputs a message* $m \in \mathcal{M}$.

Correct Functionality. *For $\rho \in \mathcal{P}$ and $m \in \mathcal{M}$, we define*

$$\texttt{SHARES}(\rho, m) = \left\{ \exists l \in \mathbb{N}_0 : \Pr\left[\begin{array}{c} (s_1, \ldots, s_N) : \\ (s_{l,1}, \ldots, s_{l,n}) = (s_1, \ldots, s_N) : \\ \mathsf{Share}\langle \mathfrak{D}(\rho, m), \{\mathfrak{S}_i(\rho) \to s_{0,i}\}_{i\in[N]} \rangle, \\ \mathsf{Reshare}\langle \{\mathfrak{S}_i(\rho, s_{0,i}) \to s_{1,i}\}_{i\in[N]} \rangle, \\ \ldots, \\ \mathsf{Reshare}\langle \{\mathfrak{S}_i(\rho, s_{l-1,i}) \to s_{l,i}\}_{i\in[N]} \rangle \end{array} \right] > 0 \right\}$$

as the set of all possible share configurations at which the shareholders can arrive after sharing and resharing m under parameter ρ. We say PSS *is correct if for all $\rho \in \mathcal{P}$, $m \in \mathcal{M}$, $(s_1, \ldots, s_N) \in \texttt{SHARES}(\rho, m)$,*

$$\Pr\left[\begin{array}{c} m = m' : \\ \mathsf{Reconstruct}\langle m' \leftarrow \mathfrak{D}(\rho), \{\mathfrak{S}_i(\rho, s_i)\}_{i\in[N]} \rangle \end{array} \right] = 1 \ .$$

Secrecy. *Let $\epsilon : \mathbb{N}^2 \to \mathbb{R}$ be a function. We say* PSS *is ϵ-secret if for all probability distributions $\mathcal{D}$ over $\mathcal{M}$, functions $F : \mathcal{M} \to \{0,1\}^*$, $\tau_{\mathcal{A}}, \tau_{\mathcal{B}} \in \mathbb{N}$, and $\mathcal{A} \in \mathsf{ProbAlgo}(\tau_{\mathcal{A}})$, there exists $\mathcal{B} \in \mathsf{ProbAlgo}(\tau_{\mathcal{B}})$ such that*

$$\Pr\left[\begin{array}{c} F(m) = y : \\ \mathcal{D} \to m, G_1(\mathcal{A}; m) \to y \end{array} \right] \leq \Pr\left[\begin{array}{c} F(m) = y : \\ \mathcal{D} \to m, \mathcal{B} \to y \end{array} \right] + \epsilon(\tau_{\mathcal{A}}, \tau_{\mathcal{B}}) \ ,$$

where $G_1(\mathcal{A}; m)$ is defined in Fig. 1.

Game $G_1(\mathcal{A}; m)$

Initialize

1 : $I \leftarrow \{\}, S \leftarrow \bot$
2 : $\mathsf{Setup}() \to \rho$
3 : **return** ρ

Share (I')

1 : $I \leftarrow (I'$ **if** $|I'| \leq T$ **else** $\emptyset)$
Run $\mathsf{Share}\langle \rho, m \rangle \to S$, where
2 : shareholders I are controlled by $\mathcal{A}$ until reboot

Reshare (I')

1 : $I \leftarrow (I'$ **if** $|I \cup I'| \leq T$ **else** $\emptyset)$
Run $\mathsf{Reshare}\langle \rho, S \rangle \to S$, where
2 : shareholders I are controlled by $\mathcal{A}$ until reboot

Finalize (y)

1 : **return** y

Fig. 1. The game used in the secrecy definition for proactive secret sharing.

Robustness. *Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function. We say* PSS *is ϵ-robust if for all $m \in \mathcal{M}$, $\tau \in \mathbb{N}$, $\mathcal{A} \in \mathsf{ProbAlgo}(\tau)$,*

$$\Pr\left[\begin{array}{c} m \neq m' : \\ G_2(\mathcal{A}, m) \to m' \end{array} \right] \leq \epsilon(\tau) \ ,$$

where $G_2(\mathcal{A}, m)$ is defined in Fig. 2.

Game $G_2(\mathcal{A}; m)$

Initialize

1 : $I \leftarrow \{\}, S \leftarrow \perp$
2 : $\mathsf{Setup}() \rightarrow \rho$
3 : **return** ρ

Share (I')

1 : $I \leftarrow (I'$ **if** $|I \cup I'| \leq T$ **else** $\emptyset)$
2 : Run $\mathsf{Share}\langle \rho, m \rangle \rightarrow S$, where shareholders I are controlled by $\mathcal{A}$ until reboot

Reshare (I')

1 : $I \leftarrow (I'$ **if** $|I \cup I'| \leq T$ **else** $\emptyset)$
2 : Run $\mathsf{Reshare}\langle \rho, S \rangle \rightarrow S$, where shareholders I are controlled by $\mathcal{A}$ until reboot

Finalize (I')

1 : $I \leftarrow (I'$ **if** $|I \cup I'| \leq T$ **else** $\emptyset)$
2 : Run $\mathsf{Reconstruct}\langle \rho, S \rangle \rightarrow m'$, where shareholders I are controlled by $\mathcal{A}$ until reboot
3 : **return** m'

Fig. 2. The game used in the robustness definition for proactive secret sharing.

3 Proactive Secret Sharing with Vector Commitments

We now present our construction of a proactive secret sharing scheme that uses vector commitments for improving efficiency. Our construction is based on the construction of [12] and enhances it so that in each sharing a vector of messages can be stored instead of only a single message. We first present the description of our vector proactive secret sharing scheme in Subsect. 3.1 and then prove its security in Subsect. 3.2.

3.1 Scheme Description

Overview of the Scheme. Our proactive secret sharing scheme follows the construction of [12], but uses a homomorphic vector commitment scheme VC instead of a single message homomorphic commitment scheme. Algorithm Setup of our scheme simply generates commitment parameters ρ by running the setup algorithm of the vector commitment scheme.

Protocol Share works as follows. On input a message vector $(m_1, \ldots, m_L)$, the dealer first generates secret shares of each m_i using Shamir's Secret Sharing Scheme [18] by sampling $D = N - T - 1$ secret polynomial coefficients, where N is the number of shareholders and T is the corruption threshold. Then, it creates a commitment c_0 to the message vector and a commitment c_i to each of the secret coefficient vectors. The corresponding decommitments $(d_0, \ldots, d_D)$ are used to compute a share of a decommitment r_i corresponding to the message vector. Finally, the dealer broadcasts all the commitments $(c_0, \ldots, c_D)$ and sends share vector $(s_{i,1}, \ldots, s_{i,L})$ and the decommitment share r_i to shareholder $\mathfrak{S}_i$.

Protocol $\mathsf{Reshare}$ works as follows. At first, the shareholders engage in sub protocol $\mathsf{ShareRecovery}$ in order to detect parties that hold invalid input shares. If

such parties are detected, then these will be rebooted and their shares be recovered so that after the execution of sub protocol ShareRecovery the shareholders hold a consistent share configuration. Now, each of the shareholders creates L verifiable sharings of the identity of the finite field message space using sub protocol ShareIdentity. Next, each shareholder asserts that the received shares of the identity are consistent by verifying the received commitments. If this is the case, then it combines the commitments, decommitments, and shares of the identity sharings with the existing secret shares in a way that the new shares still reconstruct to the original message vector. Here, only commitment c_0 is kept unchanged as an invariant referring to the original message vector. In the other case, i.e., if an inconsistency after ShareIdentity is detected, the faulty parties are determined and rebooted, their shares are recovered, and protocol Reshare is started from the beginning.

Protocol Reconstruct works as follows. The dealer $\mathfrak{D}$ retrieves all shares, commitments, and decommitment shares from the shareholders. It then determines a subset G of parties whose shares are qualified for reconstruction, i.e., with $|G| = D + 1$ and such that the shares are consistent with the commitments and decommitments. If such a subset is found, Lagrange Interpolation is used to reconstruct the message vector. If such a subset is not found, then the protocol aborts and outputs $\perp$. The latter case, however, is guaranteed not to occur if not more than T parties are corrupted.

Detailed Description. We now present our vector proactive secret sharing scheme in detail.

