Panetière: Efficient Anonymous Broadcast

Nils Fleischhacker[2], Mateusz Morusiewicz[1], and Mark Simkin[1,3]

[1] Flashbots
[2] Ruhr University Bochum
[3] Aarhus University

1 Introduction

In this document, we present a new anonymous broadcast protocol optimized for low latency and high anonymity. This is a draft of a paper that will appear online in the very near future. The few missing security proofs will soon be added and minor technical issues addressed.

1.1 Technical Overview

In the following, let us provide an overview of the system architecture underlying Panetière and the main technical ideas behind our construction.

System Overview. Our system is comprised of clients and servers. Clients hold messages they want to publish and servers provide the service through which they can do so anonymously. We assume there are ρ clients, each running their part of the protocol inside a TEE, thereby ensuring that clients follow the protocol faithfully. We assume there are γ servers of which at most $\gamma/2 - 1$ are arbitrarily corrupt. At the end of the protocol execution, we would like to guarantee that all honest clients' messages get published and that the adversary is not able to tell which honest client published which honest message. Importantly, we require that a client breaking their TEE can result in a liveness failure of the protocol, but it cannot result in a privacy failure, i.e. it should not allow a malicious client to deanonymize any of the other clients. For the rest of this overview, we assume that clients follow their protocol specifications honestly.

Secure Vector Addition. The starting point of our work, is the standard secure addition protocol from multiparty computation [BGW88]. Assuming client i holds input m_i, secure addition allows the clients to jointly compute $\sum_{i=1}^{\rho} m_i$ without revealing any information beyond the sum itself. Secure addition can be constructed from any linear threshold secret sharing scheme. Each client i secret shares their message m_i into shares $s_1^i, \ldots, s_\gamma^i$ and sends s_j^i to server j, who reveals $\sum_{i=1}^{\rho} s_j^i$. Since the secret sharing is linear, it holds that $\left(\sum_{i=1}^{\rho} s_1^i, \ldots, \sum_{i=1}^{\rho} s_\gamma^i\right)$ is a secret sharing of the sum of messages and thus the sum can be reconstructed from sufficiently many of the servers' messages. Naturally, the above protocol can be extended to the setting, where clients hold vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\rho$ instead of individual messages by simply performing secure addition component-wise.

To obtain a simple anonymous broadcast protocol, each client i locally computes vector $\boldsymbol{v}_i$, which is zero everywhere apart from a random position at which the vector contains m_i. The parties perform secure vector addition and assuming no two messages were accidentally placed in the same position in two different vectors, we can extract all messages from the sum of vectors. We show how to deal with collisions later, so for now let us ignore this issue.

Assuming all parties follow the protocol, we correctly compute the sum of all vectors and since the location of each message in the vector is chosen at random, no adversarial observer can determine which client sent which message.

The above protocol idea as is, whoever, has two main issues. First, since the sum of vectors contains ρ distinct messages, the vector length must be at least ρ and thus we need to perform at least ρ many secure additions, resulting in a prohibitively large communication overhead. Second, we assumed that each server reveals the sum of their received shares, but if a server is malicious, they may publish arbitrary malformed messages, thereby potentially preventing the protocol from computing the correct output.

Let us first deal with the former and then take care of the latter.

Reducing Bandwidth Overhead. To reduce the bandwidth overhead, we make use of a primitive called key-additively homomorphic encryption [AIKW13, BCGL+25] in combination with ideas similar to those of Karthikeyan and Polychroniadou [KP25] as well as those of Bell-Clark et al. [BCGL+25]. A key-additively homomorphic encryption scheme is a symmetric key encryption schemes that is additively homomorphic on both the key and message space, i.e. for their encryption algorithm Enc, it holds that $\mathsf{Enc}(k_1, m_1) + \mathsf{Enc}(k_2, m_2) = \mathsf{Enc}(k_1 + k_2, m_1 + m_2)$.

Now instead of performing secure addition directly on the long vectors, we let each party publish an encryption of their vector and only perform secure addition on the corresponding short encryption keys. That is, we let each client i, pick a key k_i, publish $\mathsf{ct}_i = \mathsf{Enc}(k_i, \boldsymbol{v}_i)$ and secret share k_i among the servers. The servers jointly reveal $k = \sum_{i=1}^{\rho} k_i$, which allows for decrypting $\sum_{i=1}^{\rho} \mathsf{ct}_i$. Invoking a security property called addition privacy of the underlying encryption scheme, we can guarantee that the sum of keys can decrypt *only* the sum of the ciphertexts. The bandwidth overhead is now only comprised of each client publishing an encryption of their vector $\boldsymbol{v}_i$ and the bandwidth overhead for performing secure addition on the keys.

Dealing with Malicious Servers. To prevent the servers from misbehaving, we need a way to ensure that they reveal the correct sum of their key shares. Towards this goal, we develop a cryptographic object which we call additive vector commitments with addition hiding. These commitments are additive in the sense that the sum of two commitments is a commitment to the sum of the two individually committed vectors. A bit more formally, say c_1 and c_2 are commitments to vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, then $c_1 + c_2$ is a commitment to the vector $\boldsymbol{s}_1 + \boldsymbol{s}_2$. We say a vector commitment is addition hiding, if for any position j, the sum of openings for position j in c_1 and c_2 only allows opening the commitment $c_1 + c_2$ at position j, but reveals nothing else about what was individually committed in c_1 or c_2 at that position.

With such commitments at our disposal, we extend our anonymous broadcast approach from above as follows: When client i secret shares their key k_i into shares $\boldsymbol{s}_i = (s_1, \ldots, s_\gamma)$ during the secure addition protocol, we let the client publicly commit to $\boldsymbol{s}_i$ and provide each server j with the opening of the commitment to at position j. Now the servers can reveal the sum of shares as they are supposed to and additionally they can provide an opening to that sum, which verifies against the sum of the clients' commitments, thereby preventing the servers from misbehaving. Addition hiding of the commitment scheme ensures that only the sum of the individual keys is revealed and nothing else.

Constructing the desired vector commitment scheme turns out to be rather technically involved. While the construction itself looks like a simple combination of the non-hiding vector commitment

of Fleischhacker et al. [FHSZ23a] with a variant of the hiding commitment of Baum et al. [BDL+18], proving that it is addition hiding is quite subtle, especially when practically useful, concretely small parameters are needed.

Dealing with Collisions. Conceptually, the currently outlined construction allows for securely computing the sum of arbitrary long vectors in a communication efficient manner, even when some of the servers are malicious. We turn this into an anonymous broadcast by letting each client i place their message into a random location of a vector $\boldsymbol{v}_i$ that is zero in all other positions. Assuming that by chance no two clients choose the same vector position, then $\sum_{i=1}^{\rho} \boldsymbol{v}_i$ allows recovering all messages. Note that for anonymity, it is important that each client chooses their position independently and uniformly at random, to ensure that the message location in the output does not reveal the sender of the message.

The probability that no two clients place their message in the same location, however, is low even when making the vectors very long. Towards addressing this issue, there are two approaches. The first heuristic approach is to pick the vectors slightly larger than the number of clients ρ, say vectors of length 2ρ and argue that a large constant fraction of all messages will not collide and thus that during each execution of the anonymous broadcast, most messages are delivered. In a setting, where the anonymous broadcast protocol is repeatedly executed, each message will be delivered after an expected constant number of protocol executions.

A second, slightly less efficient approach that recovers *all* messages with very high probability in every single broadcast execution, but leads to vectors of length around 3ρ, is to make the vectors $\boldsymbol{v}_i$ be slightly more complex encodings of the message m_i, such that even when collisions occur, we can extract all messages. The encodings we use in this approach are based on invertible bloom filters [GM11, FLOS24].

Protocol Extensions. To further improve performance of our anonymous broadcast protocol, we consider two extensions. The main efficiency bottleneck of our protocol is the bandwidth overhead incurred from each client needing to disseminate their encrypted vector. Any third party that wants to recover the output of the anonymous broadcast, needs to download all encrypted vectors as well as the opened sum of keys along with their commitments. Both of our optimizations are concerned with reducing the overhead for downloading the large encrypted vectors.

The first optimization is to introduce a third role into the system, called the aggregator. There may be one or multiple aggregators and we place no trust assumptions on them. For liveness, we only require one aggregator to faithfully do their task. Each aggregator obtains all encrypted vectors and publishes their sum. A third party that wants to recover the output from the broadcast protocol, can now simply directly obtain the sum of encrypted vectors from the aggregator, rather than computing it themselves. Should an aggregator misbehave, anybody can verify this by simply computing the sum of encrypted vectors themselves and thereby exclude the dishonest aggregator from their set of trusted aggregators.

The second optimization is to make use of cover traffic. Naively, the length of each individually encrypted vector is proportional to the total number of messages that will be published in a anonymous broadcast protocol execution. For anonymity, we would like that number to be large, but for efficiency, we would like that number to be small. Using cover traffic, smaller sets of clients can send messages, thereby reducing the length of each encrypted vector, while still enjoying a large anonymity set, assuming that parties without a message send cover traffic in the form of an

encryption of the all zero vector. One way of selecting smaller sets of clients, is to let each client internally flip a random coin and send a message when it comes out heads and send cover traffic when it comes out tails. This slightly increases the expected latency of each client, but reduces the communication complexity per broadcast execution.

2 Preliminaries

Let us recall some basic notation and objects that we will be using throughout the paper.

2.1 Notation.

We write $[n]$ to denote the set $\{1, \ldots, n\}$. For a set S, we write $x \leftarrow S$ to denote the process of sampling a random element and assigning it to x. We write $y \leftarrow A(x)$ to denote the process of running a possibly randomized algorithm A on input x and assigning the output to y. Unless otherwise stated logarithms are base 2, i.e. $\log x$ is the logarithm of x with base 2.

For notational convenience, we define $\mathsf{filter}, \mathsf{merge}, \mathsf{zip}$ and unzip functions below which will simplify notation when working with sets and vectors later on in the paper.

$$\mathsf{filter}(\boldsymbol{s}, C) := \boldsymbol{s'} \qquad s'_i := \begin{cases} \perp & i \notin C \\ s_i & i \in C \end{cases}$$

$$\mathsf{merge}(\boldsymbol{s}, \boldsymbol{s'}, C) := \boldsymbol{s''} \qquad s''_i := \begin{cases} s_i & i \notin C \\ s'_i & i \in C \end{cases}$$

$$\mathsf{zip}(\boldsymbol{u}, \boldsymbol{v}) := ((u_1, v_1), \ldots, (u_\ell, v_\ell))^\intercal$$

$$\mathsf{unzip}(((u_1, v_1), \ldots, (u_\ell, v_\ell))^\intercal) := ((u_1, \ldots, u_\ell)^\intercal, (v_1, \ldots, v_\ell)^\intercal)$$

2.2 Lattices

Our lattice based construction works over a power-of-two cyclotomic polynomial ring. Let $\Phi_{2n} = X^n + 1$ be the cyclotomic polynomial with n a power of two. We work in the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1 \rangle$. For the purpose of taking norms and transmitting data, we represent elements of $\mathcal{R}$ as n-dimensional vectors $\mathbb{Z}^n$ with $(c_1, \ldots, c_n)^\intercal \in \mathbb{Z}^n$ representing the ring element $\sum_{i=1}^{n} c_i X^{i-1}$. For any odd prime number q, we always represent $\mathbb{Z}_q$ by the set $\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$ centered around 0. For an odd prime q, we denote by $\mathcal{R}_q$ the quotient ring of $\mathcal{R}$ modulo q, represented by vectors in $\mathbb{Z}_q^n$. Whenever we need to take representatives to view these as elements from $\mathcal{R}$, we do so by taking representatives centered around 0 as above. For efficiency reasons, our parameter choice will always satisfy $q \equiv 1 \bmod 2n$, so we can use more efficient NTT-based multiplication in $\mathcal{R}_q$.

Let $c \in \mathcal{R}$ be a ring element with coefficients $(c_1, \ldots, c_n)$. Unless specified otherwise, we always work with the $\|.\|_\infty$-norm. We define $\|c\| := \|c\|_\infty = \max_i |c_i|$ and $\|c\|_1 = \sum_i |c_i|$ on $\mathcal{R}$ by taking the norm of the coefficient vector in the monomial basis. We extend these definitions to norms on $\mathcal{R}_q$ by taking representatives in $\mathcal{R}$ using coefficients in $\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$. We also extend the definition of $\|.\|_\infty$ to $\mathcal{R}^m$ for any m by defining $\|\boldsymbol{v}\|_\infty = \max_i \|v_i\|_\infty$ for any $\boldsymbol{v} = (v_1, \ldots, v_m)^\intercal \in \mathcal{R}^m$.

Our convention is that mixed multiplication of an element from $\mathcal{R}$ with an element from $\mathcal{R}_q$ gives an element from $\mathcal{R}_q$, thereby viewing $\mathcal{R}_q$ as an $\mathcal{R}$-module. We denote by $\mathcal{B}_{\beta,q}$ the ball $\mathcal{B}_{\beta,q} = \{a \in \mathcal{R}_q \mid \|a\| \leq \beta\}$. We are only interested in the case $\beta < \frac{q}{2}$.

The security of our constructions relies on the hardness of the short integer solution problem defined over rings as follows.

Definition 1 (Ring Short Integer Solution Problem). *For a ring $\mathcal{R}$ and parameters $\mu, q, \beta \in \mathbb{N}$, the $\mathsf{SIS}_{\mathcal{R},q,\mu,\beta}$ problem is hard if for all PPT algorithms $\mathcal{A}$ it holds that*

$$\Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^{\mu}; \boldsymbol{s} \leftarrow \mathcal{A}(\boldsymbol{a}) \colon \boldsymbol{s} \in \mathcal{B}_{\beta,q}^{\mu} \setminus \{\boldsymbol{0}\} \wedge \boldsymbol{a}^{\intercal}\boldsymbol{s} = 0] \leq \mathsf{negl}(\lambda).$$

We denote by $\mathcal{D}_\sigma$ the distribution over $\mathcal{R}$, such that each coefficient is independently distributed according to a discrete gaussian distribution with standard deviation σ.

The following lemma states that the sum of discrete gaussian distributions is statistically close to a discrete gaussian distribution and follows from [GMPW20, Corollary 4.8].

Lemma 1. *If $\sigma \geq \frac{\eta_{\epsilon(\lambda)}(\mathbb{Z}^n)}{\sqrt{\pi}}$ for some $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, then $\mathsf{SD}(\mathcal{D}_\sigma + \mathcal{D}_\sigma, \mathcal{D}_{\sqrt{2}\sigma}) \leq \mathsf{negl}(\lambda)$.*

Lemma 2. *Let $\boldsymbol{e} \leftarrow \mathcal{D}_\sigma^{\mu}$, then*

$$\Pr[\|\boldsymbol{e}\| \leq k\sigma] \geq \left(1 - 2\exp\left(-\frac{k^2}{2}\right)\right)^{n\mu}$$

Proof. The tail bound follows from the law of total probability and by applying [Lyu11, Lemma 4.4] to each coefficient of each polynomial independently. □

The following lemma follows immediately as a special case of [LS18, Corollary 1.2].

Lemma 3. *Let n be a power of 2 and let $q \equiv 5 \pmod 8$ be prime. Then any $y \in \mathcal{R}_q$ with $0 < \|y\| < \sqrt{\frac{q}{2}}$ is invertible in $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$.*

Below we prove that Ajtai's hash function [Ajt99] over specific rings is universal for short inputs.

Lemma 4. *Let n, m, q, β be positive integers such that n is a power of two, $q \equiv 5 \pmod 8$ is prime, and $2\beta < \sqrt{\frac{q}{2}}$. Let $\mathcal{R}_q$ be the polynomial ring $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$. Then the family of hash functions $\mathcal{H} := \{h_{\boldsymbol{a}}\}_{\boldsymbol{a} \in \mathcal{R}_q}$ with*

$$\begin{aligned} h_{\boldsymbol{a}} : \mathcal{B}_{\beta,q}^{m} &\to \mathcal{R}q \\ \boldsymbol{r} &\mapsto \boldsymbol{a}^{\intercal}\boldsymbol{r} \end{aligned}$$

is universal.

