

Network-Agnostic VSS with Quadratic Communication in Computational Setting

Yusong Yao, Qi Feng[✉], Cong Peng[✉], Min Luo, and Debiao He

School of Cyber Science and Engineering, Wuhan University
{yusongyao,fengqi,cpeng,mluo,hedebiao}@whu.edu.cn

Abstract. Verifiable secret sharing (VSS) is a core primitive in multiparty computation (MPC). Bhimrajka et al. [PKC’24, TIT’26] proposed the first computationally network-agnostic VSS and VSS-based MPC, which seamlessly accommodate both synchronous and asynchronous network models. However, their VSS requires Byzantine Agreement (BA) and incurs $O(n^5)$ bits of communication per sharing among n parties, and their MPC requires honest participation from nearly all parties.

We revisit the network-agnostic VSS architecture and propose a more efficient and general network-agnostic VSS structure. Our key technique is a new virtual-party strategy, which introduces more *honest* parties for reconstruction. With this technique, network-agnostic VSS is BA-free and performs as efficiently as synchronous VSS. Specifically, in terms of communication, our VSS costs $O(n^2)$ bits in the one-shot case and $O(n)$ bits in the round-by-round case with the dispute-control technique, yielding the first computationally network-agnostic VSS protocols with amortized linear communication complexity. Besides, we further address a series of subtle yet necessary hurdles in adapting existing results to network-agnostic models, including the erasure coding techniques in Reliable Broadcast (RBC), packed secret sharing techniques in VSS, and the correlated polynomial techniques in MPC.

Keywords: Verifiable Secret Sharing, Network-Agnostic, Multiparty Computation, Cryptographic Security

1 Introduction

The network-agnostic model [22] has attracted extensive attention (see, e.g., [18,19,38,22,8,47,48,37,17,59]), as it resists up to high corruptions under synchronous delivery while maintaining robustness under asynchronous delivery. In contrast, the classical synchronous model, common in current MPC, assumes a known upper bound on message delay. This simplifies design and provides a high corruption threshold, but is fragile in practice: any message exceeding

[✉] Corresponding authors: Cong Peng and Qi Feng.

that bound compromises security. The asynchronous model allows the adversary to arbitrarily delay messages, thereby providing better network robustness. However, asynchronous protocols must sacrifice threshold resilience, as theoretical analysis shows [32,15,13]. To overcome these bottlenecks, the network-agnostic model aims to achieve some network robustness for asynchronous network while maintaining a high threshold under synchronous network conditions. The network-agnostic model is proposed to more precisely capture the nature of real-world networks, i.e., the Internet typically operates under synchronous conditions most of the time but occasionally experiences asynchronous message arrivals. The model also applies to other practical scenarios with similar occasion, such as drone swarms where connections are occasionally dynamic, and multi-site command centers where networks are occasionally separated by gateways.

VSS enables robust secret sharing even with a malicious dealer and serves as a core primitive in MPC [67,44,14,54,68,51] for both synchronous and asynchronous networks. Beyond its foundational role in cryptography, MPC has found broad applications in practice [65,3,63]. However, network-agnostic VSS is still nascent and cannot directly leverage existing synchronous and asynchronous VSS results. Synchronous VSS can be divided into two types: NIZK-based [10,9,36] and reveal-based [40,11,49,43]. NIZK-based schemes require broadcasting the entire ciphertext along with the proof, which introduces additional assumptions (e.g., the Random Oracle), high computational cost, and relatively high communication cost of at least $O(n^2)$. Reveal-based schemes, on the other hand, treat all participants whose messages are delayed as malicious. This leads to false positives even with a single delayed message. On the other side, asynchronous VSS [1,46,16,36,45,52,33] typically introduces bivariate polynomials and employs honest parties to help others reconstruct the shares. However, in network-agnostic settings with potentially more corrupt parties, the honest dealer cannot find enough honest parties. In summary, network-agnostic VSS faces both asynchronous message and high-threshold challenges, which makes current network-agnostic VSS [18,19] rely on Byzantine Agreement (BA) and incur $O(n^5)$ overhead.

In this work, we propose a new virtual-party technique and present the first computational network-agnostic VSS protocol with amortized $O(n^2)$ communication overhead in the one-shot case and $O(n)$ in the round-by-round case. The virtual-party technique only requires the dealer to reveal the virtual parties' points, avoiding broadcasting any ciphertext or proof and preventing leakage of honest parties' shares. Moreover, with the help of virtual parties, the dealer can now gather enough honest parties for reconstruction in network-agnostic settings. We mainly propose two network-agnostic VSS protocols, Π_{VSS} and Π_{LVSS} . Π_{VSS} follows a clean and simple structure with quadratic communication, relying on DLog and hash functions. Π_{LVSS} achieves linear communication using only $O(n)$ round-by-round filtering, relying on the q-SDH assumption, the AGM adversarial model, and pairings. Both protocols work under the optimal threshold in the PKI model, without reliance on homomorphic encryption or

zero-knowledge proofs, similar to [18,19]. We further show that, with modifications, existing techniques can help achieve network-agnostic RBC with linear communication and MPC with quadratic communication.

The primary challenge in the network-agnostic VSS is the floating threshold from t_a to t_s . Let the adversary corrupt $t_s < n/2$ parties in the synchronous setting and $t_a < n/3$ parties in the asynchronous setting with $2t_s + t_a + 1 \leq n$. If we set the degree of the shared polynomial as t_a (which is low), the secret leaks if the adversary corrupts t_s parties. If we set the degree as t_s (which is high), the asynchronous adversary may schedule the network in a way that prevents honest parties from reconstructing a polynomial of degree t_s . Bhimrajka et al. [18,19] solved this problem by invoking a re-sharing-based technique, which is expensive and relies on the BA protocol. To address this problem, we introduce a new virtual party technique and propose a more efficient network-agnostic VSS.

To further reduce communication complexity, we adapt several techniques to the network-agnostic setting, mainly in 3 steps. First, we design a RBC protocol with linear message complexity. Second, we propose a simplified network-agnostic VSS structure and further leverage the packed secret-sharing techniques [60]. Third, we adapt the dispute-control technique [34] to the network-agnostic setting, along with the KZG commitment scheme [2]. Finally, our protocol achieves amortized $O(n\kappa)$ bits in total communication.

We also propose a robust computational VSS-based MPC protocol in the network-agnostic setting. The motivation is that [18,19] requires at least $2t_s + 1$ (close to total) parties to behave honestly. We revisit this assumption and show that near-complete participation is unnecessary. Our MPC requires only $n - t_s$ active participants to ensure correctness and liveness.

Contributions The main contribution of this paper is a simplified VSS protocol with linear communication complexity. In detail,

1. We revisit the paradigm of the current network-agnostic VSS protocol, point out that the BA and re-sharing phases are inherently not necessary, and finally propose the first simplified network-agnostic VSS.
2. On communication complexity, we seamlessly adapt a series of techniques to the network-agnostic setting, and thereby propose the RBC and VSS with linear communication.