Scheme 1 (VPSS). *Let* $(N, T) \in \mathbb{N} \times \mathbb{N}_0$ *such that* $N < p$ *and* $T < \frac{N}{2}$. *Let* $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *be a homomorphic vector commitment scheme such that* $\mathcal{M}$ *is a finite field of prime order* p. *Let* $D = N - T - 1$ *and* $\mathcal{S} = \mathcal{M}^L \times \mathcal{C}^{1+D} \times \mathcal{D}$. *For a given sharing* $((s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i))_{i\in[N]} \in \mathcal{S}^n$, *we define the subset of parties qualified for reconstruction by*

$$\mathtt{QUALI}(((s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i))_{i\in[N]}) = \left\{ \begin{array}{c} G \subseteq [N] : \\ |G| = D + 1 \wedge (\forall (i,j,k) \in G \times G \times [0,D] : c_{i,k} = c_{j,k}) \\ \wedge\, \forall i \in G : \mathsf{Open}(\rho, (s_{i,1}, \ldots, s_{i,L}), \bigcirc_{j\in[0,D]} \mathsf{EXP}(c_{i,j}, i^j), r_i) = 1 \end{array} \right\}.$$

We define the proactive secret sharing scheme $\mathsf{VPSS}_{N,T,\mathsf{VC}} = (N, T, \mathcal{P}, \mathcal{M}^L, \mathcal{S}, \mathsf{Setup}, \mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct})$, *where* Share, Reshare, *and* Reconstruct *are defined with sub protocols* ShareRecovery *and* ShareIdentity *as follows:*

Main protocols:

$\mathsf{Share}\langle \mathfrak{D}(\rho \in \mathcal{P}, m \in \mathcal{M}^L), \{\mathfrak{S}_i(\rho \in \mathcal{P}) \to s_i \in \mathcal{S}\}_{i\in[N]} \rangle$:

The dealer $\mathfrak{D}$ *does the following:*

1. *Let* $m = (m_1, \ldots, m_L) \in \mathcal{M}^L$.
2. *For* $(i, j) \in [L] \times [D]$, *sample* $U(\mathcal{M}) \to a_{i,j}$.

3. *For* $(i,j) \in [N] \times [L]$, *compute* $s_{i,j} \leftarrow m_j \bigcirc_{k\in[D]} \mathsf{EXP}(a_{j,k}, i^k)$.
4. *Compute* $\mathsf{Commit}(\rho, (m_1, \ldots, m_L)) \rightarrow (c_0, d_0)$, *and for* $i \in [D]$, *compute* $\mathsf{Commit}(\rho, (a_{1,i}, \ldots, a_{L,i})) \rightarrow (c_i, d_i)$.
5. *For* $i \in [N]$, *compute* $r_i \leftarrow d_0 \bigcirc_{j\in[D]} \mathsf{EXP}(d_j, i^j)$.
6. *Broadcast* $(c_0, \ldots, c_D)$ *and for* $i \in [N]$, *send* r_i *and* $(s_{i,1}, \ldots, s_{i,L})$ *to shareholder* $\mathfrak{S}_i$.

For $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *sets* $s_i \leftarrow (s_{i,1}, \ldots, s_{i,L}, c_0, \ldots, c_D, r_i)$.

$\mathsf{Reshare}\langle\{\mathfrak{S}_i(\rho \in \mathcal{P}, s_i \in \mathcal{S}) \rightarrow s'_i \in \mathcal{S} \cup \{\bot\}\}_{i\in[N]}\rangle$:

Run protocol $\mathsf{ShareRecovery}\langle\{\mathfrak{S}_i(\rho \in \mathcal{P}, s_i \in \mathcal{S}) \rightarrow s_i \in \mathcal{S}\}_{i\in[N]}\rangle$.

For $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *does the following.*

1. *If* $s_i = \bot$, *set* $s'_i \leftarrow \bot$ *and return.*
2. *Let* $s_i = (s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i)$.
3. *Run protocol* $\mathsf{ShareIdentity}\langle\mathfrak{S}_i(\rho), \{\mathfrak{S}_j(\rho) \rightarrow \hat{s}_{i,j}\}_{j\in[N]}\rangle$ *and let* $\hat{s}_{i,j} = (\hat{s}_{i,j,1}, \ldots, \hat{s}_{i,j,L}, \hat{c}_{i,j,1}, \ldots, \hat{c}_{i,j,D}, \hat{r}_{i,j})$.
4. *Wait until for all* $j \in [N]$, $\hat{s}_{j,i}$ *has been received or a timeout occurs. In case of a timeout of party* j, *set* $\hat{s}_{j,i} \leftarrow \bot$.
5. *For* $j \in [N]$, *compute* $\hat{c}_{i,j} \leftarrow \bigcirc_{k\in[D]} \mathsf{EXP}(\hat{c}_{j,i,k}, i^k)$ *and* $b_{i,j} \leftarrow \mathsf{Open}(\rho, (\hat{s}_{i,j,1}, \ldots, \hat{s}_{i,j,L}), \hat{c}_{i,j}, \hat{r}_{i,j})$, *and broadcast* $B_i = (b_{i,1}, \ldots, b_{i,N})$.
6. *Wait until for all* $j \in [N]$, B_j *has been received or a timeout occurs. In case of a timeout of party* j, *set* $B_j \leftarrow 0^N$.
7. *If for all* $j \in [N]$, $B_j = 1^N$, *then all shareholders behaved consistently. In this case, recompute the shares as follows:*
   (a) *For* $j \in [L]$, *compute* $s'_{i,j} \leftarrow s_{i,j} \bigcirc_{k\in[N]} \hat{s}_{k,i,j}$.
   (b) *For* $j \in [D]$, *compute* $c'_{i,j} \leftarrow c_{i,j} \bigcirc_{k\in[N]} \hat{c}_{k,i,j}$.
   (c) *Compute* $r'_i \leftarrow r_i \bigcirc_{j\in[N]} \hat{r}_{j,i}$.
   (d) *Set* $s'_i \leftarrow (s'_{i,1}, \ldots, s'_{i,L}, c_{i,0}, c'_{i,1}, \ldots, c'_{i,D}, r'_i)$.

   If there exists $j \in [N]$ *such that* $0 \in B_j$, *then the shareholders behaved inconsistently. In this case, determine the set of faulty shareholders, reboot them, recover their message and decommitment shares as described in [12], and restart the resharing protocol.*

$\mathsf{Reconstruct}\langle\mathfrak{D}(\rho \in \mathcal{P}) \rightarrow m \in \mathcal{M}^L \cup \{\bot\}, \{\mathfrak{S}_i(\rho \in \mathcal{P}, s_i \in \mathcal{S})\}_{i\in[N]}\rangle$:

For $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *sends* s_i *to* $\mathfrak{D}$.

The receiver $\mathfrak{D}$ *waits until it received* s_i *for* $i \in [N]$ *or a timeout occurs. In case of a timeout of party* i, *set* $s_i \leftarrow \bot$. *Then,* $\mathfrak{D}$ *does the following:*

1. *For* $i \in [N]$, *let* $s_i = (s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i)$.
2. *If* $\mathtt{QUALI}((s_1, \ldots, s_N)) = \emptyset$, *set* $m \leftarrow \bot$ *and return. Otherwise find* $G \in \mathtt{QUALI}((s_1, \ldots, s_N))$.
3. *For* $i \in G$, *compute* $l_i \leftarrow \prod_{j\in G\setminus\{i\}} j * \mathsf{MODINV}(j - i, p)$.
4. *For* $i \in [L]$, *compute* $m_i \leftarrow \bigcirc_{j\in G} \mathsf{EXP}(s_{j,i}, l_j)$.
5. *Set* $m \leftarrow (m_1, \ldots, m_L)$.

Sub protocols:

$\mathsf{ShareRecovery}\langle\{\mathfrak{S}_i(\rho \in \mathcal{P}, s_i \in \mathcal{S}) \rightarrow s'_i \in \mathcal{S} \cup \{\bot\}\}_{i\in[N]}\rangle$:

For $i \in [N]$, *shareholder* $\mathfrak{S}_i$ *does the following:*

1. *Let* $s_i = (s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i)$.
2. *Broadcast* $(c_{i,0}, \ldots, c_{i,D})$.