Proof. Let $\boldsymbol{r} \neq \boldsymbol{r}'$ be fixed but arbitrary. Let j be an arbitrary index, such that $r_j \neq r'_j$. Then

$$0 < \|(r_j - r'_j)\| \leq \|r_j\| + \|r'_j\| \leq 2\beta < \sqrt{\frac{q}{2}}$$

and by Lemma 3 $(r_j - r'_j)$ is therefore invertible in $\mathcal{R}_q$. Thus, we have

$$\Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^{m} : \boldsymbol{a}^{\intercal}\boldsymbol{r} = \boldsymbol{a}^{\intercal}\boldsymbol{r}']$$

$$
\begin{aligned}
&= \Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^m : \boldsymbol{a}^\intercal(\boldsymbol{r} - \boldsymbol{r}') = 0] \\
&= \Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^m : \sum_{i \in [m]} a_i(r_i - r_i') = 0] \\
&= \Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^m : a_j(r_j - r_j') = -\sum_{i \in [m] \setminus \{j\}} a_i(r_i - r_i')] \\
&= \Pr[\boldsymbol{a} \leftarrow \mathcal{R}_q^m : a_j = -(r_j - r_j')^{-1} \sum_{i \in [m] \setminus \{j\}} a_i(r_i - r_i')] \\
&= |\mathcal{R}_q|^{-1}
\end{aligned}
$$

as required. □

2.3 Statistical Distance and Entropy

Some of our proofs will make use of the leftover hash lemma. Below we recall the required mathematical concepts and the generalized version of the lemma that we will make use of.

Definition 2 (Statistical Distance). *For two distributions X, X' both defined over the same set $\mathcal{X}$, we define the statistical distance as*

$$\mathsf{SD}(X, X') := \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr[X = x] - \Pr[X' = x]|.$$

For joint distributions $(X, Y), (X', Y)$ we also write

$$\mathsf{SD}(X, X' \mid Y) := \mathsf{SD}((X, Y), (X', Y))$$

for shortness.

Definition 3 (Conditional Min-Entropy). *Let (X, Z) be a joint distribution on $\mathcal{X} \times \mathcal{Z}$. The conditinal min-entropy of X given Z is defined as*

$$\mathbf{H}_\infty(X \mid Z) := -\log \sum_{z \in \mathcal{Z}} \Pr[Z = z] \max_{x \in \mathcal{X}} \Pr[X = x \mid Z = z].$$

Lemma 5 (Generalized Leftover Hash Lemma [DORS08]). *Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be sets. Let U be the uniform distribution over $\mathcal{Y}$ and let (X, Z) be a joint distribution on $\mathcal{X} \times \mathcal{Z}$. Let $\mathcal{H} = h : \mathcal{X} \to \mathcal{Y}$ be a universal hash family, and let H be a random member of $\mathcal{H}$. Then it holds that*

$$\mathsf{SD}(H(X), U \mid H, Z) \leq \frac{1}{2} \sqrt{2^{-\mathbf{H}_\infty(X|Z) + \log |Y|}}$$

2.4 Threshold Secret Sharing

A t-out-of-γ threshold secret sharing scheme allows splitting a secret message m into shares $s_1, \ldots, s_\gamma$, such that any t shares allow for reconstructing m, while any subset of less than t shares reveals no information about m.

Definition 4 (Threshold Secret Sharing Scheme). *Let $t, \gamma \in \mathbb{N}$ with $t \leq \gamma$. A t-out-of-γ threshold secret sharing scheme over finite field $\mathbb{F}$ is a pair $(\mathsf{Share}, \mathsf{Rec})$ of efficient algorithms, which are defined as follows:*

- $\mathsf{Share}(m) \to (s_1, \ldots, s_\gamma)$*: The randomized sharing algorithm takes a message $m \in \mathbb{F}$ as input and returns shares $s_1, \ldots, s_\gamma \in \mathbb{F}$.*
- $\mathsf{Rec}(s_1, \ldots, s_\gamma) \to m$*: The deterministic reconstruction algorithm takes shares $s_\gamma, \ldots, s_\gamma \in \mathbb{F} \cup \{\bot\}$ as input and returns a message $m \in \mathbb{F} \cup \{\bot\}$.*

Definition 5 (Correctness). *Let $t, \gamma \in \mathbb{N}$ with $t \leq \gamma$. We say a t-out-of-γ threshold secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ over $\mathbb{F}$ is correct, if for all $m \in \mathbb{F}$ and all $T \subset [\gamma]$ with $|T| \geq t$, it holds that*

$$\Pr\left[\mathsf{Rec}(\mathsf{Share}(m)_T) = m\right] = 1.$$

Definition 6 (Privacy). *Let $t, \gamma \in \mathbb{N}$ with $t \leq \gamma$. We say a t-out-of-γ threshold secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ over $\mathbb{F}$ is perfectly private, if for all adversaries $\mathcal{A}$ and all $m_0, m_1 \in \mathbb{F}$, and all $T \subset [\gamma]$ with $|T| < t$, it holds that*

$$\Pr\left[\begin{matrix} b \leftarrow \{0,1\} \\ b = b' : \boldsymbol{s} \leftarrow \mathsf{Share}(m_b) \\ b' \leftarrow \mathcal{A}(\boldsymbol{s}_T) \end{matrix}\right] = 1/2,$$

where the probability is taken over the random coins of the experiment and the adversary.

Definition 7 (Linearity). *Let $t, n \in \mathbb{N}$ with $t \leq n$. We say a t-out-of-n threshold secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ over $\mathbb{F}$ is linear, if for all all $m_0, m_1 \in \mathbb{F}$, and all $T \subset [\gamma]$ with $|T| \geq t$, it holds that*

$$\mathsf{Share}(m_0) + \mathsf{Share}(m_1)$$

and

$$\mathsf{Share}(m_0 + m_1)$$

are identically distributed.

The arguably most famous threshold secret sharing scheme is due to Shamir [Sha79]. To secret share a message m, it picks a uniformly random polynomial f of degree $t-1$ with $f(0) = m$ and defines the i share to be $f(i)$.

Theorem 6 ([Sha79]). *Let $t, \gamma, q \in \mathbb{N}$ with $t \leq \gamma$ and q prime with $q > \rho$. Then there exists a correct, private, and linear t-out-of-γ threshold secret sharing over $\mathbb{Z}_q$.*

Naturally, we can extend Shamir's scheme to secret share vectors of finite field elements and thus also elements of $\mathcal{R}_q$, where q is a prime with $q > \gamma$.

Corollary 7. *Let $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and let $t, \gamma, q \in \mathbb{N}$ with $t \leq \gamma$ and q prime with $q > \gamma$. Then there exists a correct, private, and linear t-out-of-γ threshold secret sharing over $\mathcal{R}_q$.*

2.5 Additive Set Encodings

A multiset encoding is a data structure that allows for encoding a multiset and later on recover it from the encoding, assuming the number of set elements does not exceed the capacity of the data structure. What makes these encodings interesting and non-trivial is the fact that the location of each individual element in the data structure does not depend on which other elements are present. Furthermore, we will rely on *additive* encodings, which allow separately encodings multiple sets, then adding their encodings to obtain a valid encoding of the sum of the multisets.

A multiset $X = (U, m)$ is defined by an underlying set U and a function $m : U \to \mathbb{N}_0$. For any element $x \in U$, $m(x)$ denotes the multiplicity of x in X. We say $x \in X$ iff $x \in U$ and $m(x) > 0$. We denote the cardinality by $|X| = \sum_{x \in U} m(x)$ and the support by $\mathrm{Supp}(X) = \{x \in U \mid m(x) > 0\}$. Let $X = (U, m)$ and $X' = (U, m')$ be multisets. We define the sum of X and X' as $X + X' = (U, m'')$ with $m''(x) = m(x) + m'(x)$ for all $x \in U$.

Definition 8 (Additive Multiset Encodings). *Let $\mathbb{G}$ be a group. An additive multiset encoding for a universe U and with an encoding space $\mathcal{E} \subseteq \mathbb{G}$ is comprised of the following algorithms:*

$\mathsf{Setup}(1^\lambda, \rho) \to \mathsf{pp}$ *is a PPT algorithm that takes the security parameter $1^\lambda \in \mathbb{N}$ and capacity $\rho \in \mathbb{N}$ as input and returns public parameters pp.*

$\mathsf{Encode}(\mathsf{pp}, X) \to y$ *is a PPT algorithm that takes public parameters pp and multiset X with $\mathrm{Supp}(X) \subseteq U$ as input and returns an encoding $y \in \mathcal{E}$.*

$\mathsf{Decode}(\mathsf{pp}, y) \to X'/\perp$ *is an efficient deterministic algorithm that takes public parameters pp and an encodings $y \in \mathcal{E}$ as input and returns a multiset X' with $\mathrm{Supp}(X') \subseteq U$ or $\perp$.*

An additive multiset encoding is additively ϵ-correct, if for all $\lambda, \rho \in \mathbb{N}$ it holds that

$$\Pr\left[\begin{array}{rl} & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \rho) \\ & (X_1, \ldots, X_\ell) \leftarrow \mathcal{A}(\mathsf{pp}) \\ y_i \leftarrow \mathsf{Encode}(\mathsf{pp}, X_i)\ \forall i \in [\ell] & \mathrm{Supp}(X) \subseteq U \\ X := \sum_{i \in [\ell]} X_i \ : & \wedge\ |X| \le \rho \\ & \wedge\ X' \neq X \\ X' := \mathsf{Decode}(\mathsf{pp}, \sum_{i \in [\ell]} y_i) & \end{array}\right] \le \epsilon(\lambda),$$

where the probability is taken over the random coins of $\mathsf{Setup}, \mathsf{Encode}$ and $\mathcal{A}$.

Construction. Our required multiset encodings can easily be constructed from data structures for encoding key/value pairs, specifically invertible bloom tables [GM11, FLOS24]. For a given multiset, we assign each element a uniformly random key and insert the resulting set of key/value pairs into an invertible bloom filter. The location of each element in the data structure only depends on the corresponding key, not the value and thus allowing the adversary to choose arbitrarily malicious sets, does not affect the correctness guarantees of our data structure. We provide the construction explicitly in Figure 1.

Theorem 8. *Let $\rho, \lambda, \xi, \gamma, \delta$ with $\delta \ge c\rho$ for some large enough constant c be positive integers, let $\mathbb{F}$ be a finite field. Let*

$$\mathsf{PRF} : \{0,1\}^\lambda \times [\gamma] \times \mathbb{F} \to [\delta]$$

$\mathsf{Setup}(1^\lambda, \rho)$	$\mathsf{Encode}(\mathsf{pp}, X)$	$\mathsf{Decode}(\mathsf{pp}, (\boldsymbol{C}, \boldsymbol{K}, \boldsymbol{V}))$
$k \leftarrow \{0,1\}^\lambda$	$\boldsymbol{C} := 0^{\gamma \times \delta}$	$X' := \emptyset$
return $\mathsf{pp} = (k, \rho)$	$\boldsymbol{K} := 0^{\gamma \times \delta}$	**while** $\exists (i^*, j^*) \in [\gamma] \times [\delta]$. $\boldsymbol{C}[i^*, j^*] = 1$
	$\boldsymbol{V} := (0^\xi)^{\gamma \times \delta}$	$r := \boldsymbol{K}[i^*, j^*]$
	foreach $\boldsymbol{x} \in X$:	$x := \boldsymbol{V}[i^*, j^*]$
	$r \leftarrow \mathbb{F}$	$X' := X' + \{x\}$
	for $i \in [\gamma]$:	**foreach** $i \in [\gamma]$
	$j := \mathsf{PRF}(k, (i, r))$	$j := \mathsf{PRF}(k, (i, r))$
	$\boldsymbol{C}[i, j] = \boldsymbol{C}[i, j] + 1$	$\boldsymbol{C}[i, j] = \boldsymbol{C}[i, j] - 1$
	$\boldsymbol{K}[i, j] = \boldsymbol{K}[i, j] + r$	$\boldsymbol{K}[i, j] = \boldsymbol{K}[i, j] - r$
	$\boldsymbol{V}[i, j] = \boldsymbol{V}[i, j] + x$	$\boldsymbol{V}[i, j] = \boldsymbol{V}[i, j] - x$
	return $(\boldsymbol{C}, \boldsymbol{K}, \boldsymbol{V})$	**return** X'

Fig. 1. Additive multiset encoding construction based on the invertible bloom lookup tables of Goodrich and Mitzenmacher [GM11].

be a pseudorandom function. Then the construction specified in Figure 1 is an additively ϵ-correct multiset encoding for universe $U = \mathbb{F}^\xi$, where

$$\epsilon(\lambda) = \rho^{-\gamma+2} + \rho^2/|\mathbb{F}| + \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be an arbitrary adversary. Let win be the event that the adversary wins the security experiment from Definition 8.

Consider a hybrid experiment, where we replace the pseudorandom function by a truly random function. By the security of PRF, this hybrid is computationally indistinguishable from the real experiment. From now on, we consider the experiment with a truly random function. Let win' be the event that the adversary wins the hybrid experiment.

During Encode, we pick a random value $r \in \mathbb{F}$ for each x in the input set. Let coll be the event that there exists two $x, x' \in \bigcup_{i \in [\ell]} X_i$, such that the for the corresponding random values r, r' generated during the execution of Encode of their respective sets, it holds that $r = r'$. By a simple birthday bound, we have that

$$\Pr[\mathsf{coll}] \leq \frac{\rho^2}{|\mathbb{F}|}.$$

Thus we have that

$$\begin{aligned}\Pr[\mathsf{win}] = \Pr[\mathsf{win}'] + \mathsf{negl}(\lambda) &= \Pr[\mathsf{win}' \mid \mathsf{coll}] \Pr[\mathsf{coll}] + \Pr[\mathsf{win}' \mid \neg\mathsf{coll}] \Pr[\neg\mathsf{coll}] + \mathsf{negl}(\lambda) \\ &\leq \Pr[\mathsf{coll}] + \Pr[\mathsf{win}' \mid \neg\mathsf{coll}] + \mathsf{negl}(\lambda) \\ &\leq \frac{\rho^2}{|\mathbb{F}|} + \rho^{-\gamma+2} + \mathsf{negl}(\lambda),\end{aligned}$$

where

$$\Pr[\mathsf{win}' \mid \neg\mathsf{coll}] \leq \rho^{-\gamma+2}$$

directly follows from the correctness guarantees of the invertible bloom lookup tables of Goodrich and Mitzenmacher [GM11]. □

We note that completely analogous statements can be derived using the stacked invertible bloom lookup tables of Fleischhacker et al. [FLOS24]. Furthermore, one can construct an optimally space-efficient, but computationally more expensive additive multiset encoding from Newton-Girard polynomial identities as implicitly shown by Ruffing, Moreno-Sanchez, and Kate [RMK17].

3 Additive Commitments with Addition Hiding

A key component of our anonymous broadcast construction are additively homomorphic vector commitments that satisfy an appropriate notion of hiding, which we define below. Vector commitments allow for succinctly committing to a vector of values and then opening individual vector entries. Normally we just require vector commitments to be binding and hiding in each position. In our context, we require a stronger notion of *addition hiding*, which ensures that opening the sum of commitments at some position should only reveal the sum of values at the position, but nothing about the values that were summed.

In this section, we construct vanilla commitments that satisfy our security notions and in Section 4, we use these commitments to construct our desired vector commitments.

3.1 Definition

Below we precisely define our notion of commitments along with the corresponding security properties. Specifically, we define correctness, addition hiding, and binding. We note that addition hiding implies regular hiding.

Definition 9 (Additive Commitments). *Let $\mathcal{R}$ be a ring and let $A_{\mathsf{msg}}, A_{\mathsf{com}}, A_{\mathsf{op}}$ be $\mathcal{R}$-modules. A commitment scheme CS with message space A_{msg}, commitment space A_{com}, and decommitment space A_{op} is comprised of the following algorithms:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *is a PPT algorithm that takes the security parameter $\lambda \in \mathbb{N}$ as input and returns public parameters pp.*

$\mathsf{Com}(\mathsf{pp}, m) \to (c, d)$ *is a PPT algorithm that takes public parameter pp and a message $m \in A_{\mathsf{msg}}$ as input and returns a commitment $c \in A_{\mathsf{com}}$ and a decommitment $d \in A_{\mathsf{op}}$.*

$\mathsf{Vf}(\mathsf{pp}, c, d) \to m/\bot$ *is an efficient deterministic algorithm that takes public parameters pp, a commitment $c \in A_{\mathsf{com}}$, and a decommitment $d \in A_{\mathsf{op}}$ and returns a message $m \in A_{\mathsf{msg}}$ or $\bot$.*

Addition Correctness. Let $\rho \in \mathbb{N}$. A commitment scheme CS is ρ-addition correct, if for all $\lambda, \rho' \in \mathbb{N}$ with $\rho' \leq \rho$, all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $m_1, \ldots, m_{\rho'} \in A_{\mathsf{msg}}$, and all $(c_i, d_i) := \mathsf{Com}(\mathsf{pp}, m_i)$, it holds that

$$\mathsf{Vf}\Big(\mathsf{pp}, \sum_{j \in [\rho]} c_i, \sum_{i \in [\rho]} d_j\Big) = \sum_{i \in [\rho]} m_j.$$

Statistical Addition Hiding. Let $\rho \in \mathbb{N}$. A commitment scheme CS is statistically ρ-addition hiding, if for all $\lambda \in \mathbb{N}$ and all (potentially unbounded) adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{rl} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) & \\ ((m_1^0,\ldots,m_{\rho'}^0),(m_1^1,\ldots,m_{\rho'}^1)) \leftarrow \mathcal{A}(\mathsf{pp}) & \rho' \le \rho \\ b \leftarrow \{0,1\} & : \ \wedge \sum_{i\in[\rho']} m_i^0 = \sum_{i\in[\rho']} m_i^1 \\ (c_i,d_i) \leftarrow \mathsf{Com}(\mathsf{pp},m_i^b)\ \forall i \in [\rho'] & \\ b' \leftarrow \mathcal{A}((c_1,\ldots,c_{\rho'}),\sum_{i\in[\rho']} d_i) & \wedge\ b' = b \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

Binding. A commitment scheme CS is binding, if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ it holds for that

$$\Pr\left[\begin{array}{rl} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) & \\ (c,d_0,d_1) \leftarrow \mathcal{A}(\mathsf{pp}) & : \ \bot \notin \{m_0,m_1\} \wedge m_0 \neq m_1 \\ m_0 := \mathsf{Vf}(\mathsf{pp},c,d_0) & \\ m_1 := \mathsf{Vf}(\mathsf{pp},c,d_1) & \end{array}\right] \le \mathsf{negl}(\lambda)$$

3.2 Construction

We construct our desired commitment scheme in Figure 2. Conceptually our construction is a variant of the commitment scheme of Baum et al. [BDL+18]. While the scheme itself is similar to prior work, it turns out that proving addition hiding is quite subtle and technically involved.