1.1 Technical Overview

We begin with the network-agnostic VSS protocol of [18,19], which introduced the first computational network-agnostic VSS and VSS-based MPC. We compare it with recent efficient asynchronous VSS schemes [1,31,45,46,62] and synchronous VSS [36,10,11,49,43,9] and found that neither asynchronous VSS nor synchronous VSS rely on the BA protocol, and instead in the asynchronous setting, VSS itself is a building block in BA [54]. This motivates us to revisit the VSS structure and eliminate the BA protocol. Hereafter, all protocols in the

network-agnostic setting satisfy the following optimal threshold [22]. Let the adversary corrupt up to t_s parties in the synchronous setting and t_a parties in the asynchronous setting among n parties:

$$n \geq 3t_a + 1, n \geq 2t_s + 1, n \geq 2t_s + t_a + 1, t_s \geq t_a$$

Overview of Previous Approach The VSS in Bhimrajka et al. [18,19] works following the AVSS paradigm, augmented with a network-distinguishing phase. If it cannot distinguish the network ($\text{BA} \rightarrow 1$), it invokes a lower-degree re-sharing sub-protocol. Specifically,

1. The dealer shares a bivariate polynomial $F(x, y)$ of degree (t_s, t_s) , commits F as $\mathcal{C}$ and broadcasts the commitment $\mathcal{C}$; Other parties broadcast **Verified** only if they receive valid $F(i, y)$, $F(x, i)$ and $\mathcal{C}$.
2. The dealer collects and broadcasts a set $\mathcal{CS} = \{P_i\}$ of $n - t_s$ parties that each P_i claims (has broadcasted) **Verified**.
3. Distinguish the network: If P_i finds the dealer successfully broadcasts the set $\mathcal{CS}$ **on time**, it puts 1 into BA ; otherwise, it puts 0 into BA .
4. If $\text{BA} \rightarrow 0$, the network is identified as asynchronous, and AVSS works: the dealer must enlarge the size of $\mathcal{CS}$ to $n - t_a$, the parties $P_i \in \mathcal{CS}$ send $F(i, j)$ to those $P_j \notin \mathcal{CS}$, and P_j interpolates $F(x, j)$.
5. If $\text{BA} \rightarrow 1$, the dealer re-shares $F(0, i)$, using a set of t_a degree polynomials $\{m_i(x) | P_i \notin \mathcal{CS}, m_i(0) = F(0, i)\}$, with proof that $m_i(0) = F(0, i)$. The dealer again broadcasts a set $\mathcal{CS}_2$ of $n - t_s$ parties who say $m_i(j)$ is shared correctly, implying at least $n - 2t_s \geq t_a + 1$ honest parties. Finally, every P_j sends $\{m_i(j)\}$ to P_i , so that P_i can reconstruct $m_i(0)$ of degree t_a , and outputs its share $F(0, i) = m_i(0)$.

In the construction of [18,19], regardless of the network, re-sharing the secrets with a t_a -degree polynomial leaks no more information beyond what the adversary already knows. However, it incurs a communication cost of $O(n^5\kappa)$.

- if the network is synchronous, the parties outside $\mathcal{CS}$ must be corrupt. As a result, it's secure to reveal their secrets.
- if the network is asynchronous, the adversary corrupts at most t_a parties, and it's secure to re-share the secrets using a polynomial of degree t_a .

Another common strategy utilizes ZK-based VSS [8], which works simply in the network-agnostic setting. The dealer broadcasts the polynomial commitment $\mathcal{C}$, a list of ciphertexts $\mathbf{C}$ (each encrypted shares for P_i under P_i 's public key), and NIZK proofs π for correct encryption. However, the NIZK always introduces additional cryptographic assumptions (such as Random Oracle) and computational costs, and broadcasting π incurs high communication costs (at least $O(n^2)$).

Our Solution As mentioned, the AVSS does not work in the network-agnostic model, in which an honest dealer can only collect authentication from $n - t_s <$

$n - t_a$ parties. We solve this by incorporating a revealing technique into the AVSS paradigm. The revealing technique [49, 43] is traditionally used in synchronous settings, where the dealer broadcasts the shares belonging to parties that dispute the dealer. However, in the asynchronous case, parties that dispute an honest dealer are not necessarily corrupt, and revealing their shares leaks information. To avoid leakage, we let the dealer reveal points of virtual parties, helping actual parties reconstruct their shares.

Specifically, we introduce $t_s - t_a$ virtual parties $P_{n+1}, \dots, P_{n+t_s-t_a}$. Along with at least $n - t_s - t_s \geq t_a + 1$ honest parties in $\mathcal{CS}$, there are at least $t_a + 1 + t_s - t_a = t_s + 1$ honest parties, and thereby the polynomial of degree t_s can be reconstructed. Besides, this does not leak information about secrets for honest dealers in both synchronous and asynchronous settings. In the synchronous setting, as long as the honest dealer collects the set after a synchronous timeout, the parties outside the set must be corrupt. In the asynchronous case, the adversary corrupts at most t_a parties, even with the revealed $t_s - t_a$ parties, it only obtains at most t_s parties' view. Specifically:

1. In the synchronous setting, each party P_i outside $\mathcal{CS}$ is corrupt, and it is secure to reveal the points that the adversary has obtained.
2. In the asynchronous setting, the adversary corrupts only t_a parties. Even with the view of $t_s - t_a$ virtual parties, this is equivalent to the adversary corrupting t_s parties, yet it leaks no more information than a synchronous adversary.
3. In both cases, the set includes at least $n - 2t_s \geq t_a + 1$ honest parties, with $t_s - t_a$ supplementary points from the virtual parties, each party outside the $\mathcal{CS}$ can collect at least $t_a + 1 + t_s - t_a = t_s + 1$ points to reconstruct the polynomial of degree t_s .

Sketch of Π_{VSS}

Let the dealer share s using $F(x, y)$ of degree (t_s, t_s) , commit F as $\mathcal{C}$.

1. **Share:** The dealer broadcasts $\mathcal{C}$ on F , opens $F(i, y), F(x, i)$ to P_i .
2. **Check:** Upon receiving valid shares $F(i, y), F(x, i)$ that committed on $\mathcal{C}$, everyone broadcasts **Checked**.
3. **Broadcast set:** The dealer collects valid signature on $\mathcal{C}$. After the synchronous clock passed, broadcast a set $\mathcal{CS}$ including at least $n - t_s$ parties that have claimed **Checked**.
4. **Reveal set:** The dealer must also reveal $F(i, j), F(j, i)$ for $P_i \notin \mathcal{CS}, j \in [n + 1, \dots, n + t_s - t_a]$, i.e., reveal $t_s - t_a$ virtual parties' points located for those who does not sign.
5. **Reconstruct:** In this step, each $P_i \in \mathcal{CS}$ reveals $F(i, j), F(j, i)$ to $P_j \notin \mathcal{CS}$. Then every honest party outside $\mathcal{CS}$ needs to reconstruct the shares from those points revealed by the dealer and others.