3. *Wait until for* $j \in [N]$, $(c_{j,0}, \ldots, c_{j,D})$ *has been received or a timeout occurs. In case of a timeout of party* j, *set* $c_{j,k} \leftarrow \perp$ *for* $k \in [0, D]$.
4. *Determine a set* $G_i \subseteq [N]$ *such that:*
   (a) $|G_i| = D + 1$
   (b) For $(j,k) \in G_i^2$, $(c_{j,0}, \ldots, c_{j,D}) = (c_{k,0}, \ldots, c_{k,D})$.
   If such a set G_i *does not exist, set* $s_i' \leftarrow \perp$ *and return.*
5. *Let* $j \in G_i$ *and for* $k \in [0, D]$, *set* $c'_{i,k} \leftarrow c_{j,k}$.
6. *Compute* $\hat{c}_i \leftarrow \bigcirc_{k\in[0,D]} \mathsf{EXP}(c'_{i,k}, i^k)$, $b_i \leftarrow \mathsf{Open}(\rho, s_{i,1}, \ldots, s_{i,L}, \hat{c}_i, r_i)$, *and broadcast* b_i.
7. *Wait until for all* $j \in [N]$, b_j *has been received or a timeout occurs. In case of a timeout of party* j, *set* $b_j \leftarrow 0$.
8. *Let* $B_i = \{j \in [N] : b_j = 0\}$. *If* $B_i \neq \emptyset$, *vote for rebooting shareholders* B_i *and recover the message and decommitment shares of the rebooted shareholders as described in [12].*
9. *Set* $s_i' \leftarrow (s_{i,1}, \ldots, s_{i,L}, c'_{i,0}, \ldots, c'_{i,D}, r_i)$.

$\mathsf{ShareIdentity}\langle \mathfrak{D}(\rho \in \mathcal{P}), \{\mathfrak{S}_i(\rho \in \mathcal{P}) \rightarrow s_i \in \mathcal{M}^L \times \mathcal{C}^D \times \mathcal{D}\}_{i\in[N]}\rangle$:

The dealer $\mathfrak{D}$ *does the following:*

1. *For* $(i,j) \in [L] \times [D]$, *sample* $U(\mathcal{M}) \rightarrow a_{i,j}$.
2. *For* $(i,j) \in [N] \times [L]$, *compute* $s_{i,j} \leftarrow \bigcirc_{k\in[D]} \mathsf{EXP}(a_{j,k}, i^k)$.
3. *For* $i \in [D]$, *compute* $\mathsf{Commit}(\rho, (a_{1,i}, \ldots, a_{L,i})) \rightarrow (c_i, d_i)$.
4. *For* $i \in [N]$, *compute* $r_i \leftarrow \bigcirc_{j\in[D]} \mathsf{EXP}(d_j, i^j)$.
5. *Broadcast* $(c_1, \ldots, c_D)$ *and for* $i \in [N]$, *send* r_i *and* $(s_{i,1}, \ldots, s_{i,L})$ *to party* $\mathfrak{S}_i$.

For $i \in [N]$, *party* $\mathfrak{S}_i$ *sets* $s_i \leftarrow (s_{i,1}, \ldots, s_{i,L}, c_1, \ldots, c_D, r_i)$.

3.2 Scheme Analysis

We analyze the security of the vector proactive secret sharing scheme VPSS proposed in Subsect. 3.2. We first prove the correct functionality. Then, we show that if the used vector commitment schemes is information-theoretically hiding, our vector proactive secret sharing provides information-theoretic secrecy. Finally, we show that the robustness of our vector commitments scheme can be reduced the binding security of the used vector commitment scheme.

Theorem 1 (Correctness). *Let* $(N,T) \in \mathbb{N} \times \mathbb{N}_0$ *such that* $N < p$ *and* $T < \frac{N}{2}$. *Let* $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *be a homomorphic vector commitment scheme such that* $\mathcal{M}$ *is a finite field of prime order. The proactive secret sharing scheme* $\mathsf{VPSS}_{N,T,\mathsf{VC}}$ *is correct.*

Proof. Let $(N,T) \in \mathbb{N} \times \mathbb{N}_0$ such that $N < p$ and $T < \frac{N}{2}$. Let $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ be a homomorphic vector commitment scheme such that $\mathcal{M}$ is a finite field of prime order p. Let $\mathsf{VPSS}_{N,T,\mathsf{VC}} = (N, T, \mathcal{P}, \mathcal{M}^L, \mathcal{S}, \mathsf{Setup}, \mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct})$ and $D = N - T - 1$.

For $\rho \in \mathcal{P}$, $m \in \mathcal{M}$, and $i \in \mathbb{N}_0$, define

$$\texttt{SHARES}(\rho, m, l) = \left\{ \Pr\left[\begin{array}{c} (s_1, \ldots, s_N) : \\ (s_{l,1}, \ldots, s_{l,n}) = (s_1, \ldots, s_N) : \\ \mathsf{Share}\langle \mathfrak{D}(\rho, m), \{\mathfrak{S}_i(\rho) \rightarrow s_{0,i}\}_{i\in[N]}\rangle, \\ \mathsf{Reshare}\langle \{\mathfrak{S}_i(\rho, s_{0,i}) \rightarrow s_{1,i}\}_{i\in[N]}\rangle, \\ \ldots, \\ \mathsf{Reshare}\langle \{\mathfrak{S}_i(\rho, s_{l-1,i}) \rightarrow s_{l,i}\}_{i\in[N]}\rangle \end{array} \right] > 0 \right\} .$$

Let $\rho \in \mathcal{P}$ and $m = (m_1, \ldots, m_L) \in \mathcal{M}^L$. By the definition of protocol Share, we observe that for $(s_1, \ldots, s_N) \in \texttt{SHARES}(\rho, m, 0)$ we have:

$$\begin{aligned} &\exists A \in \mathcal{M}^{L\times D} : \forall i \in [N] : \\ &s_i = (s_{i,1}, \ldots, s_{i,L}, c_0, \ldots, c_t, r_i) \in \mathcal{S} \\ &\wedge \forall j \in [L] : s_{i,j} = m_j \bigcirc_{k\in[D]} \mathsf{EXP}(A_{j,k}, i^k) \\ &\wedge (c_{i,0}, d_{i,0}) \in \mathsf{Commit}(\rho, (m_1, \ldots, m_L)) \\ &\wedge \forall j \in [D] : (c_{i,j}, d_{i,j}) \in \mathsf{Commit}(\rho, (a_{1,j}, \ldots, a_{L,j})) \\ &\wedge r_i = \bigcirc_{j\in[0,D]} \mathsf{EXP}(d_{i,j}, i^j) \end{aligned}$$

Furthermore, we observe that if the conditions above hold, then $G = [D+1] \in \texttt{QUALI}((s_1, \ldots, s_n))$ and for $i \in [L]$, we have $m_i = \bigcirc_{j\in G} \mathsf{EXP}(s_{j,i}, l_j)$, where $l_j = \prod_{k\in G\setminus\{j\}} k * \mathsf{MODINV}(k - j, p)$.

Next, we observe that by the definition of $\mathsf{Reshare}$ and the homomorphic properties of the shares and the commitments we have $\texttt{SHARES}(\rho, m, 0) = \texttt{SHARES}(\rho, m, 1)$. It follows that for all $l \in \mathbb{N}_0$, $\texttt{SHARES}(\rho, m) = \texttt{SHARES}(\rho, m, l)$. We obtain that for any $\rho \in \mathcal{P}$, $(s_1, \ldots, s_N) \in \texttt{SHARES}(\rho, m)$ we have

$$\Pr\left[\begin{array}{c} m = m' : \\ \mathsf{Reconstruct}\langle m' \leftarrow \mathfrak{D}(\rho), \{\mathfrak{S}_i(\rho, s_i)\}_{i\in[N]}\rangle \end{array} \right] = 1 .$$

□

Theorem 2 (Secrecy). *Let $(N, T) \in \mathbb{N} \times \mathbb{N}_0$ such that $N < p$ and $T < \frac{N}{2}$. Let $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ be a perfectly hiding homomorphic vector commitment scheme such that $\mathcal{M}$ is a finite field of prime order. Then there exists $\alpha \in \mathbb{R}$ such that $\mathsf{VPSS}_{N,T,\mathsf{VC}}$ is ϵ-secret with*

$$\epsilon(\tau_{\mathcal{A}}, \tau_{\mathcal{B}}) = \begin{cases} 0 & if \tau_{\mathcal{B}} \geq \alpha * \tau_{\mathcal{A}}, \\ 1 & if \tau_{\mathcal{B}} < \alpha * \tau_{\mathcal{A}}. \end{cases}$$

Proof. Let $(N, T) \in \mathbb{N} \times \mathbb{N}_0$ such that $N < p$ and $T < \frac{N}{2}$. Let $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ be a homomorphic vector commitment scheme such that $\mathcal{M}$ is a finite field of prime order p. Let $\mathsf{VPSS}_{N,T,\mathsf{VC}} = (N, T, \mathcal{P}, \mathcal{M}, \mathcal{S}, \mathsf{Setup}, \mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct})$. Let $\mathcal{D}$ be a probability distribution over $\mathcal{M}$, $F : \mathcal{M} \rightarrow \{0, 1\}^*$ be a function, $\tau_{\mathcal{A}} \in \mathbb{N}$, and $\mathcal{A} \in \mathsf{ProbAlgo}(\tau_{\mathcal{A}})$.