$\mathsf{Setup}(1^\lambda)$	$\mathsf{Com}(\mathsf{pp},\boldsymbol{m};\boldsymbol{r})$	$\mathsf{Vf}(\mathsf{pp},(c^1,\boldsymbol{c}^2),\boldsymbol{r})$
$\boldsymbol{a} \leftarrow \mathcal{R}_q^\kappa$	$\boldsymbol{r} \leftarrow \mathcal{B}_{\beta,q}^\kappa$	**if** $\|\boldsymbol{r}\| > \beta_{\mathsf{agg}}$
$\boldsymbol{B} \leftarrow \mathcal{R}_q^{\mu\times\kappa}$	**return** $((\boldsymbol{a}^\intercal\boldsymbol{r}, \boldsymbol{B}\boldsymbol{r}+\boldsymbol{m}),\boldsymbol{r})$	**return** $\bot$
return $(\boldsymbol{a},\boldsymbol{B})$		**if** $c^1 \neq \boldsymbol{a}^\intercal\boldsymbol{r}$
		return $\bot$
		else
		return $\boldsymbol{c}^2 - \boldsymbol{B}\boldsymbol{r}$

Fig. 2. Commitment scheme with addition hiding.

Before we consider the security of the construction, we prove the following useful lemma specifying the min-entroy of a randomness $r \in [-\beta,\beta]$ conditioned on the sum $r + r'$ with a second unkown randomness.

Lemma 9. *Let $\beta \in \mathbb{N}$ and let $(r,r') \in [-\beta,\beta]$ be distributed uniformly. Then*

$$\mathbf{H}_\infty(r \mid r+r') = -\log \frac{4\beta+1}{(2\beta+1)^2}$$

Proof. Note that $\mathsf{Supp}(r+r') = [-2\beta,2\beta]$. By Definition 3 it holds that

$$\mathbf{H}_\infty(r \mid r+r') = -\log \sum_{s\in[-2\beta,2\beta]} \Pr[r+r'=s] \cdot \max_{t\in[-\beta,\beta]} \Pr[r=t \mid r+r'=s].$$

Applying Bayes theorem, this gives us

$$\mathbf{H}_\infty(r \mid r+r') = -\log \sum_{s\in[-2\beta,2\beta]} \Pr[r+r'=s] \cdot \max_{t\in[-\beta,\beta]} \frac{\Pr[r+r'=s \mid r=t]\cdot\Pr[r=t]}{\Pr[r+r'=s]}.$$

Since the denominator is independent of t and r is uniform, this simplifies to

$$\begin{aligned}\mathbf{H}_\infty(r \mid r+r') &= -\log \frac{1}{2\beta+1}\cdot\sum_{s\in[-2\beta,2\beta]} \max_{t\in[-\beta,\beta]} \Pr[r+r'=s \mid r=t] \\ &= -\log \frac{1}{2\beta+1}\cdot\sum_{s\in[-2\beta,2\beta]} \max_{t\in[-\beta,\beta]} \Pr[r'=s-t]. \end{aligned} \tag{1}$$

Now, note that $\Pr[r'=s-t] = 1/(2\beta+1)$ for $-\beta \le s-t \le \beta$ and $\Pr[r'=s-t]=0$ otherwise. Also note that for any $s\in[-2\beta,2\beta]$ there always exists a t, such that $-\beta \le s-t \le \beta$. Therefore, for any $s\in[-2\beta,2\beta]$

$$\max_{t\in[-\beta,\beta]} \Pr[r'=s-t] = \frac{1}{2\beta+1}.$$

Plugging this back into Equation 1, we get

$$\mathbf{H}_\infty(r \mid r+r') = -\log \frac{1}{2\beta+1}\cdot\sum_{s\in[-2\beta,2\beta]} \frac{1}{2\beta+1} = -\log\frac{4\beta+1}{(2\beta+1)^2}$$

as claimed. □

With the above lemma, we are now ready to prove our construction secure.

Theorem 10. *Let $n,\kappa,\mu,q,\beta,\beta_{\mathsf{agg}},\lambda,\rho$ be positive integers such that n is a power of two, $q\equiv 5 \pmod 8$ is prime, $2\beta < \sqrt{\frac{q}{2}}$, $\beta_{\mathsf{agg}} \ge \rho\beta$, and*

$$(\rho-1)\cdot\left(\frac{4\beta+1}{(2\beta+1)^2}\right)^{\frac{1}{2}\kappa n}\cdot q^{\frac{1}{2}n(\mu+1)} \le 2^{-\lambda}.$$

If the $\mathsf{SIS}_{\mathcal{R},q,\kappa,2\beta_{\mathsf{agg}}}$ problem is hard, then CS_{Leaf} from Figure 2 is ρ-additively correct, computationally binding, and statistically ρ-addition hiding.

Proof. The theorem follows from Lemma 11, Lemma 12, and Lemma 13 proven below. □

Lemma 11. *Let $n,\kappa,\mu,q,\beta,\beta_{\mathsf{agg}},\lambda,\rho$ be positive integers such that n is a power of two, q is prime and $\beta_{\mathsf{agg}} \ge \rho\beta$. Then CS_{Leaf} is ρ-additively correct.*

Proof. Let $\rho' \in [\rho]$, $\boldsymbol{a},\boldsymbol{B} \leftarrow \mathsf{Setup}(1^\lambda)$, and $\boldsymbol{m}^1,\dots,\boldsymbol{m}^{\rho'} \in \mathcal{R}_q^\mu$ be arbitrary. For $i\in[\rho']$ we have $\boldsymbol{c}^i = (\boldsymbol{a}^\intercal\boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i+\boldsymbol{m}^i$ and $\boldsymbol{r}^i = \boldsymbol{r}^i$ for $\boldsymbol{r}^i \in \mathcal{B}^\kappa_{\beta,q}$. We have

$$\sum_{i\in[\rho']} \boldsymbol{d}^i = \sum_{i\in[\rho']} \boldsymbol{r}^i$$

and since $\boldsymbol{r}^i \in \mathcal{B}^{\kappa}_{\beta,q}$, $\rho' \leq \rho$, and $\rho\beta \leq \beta_{\mathsf{agg}}$, it holds that

$$\Big\|\sum_{i\in[\rho']} \boldsymbol{r}^i\Big\| \leq \sum_{i\in[\rho']} \|\boldsymbol{r}^i\| \leq \rho'\beta \leq \rho\beta \leq \beta_{\mathsf{agg}}.$$

Therefore the norm-check in the verification algorithm is successful. Further,

$$\sum_{i\in[\rho']} \boldsymbol{c}^i = \Big(\sum_{i\in[\rho']} \boldsymbol{a}^\intercal \boldsymbol{r}^i, \sum_{i\in[\rho']} \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}^j\Big) = \Big(\boldsymbol{a}^\intercal \sum_{i\in[\rho']} \boldsymbol{r}^i, \boldsymbol{B}\sum_{i\in[\rho']} \boldsymbol{r}^i + \sum_{i\in[\rho']} \boldsymbol{m}^j\Big).$$

Therefore, the second check of the verification algorithm also goes through and it will output

$$\boldsymbol{B}\sum_{i\in[\rho']} \boldsymbol{r}^i + \sum_{i\in[\rho']} \boldsymbol{m}^j - \boldsymbol{B}\sum_{i\in[\rho']} \boldsymbol{r}^i = \sum_{i\in[\rho']} \boldsymbol{m}^j$$

as required. □

Lemma 12. *Let $n, \kappa, \mu, q, \beta, \beta_{\mathsf{agg}}, \lambda$ be positive integers such that n is a power of two and q is prime. If the $\mathsf{SIS}_{\mathcal{R},q,\kappa,2\beta_{\mathsf{agg}}}$ problem is hard, then CS_{Leaf} is computationally binding.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary that breaks binding of $\mathsf{CS}_{\mathrm{Leaf}}$ with probability $\varepsilon(\lambda)$. We construct an algorithm $\mathcal{B}$ solving $\mathsf{SIS}_{\mathcal{R},q,\kappa,2\beta_{\mathsf{agg}}}$ as follows. On input $\boldsymbol{a} \in \mathcal{R}_q^{\kappa}$, $\mathcal{B}$ samples $\boldsymbol{B} \leftarrow \mathcal{R}_q^{\mu\times\kappa}$ and runs $((c^1, \boldsymbol{c}^2), \boldsymbol{r}^0, \boldsymbol{r}^1) \leftarrow \mathcal{A}((\boldsymbol{a}, \boldsymbol{B})$. It then outputs $\boldsymbol{r}^0 - \boldsymbol{r}^1$.

Whenever $\mathcal{A}$ is successful, it outputs $((c^1, \boldsymbol{c}^2), (\boldsymbol{m}^0, \boldsymbol{r}^0), (\boldsymbol{m}^1, \boldsymbol{r}^1))$ such that

$$\|\boldsymbol{r}^0\| \leq \beta_{\mathsf{agg}}, \quad \|\boldsymbol{r}^1\| \leq \beta_{\mathsf{agg}}, \quad \boldsymbol{a}^\intercal\boldsymbol{r}^0 = c^1 = \boldsymbol{a}^\intercal\boldsymbol{r}^1$$

and

$$\boldsymbol{c}^2 - \boldsymbol{B}\boldsymbol{r}^0 \neq \boldsymbol{c}^2 - \boldsymbol{B}\boldsymbol{r}^1 \implies \boldsymbol{B}\boldsymbol{r}^0 \neq \boldsymbol{B}\boldsymbol{r}^1 \implies \boldsymbol{r}^0 \neq \boldsymbol{r}^1$$

Whenever $\mathcal{A}$ is successful, it thus holds that

$$\|\boldsymbol{r}^0 - \boldsymbol{r}^1\| \leq \|\boldsymbol{r}^0\| + \|\boldsymbol{r}^1\| \leq 2\beta_{\mathsf{agg}}$$

and

$$\boldsymbol{a}^\intercal(\boldsymbol{r}^0 - \boldsymbol{r}^1) = \boldsymbol{a}^\intercal\boldsymbol{r}^0 - \boldsymbol{a}^\intercal\boldsymbol{r}^1 = c^1 - c^1 = 0.$$

Therefore $\mathcal{B}$ successfully breaks $\mathsf{SIS}_{\mathcal{R},q,\kappa,2\beta_{\mathsf{agg}}}$ with probability $\varepsilon(\lambda)$. Since $\mathsf{SIS}_{\mathcal{R},q,\kappa,2\beta_{\mathsf{agg}}}$ is hard, it must hold that $\varepsilon(\lambda) \leq \mathsf{negl}(\lambda)$. □

Lemma 13. *Let $n, \kappa, \mu, q, \beta, \lambda, \rho$ be positive integers such that n is a power of two, $q \equiv 5 \pmod 8$ is a prime, $2\beta < \sqrt{\frac{q}{2}}$ and*

$$(\rho - 1) \cdot \Big(\frac{4\beta + 1}{(2\beta + 1)^2}\Big)^{\frac{1}{2}\kappa n} \cdot q^{\frac{1}{2}n(\mu+1)} \leq 2^{-\lambda}$$

Then CS_{Leaf} is statistically ρ-addition hiding.

Proof. Let $\mathcal{A}$ be an arbitrary – possibly unbounded – adversary against ρ-addition hiding of $\mathsf{CS}_{\mathrm{Leaf}}$ with success probability $\frac{1}{2} + \varepsilon(\lambda)$. Let $(\boldsymbol{a}, \boldsymbol{B}) \leftarrow \mathsf{Setup}(\lambda)$ and let $(\boldsymbol{m}_1^0, \ldots, \boldsymbol{m}_{\rho'}^0), (\boldsymbol{m}_1^1, \ldots, \boldsymbol{m}_{\rho'}^1) \leftarrow \mathcal{A}((\boldsymbol{a}, \boldsymbol{B}))$. We assume wlog that $\rho' \in [\rho]$ and

$$\boldsymbol{m} := \sum_{i \in [\rho']} \boldsymbol{m}_i^0 = \sum_{i \in [\rho']} \boldsymbol{m}_i^1.$$

We note that $(\boldsymbol{a}, \boldsymbol{B})$ are distributed uniformly over $\mathcal{R}_q^\kappa \times \mathcal{R}_q^{\mu \times \kappa}$.

Now let $\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{\rho'}$ be distributed uniformly over $\mathcal{B}_{\beta,q}^\kappa$ and define

$$\boldsymbol{r} := \sum_{i \in [\rho']} \boldsymbol{r}^i.$$

It now holds that

$$\varepsilon(\lambda) \leq \mathsf{SD}\big(((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [\rho']}, \boldsymbol{r}), ((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [\rho']}, \boldsymbol{r}) \,\big|\, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big).$$

Since, given $\boldsymbol{a}, \boldsymbol{B}, \boldsymbol{r}, \boldsymbol{m}$ – and therefore $(\boldsymbol{a}^\intercal \boldsymbol{r}, \boldsymbol{B}\boldsymbol{r} + \boldsymbol{m})$ – the first commitment is redundant information, it holds that

$$\begin{aligned}
&\mathsf{SD}\big(((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [\rho']}, \boldsymbol{r}), ((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [\rho']}, \boldsymbol{r}) \,\big|\, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&= \mathsf{SD}\big((\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [2,\rho']}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [2,\rho']} \,\big|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big)
\end{aligned}$$

Let $\boldsymbol{u}^i$ for $i \in [2, rho']$ be distributed uniformly over $\mathcal{R}_q^\mu$. We now define a series of hybrid distributions, where we successively replace each $\boldsymbol{B}\boldsymbol{r}^i$ by $\boldsymbol{u}^i$. By the triangle inequality it then holds that

$$\begin{aligned}
&\mathsf{SD}\big(((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [\rho']}, \boldsymbol{r}), ((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [\rho']}, \boldsymbol{r}) \,\big|\, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&= \mathsf{SD}\big((\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [2,\rho']}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [2,\rho']} \,\big|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&\leq \sum_{j \in [2,\rho']} \mathsf{SD}\left(\begin{matrix}((\boldsymbol{u}^i + \boldsymbol{m}_i^0)_{i \in [2,j-1]}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [j,\rho']}), \\ ((\boldsymbol{u}^i + \boldsymbol{m}_i^0)_{i \in [2,j]}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [j+1,\rho']})\end{matrix} \,\middle|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\right) \\
&\quad + \sum_{j \in [2,\rho']} \mathsf{SD}\left(\begin{matrix}((\boldsymbol{u}^i + \boldsymbol{m}_i^1)_{i \in [2,j-1]}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [j,\rho']}), \\ ((\boldsymbol{u}^i + \boldsymbol{m}_i^1)_{i \in [2,j]}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [j+1,\rho']})\end{matrix} \,\middle|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\right) \\
&\quad + \mathsf{SD}\big((\boldsymbol{u}^i + \boldsymbol{m}_i^0)_{i \in [2,\rho']}, (\boldsymbol{u}^i + \boldsymbol{m}_i^1)_{i \in [2,\rho']} \,\big|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big).
\end{aligned}$$