Towards Linear Communication At the start, we aim to settle the bound on the communication complexity of network-agnostic VSS. As one might expect,

network-agnostic VSS can never perform better than synchronous VSS. Even in the asynchronous case, the asynchronous adversary can delay t_s parties, such that it performs like the synchronous case. Over the years, synchronous VSS with optimal threshold has achieved $O(n^2)$ communication complexity in the one-shot case and $O(n)$ communication complexity with dispute-control. This sets the benchmark for our goals.

The intuition of using asynchronous communicational efficient building blocks does not always help. The main obstacle is the dual threshold nature of the latter. For instance, amortizing t_a does not reduce the amortized overhead (which works in the asynchronous model): in the network-agnostic setting, t_a can be as small as a constant, unlike in the asynchronous model where it can grow with n . We optimize it in the following aspect:

1. How to improve the efficiency of RBC in the network-agnostic setting;
2. How to achieve consistency when packing $O(n)$ secrets once;
3. How the dispute-control technique works in the network-agnostic setting.

First, current network-agnostic RBC[58] builds by combining Bracha's broadcast protocol [23] and the synchronous waiting mechanism, with communication complexity $O(n^2|m| + n^3\kappa)$. An intuitive strategy is transplanting the erasure coding technique, like Asynchronous Data Dissemination (ADD) [35], into the network-agnostic setting. However, subtly, this can only reduce the communication complexity from $O(n^2|m|)$ to $O(n^2|m|/(n - 2t))$, where t is the optimal threshold. This works in the asynchronous setting, as $t \leq n/3$ and $n - 2t \geq n/3$. But, it fails if $t = t_s \leq 1/2n$, as $n - 2t \geq 1$ could be constant and it still costs $O(n^2|m|/(n - 2t)) = O(n^2|m|)$.

We optimize the messages' reconstruction process by combining the known Merkle tree [26] and data dissemination techniques [35]. Specifically, we set the codeword dimension as $n - t_s$ instead of $n - 2t_s$. This requires all honest parties to provide valid codeword symbols for decoding. To achieve this, we use a Merkle tree to bind all the codeword symbols. As long as one party receives the message, it can verifiably distribute all codeword symbols to others. In each step for reconstruction, there are at most $O(n)$ parties sending $O(n)$ messages of size $O(\frac{1}{(n-t_s)}|m| + \log(n)\kappa)$, causing $O(n|m| + n^2 \log(n)\kappa)$ communication complexity. As a result, the total protocol costs $O(n|m| + n^3\kappa)$ bits in communication.

Second, we introduce the packed sharing into the network-agnostic setting. The common paradigm is lifting the polynomial degree from $t+1$ to $n-t$, where t is the optimal threshold. In the asynchronous setting with $n \geq 3t+1$, this works as it packs $n - 2t \geq 1/3n$ secrets once, thereby reducing the communication complexity. However, in the network-agnostic setting with $n \geq 2t_s + 1$, the packed sharing technique only leaves $n - t_s - t_s \geq 1$ degrees for packing. As a result, the communication may remain high.

We address this by further lifting the degree of the polynomial to $n - t_a$ instead of $n - t_s$. In the row dimension, the polynomial leaves $n - t_a - t_s \geq n/3$ degree for packing. In the synchronous setting with t_s corrupt parties, it seems the left $n - t_s$ honest parties can never reconstruct the row polynomials of degree

$n - t_a - 1$. We solve this problem by utilizing the virtual party technique again, which supplements $t_s - t_a$ points not only in the column dimension, but also in the row dimension. In the reconstruction of the row polynomials, there are at least $n - t_s + (t_s - t_a) = n - t_a$ points provided, which are sufficient to reconstruct polynomials of degree $n - t_a - 1$.

Third, with the above technique, the network-agnostic VSS can only achieve $O(n^2)$ communication complexity, fitting the performance of synchronous VSS in the one-shot case. This cost mainly comes from the *Commit* phase and *Reveal* phase, as the commitments $\mathcal{C}$ and the revealed set $\mathbf{\Omega}$ are of size $O(n^2\kappa)$.

The commitment size can be reduced to $O(n)$ by replacing the Pedersen commitment with the KZG commitment [50, 2, 57]. The *Reveal* phase can be optimized using the dispute-control techniques. Intuitively, once the honest dealer knows who is corrupt before sharing, it can directly set their shares as 0 and avoid revealing. Unlike previous dispute-control technique [34, 12, 47], our strategy serves a more restricted, dealer-centric purpose: the dispute set only helps the dealer predict which parties may fail to provide signatures. In contrast, in previous works, consistently identifying malicious parties to all honest parties generally requires additional mechanisms, such as Byzantine agreement in [47] or the verifiable accusation mechanism of [12].

Specifically, in each instance of sharing, the dealer can avoid revealing or identify at least one new corrupt party (who does not say **Checked**) in the synchronous setting. After sacrificing at most $2t_s$ rounds for identifying, the network-agnostic VSS can achieve linear communication in the synchronous setting. Besides, in the asynchronous case, after sacrificing at most $2t_s + 2$ rounds, the dealer knows it can collect $n - t_a$ signatures, and directly sets $F(i, y) = 0$ (to afford packing) for $i \in [n + 1, n + t_s - t_a]$. The remaining problem is that the RBC protocol costs $O(n|m| + n^3\kappa)$ in communication. We solve this using batched secret sharing, i.e. merging $O(n)$ VSS protocols with the same dealer into one instance, then the dealer can use one RBC instance to broadcast all the messages among $O(n)$ Π_{VSS} instances, to amortize the constant communication cost.

We mainly propose two VSS protocols and present the comparison of VSS protocols in Table 1. The first one Π_{VSS} , in the one-shot case, is based on Pedersen commitments and the packed sharing technique to achieve $O(n^2\kappa)$ bits per sharing in the one-shot case. The second one Π_{LVSS} , with round-by-round amortization, using the dispute-control technique and KZG commitment, achieves linear communication complexity. Let CC denote the communication complexity and RC denote the rounds. The rounds wasted in dispute-control, denoted $\mathcal{R}_{dc}$, can be amortized across $\mathcal{L}$ VSS instances. Besides, denote $\mathcal{BA}$ and $\mathcal{RBC}$ as the round in the RBC and BA protocols.