We construct an algorithm $\mathcal{B}$ that simulates $G_1(\mathcal{A}; 0^L)$. First, $\mathcal{B}$ runs $\mathsf{Setup}() \rightarrow \rho$ and sets $S \leftarrow \perp$ and $I \leftarrow \{\}$. Then, $\mathcal{B}$ runs $\mathcal{A}^O(\rho)$ and answers oracle calls by $\mathcal{A}$ as follows.

Share(I'): If $|I'| \leq T$ and $S = \perp$, do the following. Set $I \leftarrow I'$ and simulate $\mathsf{Share}\langle \rho, 0^L \rangle \rightarrow S$ while giving the control over shareholders I to $\mathcal{A}$ until reboot.

Reshare(I'): If $|I \cup I'| \leq T$ and $S \neq \perp$, do the following. Set $I \leftarrow I'$ and simulate $\mathsf{Reshare}\langle \rho, S \rangle \rightarrow S$ while giving the control over shareholders I to $\mathcal{A}$ until reboot.

Finalize(y): Output y.

By the definition of the secrecy game we observe that $\mathcal{A}$ obtains at most T shares per sharing or resharing. Thus, by the perfect secrecy property of Shamir Secret Sharing [18], the distribution of the message shares and decommitment shares observed by $\mathcal{A}$ in game G_1 is independent of m. Furthermore, by the perfect hiding security of VC, the distribution of the commitments observed by $\mathcal{A}$ is also independent of the m. It follows that for all $m \in \mathcal{M}^L$, $y \in \Im(G_1)$,

$$\Pr[G_1(\mathcal{A}; m) = y] = \Pr[G_1(\mathcal{A}; 0^L) = y] \ . \tag{1}$$

Furthermore, by the definition of $\mathcal{B}$, we have

$$\Pr[G_1(\mathcal{A}; 0^L) = y] = \Pr[\mathcal{B} = y] \ . \tag{2}$$

By the law of total probability, (1), and (2), we obtain

$$\begin{aligned}
&\Pr\begin{bmatrix} F(m) = y : \\ \mathcal{D} \rightarrow m, G_1(\mathcal{A}; m) \rightarrow y \end{bmatrix} \\
&= \sum_{\hat{m} \in \Im(\mathcal{D})} \Pr\begin{bmatrix} F(m) = y : \\ \mathcal{D} \rightarrow m, G_1(\mathcal{A}; m) \rightarrow y, m = \hat{m} \end{bmatrix} * \Pr\begin{bmatrix} m = \hat{m} : \\ \mathcal{D} \rightarrow m \end{bmatrix} \\
&= \sum_{\hat{m} \in \Im(\mathcal{D})} \Pr\begin{bmatrix} F(\hat{m}) = y : \\ G_1(\mathcal{A}; \hat{m}) \rightarrow y \end{bmatrix} * \Pr\begin{bmatrix} m = \hat{m} : \\ \mathcal{D} \rightarrow m \end{bmatrix} \\
&= \sum_{\hat{m} \in \Im(\mathcal{D})} \Pr\begin{bmatrix} F(0^L) = y : \\ G_1(\mathcal{A}; 0^L) \rightarrow y \end{bmatrix} * \Pr\begin{bmatrix} m = \hat{m} : \\ \mathcal{D} \rightarrow m \end{bmatrix} \\
&= \sum_{\hat{m} \in \Im(\mathcal{D})} \Pr\begin{bmatrix} F(0^L) = y : \\ \mathcal{B} \rightarrow y \end{bmatrix} * \Pr\begin{bmatrix} m = \hat{m} : \\ \mathcal{D} \rightarrow m \end{bmatrix} \\
&= \sum_{\hat{m} \in \Im(\mathcal{D})} \Pr\begin{bmatrix} F(m) = y : \\ \mathcal{D} \rightarrow m, \mathcal{B} \rightarrow y, m = \hat{m} \end{bmatrix} * \Pr\begin{bmatrix} m = \hat{m} : \\ \mathcal{D} \rightarrow m \end{bmatrix} \\
&= \Pr\begin{bmatrix} F(m) = y : \\ \mathcal{D} \rightarrow m, \mathcal{B} \rightarrow y \end{bmatrix} \ .
\end{aligned}$$

Finally, we observe that the running time of $\mathcal{B}$ is upper-bounded by the running time of $\mathcal{A}$ times an upper bound α on the running time of protocols Share and $\mathsf{Reshare}$. We obtain that for all $\tau_{\mathcal{A}}$, $\mathcal{A} \in \mathsf{ProbAlgo}(\tau_{\mathcal{A}})$, there exists $\mathcal{B} \in \mathsf{ProbAlgo}(\tau_{\mathcal{B}})$ such that

$$\Pr\left[\begin{matrix} F(m) = y : \\ \mathcal{D} \to m, G_1(\mathcal{A}; m) \to y \end{matrix}\right] \leq \Pr\left[\begin{matrix} F(m) = y : \\ \mathcal{D} \to m, \mathcal{B} \to y \end{matrix}\right] + \epsilon(\tau_{\mathcal{A}}, \tau_{\mathcal{B}}) \ ,$$

for

$$\epsilon(\tau_{\mathcal{A}}, \tau_{\mathcal{B}}) = \begin{cases} 0 & \text{if } \tau_{\mathcal{B}} \geq \alpha * \tau_{\mathcal{A}}, \\ 1 & \text{if } \tau_{\mathcal{B}} < \alpha * \tau_{\mathcal{A}}. \end{cases}$$

□

Theorem 3 (Robustness). *Let* $(N, T) \in \mathbb{N} \times \mathbb{N}_0$ *such that* $N < p$ *and* $T < \frac{N}{2}$. *Let* $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *be a homomorphic vector commitment scheme such that* $\mathcal{M}$ *is a finite field of prime order* p. *If* VC *is* ϵ*-binding, then the proactive secret sharing scheme* $\mathsf{VPSS}_{N,T,\mathsf{VC}}$ *is* ϵ'*-robust with*

$$\epsilon' : \mathbb{N} \to \mathbb{R}; \tau \mapsto \epsilon(\alpha * \tau) \ .$$

Proof. Let $(N, T) \in \mathbb{N} \times \mathbb{N}_0$ such that $N < p$ and $T < \frac{N}{2}$. Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function and $\mathsf{VC} = (L, \mathcal{P}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ be an ϵ-binding homomorphic vector commitment scheme such that $\mathcal{M}$ is a finite field of prime order p. Let $\mathsf{VPSS}_{N,T,\mathsf{VC}} = (N, T, \mathcal{P}, \mathcal{M}, \mathcal{S}, \mathsf{Setup}, \mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct})$, $\tau_{\mathcal{A}} \in \mathbb{N}$, $\mathcal{A} \in \mathsf{ProbAlgo}(\tau_{\mathcal{A}})$, and $m \in \mathcal{M}$.

We construct an algorithm $\mathcal{B}$ such that

$$\Pr\left[\begin{matrix} m \neq m' : \\ G_2(\mathcal{A}, m) \to m' \end{matrix}\right] = \Pr\left[\begin{matrix} b = 1 \wedge b' = 1 \wedge m \neq m' : \\ \mathsf{Setup}() \to \rho, \mathcal{B}(\rho) \to (c, m, d, m', d'), \\ \mathsf{Open}(\rho, m, c, d) \to b, \mathsf{Open}(\rho, m, c, d') \to b' \end{matrix}\right] \ ,$$

which on input $\rho \in \mathcal{P}$, algorithm $\mathcal{B}$ simulates game $G_2(\mathcal{A}, m)$ as follows. When the game is startet run G_2.**Initialize** and replace the output by ρ. When $\mathcal{A}$ calls **Share**(I'), run G_2.**Share**(I') in interaction with $\mathcal{A}$, which controls the corrupted shareholders and denote the output of shareholder i by $s_i = (s_{i,1}, \ldots, s_{i,L}, c_{i,0}, \ldots, c_{i,D}, r_i)$. Then, find $G \in \mathtt{QUALI}((s_1, \ldots, s_N))$ and set $c = c_{i,0}$, for an $i \in G$. For $i \in G$, compute $l_i \leftarrow \prod_{j \in G \setminus \{i\}} j * \mathsf{MODINV}(j - i, p)$, and compute $d \leftarrow \bigcirc_{i \in G} \mathsf{EXP}(r_i, l_i)$. When $\mathcal{A}$ calls **Reshare**, run G_2.**Reshare** in interaction with $\mathcal{A}$. When $\mathcal{A}$ calls **Finalize**, run G_2.**Finalize** in interaction with $\mathcal{A}$ and denote the share sent by shareholder i by $s'_i = (s'_{i,1}, \ldots, s'_{i,L}, c'_{i,0}, \ldots, c'_{i,D}, r'_i)$ and the output of G_2.**Finalize** by m'. Determine a set $G' \in \mathtt{QUALI}((s'_1, \ldots, s'_N))$, for $i \in G'$, compute $l'_i \leftarrow \prod_{j \in G' \setminus \{i\}} j * \mathsf{MODINV}(j - i, p)$, and compute $d' \leftarrow \bigcirc_{i \in G'} \mathsf{EXP}(r'_i, l'_i)$. Output (c, m, d, m', d').