Now, note that shifting both distribution by a fixed offset does not change the statistical distance. Therefore we can drop the messages from the distributions in the sums. Further, the uniform distribution remains invariant when it is shifted. Therefore, the last statistical distance is in fact zero. Thus,

$$\begin{aligned}
&\mathsf{SD}\big(((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [\rho']}, \boldsymbol{r}), ((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [\rho']}, \boldsymbol{r}) \,\big|\, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&= \mathsf{SD}\big((\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i \in [2,\rho']}, (\boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i \in [2,\rho']} \,\big|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&= 2 \cdot \sum_{j \in [2,\rho']} \mathsf{SD}\left(\begin{matrix}((\boldsymbol{u}^i)_{i \in [2,j-1]}, (\boldsymbol{B}\boldsymbol{r}^i)_{i \in [j,\rho']}), \\ ((\boldsymbol{u}^i)_{i \in [2,j]}, (\boldsymbol{B}\boldsymbol{r}^i)_{i \in [j+1,\rho']})\end{matrix} \,\middle|\, (\boldsymbol{a}^\intercal \boldsymbol{r}^i)_{i \in [2,\rho']}, \boldsymbol{r}, \boldsymbol{a}, \boldsymbol{B}\right). \qquad (2)
\end{aligned}$$

It now remains to bound the individual summands. Since revealing *more* information can never *reduce* the statistical distance, we have

$$\begin{aligned}
&\mathsf{SD}\left(\begin{pmatrix}((\boldsymbol{u}^i)_{i\in[2,j-1]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j,\rho']}),\\((\boldsymbol{u}^i)_{i\in[2,j]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j+1,\rho']})\end{pmatrix}\;\middle|\;(\boldsymbol{a}^\intercal\boldsymbol{r}^i)_{i\in[2,\rho']},\boldsymbol{r},\boldsymbol{a},\boldsymbol{B}\right)\\
&=\mathsf{SD}\Big(\boldsymbol{B}\boldsymbol{r}^j,\boldsymbol{u}^j\mid(\boldsymbol{u}^i)_{i\in[2,j-1]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j+1,\rho']},(\boldsymbol{a}^\intercal\boldsymbol{r}^i)_{i\in[2,\rho']},\boldsymbol{r},\boldsymbol{a},\boldsymbol{B}\Big)\\
&\leq\mathsf{SD}\Big(\boldsymbol{B}\boldsymbol{r}^j,\boldsymbol{u}^j\mid(\boldsymbol{r}^i)_{i\in[2,\rho']\setminus\{j\}},\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r},\boldsymbol{a},\boldsymbol{B}\Big)\\
&=\mathsf{SD}\Big(\boldsymbol{B}\boldsymbol{r}^j,\boldsymbol{u}^j\mid\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r}^1+\boldsymbol{r}^j,\boldsymbol{a}\Big)
\end{aligned}$$

Relying on the fact that we chose our parameters such that $\boldsymbol{B}\boldsymbol{r}^i$ is a universal hash function (Lemma 4) we can then apply the Leftover Hash Lemma (Lemma 5) while considering $\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r}^1+\boldsymbol{r}^j,\boldsymbol{a}$ as leakage about $\boldsymbol{r}^j$.

$$\begin{aligned}
&\mathsf{SD}\left(\begin{pmatrix}((\boldsymbol{u}^i)_{i\in[2,j-1]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j,\rho']}),\\((\boldsymbol{u}^i)_{i\in[2,j]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j+1,\rho']})\end{pmatrix}\;\middle|\;(\boldsymbol{a}^\intercal\boldsymbol{r}^i)_{i\in[2,\rho']},\boldsymbol{r},\boldsymbol{a},\boldsymbol{B}\right)\\
&\leq\frac{1}{2}\cdot\sqrt{2^{-\mathbf{H}_\infty(\boldsymbol{r}^j\mid\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r}^1+\boldsymbol{r}^j,\boldsymbol{a})+\log|\mathcal{R}_q^\mu|}}
\end{aligned}\tag{3}$$

To bound the conditional min-entropy of $\boldsymbol{r}^j$, we first deal with the $\boldsymbol{a}^\intercal\boldsymbol{r}^j$ by simply assuming the worst possible entropy loss.[4]

$$\begin{aligned}
&\mathbf{H}_\infty(\boldsymbol{r}^j\mid\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r}^1+\boldsymbol{r}^j,\boldsymbol{a})\\
&\geq\mathbf{H}_\infty(\boldsymbol{r}^j\mid\boldsymbol{r}^1+\boldsymbol{r}^j)-\log|\mathcal{R}_q|\\
&=\mathbf{H}_\infty(\boldsymbol{r}^j\mid\boldsymbol{r})-\log q^n
\end{aligned}$$

We then note that all coefficients of $\boldsymbol{r}^j$ are independent. We can thus apply Lemma 9 seperately for each of the κn coefficient, yielding

$$\mathbf{H}_\infty(\boldsymbol{r}^j\mid\boldsymbol{a}^\intercal\boldsymbol{r}^j,\boldsymbol{r}^1+\boldsymbol{r}^j,\boldsymbol{a})=-\log\Big(\frac{4\beta+1}{(2\beta+1)^2}\Big)^{\kappa n}-\log q^n.$$

Plugging this back into Equation 3 we get

$$\begin{aligned}
&\mathsf{SD}\left(\begin{pmatrix}((\boldsymbol{u}^i)_{i\in[2,j-1]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j,\rho']}),\\((\boldsymbol{u}^i)_{i\in[2,j]},(\boldsymbol{B}\boldsymbol{r}^i)_{i\in[j+1,\rho']})\end{pmatrix}\;\middle|\;(\boldsymbol{a}^\intercal\boldsymbol{r}^i)_{i\in[2,\rho']},\boldsymbol{r},\boldsymbol{a},\boldsymbol{B}\right)\\
&\leq\frac{1}{2}\cdot\sqrt{2^{\log\left(\frac{4\beta+1}{(2\beta+1)^2}\right)^{\kappa n}+\log q^n+\log q^{n\mu}}}\\
&=\frac{1}{2}\cdot\Big(\frac{4\beta+1}{(2\beta+1)^2}\Big)^{\frac{1}{2}\kappa n}\cdot q^{\frac{1}{2}n(\mu+1)}
\end{aligned}$$

[4] Note, that this bound on the loss is tight, iff the range of the hash function defined by $\boldsymbol{a}$ covers the entirety of $\mathcal{R}_q$.

and finally from Equation 2 that

$$\begin{aligned}
&\mathsf{SD}\big(((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^0)_{i\in[\rho']}, \boldsymbol{r}), ((\boldsymbol{a}^\intercal \boldsymbol{r}^i, \boldsymbol{B}\boldsymbol{r}^i + \boldsymbol{m}_i^1)_{i\in[\rho']}, \boldsymbol{r}) \,\big|\, \boldsymbol{a}, \boldsymbol{B}, \boldsymbol{m}\big) \\
&\leq 2 \cdot \sum_{j\in[2,\rho']} \frac{1}{2} \cdot \Big(\frac{4\beta+1}{(2\beta+1)^2}\Big)^{\frac{1}{2}\kappa n} \cdot q^{\frac{1}{2}n(\mu+1)} \\
&= (\rho'-1) \cdot \Big(\frac{4\beta+1}{(2\beta+1)^2}\Big)^{\frac{1}{2}\kappa n} \cdot q^{\frac{1}{2}n(\mu+1)} \\
&\leq (\rho-1) \cdot \Big(\frac{4\beta+1}{(2\beta+1)^2}\Big)^{\frac{1}{2}\kappa n} \cdot q^{\frac{1}{2}n(\mu+1)} \\
&\leq 2^{-\lambda}
\end{aligned}$$

where the last two inequalities hold by the conditions from the lemma statement. □

4 Additive Vector Commitments with Addition Hiding

In this section, we now construct vector commitments with addition hiding. Intuitively, we construct such vector commitments by plugging the hiding commitments from the previous sections into a variant of the non-hiding homomorphic non-hiding vector commitments of Fleischacker et al. [FHSZ23a].

4.1 Definition

Analogous to our commitment definition, we require vector commitments to be correct, addition hiding, and binding.

Definition 10 (Additive Vector Commitments). *Let $\mathcal{R}$,$\mathcal{R}'$ be a rings, let A_{msg} be a $\mathcal{R}'$-module and let $A_{\mathsf{com}}, A_{\mathsf{op}}$ be $\mathcal{R}$-modules. A vector commitment scheme for vector length $\gamma \in \mathbb{N}$ with message space A_{msg}, commitment space A_{com}, and decommitment space A_{op} is comprised of the following algorithms:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *is a PPT algorithm that takes the security parameter $\lambda \in \mathbb{N}$ as input and returns public parameters* pp.

$\mathsf{Com}(\mathsf{pp}, \boldsymbol{v}) \to (c, \boldsymbol{d})$ *is a PPT algorithm that takes public parameters* pp *and a vector of messages $\boldsymbol{v} \in A_{\mathsf{msg}}^\gamma$ as input and returns a commitment $c \in \mathcal{C}$ and decommitments $\boldsymbol{d} \in A_{\mathsf{op}}^\gamma$.*

$\mathsf{Vf}(\mathsf{pp}, i, c, d_i) \to m/\perp$ *is an efficient deterministic algorithm that takes public parameters* pp*, a commitment $c \in A_{\mathsf{com}}$, an index $i \in [\ell]$, and a decommitment $d_i \in A_{\mathsf{op}}$ and returns a message $m \in A_{\mathsf{msg}}$ or $\perp$.*

Addition Correctness. Let $\rho \in \mathbb{N}$. An additive vector commitment is ρ-addition correct, if for all $\lambda, \rho' \in \mathbb{N}$, with $\rho' < \rho$, all $\boldsymbol{v}^1, \ldots \boldsymbol{v}^{\rho'} \in A_{\mathsf{msg}}^\gamma$, all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(c_j, \boldsymbol{d}^j) := \mathsf{Com}(\mathsf{pp}, \boldsymbol{v}^j)$, and all $i \in [\ell]$ it holds that

$$\mathsf{Vf}\Big(\mathsf{pp}, i, \sum_{j\in[\rho']} c_j, \sum_{j\in[\rho']} d_i^j\Big) = \sum_{j\in[\rho']} v_i^j.$$

Statistical Addition Hiding. Let $\rho \in \mathbb{N}$. An additive vector commitment is statistically ρ-addition hiding, if for all $\lambda \in \mathbb{N}$ and all (potentially unbounded) adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{rl} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) & \rho' \leq \rho \\ ((\boldsymbol{v}^{1,0},\ldots,\boldsymbol{v}^{\rho',0}),(\boldsymbol{v}^{1,1},\ldots,\boldsymbol{v}^{\rho',1}),C) \leftarrow \mathcal{A}(\mathsf{pp}) & \wedge \|\boldsymbol{v}^{j,b}\| = \gamma \ \forall (j,b) \in [\rho'] \times \{0,1\} \\ b \leftarrow \{0,1\} & \wedge\ v_i^{j,0} = v_i^{j,1} \ \forall (j,i) \in [\rho'] \times C \\ (c_j, \boldsymbol{d}^j) \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{v}^{j,b}) \ \forall j \in [\rho'] & \wedge \sum_{j\in[\rho']} \boldsymbol{v}^{j,0} = \sum_{j\in[\rho']} \boldsymbol{v}^{j,1} \\ b' \leftarrow \mathcal{A}\Big(\big(c_j, \mathsf{filter}(\boldsymbol{d}^j, C)\big)_{j\in[\rho']}, \sum_{j\in[\rho']} \boldsymbol{d}^j\Big) & \wedge\ b' = b \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Position Binding. An additive vector commitment is position binding, if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (i, c, d_0, d_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ m_0 := \mathsf{Vf}(\mathsf{pp}, i, c, d_0) \\ m_1 := \mathsf{Vf}(\mathsf{pp}, i, c, d_1) \end{array} : \perp \notin \{m_0, m_1\} \wedge m_0 \neq m_1\right] \leq \mathsf{negl}(\lambda)$$

4.2 Construction from Lattices

As mentioned above, we start with a non-hiding additive vector commitment, which we make hiding by committing to the individual entries with a hiding commitment before using the non-hiding vector commitment itself. To construct the non-hiding vector commitment, we use a very slightly modified version of the vector commitment used in the Chipmunk multi-signature scheme [FHSZ23a].

The main difference is that Chipmunk used a slightly convoluted three-tiered system of verification algorithms, where commitments/openings could be "individually", "strongly", or "weakly" verifying. This was necessary because Chipmunk required correctness to hold for certain random linear combinations of individually verifying openings. We, however, only require correctness to hold for a simple sum of honestly generated commitments/openings and thus we only require a single regular verification algorithm.

The other modification is purely syntactic. While the Chipmunk interface has an opening algorithm to open individual positions, we never use this functionality and thus don't define such an interface. In our case, we commit to a vector and immediately open the commitment at all positions in one go. Therefore, we simply let the commit algorithm directly output the commitment along with all individual openings.

For sake of completeness and to avoid confusion due to the changes mentioned above, we will restate the full construction here. Except for the modifications described above, the following definitions and exposition follows [FHSZ23a] essentially verbatim.

The construction requires decomposition of $\mathcal{R}_q$ elements into low norm vectors of $\mathcal{R}$ elements, as well a projecting vectors of low norm $\mathcal{R}$ elements back into $\mathcal{R}_q$. Following Chipmunk we use balanced $(2\eta+1)$-ary decomposition, which allows to use the greatest arity while minimizing the infinity norm of decomposed elements.

Definition 11 (Projection onto $\mathcal{R}$ elements). *Let $\eta, \kappa \in \mathbb{N}$. For any $\boldsymbol{b} \in \mathbb{Z}^\kappa$ we define the function*

$$\mathsf{proj}_{\eta,\kappa} \colon \mathbb{Z}^\kappa \to \mathbb{Z}, \quad \mathsf{proj}_{\eta,\kappa}(\boldsymbol{b}) = \sum_{j=1}^{\kappa} b_j \cdot (2\eta+1)^{j-1} \ .$$

We can extend this to a map

$$\mathsf{proj}_{\eta,\kappa} \colon \mathcal{R}^\kappa \to \mathcal{R}, \quad \mathsf{proj}_{\eta,\kappa}(\boldsymbol{b}) = \sum_{j=1}^{\kappa} b_j \cdot (2\eta+1)^{j-1} \ .$$

Definition 12 (Balanced $(2\eta+1)$-ary decomposition of $\mathcal{R}$ elements). *Fix some odd arity $2\eta+1$ and let $\kappa \in \mathbb{N}$ be the number of limbs. Then we define the balanced $(2\eta+1)$-ary decomposition with κ limbs of any $a \in \mathbb{Z}$ as*

$$a = \sum_{i=1}^{\kappa} a_i \cdot (2\eta+1)^{i-1}$$

where $a_i \in \{-\eta, \ldots, \eta\}$ for all $1 \leq i < \kappa$.

We can extend this to a map on $\mathcal{R}$ by essentially decomposing each coefficient, mapping a polynomial $a \in \mathcal{R}$ to limbs $a_1 \ldots, a_\kappa \in \mathcal{R}$ such that

$$a = \sum_{i=1}^{\kappa} a_i \cdot (2\eta+1)^{i-1}$$

where $\|a_i\|_\infty \leq \eta$ for all $1 \leq i < \kappa$. If $\|a\|_\infty < \frac{(2\eta+1)^\kappa}{2}$, we also have the bound $\|a_\kappa\|_\infty \leq \eta$ for the most significant limb.

Matching the notation from Definition 11, we denote this decomposition map by $\mathsf{dec}_{\eta,\kappa}$, giving a map

$$\mathsf{dec}_{\eta,\kappa} \colon \mathcal{R} \to \mathcal{R}^\kappa, \quad a \mapsto (a_1, \ldots, a_\kappa) \ .$$

Remark 1. Fleischhacker et al. [FHSZ23a] show that any $a \in \mathbb{Z}$ and any $a \in \mathbb{R}$ indeed always has a balanced $(2\eta+1)$-ary decomposition with κ limbs as defined above. We note that it is notationally convenient to allow arbitrarily sized a in the definition and to not bound the last component of the decomposition a_κ, thereby putting all higher-order terms into a_κ. If we have the bound $|a| < \frac{(2\eta+1)^\kappa}{2}$, then the most significant limb a_κ will also be in $\{-\eta, \ldots, \eta\}$.

Definition 13 (Projection and Decomposition for $\mathcal{R}_q$). *Fix some odd arity $(2\eta+1)$ and let q be prime. Set $\kappa := \lceil \log_{2\eta+1} q \rceil$ We denote by*

$$\mathsf{proj}_q \colon \mathcal{R}^\kappa \to \mathcal{R}_q, \quad \mathsf{proj}_q(\boldsymbol{a}) := \mathsf{proj}_{\eta,\kappa}(\boldsymbol{a}) \bmod q \in \mathcal{R}_q$$

and by

$$\mathsf{dec}_q \colon \mathcal{R}_q \to \mathcal{R}^\kappa, \quad \mathsf{dec}_q(a) := \mathsf{dec}_{\eta,\kappa}(a'), \ \ ,$$

where a' is the representative of a in $\mathcal{R}$ with coefficients in $\{-\frac{q-1}{2}, \ldots, +\frac{q-1}{2}\}$.