From VSS to MPC Finally, we show how to construct an MPC protocol based on our VSS protocol. We mainly follow the framework in [18, 19, 30], and utilize the correlated polynomial generation strategy to produce $f_a(x) \cdot f_b(x) = f_c(x)$. Then, the correctness of Beaver triples can be checked using the Schwartz-

Table 1. Comparison of VSS Protocols

Reference	CC	RC	Setup	Network
Cachin et al. [25]	$O(n^3\kappa)$	4	PKI	Asynchronous
Abraham et al. [2]	$O(n\kappa)$	$4 + \mathcal{BC}$	PKI AGM	Asynchronous
Bhimrajka et al. [18,19]	$O(n^5\kappa)$	$4\mathcal{BC} + \mathcal{BA} + 3$	PKI BA	Agnostic
This work (Π_{VSS})	$O(n^2\kappa)$	$\mathcal{BC} + 4$	PKI CRS	Agnostic
This work (Π_{LVSS})	$O(n\kappa + \frac{n \cdot \mathcal{R}_{dc}}{\mathcal{L}} \kappa)$	$\frac{\mathcal{R}_{dc}}{\mathcal{L}} + \mathcal{BC} + 4$	PKI CRS AGM	Agnostic

Zippel Lemma, and we can also extract fresh Beaver triples. A limitation in [18,19] is that they require at least $2t_s + 1$ parties to finish the Beaver triple sharing procedure, which is unrealistic. In the synchronous setting, the adversary who corrupts t_s parties can block this. They rely on the assumption that the extracted Beaver triples must be independent of the adversary’s view, and the triple extraction phase itself sacrifices near-half triples. As a result, in the $2t_s + 1$ triples, they sacrifice t_s triples and tolerate t_s triples, meaning it leaves 1 triple to extract.

We solve this problem based on the key observation: those parties who do not complete the Beaver triple sharing must be corrupt in the synchronous setting. Then we infer that among the set $\mathcal{CS}$ of parties that complete the sharing, the number of corrupt parties is at most $\max(t_a, t_s - (n - |\mathcal{CS}|))$. The detailed analysis in Section 4 demonstrates that these are sufficient to extract at least one independent Beaver triple as long as $|\mathcal{CS}| \geq n - t_s$. As a result, we can directly transplant the framework in [30] into the network-agnostic setting.

The comparison of MPC protocols is presented in Table 2. Let $|C|$ denote the number of gates in the circuit and R_{MPC} denote the underlying MPC round complexity in [47]. We compare our protocol with two threshold homomorphic encryption-based MPC schemes, one channel-based MPC and one VSS-based MPC scheme. As shown, our protocol is also VSS-based and reduces communication complexity from $O(n^6|C|)$ to $O(n^2|C|)$. For round complexity, all protocols require $O(n + D)$ rounds when using BA as a building block. Thanks to this, the required $O(n)$ -round process of establishing dispute-control does not increase the overall round complexity to a higher order.

Table 2. Comparison of Network-Agnostic MPC Protocols

Reference	CC	RC	Setup	Type
Blum et al. [22]	$O(n^3 C \kappa)$	$O(n + D)$	PKI, Trusted Setup	THE-based
Bacho et al. [8]	$O(n^2 C \kappa)$	$O(n + D)$	PKI, CRS	THE-based
Ji et al. [47]	$O(n C \kappa)$	$O(n \cdot R_{MPC})$	PKI, CRS	Channel-based
Bhimrajka et al. [18,19]	$O(n^6 C \kappa)$	$O(n + D)$	PKI	VSS-based
This work	$O(n^2 C \kappa)$	$O(n + D)$	PKI, CRS	VSS-based

1.2 Related work

In the synchronous setting, RBC protocols [20,8,22,66] build on the Dolev-Strong protocol [39] with $O(n)$ rounds and tolerate up to $n - 1$ corruptions. In the asynchronous setting, Bracha [23] gave the first RBC with $O(n^2|m|)$ communication; subsequent works [26,24] used erasure coding and Merkle trees to achieve $O(n|m| + n^2 \log n \kappa)$, while Das et al. [35] eliminated the $\log n$ factor, reaching $O(n|m| + n^2 \kappa)$. In the network-agnostic setting, Momose et al. [58] combined Bracha’s broadcast with a synchronous waiting mechanism, achieving $O(n^2|m| + n^3 \kappa)$ complexity, and noted that threshold signatures can reduce it to $O(n^2)$ under a trusted setup. This suggests that further optimizations to Bracha’s mechanism could improve upon [58], which motivates our work.

Synchronous VSS mainly have two types: the revealing based [40,11,49,43] and the zero-knowledge (ZK) proof based [36,10]. In the revealing-based approach, after the dealer sends each share over a secure channel, any party that fails to respond within the synchronous bound is deemed malicious, and the honest parties directly broadcast its share to reveal the secret. The ZK-based strategy works by letting the dealer broadcast the encryption of shares for all parties, along with a proof that these shares are correct ciphertexts and lie on the polynomial; because the dealer must broadcast the full ciphertext and proof for each party, the ZK-based approach suffers from high computational and communication overhead. Both strategies require $O(n^2)$ communication complexity.

Research on asynchronous VSS began with Feldman’s scheme [40]. Ben-Or et al. [13] proposed a perfectly-secure AVSS in the $n \geq 4t + 1$ setting with $\Omega(n^4)$ communication per sharing. Cachin et al. [25] proposed a computational AVSS in the $n \geq 3t + 1$ setting (optimal threshold) with $O(n^3)$ bits of communication per sharing. In 2015, Patra et al. [60] leveraged packed-sharing techniques [41] to reduce the communication overhead to $O(n^2)$. Abraham et al. [2] further optimized it to $O(n)$ bits with the KZG polynomial commitment scheme. Recently, Shoup et al. [62] proposed the hash-based AVSS, and Bandrupalli et al. [12] proposed the hash-based MPC with linear communication. In addition, a considerable number of recent studies [1,46,16,36,45,52,33] have investigated various kinds of AVSS and AMPC.

Research on network-agnostic VSS started recently. In 2024, Bhimrajka et al. [18] (further proposed an optimized version in 2026 [19]) proposed the first VSS protocol, using a standard approach that relies on Pedersen’s commitment and does not employ packing. As a result, their protocol requires $O(n^5)$ per sharing. This motivates our work to adapt recent advances in asynchronous VSS to the network-agnostic setting. Network-agnostic MPC approaches fall into three categories: GC-based [38], THE-based [22,8], and VSS-based [18,19]. Blum et al. [22] pioneered THE-based MPC with optimal resilience but high computational overhead, while GC-based approaches face similar efficiency issues. Very recently, Ji et al. [48,47] constructed a channel that bridges the synchronous and asynchronous settings, yielding an MPC protocol with $O(n)$ communication complexity per gate. In their protocol, each round of the underlying MPC is emulated through a network-agnostic channel whose invocation includes at least

one BA layer with $O(n)$ expected rounds. Thus, compiling an R -round MPC protocol requires $O(nR)$ expected rounds. In the $n \geq 2t_s + t_a + 1$ setting, Bhimrajka et al. [18,19] introduced the first VSS-based solution, but it incurs $O(n^6)$ bits in communication per gate. In the $n \geq 3t_s + t_a + 1$ setting, Gilad et al. [7] propose a perfectly secure network-agnostic VSS with $O(n^3 \log(n))$ communication complexity.