We now derive an upper bound on

$$\Pr\left[\begin{matrix} m \neq m' : \\ G_2(\mathcal{A}, m) \to m' \end{matrix}\right] \ .$$

We observe that by the definition of protocol Share, the properties of the broadcast channel, and because the majority of the shareholders are honest, we have for $i \in [N]$, $\hat{c}_i \leftarrow \bigcirc_{k \in [0,D]} \mathsf{EXP}(c_{i,k}, i^k)$, that $\mathsf{Open}(\rho, (s_{i,1}, \ldots, s_{i,L}), \hat{c}_i, r_i) = 1$. Furthermore, we observe that for $i \in [L]$, $m_i = \bigcirc_{j \in G} \mathsf{EXP}(s_{j,i}, l_j)$,

we have $m = (m_1, \ldots, m_L)$, $d = \bigcirc_{j \in G} \mathsf{EXP}(r_j, l_j)$, and $c = \bigcirc_{i \in G} \mathsf{EXP}(\hat{c}_i, l_i)$. Because VC is homomorphic, it follows that $\mathsf{Open}(\rho, m, c, d) = 1$. Analogously we obtain that $\mathsf{Open}(\rho, m', c', d') = 1$. Furthermore, we observe that by the definitions of protocols $\mathsf{Reshare}$ and $\mathsf{Reconstruct}$, the properties of the broadcast channel, and the honest majority, we have that for all $i \in G'$, $c = c'$. It follows that

$$\Pr\begin{bmatrix} m \neq m' : \\ G_2(\mathcal{A}, m) \rightarrow m' \end{bmatrix} = \Pr\begin{bmatrix} b = 1 \wedge b' = 1 \wedge m \neq m' : \\ \mathsf{Setup}() \rightarrow \rho, \mathcal{B}(\rho) \rightarrow (c, m, d, m', d'), \\ \mathsf{Open}(\rho, m, c, d) \rightarrow b, \mathsf{Open}(\rho, m, c, d') \rightarrow b' \end{bmatrix} .$$

We observe that for any $\mathcal{A}$, the running time of $\mathcal{B}_{\mathcal{A}}$ is upper-bounded by the running time of $\mathcal{A}$ times a constant α. Thus, we obtain that $\mathsf{VPSS}_{N,T,\mathsf{VC}}$ is ϵ'-robust with

$$\epsilon' : \mathbb{N} \rightarrow \mathbb{R}; \tau \mapsto \epsilon(\alpha * \tau) .$$

□

4 Instantiation, Implementation, and Evaluation

We first describe in Subsect. 4.1 how we instantiate the vector commitment scheme that is necessary for our vector proactive secret sharing scheme described in Sect. 3. Afterwards we describe in Subsect. 4.2 how we implemented our vector proactive secret sharing scheme instantiated with the described vector commitment scheme. Finally, we evaluate the performance of our scheme and its implementation in Subsect. 4.3.

4.1 Instantiation

In the following we describe a vector commitment scheme that has the properties required by our vector proactive secret sharing scheme, i.e., it is perfectly hiding, computationally binding, and homomorphic. In addition, it is concise, which means that commitment and decommitment are potentially much shorter then the committed message vector. The construction is an extension of the commitment scheme proposed in [15] and is sometimes referred to by generalized Pedersen commitment [10]. Here we cast the construction into our definition of a vector commitment scheme and show that its security can be based on the fixed generator discrete logarithm problem.

Scheme 2 (DLVC). *Let $\mathbb{G}$ be a finite cyclic group, p be the order of $\mathbb{G}$, $\circ$ denote the operation associated with $\mathbb{G}$, and $L \in \mathbb{N}$. We define the vector commitment scheme $\mathsf{DLVC}_{\mathbb{G},L} = (L, \mathsf{GEN}(\mathbb{G})^L, \mathbb{Z}_p, \mathbb{G}, \mathbb{Z}_p, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ as follows.*

$\mathsf{Setup}() \rightarrow (g_0, \ldots, g_L)$*: For $i \in [0, L]$, sample $U(\mathsf{GEN}(\mathbb{G})) \rightarrow g_i$.*
$\mathsf{Commit}(\rho, m) \rightarrow (c, d)$*: Let $\rho = (g_0, \ldots, g_L)$ and $m = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$. Sample $U(\mathbb{Z}_p) \rightarrow d$ and compute $c \leftarrow \mathsf{EXP}(g_0, d) \bigcirc_{i \in [L]} \mathsf{EXP}(g_i, m_i)$.*

$\mathsf{Open}(\rho, m, c, d) \rightarrow b$: *Let* $\rho = (g_0, \ldots, g_L)$ *and* $m = (m_1, \ldots, m_L)$. *Compute* $c' \leftarrow \mathsf{EXP}(g_0, d) \bigcirc_{i \in [L]} \mathsf{EXP}(g_i, m_i)$. *If* $c = c'$, *set* $b \leftarrow 1$. *If* $c \neq c'$, *set* $b \leftarrow 0$.

Theorem 4. *Let* $\mathbb{G}$ *be a finite cyclic group and* $L \in \mathbb{N}$. *The vector commitment scheme* $\mathsf{DLVC}_{\mathbb{G},L}$ *is correct.*

Theorem 5. *Let* $\mathbb{G}$ *be a finite cyclic group and* $L \in \mathbb{N}$. *The vector commitment scheme* $\mathsf{DLVC}_{\mathbb{G},L}$ *is perfectly hiding.*

Theorem 6. *Let* $\mathbb{G}$ *be a finite cyclic group of prime order* p, $g \in \mathsf{GEN}(\mathbb{G})$, *and* $L \in \mathbb{N}$. *If* $\mathsf{DLOG}(\mathbb{G}, g)$ *is* ϵ*-hard, then there exists* $\alpha \in \mathbb{N}$ *such that* $\mathsf{DLVC}_{\mathbb{G},L}$ *is* ϵ'*-binding with*

$$\epsilon' : \mathbb{N} \rightarrow \mathbb{R}; \tau \mapsto \epsilon(\tau + \alpha) + \frac{1}{p} .$$

Theorem 7. *Let* $\mathbb{G}$ *be a finite cyclic group and* $L \in \mathbb{N}$. *The commitment scheme* $\mathsf{DLVC}_{\mathbb{G},L}$ *is homomorphic.*

The proofs of the theorems can be found in Appendix A.

4.2 Implementation

We implemented a proactive secret sharing system based on the proactive secret sharing scheme VPSS (Subsect. 3.1) instantiated with the vector commitment scheme DLVC (Subsect. 4.1) using the programming language Java 8. In order to support storage of large byte arrays, we use a data encoding that maps byte arrays to message vectors of the secret sharing scheme and then run multiple instances of the scheme per byte array.

System Parameters. Our proactive secret sharing system uses the following parameters:

Number of shareholders N**:** This parameter specifies the total number of shareholders that are involved in the secret sharing protocols.

Corruption threshold T**:** This parameters specifies the maximum number of corrupted shareholders that can be tolerated. We require that $T < \frac{N}{2}$.

Vector length L**:** This parameter specifies the length of the message vectors of the secret sharing scheme and vector commitment scheme.

Message space size M**:** This parameter represents the size in bytes of an element of a message vector for the secret sharing scheme and the vector commitment scheme. The message space size M is determined by the parameters of the commitment scheme and our implementation supports $M \in \{32, 64\}$. We instantiate the commitment space $\mathbb{G}$ as the unique p-order subgroup of $\mathbb{Z}_q$ for primes p and q with $\log_2(p) > M * 8 \geq 256$, $\log_2(q) \geq 2048$, and $(p-1) \bmod q = 0$.