For proj_q and dec_q, the value of η is not denoted explicitly. In our constructions, all uses of proj_q and dec_q will use the same value for η, even if the values of q differ. The following proposition immediately follow from the definitions.

Proposition 14. *Let q be an odd integer and fix some odd arity $2\eta+1$. The maps proj_q and $\mathsf{proj}_{\eta,\kappa}$ defined above are $\mathcal{R}$-linear. The map $\mathsf{dec}_{\eta,\kappa}$ is a one-sided inverse to $\mathsf{proj}_{\eta,\kappa}$, meaning that $\mathsf{proj}_{\eta,\kappa}(\mathsf{dec}_{\eta,\kappa}(a)) = a$ for any $a \in \mathcal{R}$. Similarly, dec_q is a one-sided inverse to proj_q, meaning that $\mathsf{proj}_q(\mathsf{dec}_q(a)) = a$ for any $a \in \mathcal{R}_q$. For $a \in \mathcal{R}_q$, we also have $\|\mathsf{dec}_q(a)\|_\infty \leq \eta$.*

Following the exposition in Chipmunk we will at times abuse notation slightly and apply dec_q resp. $\mathsf{dec}_{\eta,\kappa}$ to *vectors* of $\mathcal{R}_q$, respectively $\mathcal{R}$ elements, which is to be understood as the component-wise application of dec_q resp. $\mathsf{dec}_{\eta,\kappa}$ with subsequent concatenation of the resulting vectors. Similarly, proj_q respectively $\mathsf{proj}_{\eta,\kappa}$ may be applied to vectors of a length that is a *multiple* of κ to result in a vector of $\mathcal{R}_q$ respectively $\mathcal{R}$ elements. The above generalizes to this extension.

This lets us define a labeling function for a full binary tree as follows.

Definition 14 (Labeled Full Binary Tree). *Let $n, q, q', \xi \in \mathbb{N}$ with n a power of two and q, q' primes. Let $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{2^\tau})^\intercal \in (\mathcal{R}_{q'}^\xi)^{2^\tau}$, $\boldsymbol{g} \in \mathcal{R}_q^{\xi\lceil \log_{2\eta+1} q' \rceil}$ and $\boldsymbol{h}_0, \boldsymbol{h}_1 \in \mathcal{R}_q^{\lceil \log_{2\eta+1} q \rceil}$ be fixed. We define the labeling function $\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1} \colon (\mathcal{R}_{q'}^\xi)^{2^\tau} \times \{0,1\}^{\leq \tau} \to \mathcal{R}^{\lceil \log_{2\eta+1} q \rceil}$ for a labeled full binary tree of depth τ as*

$$\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, v) \coloneqq \begin{cases} \mathsf{dec}_q(\boldsymbol{g}^\intercal \cdot \mathsf{dec}_{q'}(\boldsymbol{m}_{v+1})) & \text{if } |v| = \tau \\ \mathsf{dec}_q\begin{pmatrix} \boldsymbol{h}_0^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, v\|0) \\ +\boldsymbol{h}_1^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, v\|1) \end{pmatrix} & \text{if } |v| < \tau \end{cases}.$$

For this, remember that multiplication of elements from $\mathcal{R}_q$ and $\mathcal{R}$ is always understood[5] *to give an element in $\mathcal{R}_q$. For $\boldsymbol{m}_{v+1}$, we interpret $v \in \{0,1\}^\tau$ as an integer in $\{0 \ldots, 2^\tau - 1\}$ in big-endian encoding and add 1 (i.e. the inverse of taking $\tilde{t} = \mathsf{bin}_\tau(t-1)$).*

Theorem 15. *Let $n, q, q', \rho, \eta, \tau, \xi, \beta_{\mathsf{agg}}$ be positive integers n is a power of two, q, q' are prime, and $\beta_{\mathsf{agg}} \geq \eta\rho$ If the $\mathsf{SIS}_{\mathcal{R},q,2\lceil \log_{2\eta+1} q \rceil, 2\beta_{\mathsf{agg}}}$ problem and the $\mathsf{SIS}_{\mathcal{R},q,\xi\lceil \log_{2\eta+1} q' \rceil, 2\beta_{\mathsf{agg}}}$ problem are hard, then CS_{Chip} from Figure 3 is a ρ-addition correct and position binding additive vector commitment for domain $A_{\mathsf{msg}} = \mathcal{R}_{q'}^\xi$ and vector length 2^τ.*

Proof. The theorem follows from Lemma 16 and Lemma 17 proven below. □

Lemma 16. *Let $n, q, q', \rho, \eta, \tau, \xi, \beta_{\mathsf{agg}}$ be positive integers n is a power of two, q, q' are prime, and $\beta_{\mathsf{agg}} \geq \eta\rho$. Then CS_{Chip} is ρ-addition correct.*

Proof. Let $\lambda \in \mathbb{N}$, $\gamma = 2^\tau$, $\rho' < \rho$. Let $\boldsymbol{v}^1, \ldots \boldsymbol{v}^{\rho'} \in \mathcal{M}^\gamma$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \gamma)$ be arbitrary. For all $i \in [\rho']$, let $(\boldsymbol{c}^i, (\boldsymbol{d}_1^i, \ldots, \boldsymbol{d}_\gamma^i)) \leftarrow \mathsf{Com}(\mathsf{pp}, \boldsymbol{v}^i)$ be arbitrary. Define

$$\boldsymbol{c} \coloneqq \sum_{i \in [\rho']} \boldsymbol{c}^i \quad \boldsymbol{d}_t = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_\tau, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_\tau, \boldsymbol{u}) \coloneqq \sum_{i \in [\rho']} \boldsymbol{d}_t^i$$

We first observe that since proj_q is $\mathcal{R}$-linear

$$\mathsf{proj}_q(\boldsymbol{p}_\tau) = \mathsf{proj}_q\Big(\sum_{i \in [\rho']} \boldsymbol{p}_\tau^i\Big) = \sum_{i \in [\rho']} \mathsf{proj}_q(\boldsymbol{p}_\tau^i)$$

[5] as opposed to taking some canonical representative of $\mathcal{R}_q$-elements in $\mathcal{R}$ and then multiplying in $\mathcal{R}$

$\mathsf{Setup}(1^\lambda)$

$\boldsymbol{g} \leftarrow \mathcal{R}_q^{\xi\kappa'}$
$\boldsymbol{h}_0 \leftarrow \mathcal{R}_q^{\kappa}$
$\boldsymbol{h}_1 \leftarrow \mathcal{R}_q^{\kappa}$
return $(\boldsymbol{g}, \boldsymbol{h}_0, \boldsymbol{h}_1)$

$\mathsf{Com}(\mathsf{pp}, \boldsymbol{m})$

$\boldsymbol{p}_0 \coloneqq \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, \epsilon)$
$c \coloneqq \mathsf{proj}_q(\boldsymbol{p}_0)$
for $t \in [2^\tau]$
  $\tilde{t} \coloneqq \mathsf{bin}_\tau(t-1)$
  for $j \in [\tau]$
    $\boldsymbol{p}_{t,j} \coloneqq \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, \tilde{t}_{<j} \| \tilde{t}_j)$
    $\boldsymbol{s}_{t,j} \coloneqq \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, \tilde{t}_{<j} \| (\tilde{t}_j \oplus 1))$
  $\boldsymbol{u}_t \coloneqq \mathsf{dec}_{q'}(\boldsymbol{m}_t)$
  $\boldsymbol{d}_t \coloneqq (\boldsymbol{p}_{t,1}, \ldots, \boldsymbol{p}_{t,\tau}, \boldsymbol{s}_{t,1}, \ldots, \boldsymbol{s}_{t,\tau}, \boldsymbol{u}_t)$
return $(c, (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{2^\tau}))$

$\mathsf{Vf}(\mathsf{pp}, c, t, \boldsymbol{d})$

parse $\boldsymbol{d}$ **as** $(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_\tau, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_\tau, \boldsymbol{u})$
$\tilde{t} \coloneqq \mathsf{bin}_\tau(t-1)$
if $\|\boldsymbol{u}\| > \beta_{\mathsf{agg}}$ **or** $\mathsf{proj}_q(\boldsymbol{p}_\tau) \neq \boldsymbol{g}^\intercal \cdot \boldsymbol{u}$
  return $\perp$
if $c \neq \boldsymbol{h}_{\tilde{t}_1}^\intercal \cdot \boldsymbol{p}_1 + \boldsymbol{h}_{\tilde{t}_1 \oplus 1}^\intercal \cdot \boldsymbol{s}_1$
  return $\perp$
for $j \in [2, \tau]$
  if $\mathsf{proj}_q(\boldsymbol{p}_{j-1}) \neq \boldsymbol{h}_{\tilde{t}_j}^\intercal \cdot \boldsymbol{p}_j + \boldsymbol{h}_{\tilde{t}_j \oplus 1}^\intercal \cdot \boldsymbol{s}_j$
    return $\perp$
for $j \in [\tau]$
  if $\|\boldsymbol{p}_j\| > \beta_{\mathsf{agg}}$ **or** $\|\boldsymbol{s}_j\| > \beta_{\mathsf{agg}}$
    return $\perp$
return $\mathsf{proj}_{q'}(\boldsymbol{u}) \in \mathcal{R}_{q'}^{\xi}$

Fig. 3. The construction of the Chipmunk additive vector commitment $\mathsf{CS}_{\mathrm{Chip}}$ for message space $A_{\mathsf{msg}} = \mathcal{R}_{q'}^{\xi}$, and vector length 2^τ. Commitments c are in $A_{\mathsf{com}} = \mathcal{R}_q$. Openings are small elements in $A_{\mathsf{op}} = (\mathcal{R}^\kappa)^{2\tau} \times (\mathcal{R}^{\kappa'})^\xi$, where $\kappa = \lceil \log_{2\eta+1} q \rceil$, $\kappa' = \lceil \log_{2\eta+1} q' \rceil$. Multiplication of $\mathcal{R}_q$ with $\mathcal{R}$ elements as done in the Ajtai hashes like $\boldsymbol{g}^\intercal \cdot \boldsymbol{u}$ is understood to give an element in $\mathcal{R}_q$, i.e. we perform modular reduction here.

Since the commitment and opening are honestly computed, it then follows from the definition of Com and Definition 14 that

$$\mathsf{proj}_q(\boldsymbol{p}_\tau) = \sum_{i \in [\rho']} \mathsf{proj}_q(\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}) = \sum_{i \in [\rho']} \mathsf{proj}_q(\mathsf{dec}_q(\boldsymbol{g}^\intercal \cdot \mathsf{dec}_{q'}(\boldsymbol{v}_t^i)))$$

Finally, Proposition 14, the definition of Com, and $\mathcal{R}$-linearity of proj_q gives us

$$\mathsf{proj}_q(\boldsymbol{p}_\tau) = \sum_{i \in [\rho']} \boldsymbol{g}^\intercal \cdot \mathsf{dec}_{q'}(\boldsymbol{v}_t^i) = \sum_{i \in [\rho']} \boldsymbol{g}^\intercal \cdot \boldsymbol{u}^i = \boldsymbol{g}^\intercal \cdot \sum_{i \in [\rho']} \boldsymbol{u}^i = \boldsymbol{g}^\intercal \cdot \boldsymbol{u}$$

as required. Similarly, for all $j \in [\tau]$ it holds that

$$\begin{aligned}
\mathsf{proj}_q(\boldsymbol{p}_{j-1}) &= \mathsf{proj}_q\Big(\sum_{i \in [\rho']} \boldsymbol{p}_{j-1}^i\Big) \\
&= \sum_{i \in [\rho']} \mathsf{proj}_q(\boldsymbol{p}_{j-1}^i) \\
&= \sum_{i \in [\rho']} \mathsf{proj}_q\left(\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j})\right) && \text{(Def. of } \mathsf{Com}\text{)} \\
&= \sum_{i \in [\rho']} \mathsf{proj}_q\left(\mathsf{dec}_q\begin{pmatrix} \boldsymbol{h}_0^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| 0) \\ + \boldsymbol{h}_1^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| 1) \end{pmatrix}\right) && \text{(Definition 14)} \\
&= \sum_{i \in [\rho']} \big(\boldsymbol{h}_0^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| 0) + \boldsymbol{h}_1^\intercal \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| 1)\big) && \text{(Proposition 14)}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i\in[\rho']} \boldsymbol{h}_{\tilde{t}_j}^{\mathsf{T}} \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, \tilde{t}_{<j} \| \tilde{t}_j) + \sum_{i\in[\rho']} \boldsymbol{h}_{\tilde{t}_j\oplus 1}^{\mathsf{T}} \cdot \mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{m}, \tilde{t}_{<j} \| (\tilde{t}_j \oplus 1)) \\
&= \boldsymbol{h}_{\tilde{t}_j}^{\mathsf{T}} \cdot \sum_{i\in[\rho']} \boldsymbol{p}_j^i + \boldsymbol{h}_{\tilde{t}_j\oplus 1}^{\mathsf{T}} \cdot \sum_{i\in[\rho']} \boldsymbol{s}_j^i && \text{(Def. of } \mathsf{Com}) \\
&= \boldsymbol{h}_{\tilde{t}_j}^{\mathsf{T}} \cdot \boldsymbol{p}_j + \boldsymbol{h}_{\tilde{t}_j\oplus 1}^{\mathsf{T}} \cdot \boldsymbol{s}_j.
\end{aligned}
$$

Observe that for $j = 1$, this gives $\boldsymbol{c} = \boldsymbol{h}_{\tilde{t}_1}^{\mathsf{T}} \cdot \boldsymbol{p}_1 + \boldsymbol{h}_{\tilde{t}_1\oplus 1}^{\mathsf{T}} \cdot \boldsymbol{s}_1$. It now remains to check that the norm bounds are satisfied. It immediately follows from the definition of $\mathsf{dec}_{q'}$ that

$$
\|\boldsymbol{u}\| = \Big\|\sum_{i\in[\rho']} \boldsymbol{u}^i\Big\| \le \sum_{i\in[\rho']} \|\boldsymbol{u}^j\| = \sum_{i\in[\rho']} \|\mathsf{dec}_{q'}(\boldsymbol{v}^i)\| \le \rho'\eta \le \rho\eta \le \beta_{\mathsf{agg}}.
$$

And similarly, since the output of $\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}$ is always an output of dec_q that

$$
\|\boldsymbol{p}_j\| = \Big\|\sum_{i\in[\rho']} \boldsymbol{p}_j^i\Big\| \le \sum_{i\in[\rho']} \|\boldsymbol{p}_j^i\| = \sum_{i\in[\rho']} \|\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| \tilde{t}_j)\| \le \rho'\eta \le \rho\eta \le \beta_{\mathsf{agg}}
$$

and

$$
\|\boldsymbol{s}_j\| = \Big\|\sum_{i\in[\rho']} \boldsymbol{s}_j^i\Big\| \le \sum_{i\in[\rho']} \|\boldsymbol{s}_j^i\| = \sum_{i\in[\rho']} \|\mathsf{label}_{\boldsymbol{g},\boldsymbol{h}_0,\boldsymbol{h}_1}(\boldsymbol{v}^i, \tilde{t}_{<j} \| (\tilde{t}_j \oplus 1))\| \le \rho'\eta \le \rho\eta \le \beta_{\mathsf{agg}}.
$$

Therefore, all checks in the verification algorithm go through and the algorithm outputs

$$
\mathsf{proj}_{q'}(\boldsymbol{u}) = \mathsf{proj}_{q'}\Big(\sum_{i\in[\rho']} \boldsymbol{u}^i\Big) = \mathsf{proj}_{q'}\Big(\sum_{i\in[\rho']} \boldsymbol{u}^i\Big) = \sum_{i\in[\rho']} \mathsf{proj}_{q'}(\boldsymbol{u}^i) = \sum_{i\in[\rho']} \mathsf{proj}_{q'}(\mathsf{dec}_{q'}(\boldsymbol{v}_t^i)) = \sum_{i\in[\rho']} \boldsymbol{v}_t^i
$$

as required. □

Lemma 17. *Let $n, q, q', \rho, \eta, \tau, \xi, \beta_{\mathsf{agg}}$ be positive integers, such that n is a power of two, and q, q' are prime. If the $\mathsf{SIS}_{\mathcal{R},q,2\lceil\log_{2\eta+1} q\rceil,2\beta_{\mathsf{agg}}}$ problem and the $\mathsf{SIS}_{\mathcal{R},q,\xi\lceil\log_{2\eta+1} q'\rceil,2\beta_{\mathsf{agg}}}$ problem are hard, then CS_{Chip} is position binding.*

Proof. Position binding follows directly from the position binding proof in the full version of the original Chipmunk paper [FHSZ23b, Lemma 21] by merely considering our regular verification algorithm instead of their "weak" verification algorithm. □

4.3 Making it Hiding

Let us now prove that using the hiding commitment scheme $\mathsf{CS}_{\text{leaf}}$ in combination with the non-hiding vector commitment $\mathsf{CS}_{\text{Chip}}$ from above, gives us the desired hiding vector commitment.