1.3 Paper Organization

In Section 2 we provide the mathematical and security preliminaries. Section 3 is the main chapter, which presents RBC protocol with linear communication, the simplified VSS protocol with $O(n^2)$, and the amortized linear communication VSS protocol Π_{LVSS} . Furthermore, Section 4 presents an MPC protocol constructed from the VSS protocol. Appendix A presents the building blocks in the MPC protocol. Appendix B provides the detailed proof for RBC and VSS, especially in UC. Appendix C presents the building blocks and security analysis in the linear communication Π_{LVSS} .

2 Preliminary

2.1 Notations

Let κ be the security parameter and n be the number of parties. Let the parties be $\{P_1, \dots, P_n\}$. We abuse notation by identifying P_i with i . The adversary can corrupt at most t_a nodes in the asynchronous setting and t_s nodes in the synchronous setting. Besides, we denote Π as a protocol, while $\Pi^{(i)}$ denotes the i -th protocol instance. Let $[n]$ denote the set $\{1, 2, \dots, n\}$, and $\{m_i\}$ denotes the set in which m_i corresponds to P_i . Boldface letters denote vectors, and the i -th element of a vector $\mathbf{A}$ is written as $\mathbf{A}[i]$. Besides, let Δ be the upper bound on the message delay in the synchronous setting and T_Π be the time window Π requires in the synchronous setting. For simplicity, we assume that each cryptographic field (or group) is size-bounded by 2^κ , including the signatures, hash values, and shared secrets.

Also, we use uppercase letters to denote bivariate polynomials like $F(x, y)$, while we use lowercase letters for univariate polynomials like f . All the polynomials are over a field $\mathbb{F}$, degree- d (univariate) polynomials belong to $\mathbb{F}[X]_{\leq d}$, and bivariate degree- (d_1, d_2) polynomials belong to $\mathbb{F}[X, Y]_{\leq d_1, \leq d_2}$. Also, we use the sans-serif font like RBC to denote specific protocols (e.g., reliable broadcast).

We use $[x]$ to denote a degree- t_s Shamir sharing of a secret x . Additionally, we use $\llbracket x \rrbracket$ to denote an authenticated Shamir sharing, i.e., $\llbracket x \rrbracket = ([x], [r], \mathcal{C})$, where $\mathcal{C}$ is a polynomial commitment to x with key r , following [34].

2.2 Security models

We utilize the simple UC (SUC) framework [29] to describe the security of our protocol. SUC models the network directly, by introducing a machine **Router**

that captures the ability of how the adversary can control the communication and network. Notably, it is proved in [29] that if the security holds in the SUC framework, it also holds in the UC framework. In the following, we abuse the notation UC by referring to the SUC framework simply as UC. Besides, when describing the functionality and protocol, we omit the session identifier (sid) in our descriptions and assume all parties operate within a single instance.

Adversary Let the adversary $\mathcal{A}$ corrupt the nodes in a static way. $\mathcal{A}$ can corrupt up to t_a parties in the asynchronous setting and t_s parties in the synchronous setting. Our protocol works under the optimal threshold: $n \geq 3t_a + 1, n \geq 2t_s + 1, n \geq t_a + 2t_s + 1$ along with $t_s \geq t_a$. Both $\mathcal{A}$ and $\mathcal{Z}$ are probabilistic polynomial-time (PPT) machines.

Communication Recent works [6,22] analyze security in the network-agnostic model. We formally model the network as a router $\mathcal{R}$. $\mathcal{R}$ receives the tag *asyn* or *syn* from the environment before the protocol starts. $\mathcal{R}$ delivers the message to the receiver after receiving (Allowed, header) from the adversary or once the delay bound Δ is reached in the synchronous setting.

In the asynchronous setting, the adversary $\mathcal{A}$ may delay message delivery arbitrarily. In the synchronous setting, it can delay delivery for at most Δ time. In the agnostic setting, before the protocol starts, the environment $\mathcal{Z}$ sends an initialization flag $flag \in \{\text{syn}, \text{asyn}\}$ to $\mathcal{R}$ and $\mathcal{A}$. If $flag = \text{syn}$, $\mathcal{A}$ may corrupt up to t_s parties, and $\mathcal{R}$ imposes a maximum delay of Δ . If $flag = \text{asyn}$, $\mathcal{A}$ may corrupt up to t_a parties, and $\mathcal{R}$ may delay messages arbitrarily.

We define $\text{REAL}_{\mathcal{A}, \pi, \mathcal{Z}}$ as the view of the real-world adversary $\mathcal{A}$, which includes the random tape and all the transcript messages of $\mathcal{A}$ during the execution of π . We also define $\text{IDEAL}_{\mathcal{S}, \mathcal{F}, \mathcal{Z}}$ as the output of the ideal-world simulator $\mathcal{S}$. Given two probability ensembles X, Y , we use the symbol $X \equiv Y$ to denote that they are computationally indistinguishable.

Definition 1. Let π be a protocol and let $\mathcal{F}$ be an ideal functionality. We say π UC-emulates $\mathcal{F}$ in the network-agnostic setting if for every real-world adversary $\mathcal{A}$ that executes π , there exists an ideal-world adversary $\mathcal{S}$ that runs in polynomial time such that for every PPT $\mathcal{Z}$:

$$\text{REAL}_{\mathcal{A}, \pi, \mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{S}, \mathcal{F}, \mathcal{Z}}$$

2.3 Existing Building Blocks

Cryptographic Assumption In our construction, we assume the DLog problem, the existence of a collision-resistant hash function $hash : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$, and an EUF-CMA secure digital signature scheme with PKI.

Definition 2 (Discrete Logarithm (DLog) Assumption). Let $\mathbb{G}$ be a cyclic group of prime order p with generator g . The Discrete Logarithm problem is: given $h = g^x \in \mathbb{G}$ for a random $x \in \mathbb{Z}_p$, compute x . We say the DLog assumption holds if for all polynomial-time adversaries $\mathcal{A}$,

$$\text{Adv}_{DL}(\kappa) = \Pr[(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\kappa); x \leftarrow \mathbb{Z}_p; x' \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^x) : x' = x] \leq \text{negl}(\kappa).$$

Definition 3 (EUF-CMA Security for Digital Signatures). A digital signature scheme $\Pi = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is existentially unforgeable under adaptive chosen message attacks (EUF-CMA) if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the following advantage is negligible:

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{EUF-CMA}}(\kappa) = \Pr \left[\begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}(1^\kappa); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}_{sk}(\cdot)}(pk) : \\ \text{Verify}(pk, m^*, \sigma^*) = 1 \wedge m^* \notin Q \end{array} \right] \leq \text{negl}(\kappa),$$

where Q is the set of messages submitted by $\mathcal{A}$ to its signing oracle.