Commitment space size C**:** This parameter represents the size in bytes of commitments and is determined by $C = \lceil \log_2(q)/8 \rceil$.

Data Encoding. We use the following data encoding to map byte arrays to the message space of VPSS. Let $\mathcal{M}^L$ be the message space of the secret sharing scheme. We use the algorithms Encode and Decode for encoding byte arrays of $\mathcal{B} = \{b \in \{0,\ldots,255\}^* : |b| \leq \mathsf{INTMAX}\}$ to message matrices of $\mathcal{M}^{L\times R^*} = \{m \in \mathcal{M}^{L\times *} : \mathsf{Cols}(m) \leq \lceil \frac{R}{L} \rceil, R = \lceil \frac{\mathsf{INTSIZE}+\mathsf{INTMAX}}{M} \rceil\}$, where $\mathsf{INTSIZE} = 4$ and $\mathsf{INTMAX} = 2^{31}-1$ for Java 8. Our byte array encoding requires two other types of encodings: $(\mathsf{Encode}_{\mathsf{Integer},\mathbb{B}^{\mathsf{INTSIZE}}}, \mathsf{Decode}_{\mathsf{Integer},\mathbb{B}^{\mathsf{INTSIZE}}})$ is an encoding from Java Integers to byte arrays of length INTSIZE, which is supported natively by Java, and $(\mathsf{Encode}_{\mathbb{B}^M,\mathcal{M}}, \mathsf{Decode}_{\mathbb{B}^M,\mathcal{M}})$ is an encoding from byte arrays of length M to message space elements of $\mathcal{M} = \mathbb{Z}_p$, for $p \in \mathbb{N}$, which we implement using Java Big Integers.

$\mathsf{Encode}(b \in \mathcal{B}) \rightarrow m \in \mathcal{M}^{L\times R^*}$:

1. Let $\texttt{length} = \mathsf{Encode}_{\mathsf{Integer},\mathbb{B}^{\mathsf{INTSIZE}}}(|b|)$ and set $b' \leftarrow \texttt{length}\|b$.
2. Let $b'' = b'\|0^{|b'| \bmod M}$ and $b'' = a_1\| \ldots \|a_n$ such that for $i \in [n]$, a_i is a byte array of length M.
3. For $i \in [n]$, let $m_i = \mathsf{Encode}_{\mathbb{B}^M,\mathcal{M}}(a_i)$, where $\mathsf{Encode}_{\mathbb{B}^M,\mathcal{M}}$ is an algorithm that encodes elements of $\mathbb{B}^M$
4. Reshape the vector $(m_1,\ldots,m_n) \in \mathcal{M}^n$ into a matrix $m \in \mathcal{M}^{L\times\lceil\frac{n}{L}\rceil}$, that is, let $m = (m_{i,j})_{(i,j)\in[L]\times[\lceil\frac{n}{L}\rceil]}$, where $m_{i,j} = m_k$ for $k = i + (j-1) * L$ and $m_k = 0$ if $k > n$.

$\mathsf{Decode}(m \in \Im(\mathsf{Encode})) \rightarrow b \in \mathcal{B}$:

1. Reshape the matrix $m = (m_{i,j}) \in \mathcal{M}^{L\times L'}$ into vector $(m_1,\ldots,m_n) \in \mathcal{M}^{L*L'}$. That is, for $i \in [L * L']$, let $m_i = m_{j,k}$, where $j = i \bmod k$ and $j = \lfloor \frac{i}{L} \rfloor$.
2. For $i \in [L * L']$, let $b_i = \mathsf{Decode}_{\mathbb{B}^M,\mathcal{M}}(m_i)$.
3. Let $b'' = a_1\| \ldots \|a_{L*L'} = b_1\| \ldots \|b_{L*L'*S}$, where $b_i \in \mathbb{B}$ for $i \in [L*L'*M]$.
4. Let $l = \mathsf{Decode}_{\mathsf{Integer},\mathbb{B}^{\mathsf{INTSIZE}}}(b_1\| \ldots \|b_{\mathsf{INTSIZE}})$.
5. Let $b = b_{\mathsf{INTSIZE}+1}\| \ldots \|b_{\mathsf{INTSIZE}+l}$.

This encoding fulfills the requirement that for all $b \in \mathcal{B}$, $\mathsf{Decode}(\mathsf{Encode}(b)) = b$. In our implementation, we store a byte array $b \in \mathcal{B}$ with $m \leftarrow \mathsf{Encode}(B)$ and $\mathsf{Cols}(m) > 1$ by running for each column of m a separate instance of the secret sharing system.

4.3 Evaluation

In this section we evaluate the theoretical and practical performance of our proactive secret sharing system based on the proactive secret sharing scheme VPSS, the vector commitment scheme DLVC, and the data encoding described in Subsect. 4.2. For the theoretical performance evaluation we distinguish between broadcast communication and direct point-to-point communication. For our experimental performance evaluation we focus on measuring the computation time of the protocols. Practical communication times highly depend on the network infrastructure. Our measurements are for honest executions of the protocols. Protocol runs with malicious parties may take longer as they require additional steps for resolving conflicts.

Table 1. Computation and communication complexity of the protocols Share, Reshare, and Reconstruct of our proactive secret sharing system. **COMP** denotes the computation complexity measured in the number of modular exponentiations for modulus $p \approx 2^{8M}$, **BC-OUT** denotes the outgoing broadcast traffic, **BC-IN** denotes the incoming broadcast traffic, **DIR-OUT** denotes the outgoing directed point-to-point traffic, and **DIR-IN** denotes the incoming directed point-to-point traffic, where the traffic is measured in bytes.

	COMP	BC-OUT	BC-IN	DIR-OUT	DIR-IN
Share	$\lceil \frac{D}{LM} \rceil (N-T) * (L+1)$ for $\mathcal{D}$	$\lceil \frac{D}{LM} \rceil (N-T)C$ for $\mathcal{D}$	$\lceil \frac{D}{LM} \rceil (N-T)C$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil N(L+1)M$ for $\mathcal{D}$	$\lceil \frac{D}{LM} \rceil (L+1)M$ for $\mathcal{S}_i$
Reshare	$\lceil \frac{D}{LM} \rceil (2N-T)(L+1)$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil (2(N-T)+1)C$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil N(2(N-T)+1)C$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil N(L+1)M$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil N(L+1)M$ for $\mathcal{S}_i$
Reconstruct	$\lceil \frac{D}{LM} \rceil (N-T)(L+1)$ for $\mathcal{D}$	0	0	$\lceil \frac{D}{LM} \rceil (L+1)M + (N-T)C$ for $\mathcal{S}_i$	$\lceil \frac{D}{LM} \rceil N((L+1)M + (N-T)C)$ for $\mathcal{D}$

Theoretical Performance. In Table 1 we present the computation and communication complexity of the protocols Share, Reshare, and Reconstruct of our proactive secret sharing system. For the computation complexity, we count the number of modular exponentiations during commitment generation and verification because these typically account for more than 90% of the computation time, as can be seen from the runtime profile of the implementation. For the communication complexity, we count the number of shares and commitments that are transmitted and multiply these counts with the respective sizes of these elements. In Fig. 3 we plot the communication performance as a function of the vector length L. We observe that especially the broadcast communication per party can be drastically reduced by increasing the vector length L. The effect of increasing L on direct communication is noticeable for small L. We observe that in comparison to standard proactive secret sharing (i.e., $L = 1$) our vector proactive secret sharing scheme uses only $\frac{1}{L}$ the communication, that is, for large L the communication complexity is comparable with the optimal communication complexity of standard secret sharing [18].

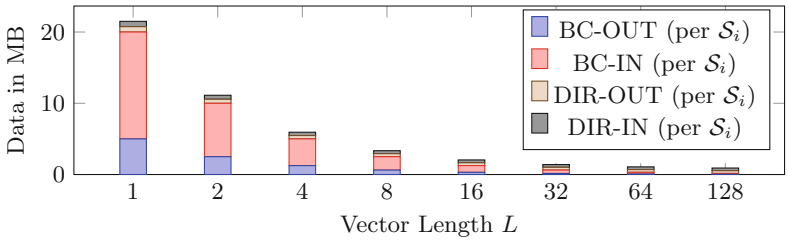

Fig. 3. Network communication during protocol Reshare plotted over the vector length L for $N = 3$, $T = 1$, $D = 128$ kB, $M = 32$ B, where $L = 1$ represents [12].