Theorem 18. *Let CS_{leaf} be a ρ-addition correct, statistically ρ-addition hiding, and binding commitment scheme with message space A_{msg} and commitment space A_{com}. Let VC be a ρ-addition correct and position binding vector commitment scheme for vector length γ with message space A_{com}. Then VC_{hide} is a ρ-addition correct, statistically ρ-addition hiding, and position binding additive vector commitment for vector length γ with message space A_{msg}.*

$\mathsf{Setup}(1^\lambda)$	$\mathsf{Com}(\mathtt{pp}, \boldsymbol{v})$	$\mathsf{Vf}(\mathtt{pp}, c, i, (d, \hat{d}))$
$\mathtt{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	**for** $i \in [\gamma]$:	$\hat{c} \leftarrow \mathsf{VC.Vf}(\mathtt{pp}_{\mathsf{VC}}, c, i, d)$
$\mathtt{pp}_{\mathrm{leaf}} \leftarrow \mathsf{CS}_{\mathrm{leaf}}.\mathsf{Setup}(1^\lambda)$	$\quad(\hat{c}_j, \hat{d}_j) \leftarrow \mathsf{CS}_{\mathrm{leaf}}.\mathsf{Com}(\mathtt{pp}_{\mathrm{leaf}}, v_i)$	**if** $\hat{c} = \bot$
return $(\mathtt{pp}_{\mathsf{VC}}, \mathtt{pp}_{\mathrm{leaf}})$	$(c, \boldsymbol{d}) \leftarrow \mathsf{VC.Com}(\mathtt{pp}_{\mathsf{VC}}, (\hat{c}_i)_{i\in[\gamma]})$	$\quad$**return** $\bot$
	return $\left(c, \left((d_i, \hat{d}_i)\right)_{i\in[\gamma]}\right)$	**return** $\mathsf{CS}_{\mathrm{leaf}}.\mathsf{Vf}(\mathtt{pp}_{\mathrm{leaf}}, \hat{c}, \hat{d})$

Fig. 4. Construction of hiding vector commitment from non-hiding vector commitment and hiding commitments.

Proof. The theorem follows from Lemma 19, Lemma 20 and Lemma 21 proven below. □

Lemma 19. *If* CS_{leaf} *and* VC *are both* ρ*-addition correct, then* VC_{hide} *is also* ρ*-addition correct.*

Proof. Let $\lambda, \rho' \in \mathbb{N}$ with $\rho' \leq \rho$ and $(\mathtt{pp}_{\mathsf{VC}}, \mathtt{pp}_{\mathrm{leaf}}) \leftarrow \mathsf{Setup}(\lambda)$. Let $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^{\rho'} \in A_{\mathsf{msg}}^\gamma$ be arbitrary and let $(c_j, (d_i^j, \hat{d}_i^j)_{i\in[\gamma]}) \leftarrow \mathsf{Com}(\mathtt{pp}, \boldsymbol{v}^j)$. It needs to hold for all $i \in [\gamma]$ that

$$\mathsf{Vf}((\mathtt{pp}_{\mathsf{VC}}, \mathtt{pp}_{\mathrm{leaf}}), i, \sum_{j\in[\rho']} c_j, \Big(\sum_{j\in[\rho']} d_i^j, \sum_{j\in[\rho']} \hat{d}_i^j\Big)) = \sum_{j\in[\rho']} v_i^j$$

By definition of the Com, we have that for $j \in [\rho']$

$$(c_j, (d_i^j)_{i\in[\gamma]}) \leftarrow \mathsf{VC.Com}(\mathtt{pp}_{\mathsf{VC}}, (\hat{c}_i^j)_{i\in[\gamma]})$$

where for $i \in [\gamma]$

$$(\hat{c}_i^j, \hat{d}_i^j) \leftarrow \mathsf{CS}_{\mathrm{leaf}}.\mathsf{Com}(\mathtt{pp}_{\mathrm{leaf}}, v_i^j).$$

By ρ-addition correctness of VC, it thus holds that in the first step of the verification algorithm

$$\hat{c} = \mathsf{VC.Vf}(\mathtt{pp}_{\mathsf{VC}}, \sum_{j\in[\rho']} c_j, i, \sum_{j\in[\rho']} d_i^j) = \sum_{j\in[\rho']} \hat{c}_i^j$$

and by ρ-addition correctness of $\mathsf{CS}_{\mathrm{leaf}}$, it then holds that the verification algorithm outputs

$$\mathsf{CS}_{\mathrm{leaf}}.\mathsf{Vf}(\mathtt{pp}_{\mathrm{leaf}}, \hat{c}, \hat{d}) = \mathsf{CS}_{\mathrm{leaf}}.\mathsf{Vf}(\mathtt{pp}_{\mathrm{leaf}}, \sum_{j\in[\rho']} \hat{c}_i^j, \sum_{j\in[\rho']} \hat{d}_i^j) = \sum_{j\in[\rho']} v_i^j$$

as required. □

Lemma 20. *If* CS_{leaf} *is statistically* ρ*-addition hiding, then* VC_{hide} *is also statistically* ρ*-addition hiding.*

Proof. Let $\mathcal{A}$ be an arbitrary adversary against the statistical ρ-addition hiding of $\mathsf{VC}_{\mathrm{hide}}$. We define the hybrid games H_ι for $\iota \in [0, \gamma]$ in Figure 5.

Observe that hybrid 0 corresponds to the security experiment with challenge bit $b = 1$ and hybrid γ corresponds to the security experiment with challenge bit $b = 0$. During intermediate hybrid ι, we switch the $(\iota + 1)$-st entry in each challenge vector from the values in $\boldsymbol{v}^{1,1}, \ldots, \boldsymbol{v}^{\rho',1}$ to those in $\boldsymbol{v}^{1,0}, \ldots, \boldsymbol{v}^{\rho',0}$. From the security experiment, we know that for all $(j, i) \in [\rho'] \times C$, we have

$\mathsf{H}_\iota(1^\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$((\boldsymbol{v}^{1,0}, \ldots, \boldsymbol{v}^{\rho',0}), (\boldsymbol{v}^{1,1}, \ldots, \boldsymbol{v}^{\rho',1}), C) \leftarrow \mathcal{A}(\mathsf{pp})$

if $\rho' > \rho$

or $\exists (j, b) \in [\rho'] \times \{0, 1\}.\ \|\boldsymbol{v}^{j,b}\| \neq \gamma$

or $\exists (j, i) \in [\rho'] \times C.\ v_i^{j,0} \neq v_i^{j,1}$

or $\sum_{j \in [\rho']} \boldsymbol{v}^{j,0} \neq \sum_{j \in [\rho']} \boldsymbol{v}^{j,1}$

return 0

$\hat{\boldsymbol{v}}_i := (v_1^{i,0}, \ldots, v_\iota^{i,0}, v_{\iota+1}^{i,1}, \ldots, v_{1,\gamma}^1)^\intercal\ \forall i \in [\rho']$

$(c_i, \boldsymbol{d}^i) \leftarrow \mathsf{Com}(\mathsf{pp}, \hat{\boldsymbol{v}}_i)\ \forall i \in [\rho']$

return $\mathcal{A}\Big(\big(c_i, \mathsf{filter}(\boldsymbol{d}^i, C)\big)_{i \in [\rho']}, \sum_{i \in [\rho']} \boldsymbol{d}^i\Big)$

Fig. 5. Intermediate hybrids used in the proof of Lemma 20.

that $v_i^{j,0} = v_i^{j,1}$ and thus the hybrids ι and $\iota + 1$ are perfectly indistinguishable, when $\iota + 1 \in C$. For each ι with $\iota + 1 \notin C$, we have that

$$\sum_{j \in [\rho']} v_{\iota+1}^{j,0} = \sum_{j \in [\rho']} v_{\iota+1}^{j,1}$$

therefore we can apply the statistical ρ-addition hiding of $\mathsf{CS}_{\text{leaf}}$ to the two message sequences $(v_t^{j,0})_{j \in [\rho']}$ and $(v_t^{j,1})_{j \in [\rho']}$. More explicitly, a distinguisher generates the leaf commitments and decommitments at every position other than t locally, and uses its $\mathsf{CS}_{\text{leaf}}$ challenge at position t and thus statistical indistinguishability of the hybrids follows from the statistical addition hiding of the leaf commitment. □

Lemma 21. *If CS_{leaf} is binding and VC is position binding, then VC_{hide} is position binding.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the position binding game and let win be the event that the adversary wins the game, i.e.

$$\Pr[\mathsf{win}] = \Pr\left[\begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (i, c, (d_0, \hat{d}_0), (d_1, \hat{d}_1)) \leftarrow \mathcal{A}(\mathsf{pp}) \\ m_0 := \mathsf{Vf}(\mathsf{pp}, i, c, (d_0, \hat{d}_0)) \\ m_1 := \mathsf{Vf}(\mathsf{pp}, i, c, (d_1, \hat{d}_1)) \end{array} : \bot \notin \{m_0, m_1\} \wedge m_0 \neq m_1\right]$$

$$= \Pr\left[\begin{array}{r} (\mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\text{leaf}}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (i, c, (d_0, \hat{d}_0), (d_1, \hat{d}_1)) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \hat{c}_0 \leftarrow \mathsf{VC}.\mathsf{Vf}(\mathsf{pp}_{\mathsf{VC}}, c, i, d_0) \\ m_0 \leftarrow \mathsf{CS}_{\text{leaf}}.\mathsf{Vf}(\mathsf{pp}_{\text{leaf}}, \hat{c}_0, \hat{d}_0) \\ \hat{c}_1 \leftarrow \mathsf{VC}.\mathsf{Vf}(\mathsf{pp}_{\mathsf{VC}}, c, i, d_1) \\ m_1 \leftarrow \mathsf{CS}_{\text{leaf}}.\mathsf{Vf}(\mathsf{pp}_{\text{leaf}}, \hat{c}_1, \hat{d}_1) \end{array} : \bot \notin \{m_0, m_1\} \wedge m_0 \neq m_1\right]$$

Then we have that

$$\Pr[\mathsf{win}] \leq \Pr\left[\begin{array}{c}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{leaf}}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (i, c, (d_0, \hat{d}_0), (d_1, \hat{d}_1)) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \hat{c}_0 \leftarrow \mathsf{VC.Vf}(\mathsf{pp}_{\mathsf{VC}}, c, i, d_0) \\ \hat{c}_1 \leftarrow \mathsf{VC.Vf}(\mathsf{pp}_{\mathsf{VC}}, c, i, d_1)\end{array} : \hat{c}_0 \neq \hat{c}_1 \wedge \perp \notin \{\hat{c}_0, \hat{c}_1\}\right] + \Pr[\mathsf{win} \wedge \hat{c}_0 = \hat{c}_1] \leq \mathsf{negl}(\lambda),$$

where the first term in the last equation is negligible due to the binding property of the vector commitment and the second term is negligible due to the binding property of the leaf commitment. □

5 Key Additive Homomorphic Encryption

In this section, we formally define and construct key-additively homomorphic encryption. The conceptual notion itself originates from the work of Applebaum et al. [AIKW13] and our construction is very close to the construction of Bell-Clark et al. [BCGL+25]. We provide a full rigorous proof of security for the presented construction.

5.1 Definition

We slightly strengthen the definition Bell-Clark et al. by allowing the adversary to choose messages based on the public parameters. Below, we define addition correctness and privacy for key-additive homomorphic encryption schemes.

Definition 15 (Key Additive Homomorphic Encryption). *A key additively homomorphic encryption scheme* KAHE *with message space* A_{msg}*, key space* $\mathcal{K} \subseteq A_{\mathsf{key}}$ *and ciphertext space* A_{ciph} *is comprised of the following algorithms:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *is a PPT algorithm that takes the security parameter* $\lambda \in \mathbb{N}$ *as input and returns public parameters* pp*.*

$\mathsf{Gen}(\mathsf{pp}) \to k$ *is a PPT algorithm that takes public parameter* pp *and returns a key* $k \in \mathcal{K}$*.*

$\mathsf{Enc}(\mathsf{pp}, k, m) \to c$ *is a PPT algorithm that takes public parameters* pp*, a key* $k \in \mathcal{K}$*, and a message* $m \in A_{\mathsf{msg}}$ *as input and returns a ciphertext* $c \in A_{\mathsf{ciph}}$*.*

$\mathsf{Dec}(\mathsf{pp}, k, c) \to m$ *is a PPT algorithm that takes public parameters* pp*, a key* $k \in \mathcal{K}$*, and a ciphertext* $c \in A_{\mathsf{ciph}}$ *as input and returns a message* $m \in A_{\mathsf{msg}}$*.*

Addition Correctness. Let $\rho \in \mathbb{N}$*. A key additively homomorphic encryption scheme is* (ρ, ε)*-addition correct if for all* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{rl}\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) & \\ (m_1, \ldots, m'_\rho) \leftarrow \mathcal{A}(\mathsf{pp}) & \rho' \leq \rho \\ k_j \leftarrow \mathsf{Gen}(\mathsf{pp})\ \forall j \in [\rho'] & \wedge\ m_j \in A_{\mathsf{msg}}\ \forall j \in [\rho'] \\ c_j \leftarrow \mathsf{Enc}(\mathsf{pp}, k_j, m_j)\ \forall j \in [\rho'] & \wedge\ m \neq \sum_{j \in [\rho']} m_j \\ m \leftarrow \mathsf{Dec}\Big(\mathsf{pp}, \sum_{j \in [\rho']} k_j, \sum_{j \in [\rho']} c_j\Big) & \end{array} : \right] \leq 2^{-\varepsilon}$$

where the probability is taken over the random coins of Setup, Gen,Enc *and* $\mathcal{A}$*.*

Addition Privacy. Let $\rho \in \mathbb{N}$. A key additively homomorphic encryption scheme is ρ-addition private, if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ ((m_1^0,\ldots,m_{\rho'}^0),(m_1^1,\ldots,m_{\rho'}^1)) \leftarrow \mathcal{A}(\mathsf{pp}) \\ k_j \leftarrow \mathsf{Gen}(\mathsf{pp})\ \forall j \in [\rho'] \\ b \leftarrow \{0,1\} \\ c_j \leftarrow \mathsf{Enc}(\mathsf{pp}, k_j, m_j^b)\ \forall j \in [\rho'] \\ b' \leftarrow \mathcal{A}\Big((c_1,\ldots,c_{\rho'}), \sum_{j\in[\rho']} k_j\Big) \end{array} : \begin{array}{l} \rho' \le \rho \\ \wedge \sum_{j\in[\rho']} m_j^0 = \sum_{j\in[\rho']} m_j^1 \\ \wedge\ b' = b \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

5.2 Construction

We are now ready to present our construction in Figure 6 and prove it secure in Theorem 22.

$\mathsf{Setup}(1^\lambda)$	$\mathsf{Gen}(\mathsf{pp})$	$\mathsf{Enc}(\mathsf{pp}, k, \boldsymbol{m})$	$\mathsf{Dec}(\mathsf{pp}, k, \boldsymbol{c})$
$\boldsymbol{a} \leftarrow \mathcal{R}_q^\mu$	$k \leftarrow \mathcal{D}_\sigma$	$\boldsymbol{e} \leftarrow \mathcal{D}_\sigma^\mu$	$\boldsymbol{m} := \boldsymbol{c} - \boldsymbol{a} \cdot k \bmod t$
return $\boldsymbol{a}$	**return** k	$\boldsymbol{c} := \boldsymbol{a} \cdot k + t \cdot \boldsymbol{e} + \boldsymbol{m}$	**return** $\boldsymbol{m}$
		return $\boldsymbol{c}$	

Fig. 6. Key-additive homomorphic encryption scheme.

Theorem 22. *Let $n, \mu, q, t, \rho, \nu, \sigma$ be positive integers such that n is a power of two,*

$$q > t(2\sigma\sqrt{2\rho((\varepsilon+1)\ln(2) + \ln(n\mu))} + \rho)$$

is a prime, and $\sigma \ge 2\sqrt{2}\eta_{2^{-\lambda}}(\mathbb{Z}^n)$. If the $\mathsf{LWE}_{\mathcal{R},q,\mathcal{D}_{\sigma/2}}$ problem is hard, then KAHE from Figure 6 is ρ-addition correct and ρ-addition private.

6 Anonymous Broadcast

6.1 Security Model

In this work we consider stand-alone security. We assume the existence of secure point-to-point channels as well as a broadcast channel. In our model a party can be either honest, actively corrupt or fail-stop corrupt. We assume all corruptions are static in the sense that the set of corrupt parties is determined before the protocol starts. The behavior of actively corrupt parties is fully controlled by the adversary. Fail-stop corrupt parties share their internal state with the adversary but behave honestly in the protocol. However, the adversary is allowed to specify, in every communication round, whether or not each fail-stop corrupt party will cease to participate in the protocol. The decision can be made *after* the adversary has seen the messages that *would* have been sent.[6]

[6] We note that our notion of fail-stop corruption is – consistent with some previous work (e.g.[MRY23]) – strictly *stronger* than passive corruptions. In some other previous work (e.g. [HM20]) fail-stop corrupted parties explicitely do *not* share their internal state with the adversary. In these models our notion corresponds to *simultaneous* passive and fail-stop corruption of the same party.