Definition 4 (Collision Resistance for Keyed Hash Functions). A keyed hash function family $\text{hash}_K = (\text{Gen}, H)$ consists of a PPT key-generation algorithm $\text{Gen}(1^\kappa)$ that outputs a public key K , and a deterministic function $\text{hash}_K : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$. We say hash_K is **collision-resistant** if for all probabilistic polynomial-time adversaries $\mathcal{A}$,

$$\text{Adv}_{\text{hash}_K, \mathcal{A}}^{\text{CR}}(\kappa) = \Pr \left[K \leftarrow \text{Gen}(1^\kappa); (m_0, m_1) \leftarrow \mathcal{A}(1^\kappa, K) : m_0 \neq m_1 \wedge \text{hash}_K(m_0) = \text{hash}_K(m_1) \right] \leq \text{negl}(\kappa).$$

Definition 5 (Merkle Tree). A Merkle tree for a list of n data blocks $D = (d_1, \dots, d_n)$ is a binary tree constructed as follows:

- Each leaf node h_i is computed as $h_i = \text{hash}_K(d_i)$ for $i \in \{1, \dots, n\}$.
- For a pair of child nodes (h_ℓ, h_r) , their parent node is $\text{hash}_K(h_\ell \| h_r)$.
- The root value root is the hash value at the top of the tree.

The Merkle tree allows efficient verification of membership via an authentication path π_i for data block d_i , which consists of the sibling hashes along the path from h_i to root . Moreover, assuming that hash_K is collision resistant, the Merkle tree is computationally binding: it is infeasible to produce two different valid openings for the same position under the same root.

Erasur Coding tools In our protocol, we employ the RS coding technique [24], which works in a Galois Field $\mathbb{F} = \text{GF}(2^a)$.

- $\text{RSEnc}_{n,t}(m) \rightarrow \mathbf{c}$: Input a message $m \in \mathbb{M}$, map m into a vector of t field elements $(f_0, \dots, f_{t-1})$ that each f_i is of size $|m|/t$ bits. Then it constructs the polynomial $f(x) = f_0 + f_1x + \dots + f_{t-1}x^{t-1}$. Evaluate the polynomial at $[0, n-1]$, let $\mathbf{c} = \{c_i \mid c_i = f(i), 0 \leq i \leq n-1\}$ and return $\mathbf{c}$.
- $\text{RSDec}_{n,t}(\mathbf{c}, \mathbf{I}) \rightarrow m$: Input a codeword $\mathbf{c}$ and a corresponding index vector $\mathbf{I}$ of the same length, interpolate the polynomial f using the set of points $\{j, f(j) \mid j = \mathbf{I}[i], f(j) = \mathbf{c}[i]\}$. Finally, extract the coefficient of f to obtain the vector $\mathbf{f} = (f_0, \dots, f_{t-1})$, obtain m from $\mathbf{f}$ and return m .

Pedersen Commitment Scheme on Bivariate Polynomial The Pedersen commitment [61] $\text{Com}(m; r) = g^m h^r$ satisfies the additive homomorphic property $\text{Com}(m_1; r_1) \cdot \text{Com}(m_2; r_2) = \text{Com}(m_1 + m_2; r_1 + r_2)$. We utilize the bivariate polynomial commitment scheme of [25], including its `VerifyPoly` and `VerifyPoint` procedures. The scheme is computationally binding under the DLog assumption and statistically hiding. Let $\mathbb{G}$ be a cyclic group of prime order p with independent generators $g, h \in \mathbb{G}$. Let $F(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} a_{i,j} x^i y^j \in \mathbb{F}_p[x, y]$ and $R(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} b_{i,j} x^i y^j \in \mathbb{F}_p[x, y]$ be random bivariate polynomials of degree d_1 in x and d_2 in y . The following algorithm also inputs a hidden parameter $\text{crs} = (\mathbb{G}, p, g, h)$.

- `Commit`(F, R) $\rightarrow \mathcal{C}$: Output the commitment matrix $\mathcal{C} = \{C_{i,j}\}_{i=0, j=0}^{d_1, d_2}$ where $C_{i,j} = g^{a_{i,j}} h^{b_{i,j}}$.
- `VerifyPoly`($\mathcal{C}, u, f, f', r, r'$): Parse the input as

$$f(x) = \sum_{i=0}^{d_1} a_i x^i, f'(y) = \sum_{j=0}^{d_2} a'_j y^j, r(x) = \sum_{i=0}^{d_1} b_i x^i, r'(y) = \sum_{j=0}^{d_2} b'_j y^j$$

Return 1 iff for all $i \in [0, d_1]$: $g^{a_i} h^{b_i} = \prod_{j=0}^{d_2} C_{i,j}^{u_j}$, and for all $j \in [0, d_2]$: $g^{a'_j} h^{b'_j} = \prod_{i=0}^{d_1} C_{i,j}^{u_i}$.

- `VerifyPoint`($\mathcal{C}, x, y, \alpha, \alpha'$): Return 1 iff $g^\alpha h^{\alpha'} = \prod_{i=0}^{d_1} \prod_{j=0}^{d_2} C_{i,j}^{x^i y^j}$.

The scheme is statistically hiding and computationally binding under the Discrete Logarithm assumption. It is formally defined in Appendix B.

KZG Polynomial Commitment Kate et al. [50] propose the KZG commitment scheme that only costs one field element. Then, **Bingo**[2] (Crypto 2023) extends KZG to support committing bivariate polynomials. We denote the KZG commitment scheme as KZG, and the bivariate KZG commitment in **Bingo** as BKZG. Given groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and a bilinear map $e(\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$, and given a polynomial $f \in \mathbb{F}_{<d}(X)$, KZG has four algorithms as follows:

- `KZG.Setup`(κ, d) $\rightarrow \text{crs}$: Let $\tau, a \xleftarrow{\$} \mathbb{F}_{2^\kappa}$, let g, h be random generators of $\mathbb{G}_1, \mathbb{G}_2$, $\hat{g} \leftarrow g^a$. Return $\text{crs} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, h^\tau, \{g^{\tau^i}, \hat{g}^{\tau^i} \mid 0 \leq i \leq d\})$.
- `KZG.Commit`($\text{crs}, f(x); f'(x) \xleftarrow{\$} \mathbb{F}_{2^\kappa}^d$) $\rightarrow \mathcal{C}$: Return $\mathcal{C} \leftarrow g^{f(\tau)} \hat{g}^{f'(\tau)}$.
- `KZG.Eval`($\text{crs}, f(x), f'(x), i$) $\rightarrow (f(i), f'(i), \pi)$: Let $m \leftarrow f(i)$, $\hat{m} \leftarrow f'(i)$. Let

$$q(x) \leftarrow \frac{f(x) - m}{x - \omega_i}, \hat{q}(x) \leftarrow \frac{f'(x) - \hat{m}}{x - \omega_i}, \pi \leftarrow g^{q(\tau)} \hat{g}^{\hat{q}(\tau)}$$

Return $(m, \hat{m}, \pi)$.