Experimental Performance. For the experimental performance evaluation we focus on measuring the computation time of the protocols, as practical communication times highly depend on the network infrastructure and would require a more advanced implementation and testbed. In Fig. 4 we show the measured running times for protocols Share, Reshare, and Reconstruct for $M = 32$ and different message vector lengths L. We observe that we reduce the computation time by up to 50% when we increase the vector length L, as predicted by the theoretical complexity evaluation. Increasing the message space size M does not improve performance significantly as modular exponentiations are more expensive for larger M.

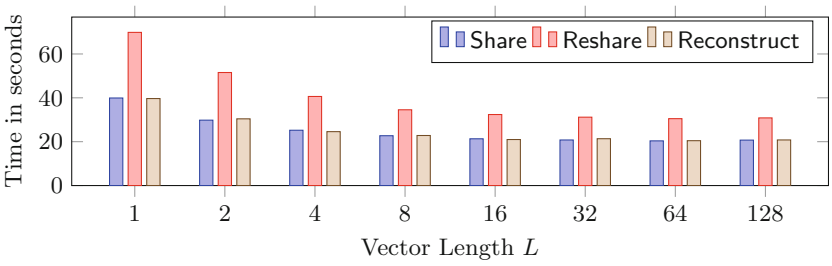

Fig. 4. Measured running times for protocols Share, Reshare, and Reconstruct plotted over the vector length L for $N = 3$, $T = 1$, $D = 128$ kB, $M = 32$ B.

5 Conclusions

We presented a vector proactive secret sharing scheme that allows for drastically reduced communication and computation costs. Concretely, when instantiated with the vector commitment scheme described in Subsect. 4.1 our scheme reduces computation costs by 50% and broadcast communication costs by a factor L, where L is the length of the commitment scheme message vectors, compared to the scheme of [12].

We see several directions for future work. While our scheme achieves almost optimal communication performance, the computation times are still a bottle neck. It would be worthwhile to explore whether there exist suitable vector commitment schemes that are computationally more efficient. Furthermore, the vector commitment scheme used by us is based on the discrete logarithm problem which is susceptible to quantum computer attacks. It would be worthwhile to explore suitable vector commitment schemes that are secure against quantum computers. In [13], Kate, Zaverucha, and Goldberg propose polynomial commitments and show how they can be used to reduce the communication complexity of verifiable secret sharing. However, they do not study the implications for proactive secret sharing. It would be interesting to see whether their techniques

can be combined with our techniques in order to further reduce the communication complexity of our vector proactive secret sharing scheme. Besides that, it would be interesting to extend our scheme to the asynchronous network setting where a global clock is not available to the participating network parties.

Acknowledgments. This work has been co-funded by the DFG as part of project S6 within the CRC 1119 CROSSING.

A Proofs

Proof (Proof of Theorem 4). Let $\mathbb{G}$ be a finite cyclic group, p be the order of $\mathbb{G}$, and $L \in \mathbb{N}$, $\mathsf{DLVC}_{\mathbb{G},L} = (L, \mathsf{GEN}(\mathbb{G})^L, \mathbb{Z}_p, \mathbb{G}, \mathbb{Z}_p, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, and $m = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$.

We observe that for $\mathsf{Setup}() \to \rho$, we have $\rho = (g_0, \ldots, g_L) \in \mathsf{GEN}(\mathbb{G})^L$. Furthermore, we observe that if $\mathsf{Commit}(\rho, m) \to (c, d)$, then $c = \mathsf{EXP}(g_0, d) \circ \bigcirc_{i=1}^{L} \mathsf{EXP}(g_i, m_i)$. It follows that $\mathsf{Open}(\rho, m, c, d) = 1$. □

Proof (Proof of Theorem 5). Let $\mathbb{G}$ be a finite cyclic group associated with operation $\circ$, $L \in \mathbb{N}$, and $\mathsf{DLVC}_{\mathbb{G},L} = (L, \mathsf{GEN}(\mathbb{G})^L, \mathbb{Z}_p, \mathbb{G}, \mathbb{Z}_p, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$. We observe that for all $\rho \in \mathsf{GEN}(\mathbb{G})^L$, $m \in \mathbb{Z}_p^L$, $c^* \in \mathbb{G}$, by the definition of Commit and because g is a generator, we have

$$\begin{aligned}
&\Pr\left[\begin{matrix} c = c^* : \\ \mathsf{Commit}(\rho, m) \to (c, d) \end{matrix}\right] \\
&= \Pr\left[\begin{matrix} c = c^* : \\ U(\mathbb{Z}_p) \to d, c \leftarrow \mathsf{EXP}(g_0, d) \bigcirc_{i=1}^{L} \mathsf{EXP}(g_i, m_i) \end{matrix}\right] \\
&= \Pr\left[\begin{matrix} c = c^* : \\ U(\mathbb{G}) \to c \end{matrix}\right],
\end{aligned}$$

where $\rho = (g_0, \ldots, g_L)$ and $m = (m_1, \ldots, m_L)$. □

Proof (Proof of Theorem 6). The following proof is adapted from Section 2.3.2 of [6].

Let $\mathbb{G}$ be a finite cyclic group of prime order p, $\mathsf{DLVC}_{\mathbb{G},L} = (L, \mathsf{GEN}(\mathbb{G})^L, \mathbb{Z}_p, \mathbb{G}, \mathbb{Z}_p, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, $g \in \mathsf{GEN}(\mathbb{G})$, $\tau \in \mathbb{N}$, and $\mathcal{A} \in \mathsf{Algo}(\tau)$. In the following, we prove an upper bound on

$$\Pr\left[\begin{matrix} \mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' : \\ \mathsf{Setup}() \to \rho, \mathcal{A}(\rho) \to (c, m, d, m', d') \end{matrix}\right].$$

Let $\mathcal{B}$ be an algorithm that takes as input $y \in \mathbb{G}$ and works as follows. Sample $U(\mathbb{Z}_p) \to a_0$ and for $i \in [L]$, $U(\mathbb{Z}_p^2) \to (a_i, b_i)$. Compute $g_0 \leftarrow \mathsf{EXP}(g, a_0)$ and for $i \in [L]$, $g_i \leftarrow \mathsf{EXP}(g, a_i) \circ \mathsf{EXP}(y, b_i)$. Run $\mathcal{A}((g_0, \ldots, g_L)) \to (c, m, d, m', d')$. If $\mathsf{Open}(\rho, m, c, d) = 0$ or $\mathsf{Open}(\rho, m', c, d') = 0$, output $\perp$. Otherwise, proceed as follows. Let $m = (m_0, \ldots, m_L) \in \mathbb{Z}_p^L$ and $m' = (m'_0, \ldots, m'_L) \in \mathbb{Z}_p^L$. Compute $a \leftarrow a_0(d - d') + \sum_{i \in [L]} a_i(m_i - m'_i)$ and $b \leftarrow \sum_{i \in [L]} b_i(m'_i - m_i)$. If $b = 0$, output $\perp$. Otherwise, compute $x \leftarrow \frac{a}{b}$ and output x.

We observe that because the a_i's are uniformly distributed and g is a generator, the g_i's are also uniformly distributed. This means that $(g_0, \ldots, g_L)$ has the same distribution as ρ generated by $\mathsf{Setup}()$. It follows that

$$\Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' : \\ \mathsf{Setup}() \to \rho, \mathcal{A}(\rho) \to (c, m, d, m', d')\end{matrix}\right] = \Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] .$$

Using sigma additivity we write

$$\begin{aligned}&\Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] \\ &= \Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' \wedge b = 0 : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] \\ &\quad + \Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' \wedge b \neq 0 : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] .\end{aligned}$$

The first term is upper-bounded by $\frac{1}{p}$, as can be seen as follows:

$$\begin{aligned}&\Pr\left[\begin{matrix}\mathsf{Open}(\rho, m, c, d) = 1 \wedge \mathsf{Open}(\rho, m', c, d') = 1 \wedge m \neq m' \wedge b = 0 : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] \\ &\leq \Pr\left[\begin{matrix}m \neq m' \wedge b = 0 : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] \\ &= \Pr\left[\begin{matrix}m \neq m' \wedge \sum_{j\in[L]} b_j(m'_j - m_j) = 0 : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] \\ &\leq \Pr\left[\begin{matrix}\exists i \in [L], m_i \neq m'_i \wedge b_i = \frac{-\sum_{j\in[L]\setminus\{i\}} b_j(m'_j - m_j)}{(m'_i - m_i)} : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] = \frac{1}{p} .\end{aligned}$$