We define security in the real/ideal-world security framework. Here we compare a real protocol execution with an ideal world, where the parties have access to a trusted party, called the ideal functionality, that computes some function on their behalf. Intuitively, we say a protocol securely instantiates some ideal functionality, if no adversary in the real world can do more harm than an adversary, called the simulator, can do in the ideal world.

To define security formally, we need to introduce some notation and nomenclature first. The *view* of a party in a real-world protocol execution, is comprised of their input, their random tape, and the messages they receive during a protocol execution. The view of a set of parties, is the union of the individual parties' views. With $\mathsf{REAL}\left[1^\lambda, \Pi, \mathcal{A}, \boldsymbol{m}, z\right]$ we denote the outputs of all honest parties along with the output of the adversary $\mathcal{A}$ in an execution of protocol Π on input vector $\boldsymbol{m}$ with security parameter λ and auxiliary input z. With $\mathsf{IDEAL}\left[1^\lambda, \mathcal{F}, \mathcal{S}, \boldsymbol{m}, z\right]$ we correspondingly denote the outputs of all honest parties along with output of the simulator in an ideal-world protocol execution.

Definition 16. *We say protocol Π securely instantiates ideal functionality $\mathcal{F}$ in the presence of corruption class $\mathbb{A}$, if for all PPT real-world adversaries $\mathcal{A} \in \mathbb{A}$, there exists an ideal-world PPT adversary $\mathcal{S}$, such that for all $\lambda \in \mathbb{N}$, all message vectors $\boldsymbol{m}$, and auxiliary inputs $z \in \{0,1\}^*$, it holds that*

$$\mathsf{REAL}\left[1^\lambda, \Pi, \mathcal{A}, \boldsymbol{m}, z\right] \approx \mathsf{IDEAL}\left[1^\lambda, \mathcal{F}, \mathcal{S}, \boldsymbol{m}, z\right].$$

6.2 Ideal Functionalities

Let us define the ideal functionalities relevant for us. In Definition 17, we define a public-key infrastructure, since we assume all parties have an associated public key. In Definition 18 and Definition 19, we define two flavors of anonymous broadcast with guaranteed output delivery.

In Definition 18, we define the strongest notion of anonymous broadcast, where the adversary learns no information about which client sent which message and additionally malicious messages may not depend on the honest messages. In Definition 19, we relax the latter requirement and allow the adversary to see the set of honest messages, before choosing their own input messages. In Section 6.3, we show how to construct our notion of relaxed anonymous broadcast and in Section 6.4 we then show how relaxed anonymous broadcast can easily be transformed into the full anonymous broadcast via extractable, equivocable commitments [Di 03].

Definition 17 (Public Key Infrastructure). *Let Gen be a key generation algorithm. Let $S_1, S_2, \ldots, S_\gamma$ be participating servers and let $C_1, C_2, \ldots, C_\rho$ be participating clients such that an adversary $\mathcal{A}$ controls some subset of parties. Then, the ideal public key infrasctructure denoted by $\mathcal{F}_{PKI}$ proceeds as follows:*

1. *Functionality $\mathcal{F}_{PKI}$ initialize an empty set K.*
2. *For every honest server S_i, $\mathcal{F}_{PKI}$ samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ and adds (i, pk_i) to K.*
3. *The functionality $\mathcal{F}_{PKI}$ sends K to the adversary and waits to receive public keys from the corrupted servers.*
4. *The adversary instructs each corrupted server S_i to send a public key pk_i.*
5. *Upon receiving each pk_i, $\mathcal{F}_{PKI}$ adds (i, pk_i) to K.*
6. *Once all public keys are received from all malicious servers, $\mathcal{F}_{PKI}$ outputs sk_i to each honest server S_i and K to each honest client C_j as well as to the ideal world adversary.*

Definition 18 (Full Anonymous Broadcast with Guaranteed Output Delivery). *Let $\mathcal{M}$ be a totally ordered set. Let $C_1, C_2, \ldots, C_\rho$ be participating clients where C_i has secret message $m_i \in \mathcal{M}$ such that an adversary $\mathcal{A}$ controls some subset of clients. Let R be the recipient. Then, the ideal anonymous broadcast with guaranteed output delivery functionality denoted by $\mathcal{F}_{AB\text{-}GOD}$ proceeds as follows:*

1. *Functionality $\mathcal{F}_{AB\text{-}GOD}$ initialize an empty multiset M and waits for a message m_i from each client.*
2. *The adversary $\mathcal{A}$ can instruct each corrupted client to send any message from $\mathcal{M} \cup \{\perp\}$ where $\perp$ is a special character indicating an invalid input message.*
3. *Upon receiving each m_i, $\mathcal{F}_{AB\text{-}GOD}$ checks if $m_i = \perp$. If so, drop it. Otherwise, add m_i to M.*
4. *Once all messages are received from all participating clients, $\mathcal{F}_{AB\text{-}GOD}$ sorts M and sends the messages to R in sorted order.*

Definition 19 (Relaxed Anonymous Broadcast with Guaranteed Output Delivery). *Let $\mathcal{M}$ be a totally ordered set. Let $C_1, C_2, \ldots, C_\rho$ be participating clients where C_i has secret message $m_i \in \mathcal{M}$ such that an adversary $\mathcal{A}$ controls some subset of clients. Let R be the recipient. Then, the ideal relaxed anonymous broadcast with guaranteed output delivery functionality denoted by $\mathcal{F}_{RAB\text{-}GOD}$ proceeds as follows:*

1. *Functionality $\mathcal{F}_{RAB\text{-}GOD}$ initialize an empty multiset M and waits for a message m_i from each* honest *client.*
2. *Upon receiving each m_i, add m_i to M.*
3. *Once all messages are received from all honest clients, $\mathcal{F}_{RAB\text{-}GOD}$ sorts M and sends the messages to $\mathcal{A}$ in sorted order and waits for a message m_i from each* corrupted *client.*
4. *The adversary $\mathcal{A}$ can instruct each corrupted client to send any message from $\mathcal{M} \cup \{\perp\}$ where $\perp$ is a special character indicating an invalid input message.*
5. *Upon receiving each m_i, $\mathcal{F}_{RAB\text{-}GOD}$ checks if $m_i = \perp$. If so, drop it. Otherwise, add m_i to M.*
6. *Once all messages are received from all corrupted clients, $\mathcal{F}_{RAB\text{-}GOD}$ sorts M and sends the messages to R in sorted order.*

6.3 Constructing Relaxed Anonymous Broadcast

1. Initialize $\mathcal{F}_{PKI}$. Each honest client C_j receives $\{(1, \mathsf{pk}_1), \ldots, (\gamma, \mathsf{pk}_\gamma)\}$ and each honest server S_i receives sk_i.
2. Each honest client C_j, given a message m_j, runs

$$(\mathsf{ct}_j, \mathsf{vc}_j, \hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j) \leftarrow \mathsf{Client}(\mathsf{sid}, j, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), (\mathsf{pk}_1, \ldots, \mathsf{pk}_\gamma), m_j)$$

   and broadcasts $\mathsf{ct}_j, \mathsf{vc}_j, (\hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)$.
3. Each honest server S_i, receives $(\mathsf{ct}^j, \mathsf{vc}^j, \hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)_{j \in [\rho]}$ on the broadcast channel, runs

$$(d_i, I_i) \leftarrow \mathsf{Server}\left(\mathsf{sid}, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), \mathsf{sk}_i, (\mathsf{ct}^j, \mathsf{vc}^j, \hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)_{j \in [\rho]}\right)$$

   and broadcasts (d_i, I_i).

$\mathsf{Client}\big(\mathsf{sid}, j, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), (\mathsf{pk}_1, \ldots, \mathsf{pk}_\gamma), m\big)$

$y := \mathsf{MSE.Encode}(\mathsf{pp}_{\mathrm{mse}}, \{m\})$
$k \leftarrow \mathsf{KAHE.Gen}(\mathsf{pp}_{\mathrm{kahe}})$
$\mathsf{ct} := \mathsf{KAHE.Enc}(k, y)$
$(k_1, \ldots, k_\gamma) \leftarrow \mathsf{SSS.Share}(k)$
$(\mathsf{vc}, (d_1, \ldots, d_\gamma)) \leftarrow \mathsf{VC.Com}(\mathsf{pp}_{\mathsf{VC}}, (k_1, \ldots, k_\gamma))$
for $i \in [\gamma]$
  $\hat{\mathsf{ct}}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, (\mathsf{sid}, j, d_i))$
return $(\mathsf{ct}, \mathsf{vc}, (\hat{\mathsf{ct}}_1, \ldots, \hat{\mathsf{ct}}_\gamma))$

$\mathsf{Recipient}\big(\mathsf{sid}, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), (\mathsf{ct}^j, \mathsf{vc}^j)_{j\in[\rho']}, (d_i, I_i)_{i\in[\gamma]}\big)$

$H := \{\}, I := \perp$
for $i \in [\gamma]$
  $H[I_i] := H[I_i] \cup \{i\}$
  if $|H[I_i]| > \gamma/2$
    $I := I_i$
if $I = \perp$
  return $\perp$
$\mathsf{vc} := 0, \mathsf{ct} := 0$
for $j \in I$:
  $\mathsf{vc} := \mathsf{vc} + \mathsf{vc}^j$
  $\mathsf{ct} := \mathsf{ct} + \mathsf{ct}^j$
for $i \in H[I]$
  $k_i \leftarrow \mathsf{VC.Vf}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{vc}, i, d_i)$
$k := \mathsf{SSS.Decode}((k_i, \ldots, k_\gamma))$
$y := \mathsf{KAHE.Dec}(k, \mathsf{ct})$
$X \leftarrow \mathsf{MSE.Decode}(\mathsf{pp}_{\mathrm{mse}}, y)$
if $X = \perp$ **or** $|X| \neq |I|$
  return $\perp$
return $\mathsf{sort}(X)$

$\mathsf{Server}\Big(\mathsf{sid}, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), \mathsf{sk}_i, (\mathsf{ct}^j, \mathsf{vc}^j, \hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)_{j\in[\rho']}\Big)$

$d_i := 0$
$I_i := \emptyset$
for $j \in [\rho']$:
  if $\mathsf{ct}^j \neq \perp$ **and** $\mathsf{vc}^j \neq \perp$ **and** $\forall i' \in [\gamma].\ \hat{\mathsf{ct}}_{i'}^j \neq \perp$
    $(\mathsf{sid}^j, j', d_i^j) \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_i, \hat{\mathsf{ct}}_i^j)$
    if $j' = j$ **and** $\mathsf{sid}^j = \mathsf{sid}$ **and** $d_i^j \neq \perp$:
      $d_i := d_i + d_i^j$
      $I_i := I_i \cup \{j\}$
return (d_i, I_i)

Fig. 7. Helper protocols for our anonymous broadcast protocol.

4. The recipient, receives $((\mathsf{ct}^j, \mathsf{vc}^j, (\hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)))_{j\in[\rho]}$ and $((d_i, I_i))_{i\in[\gamma]}$ on the broadcast channel. The recipient then runs

$$\hat{\boldsymbol{m}} \leftarrow \mathsf{Recipient}\left(\mathsf{sid}, (\mathsf{pp}_{\mathrm{kahe}}, \mathsf{pp}_{\mathsf{VC}}, \mathsf{pp}_{\mathrm{mse}}), ((\mathsf{ct}^j, \mathsf{vc}^j))_{j\in[\rho]}, (d_i, I_i))_{i\in[\gamma]}\right)$$

and outputs $\hat{\boldsymbol{m}}$.

Theorem 23. *The protocol specified above securely instantiates ideal functionality $\mathcal{F}_{RAB\text{-}GOD}$ in the $\mathcal{F}_{PKI}$-hybrid model against an adversary that fail-stop corrupts up to $\rho - 2$ clients and fully corrupts less than $\gamma/2$ server.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT real-world adversary against the protocol specified above. Let U be the set of parties corrupted by $\mathcal{A}$. We construct an ideal world adversary $\mathcal{S}$ against $\mathcal{F}_{RAB\text{-}GOD}$ as follows:

1. Initialize an empty set K.
2. For each $S_i \notin U$, sample sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ and add (i, pk_i) to K.
3. At the end of the client-broadcast round, receive the broadcast of every $C_j \in U$ that participated in the round. For every $C_j \in U$ that did not broadcast, set

$$(\mathsf{ct}^j, \mathsf{vc}^j, \hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j) := \perp.$$

4. Send K to $\mathcal{A}$ and wait to receive keys from corrupted servers.
5. Upon receiving pk_i from $S_i \in U$ add (i, pk_i) to K.
6. Receive M from $\mathcal{F}_{RAB\text{-}GOD}$.
7. Let

$$V := \{j \in [\rho] : C_j \notin U\},$$

and let $j^* := \min V$.

8. For every $j \in V$, sample an encryption key

$$k^j \leftarrow \mathsf{KAHE.Gen}(\mathsf{pp}_{\mathrm{kahe}}).$$

Define the values that will be secret-shared and committed to by

$$\widetilde{k}^j := \begin{cases} 0, & \text{if } j \neq j^*, \\ \sum_{\ell\in V} k^\ell, & \text{if } j = j^*. \end{cases}$$

For every $C_j \notin U$, compute the following:

(a) If $j \neq j^*$, set

$$y^j := \mathsf{MSE.Encode}(\mathsf{pp}_{\mathrm{mse}}, \emptyset).$$

If $j = j^*$, set

$$y^j := \mathsf{MSE.Encode}(\mathsf{pp}_{\mathrm{mse}}, M).$$

(b) Compute

$$\mathsf{ct}^j := \mathsf{KAHE.Enc}(k^j, y^j), \qquad (\widetilde{k}_1^j, \ldots, \widetilde{k}_\gamma^j) \leftarrow \mathsf{SSS.Share}(\widetilde{k}^j),$$

and

$$\left(\mathsf{vc}^j, (d_1^j, \ldots, d_\gamma^j)\right) \leftarrow \mathsf{VC.Com}\left(\mathsf{pp}_{\mathsf{VC}}, (\widetilde{k}_1^j, \ldots, \widetilde{k}_\gamma^j)\right).$$

(c) For every $i \in [\gamma]$, compute

$$\hat{\mathsf{ct}}_i^j := \begin{cases} \mathsf{PKE.Enc}(\mathsf{pk}_i, (\mathsf{sid}, j, d_i^j)), & \text{if } S_i \in U, \\ \mathsf{PKE.Enc}(\mathsf{pk}_i, (0,0,0)), & \text{if } S_i \notin U. \end{cases}$$

(d) Send

$$\left(\mathsf{ct}^j, \mathsf{vc}^j, (\hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j)\right)$$

to $\mathcal{A}$.

9. Wait to receive $(\mathsf{ct}^j, \mathsf{vc}^j, (\hat{\mathsf{ct}}_1^j, \ldots, \hat{\mathsf{ct}}_\gamma^j))$ for each $C_j \in U$.
10. Simulate the broadcasts of the honest servers as follows. For every $S_i \notin U$, initialize

$$d_i := 0 \qquad \text{and} \qquad I_i := \emptyset.$$

For every simulated honest client $C_j \notin U$, use the opening d_i^j retained when generating its vector commitment, and set

$$d_i := d_i + d_i^j \qquad \text{and} \qquad I_i := I_i \cup \{j\}.$$

For every corrupted client $C_j \in U$, process its broadcast exactly as the honest server would. In particular, if

$$\mathsf{ct}^j \neq \bot, \qquad \mathsf{vc}^j \neq \bot, \qquad \text{and} \qquad \hat{\mathsf{ct}}_{i'}^j \neq \bot \quad \text{for every } i' \in [\gamma],$$

compute

$$(\mathsf{sid}^j, j', d_i^j) \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_i, \hat{\mathsf{ct}}_i^j).$$

If

$$\mathsf{sid}^j = \mathsf{sid}, \qquad j' = j, \qquad \text{and} \qquad d_i^j \neq \bot,$$

set

$$d_i := d_i + d_i^j \qquad \text{and} \qquad I_i := I_i \cup \{j\}.$$

Finally, send (d_i, I_i) to $\mathcal{A}$ for every honest server $S_i \notin U$.