- `KZG.Verify`($\text{crs}, \mathcal{C}, \omega_i, m, \hat{m}, \pi$) $\rightarrow 0/1$: If $e(\mathcal{C} g^{-m} \hat{g}^{-\hat{m}}, h) = e(\pi, h^{\tau - \omega_i})$, return 1. Else, return 0.

Authentic Share with Commitment Here we define the univariate authentic sharing with Commitment [34]. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be n parties. A secret $x \in \mathbb{F}_q$ is shared via a random degree- t polynomial f with $f(0) = x$. Each party P_i holds a $\llbracket x \rrbracket = (x_i, r_i, \mathcal{C})$ where:

- $x_i = f(i) \in \mathbb{F}_q$ is P_i 's Shamir secret share.
- $r_i \in \mathbb{F}_q$ is P_i 's share of commit key $r_i = r(i)$.
- $\mathcal{C} = \text{Commit}(f, r)$ is a public homomorphic polynomial commitment on polynomial $f = \sum f_i x^i$ with random key polynomial $r = \sum r_i x^i$.

Given $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$, parties can locally compute a share of any linear combination $z = \alpha x + \beta y$ ($\alpha, \beta \in \mathbb{F}_q$) as:

$$\text{share}_i(z) = (\alpha x_i + \beta y_i, \alpha r_i^{(x)} + \beta r_i^{(y)}, \mathcal{C}_z = \{\mathcal{C}_x[i]^\alpha \mathcal{C}_y[i]^\beta \mid i \in [0, d-1]\})$$

The new public commitment $\mathcal{C}_z$ is a valid commitment to z .

3 Network-Agnostic VSS

In the following, we take three steps. First, we propose the RBC protocol with linear communication; then we propose the simplified and BA-free network-agnostic VSS Π_{VSS} with quadratic communication; finally, we propose a compiler which compiles Π_{VSS} into a linear communication version Π_{LVSS} in amortized fashion.

3.1 Reliable Broadcast Protocol

The RBC protocol ensures that all honest parties finally obtain the same message from the sender, even if the sender is corrupted. In the following, we propose our RBC protocol with only linear communication complexity overhead.

Functionality: $\mathcal{F}_{RBC}$

The functionality is parameterized by a set of honest parties $H \subseteq [n]$ and a router $\mathcal{R}$.

Input(m): this operation is invoked once by the sender S , who inputs a message m . In response, $\mathcal{F}_{RBC}$ sends the message $\text{NotifyInput}(m)$ to the ideal-world adversary and sends the message $S \rightarrow P_i : \text{Output}(m)$ to P_i for $i \in [n]$ through the router $\mathcal{R}$.

Protocol Intuition and Description We propose the network-agnostic reliable broadcast protocol Π_{RBC} as follows. Our RBC protocol requires 3 phases: Input phase, Check phase, and Reconstruct Phase. The Input phase and Check phase follow the paradigm of [58]. Let the honest parties broadcast a quorum $\mathcal{CS}$ of $n - t_s$ votes (signatures) on the message m . In the synchronous setting, honest parties will not vote for inconsistent m thanks to the guaranteed-delivery message; whereas in the asynchronous setting, the adversary corrupting t_a parties cannot form a $\mathcal{CS}$ on m and a $\mathcal{CS}'$ on m' simultaneously due to threshold.

We make three main modifications to the protocol in [58].

1. First, we let the dealer encode m into a codeword $\mathbf{c}$ and build a Merkle tree with $\mathbf{c}$ as leaves. Then, we let non-sender parties only send the Merkle tree root h instead of the original message m .
2. Second, they only send $(\text{vote}, h)_{P_i}$ after receiving a consistent message from the sender P_S .
3. After getting agreement on the root h , all the parties invoke a *Reconstruct Phase* to finally obtain the message m .

The parties do not echo the full message m , but only codeword symbols of smaller size in the reconstruction phase. The main challenge in the *Reconstruct Phase* is ensuring that everyone can finally receive enough codeword symbols. We achieve this using a modified idea in [35]. As the codeword has already been bound on a common Merkle tree, all the codeword symbols are verifiable. As a result, as long as one party holds a valid $\mathbf{c}$, every party can finally receive and accept this. To maintain the linear complexity, we utilize a two-level distribution structure. Denote $(msg)_{P_i}$ as a message signed by P_i , and $(msg)_{\{P_i\}}$ as a message signed by a set of parties $\{P_i\}$, the protocol shows as follows.

Protocol Broadcast: Π_{RBC}

The dealer P_S inputs m , all parties get the public key from PKI, and each sets $\text{out} = \perp$.

Input Phase: The dealer P_S

1. Inputs m , encodes it as $\text{REnc}_{n, n-t_s}(m) \rightarrow \mathbf{c}$ and builds a Merkle tree with h as root and $\mathbf{c}$ as leaves.
2. Sends $(\text{propose}, h)_S$ along with the codeword $\mathbf{c}$ to P_i for $i \in [n]$.

Check Phase: Every party P_k

3. Upon receiving $(\text{propose}, h)_{P_S}$, broadcasts $(\text{propose}, h)_{P_S}$. Further broadcasts $(\text{vote}, h)_{P_k}$ if
 - wait for Δ , and no conflict $(\text{propose}, h')_{P_S}$ that $h' \neq h$ is found;
 - received $(\text{propose}, h)_{P_S}$ and corresponding $(\mathbf{c})$ from the P_S ;
4. After receiving $n - t_s$ $(\text{vote}, h)_{P_i}$ for different i , adds P_i into $\mathcal{C}$, broadcasts $(\text{vote}, h)_{\{P_i | P_i \in \mathcal{C}\}}$, sets $\text{out} = h$.

Reconstruct Phase: Upon $\text{out} = h \neq \perp$, every party P_k

5. If it has received valid $\mathbf{c}$ from the sender, sends $(\text{rec}, \mathbf{c}[i], \pi_i)$ to P_i for $i \in [n]$, where π_i is an authentic path from $\mathbf{c}[i]$ to h .
6. If has received valid $(\text{rec}, \mathbf{c}[k], \pi_k)$, transfers $(\text{rec}, \mathbf{c}[k], \pi_k)$ to every P_i for $i \in [n]$.
7. If it has finished step 6 and received $n - t_s$ valid $(\text{rec}, \mathbf{c}[i], \pi_i)$ from P_i , decode $\text{RSDec}_{n, n-t_s}(\{\mathbf{c}[i]\}, \{i\}) \rightarrow m$ and output m .