Next we prove that the second term is upper-bounded by

$$\Pr\left[\begin{matrix}\mathsf{EXP}(g, x) = y : \\ U(\mathbb{G}) \to y, \mathcal{B}(y) \to x\end{matrix}\right] .$$

We observe that if $b \neq 0$, then $\mathsf{Open}(\rho, m, c, d) = 1$, $\mathsf{Open}(\rho, m', c, d') = 1$, $m \neq m'$, and

$$\begin{aligned}&\mathsf{EXP}(g_0, d) \bigcirc_{i\in[L]} \mathsf{EXP}(g_i, m_i) = \mathsf{EXP}(g_0, d') \bigcirc_{i\in[L]} \mathsf{EXP}(g_i, m'_i) \\ &\iff \mathsf{EXP}(g_0, d - d') \bigcirc_{i\in[L]} \mathsf{EXP}(g_i, m_i - m'_i) = e_{\mathbb{G}} \\ &\iff \mathsf{EXP}\left(g, a_0(d - d') + \sum_{i\in[L]} a_i(m_i - m'_i)\right) \circ \mathsf{EXP}\left(y, \sum_{i\in[L]} b_i(m_i - m'_i)\right) = e_{\mathbb{G}} \\ &\iff \mathsf{EXP}\left(g, a_0(d - d') + \sum_{i\in[L]} a_i(m_i - m'_i)\right) = \mathsf{EXP}\left(y, \sum_{i\in[L]} b_i(m'_i - m_i)\right) \\ &\iff \mathsf{EXP}\left(g, \frac{a}{b}\right) = y .\end{aligned}$$

It follows that

$$\Pr\begin{bmatrix}\mathsf{EXP}(g,\frac{a}{b})=y\wedge b\neq 0:\\ U(\mathbb{G})\to y,\mathcal{B}(y)\to x\end{bmatrix}$$
$$=\Pr\begin{bmatrix}\mathsf{Open}(\rho,m,c,d)=1\wedge\mathsf{Open}(\rho,m',c,d')=1\wedge m\neq m'\wedge b\neq 0:\\ U(\mathbb{G})\to y,\mathcal{B}(y)\to x\end{bmatrix}.$$

By the fact that $b=0$ implies $x=\perp$ and $\mathsf{EXP}(g,\perp)\notin\mathbb{G}$, we have

$$\Pr\begin{bmatrix}\mathsf{EXP}(g,x)=y\wedge b\neq 0:\\ U(\mathbb{G})\to y,\mathcal{B}(y)\to x\end{bmatrix}=\Pr\begin{bmatrix}\mathsf{EXP}(g,x)=y:\\ U(\mathbb{G})\to y,\mathcal{B}(y)\to x\end{bmatrix}.$$

In summary, we obtain

$$\Pr\begin{bmatrix}\mathsf{Open}(\rho,m,c,d)=1\wedge\mathsf{Open}(\rho,m',c,d')=1\wedge m\neq m':\\ \mathsf{Setup}()\to\rho,\mathcal{A}(\rho)\to(c,m,d,m',d')\end{bmatrix}$$
$$\leq\Pr\begin{bmatrix}\mathsf{EXP}(g,x)=y:\\ U(\mathbb{G})\to y,\mathcal{B}(y)\to x\end{bmatrix}+\frac{1}{p}.$$

Finally, we observe that the running time of $\mathcal{B}$ is upper-bounded by $\tau+\alpha$, where α is the constant difference between the running time of $\mathcal{B}$ and the running time of $\mathcal{A}$. It follows that if $\mathsf{DLOG}(\mathbb{G},g)$ is ϵ-hard, then $\mathsf{DLVC}_{\mathbb{G},L}$ is ϵ'-binding-secure with

$$\epsilon':\tau\mapsto\epsilon(\tau+\alpha)+\frac{1}{p}.$$

□

Proof (Proof of Theorem 7). Let $\mathbb{G}$ be a finite cyclic group, $L\in\mathbb{N}$, and $\mathsf{DLVC}_{\mathbb{G},L}=(L,\mathsf{GEN}(\mathbb{G})^L,\mathbb{Z}_p,\mathbb{G},\mathbb{Z}_p,\mathsf{Setup},\mathsf{Commit},\mathsf{Open})$. Let $\circ$ denote the operation associated with $\mathbb{G}$, $+$ and $*$ denote addition and multiplication over $\mathbb{Z}_p$, and $\oplus$ denote addition over $\mathbb{Z}_p^L$. We observe that for any $\rho\in\mathcal{P}$, $(m_1,c_1,d_1)\in\mathtt{COMS}(\rho)$, and $(m_2,c_2,d_2)\in\mathtt{COMS}(\rho)$ we have that

$$\begin{aligned}
&(m_1,c_1,d_1)\in\mathtt{COMS}(\rho)\wedge(m_2,c_2,d_2)\in\mathtt{COMS}(\rho)\\
\implies\ &\big(\mathsf{EXP}(g_0,d_1)\bigcirc_{i\in[L]}\mathsf{EXP}(g_i,m_{1,i})\big)=c_1\\
&\wedge\big(\mathsf{EXP}(g_0,d_2)\bigcirc_{i\in[L]}\mathsf{EXP}(g_i,m_{2,i})\big)=c_2\\
\implies\ &\big(\mathsf{EXP}(g_0,d_1)\bigcirc_{i\in[L]}\mathsf{EXP}(g_i,m_{1,i})\big)\\
&\circ\big(\mathsf{EXP}(g_0,d_2)\bigcirc_{i\in[L]}\mathsf{EXP}(g_i,m_{2,i})\big)=c_1\circ c_2\\
\iff\ &\mathsf{EXP}(g_0,d_1+d_2)\bigcirc_{i\in[L]}\mathsf{EXP}(g_i,m_{1,i}+m_{2,i})=c_1\circ c_2\\
\iff\ &\mathsf{Open}(\rho,m_1\oplus m_2,c_1*c_2,d_1\circ d_2)=1.
\end{aligned}$$

□

References

1. Baron, J., Defrawy, K.E., Lampkins, J., Ostrovsky, R.: Communication-optimal proactive secret sharing for dynamic groups. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 23–41. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_2

2. Baron, J., El Defrawy, K., Lampkins, J., Ostrovsky, R.: How to withstand mobile virus attacks, revisited. In: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing, PODC 2014, pp. 293–302. ACM, New York (2014). https://doi.org/10.1145/2611462.2611474. http://doi.acm.org/10.1145/2611462.2611474
3. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004). https://eprint.iacr.org/2004/331
4. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
5. Blakley, G.R.: Safeguarding cryptographic keys. In: International Workshop on Managing Requirements Knowledge (AFIPS), December 1979. https://doi.org/10.1109/AFIPS.1979.98. doi.ieeecomputersociety.org/10.1109/AFIPS.1979.98
6. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
7. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 88–97. ACM, New York (2002). https://doi.org/10.1145/586110.586124. http://doi.acm.org/10.1145/586110.586124
8. 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
9. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984). https://doi.org/10.1016/0022-0000(84)90070-9. http://www.sciencedirect.com/science/article/pii/0022000084900709
10. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_12
11. Gupta, V.H., Gopinath, K.: g^2_{its} VSR: an information theoretical secure verifiable secret redistribution protocol for long-term archival storage. In: Fourth International IEEE Security in Storage Workshop, pp. 22–33, September 2007. https://doi.org/10.1109/SISW.2007.11
12. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: how to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_27
13. 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
14. Nikov, V., Nikova, S.: On proactive secret sharing schemes. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 308–325. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30564-4_22
15. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9

16. Sadeghi, A.-R., Steiner, M.: Assumptions related to discrete logarithms: why subtleties make a real difference. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 244–261. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_16
17. Schultz, D., Liskov, B., Liskov, M.: MPSS: mobile proactive secret sharing. ACM Trans. Inf. Syst. Secur. **13**(4), 341–3432 (2010). https://doi.org/10.1145/1880022.1880028. http://doi.acm.org/10.1145/1880022.1880028
18. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176. http://doi.acm.org/10.1145/359168.359176
19. Wong, T.M., Wang, C., Wing, J.M.: Verifiable secret redistribution for archive systems. In: Proceedings of the First International IEEE Security in Storage Workshop, pp. 94–105, December 2002. https://doi.org/10.1109/SISW.2002.1183515
20. Zhou, L., Schneider, F.B., Van Renesse, R.: APSS: proactive secret sharing in asynchronous systems. ACM Trans. Inf. Syst. Secur. **8**(3), 259–286 (2005). https://doi.org/10.1145/1085126.1085127. http://doi.acm.org/10.1145/1085126.1085127