11. For each $C_j \in U$: If $\mathsf{ct}^j = \bot$, $\mathsf{vc}^j = \bot$ or $\hat{\mathsf{ct}}_{i'}^j = \bot$ for any $i' \in [\gamma]$, ignore C_j's messages and set $m^j := \bot$. Otherwise, for each $S_i \notin U$ compute $(\mathsf{sid}^j, j', d_i^j) \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_i, \hat{\mathsf{ct}}_i^j)$ and $k_i^j \leftarrow \mathsf{Vf}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{vc}^j, i, d_i^j)$. For each $S_i \in U$, define $k_i^j := \bot$. Compute $k^j := \mathsf{SSS.Decode}((k_i^j, \ldots, k_\gamma^j))$, $y^j := \mathsf{KAHE.Dec}(k^j, \mathsf{ct}^j)$, and $\{m^j\} \leftarrow \mathsf{MSE.Decode}(\mathsf{pp}_{\mathrm{mse}}, y^j)$.
12. For each $C_j \in U$ send m^j to $\mathcal{F}_{RAB\text{-}GOD}$.
13. Output whatever $\mathcal{A}$ outputs.

We prove that the output of the ideal-world adversary is indistinguishable from that of the real-world adversary via a sequence of hybrids:

H_0 : In the first hybrid, we consider an execution of the protocol in the real world.

H_1 : We first equivalently implement the computation of every honest server as follows. For ciphertexts generated by honest clients, the experiment retains the corresponding vector-commitment openings and uses those retained openings when computing the honest server's broadcast. By correctness of the PKE scheme, this gives the same output as decrypting the honestly generated ciphertexts.

We then replace, for every honest client $C_j \notin U$ and every honest server $S_i \notin U$,

$$\hat{\mathsf{ct}}_i^j = \mathsf{PKE.Enc}(\mathsf{pk}_i, (\mathsf{sid}, j, d_i^j))$$

with

$$\hat{\mathsf{ct}}_i^j = \mathsf{PKE.Enc}(\mathsf{pk}_i, (0, 0, 0)).$$

The broadcast of an honest server continues to be computed using the retained opening d_i^j for every honest client. Ciphertexts submitted by corrupted clients remain unchanged and are decrypted exactly as in the real protocol.
Indistinguishability from the previous hybrid follows from the security of the PKE scheme.

H_2 : In this hybrid, we change how honest parties compute the secret sharing of their encryption key. Each honest client, first picks the shares of malicious servers uniformly at random and then completes this unqualified set of shares to a valid secret sharing of the desired value. This hybrid is a semantic change, thus perfectly indistinguishable from the previous hybrid.

H_3 : Let

$$T := \{i \in [\gamma] : S_i \in U\}$$

be the set of corrupted-server positions, and write

$$\boldsymbol{k}^j := (k_1^j, \ldots, k_\gamma^j)$$

for the sharing vector of honest client C_j in the previous hybrid.
For every honest client C_j with $j \neq j^*$, sample a sharing $\widetilde{\boldsymbol{k}}^j$ of zero conditioned on

$$\widetilde{k}_i^j = k_i^j \qquad \text{for every } i \in T.$$

Define

$$\widetilde{\boldsymbol{k}}^{j^*} := \sum_{j \in V} \boldsymbol{k}^j - \sum_{j \in V \setminus \{j^*\}} \widetilde{\boldsymbol{k}}^j.$$

By linearity of the secret-sharing scheme, $\widetilde{\boldsymbol{k}}^{j^*}$ is a sharing of

$$\sum_{j \in V} k^j.$$

Moreover, the old and new share vectors agree at every corrupted server position. Indeed, for every $i \in T$,

$$\widetilde{k}_i^{j^*} = \sum_{j \in V} k_i^j - \sum_{j \in V \setminus \{j^*\}} \widetilde{k}_i^j = k_i^{j^*}.$$

Their aggregate vectors are also exactly equal:

$$\sum_{j \in V} \widetilde{\boldsymbol{k}}^j = \sum_{j \in V} \boldsymbol{k}^j.$$

We replace the honest clients' vector commitments and corresponding openings by commitments and openings for $\widetilde{\boldsymbol{k}}^j$ for $j \in V$. Ciphertexts addressed to corrupted servers contain the corresponding new openings, and honest-server broadcasts are computed using the new aggregate openings. Ciphertexts addressed to honest servers remain encryptions of $(0, 0, 0)$.
Indistinguishability from the previous hybrid follows from the addition-hiding property of the vector commitment.

H_4 : In this hybrid, we change what is encrypted with the key-additively homomorphic encryption scheme. For each honest client j with $j \neq j^*$, compute $\mathsf{ct}^j \leftarrow \mathsf{KAHE.Enc}(k^j, 0)$. For j^*, we compute

$$\mathsf{ct}^{j^*} \leftarrow \mathsf{KAHE.Enc}\left(k^{j^*}, \sum_{i \in V} y^i\right).$$

Indistinguishability of this hybrid from the previous one follows from the addition privacy of the key-additively homomorphic encryption scheme.

H_5 : In this hybrid, we change the multiset encodings of honest parties. For each honest $j \neq j^*$, we compute $y^j = \mathsf{MSE.Encode}(\mathsf{pp}_{\mathrm{mse}}, \emptyset)$. For j^*, we compute

$$y^{j^*} = \mathsf{MSE.Encode}(\mathsf{pp}_{\mathrm{mse}}, M).$$

At this point, the experiment is identical to the simulator.

6.4 From Relaxed to Full Anonymous Broadcast

The above broadcast protocol only satisfies the relaxed notion of anonymous broadcast. The reason for this is that the simulator needs to reveal ct_i for each honest client i before the (rushing) adversary reveals their ciphertexts. Ideally, we would hope that we can program the output of the protocol in the last round, but the communication required to reveal the sum of encryption keys is much smaller than the amount of information that needs to be programmed, which is the sorted list of honest messages.

There is, however, an easy way to lift a relaxed anonymous broadcast protocol to a full anonymous broadcast using equivocable, extractable commitments [Di 03]. To do this, we run the relaxed anonymous broadcast protocol twice sequentially. In the first execution, each party anonymously sends a commitment to their message. In the second execution, each party sends the opening to their commitment.

After the first execution, the simulator can extract the adversary's messages from the sent commitments and send them to the ideal functionality to recover the honest parties' messages. In the second execution, the simulator can equivocate the honest commitments to reveal the the correct honest parties' messages.

Corollary 24. *The protocol specified above securely instantiates ideal functionality $\mathcal{F}_{AB\text{-}GOD}$ in the $\mathcal{F}_{PKI}$-hybrid model against an adversary that fail-stop corrupts up to $\rho - 2$ clients and fully corrupts less than $\gamma/2$ server.*

7 Benchmarks

7.1 Experimental Setup

We conducted the benchmarks on a 503 GB memory TDX-enabled machine with 16-core 3.9 GHz Intel Xeon Gold 6526Y processor. The operating system for all benchmarks is Ubuntu 26.04 with kernel 7.0.0-22-generic. The client benchmarks were run on a 64 GB, 8vCPU TDX VM with Qemu on the same machine. For the benchmark all workloads were core-pinned, including the TDX VM vCPUs, and all benchmarks ran using 8 threads. We are simulating network delays and throughput for realistic throughput estimation.

7.2 Practical Optimizations

There are several practical optimizations that can be applied to the provably secure protocol from Section 6. Let us shortly highlight them here:

Cover Traffic. The formal description of our anonymous broadcast protocol assumes that each client has a message to send. This may not necessarily be the case or the protocol may simply be too inefficient, when the number of clients is too large. A pragmatic solution is to reduce the number of clients that can participate in any one broadcast execution, but this comes at the cost of reducing the anonymity set.

Our protocol supports cover traffic from clients that do not actually send messages. For this, we set the capacity of each actively sending client's multiset encoding proportional to an upper bound on the number of actively sending clients. Clients that do not send a message, may still provide cover traffic by replacing their encryption of a multiset encoding by an encryption of the all zero string. This way, they contribute to the anonymity set, but do not inflate the size of each individual encoding and allow for a reduced bandwidth overhead.

Aggregators. The main bottleneck of our solution is the bandwidth overhead induced by the clients' encryptions of their respective mutliset encodings. To reduce the amount of data that needs to be sent through the network and received by the servers, our protocol may rely on untrusted aggregation services, which receive all ciphertexts from the clients and aggregate them before forwarding them to the network. Even if all aggregators are malicious, they cannot break the anonymity of our protocol. As long as at least one aggregator is honest, our protocol will either produce the correct output or detect that something went wrong. In case something went wrong, all parties can obtain all clients' ciphertexts and check which aggregator sent an incorrect aggregate.

Slot Reservation. Multiset encodings based on invertible bloom lookup tables as described in Section 2.5 are a small constant factor larger than the minimal size necessary for accommodating all messages. When the number of clients and messages are large, this quickly becomes a bandwidth bottleneck. Using two sequential broadcast executions, one can minimize the bandwidth overhead as follows: In the first broadcast execution, each client picks a random nonce and broadcasts this using our protocol. The clients observe the output of the first broadcast and sort all nonces. In the second broadcast execution, the clients do not use a multiset encoding, but instead encrypt an array with exactly ρ' message slots, where client i puts their message in the j-th slot, where j is the rank of their nonce in the sorted list of nonces.

7.3 Anonymous Broadcast Performance

We have implemented the protocol routines from the previous sections in Rust to measure the protocol's performance in terms of computation and bandwidth usage. The performance of the protocol depends mainly on the sizes of messages and number of clients, which increases the size of ciphertexts the clients produce as well as slightly increasing the client's computation time. Increasing the number of servers has a small but noticeable effect on the client's computation time and client's total egrees. In Table 1 we see the computation is fast even for very large broadcast channel, with even the 4.8 MB channel (300 clients, each sending a 16 KB message) finishing in just over 600ms. However, the messages grow large, and the total amount of data that needs to be transferred is considerable.

Message Size	# Clients	# Servers	Times (ms)			Message Size			
			Client	Server	Recipient	Cl.→Srv.	Cl.→Rec.	Srv.→Rec.	Rec. ingress
4 KB	100	8	33.4	15.4	19.5	0.47 MB	1.72 MB	109 KB	0.17 GB
4 KB	300/100	8	34.6	42.8	28.0	0.47 MB	1.72 MB	124 KB	0.52 GB
4 KB	300	8	60.1	43.8	76.7	0.47 MB	5.15 MB	124 KB	1.55 GB
16 KB	100	8	73.7	15.4	64.6	0.47 MB	6.84 MB	109 KB	0.69 GB
16 KB	300/100	8	73.3	42.9	93.9	0.47 MB	6.84 MB	124 KB	2.05 GB
16 KB	300	8	177.8	42.6	300.6	0.47 MB	20.5 MB	124 KB	6.15 GB
16 KB	300	16	200.9	48.3	303.5	1.12 MB	20.5 MB	147 KB	6.16 GB
16 KB	300	32	253.6	54.5	322.7	2.58 MB	20.5 MB	170 KB	6.16 GB

Table 1. CPU performance and wire transfer size benchmarks across message size, client set and server count. Computation times reflect the protocol routines.

Encoding	Ingest	Throughput (MB/s), by server—recipient link (Gbit/s)			
		1|1	4|4	1|8	1|20
Peeling	direct	0.050	0.194	0.376	0.865
	aggregated	0.525	1.279	0.713	0.736
	erasure	0.802	1.386	0.850	0.854
Newton	direct	0.142	0.540	1.011	2.119
	aggregated	1.334	2.783	1.748	1.796
	erasure	1.881	2.983	2.039	2.053
Slot res.	direct	0.131	0.505	0.963	2.118
	aggregated	1.300	2.920	1.738	1.789
	erasure	1.866	3.108	2.028	2.043

Table 2. Benchmarks for different network settings: throughput of recovered payload at 300 clients and 16 KB messages for a total of 4.8 MB plaintext size, 16 servers, with a 100 Mbit/s client uplink throughout.

7.4 Different Network Connections

For a realistic performance benchmark in Table 2 we simulate network delays with 60 ms latency, 10 ms jitter, and various link bandwidths (1Gbit, 4Gbit, and 20Gbit). For the purposes of the benchmark we assume there is just one recipient that has to ingest all of the messages to recover the broadcast output. We see that the modeled performance highly depends on the link bandwidth of the recipient, which is intuitive given how much data it has to ingest. The recipient has to download the ciphertext, which is already large, from each client. While the recipient downlink is the biggest bottleneck, with aggregation of ciphertext done by a separte set of parties, the server links quickly become the new bottleneck.

7.5 Individual Components

We examine the performance of our main building blocks, namely the key-additively homomorphic encryption and the vector commitment scheme. The benchmarks for the commitment scheme are in Table 3 and those for the encryption scheme are in Table 4.

Length	Sizes (KB)		Times (ms)	
	Commitment	Opening	Commitment	Verification
8	4.35	58.1	7.1	3.28
16	4.35	68.9	14.2	3.83
32	4.35	79.6	27.2	4.33

Table 3. Benchmarks of our vector commitments.

Plaintext	Sizes (KB)		Times (ms)	
	Ciphertext	Key	Encryption	Decryption
0.43 MB	0.60 MB	12.8	2.2	2.1
1.23 MB	1.72 MB	12.8	6.1	4.3
1.83 MB	2.55 MB	12.8	8.3	5.9
3.70 MB	5.15 MB	12.8	16.0	12.7
5.08 MB	7.05 MB	12.8	21.4	14.7
14.8 MB	20.5 MB	12.8	60.8	41.9

Table 4. Benchmarks of our key-additively homomorphic encryption, across the plaintext widths.

Acknowledgments. The authors would like to thank Katharina Boudgoust. Not all heroes wear capes, some just know lattices really well. The authors thank François-Xavier Wicht for helpful discussions during the initial stages of this protocol.

References

AIKW13. Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 166–184, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_10. 1.1, 5

Ajt99. Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99: 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9, Prague, Czech Republic, July 11–15, 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48523-6_1. 2.2

BCGL+25. James Bell-Clark, Adrià Gascón, Baiyu Li, Mariana Raykova, and Phillipp Schoppmann. Willow: Secure aggregation with one-shot clients. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part VIII*, volume 16007 of *Lecture Notes in Computer Science*, pages 285–318, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-032-01913-4_9. 1.1, 5

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385, Amalfi, Italy, September 5–7, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-98113-0_20. 1.1, 3.2

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press. doi:10.1145/62212.62213. 1.1

Di 03. Giovanni Di Crescenzo. Equivocable and extractable commitment schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 74–87, Amalfi, Italy, September 12–13, 2003. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-36413-7_6. 6.2, 6.4

DORS08. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. 38(1):97–139, 2008. doi:10.1137/060651380. 5

FHSZ23a. Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 386–400, Copenhagen, Denmark, November 26–30, 2023. ACM Press. doi:10.1145/3576915.3623219. 1.1, 4, 4.2, 1

FHSZ23b. Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. Cryptology ePrint Archive, Report 2023/1820, 2023. URL: https://eprint.iacr.org/2023/1820. 4.2

FLOS24. Nils Fleischhacker, Kasper Green Larsen, Maciej Obremski, and Mark Simkin. Invertible bloom lookup tables with less memory and randomness. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, Royal Holloway, London, United Kingdom, September 2-4, 2024*, volume 308 of *LIPIcs*, pages 54:1–54:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 1.1, 2.5, 2.5

GM11. Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *49th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2011, Allerton Park & Retreat Center, Monticello, IL, USA, 28-30 September, 2011*, pages 792–799. IEEE, 2011. 1.1, 2.5, 1, 2.5

GMPW20. Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 623–651, Edinburgh, UK, May 4–7, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-45374-9_21. 2.2

HM20. Martin Hirt and Marta Mularczyk. Efficient MPC with a mixed adversary. In Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs, editors, *ITC 2020: 1st Conference on Information-Theoretic Cryptography*, volume 163 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:23, Boston, MA, USA, June 17–19, 2020. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITC.2020.3. 6

KP25. Harish Karthikeyan and Antigoni Polychroniadou. sfOPA: One-shot private aggregation with single client interaction and its applications to federated learning. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part VIII*, volume 16007 of *Lecture Notes in Computer Science*, pages 319–353, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-032-01913-4_10. 1.1

LS18. Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 204–224, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78381-9_8. 2.2

Lyu11. Vadim Lyubashevsky. Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, 2011. URL: https://eprint.iacr.org/2011/537. 2.2

MRY23. Nikolas Melissaris, Divya Ravi, and Sophia Yakoubov. Threshold-optimal MPC with friends and foes. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *Progress in Cryptology - INDOCRYPT 2023: 24th International Conference in Cryptology in India, Part II*, volume 14460 of *Lecture Notes in Computer Science*, pages 3–24, Goa, India, December 10–13, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-56235-8_1. 6

RMK17. Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. P2P mixing and unlinkable Bitcoin transactions. In *ISOC Network and Distributed System Security Symposium – NDSS 2017*, San Diego, CA, USA, February 26 – March 1, 2017. The Internet Society. doi:10.14722/ndss.2017.23415. 2.5

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. doi:10.1145/359168.359176. 2.4, 6