Analysis For security, we have the following theorem, where the proof is presented in Appendix B.

Theorem 1. *In Π_{RBC} , assuming a collision-resistance hash function and a PKI for an EUF-CMA secure signature scheme, let the adversary follow the network-agnostic restriction, the protocol Π_{RBC} satisfies the following properties:*

- *Consistency: If any pair of honest parties output m and m' , then $m = m'$.*
- *Integrity: If the dealer is honest and inputs m , then no honest party will output m' for $m' \neq m$.*
- *Validity: If the dealer is honest, every honest party eventually terminates.*
- *Totality: If any honest party finishes the protocol, all honest parties eventually terminate.*

For the communication complexity of Π_{RBC} , we have the following main theorem:

Theorem 2. *Let the size of both the hash value and the signature be κ . To broadcast a message m , protocol Π_{RBC} incurs a total communication of $O(n|m| + n^3\kappa)$ bits.*

Proof. First, we claim that the codeword vector $\mathbf{c}$ has size $O(|m|)$, while each element $\mathbf{c}[i]$ has size $O(|m|/n)$. Let the degree of the Reed-Solomon code be set to $n - t_s$. Then m is encoded into an n -length vector, each is of $|m|/(n - t_s)$ bits. Since $n - t_s \geq n/2$, it follows that $\mathbf{c}[i]$ has size $O(|m|/(n - t_s)) = O(|m|/n)$ bits. Moreover, the authentication path has size $O(\log(n) \cdot \kappa)$ bits.

As a result, the $(\text{propose}, h)_{P_s}$ message has size 2κ bits; a (vote, h) message (from any party) has size κ bits, and there are n such votes in total; and each $(\text{rec}, \mathbf{c}[i], \pi_i)$ message has size $O(|m|/n + \log(n) \cdot \kappa)$ bits. In protocol Π_{RBC} , the sender sends $\mathbf{c}$ to all. Each party send a $(\text{propose}, \dots)$ message once to all, and then send both a $(\text{vote}, \dots)$ and a $(\text{rec}, \dots)$ message once each to all. In summary, Π_{RBC} incurs a total communication of $O(n \cdot |m| + n^2 \cdot (n \cdot \kappa + |m|/n + \log(n) \cdot \kappa)) = O(n|m| + n^3\kappa)$

3.2 Simplified and Efficient Network-Agnostic VSS

In this subsection, we propose our VSS protocol Π_{VSS} , which allows a dealer to share a secret with consistency. Our protocol achieves $O(n^2\kappa)$ communication overhead for sharing a secret $s \in \mathbb{F}_{2^\kappa}$. The functionality $\mathcal{F}_{VSS}$ is given as follows. This is somewhat different from the classical AVSS functionality [46] as we reveal additional virtual parties' view to the adversary and $\mathcal{F}_{VSS}$ outputs the commitment as an authentication simultaneously.

Functionality: Packed VSS $\mathcal{F}_{VSS}$

The functionality is parameterized by a set of honest parties $H \subset [n]$.

Share(Input, $F(x, y)$). Upon being invoked by the sender, if the sender is corrupt, it also receives $R(x, y)$ from the adversary; otherwise, it randomly samples R . Finally, let $\mathcal{C} \leftarrow \text{Commit}(F(x, y), R(x, y))$, and

- send (**share**, $F(x, i), F(i, y), R(x, i), R(i, y), \mathcal{C}$) to $\mathcal{A}$ for $i \in [n] \setminus H$. If $|[n] \setminus H| \leq t_a$, further reveal (**share**, $F(x, i), F(i, y), R(x, i), R(i, y)$) to $\mathcal{A}$ for $i \in [n + 1, n + t_s - t_a]$.
- send (**output**, $F(x, i), R(x, i), \mathcal{C}$) to P_i for $i \in [n]$, through the router.

High-level idea The main structure follows the AVSS in [36,2], which is *Commit+Share* $\rightarrow$ *Check* $\rightarrow$ *Collect* $\rightarrow$ *Reconstruct*. Specifically, the dealer commits to and shares the polynomial. Then the parties check the validity of shares and sign to the dealer as authentication. The dealer broadcasts the signatures to convince everyone that the sharing procedure is complete. Finally, the parties who signed help those who did not sign to reconstruct their shares.

The main difference between the original AVSS and our network-agnostic VSS is that the dealer reveals $t_s - t_a$ points for virtual parties to each party P_i that does not sign. Intuitively, in the synchronous setting, parties that do not sign must be corrupt, and the revealing phase leaks no more information; in the asynchronous setting, it is still secure to expand the adversary's view from t_a parties to t_s parties. Furthermore, after seeing the broadcast message, everyone ensures that those parties outside the verified set can also reconstruct the correct polynomial, so that the Termination property holds.

Protocol Description The protocol Π_{VSS} works in 4 steps. Initially, we assume a PKI setup for distributing public keys, where each party holds a secret key for signing. Besides, each party has built a secure P2P channel for communication.

In *Phase 1*, the dealer is given $n - t_s - t_a$ secrets once for packed secret sharing. This is because we can finally ensure $n - t_a$ points for reconstruction, which leaves $n - t_s - t_a$ degrees for sharing. Then the dealer selects a bivariate polynomial F of degree $(n - t_a - 1, t_s)$, along with a commitment key R with the same degree. Then, it shares $F(i, y), R(i, y)$ and $F(x, i), R(x, i)$ to P_i along with the commitment $\mathcal{C}$. The polynomial is committed using the bivariate polynomial Pedersen Commitment scheme, as $\mathcal{C} = \text{Commit}(F, R)$. As a result, $\mathcal{C}$ is a matrix including $(t_s + 1) \cdot (n - t_a)$ elements.

In *Phase 2*, each party P_i receives the shares from the dealer. It then checks if the share is consistent with the commitment $\mathcal{C}$. If it is, P_i accepts $F(i, y), R(i, y)$ and $F(x, i), R(x, i)$ from the dealer, and further signs a message containing $\mathcal{C}$ and transfers the signature to the dealer. We borrow this idea from [36], utilizing the signing to the dealer instead of broadcasting the flag **Checked**. Benefiting from this, the instances of RBC protocols are decreased from $O(n)$ to $O(1)$.

In *Phase 3*, the dealer needs to collect signatures to convince all parties that the *Share* procedure has been completed successfully. However, it can only collect

$n - t_s$ signatures in the synchronous setting. Specifically, the dealer waits for 2Δ time units, which is sufficient for any honest party to send its signature to the dealer. After receiving $n - t_s$ valid signatures, which implies that either those who did not sign are corrupt (in the synchronous setting) or the adversary is operating in the asynchronous setting, the dealer broadcasts the commitment along with these signatures and a set $\mathcal{CS}$ that indicates the signatories. Additionally, it publicly reveals $t_s - t_a$ points of the polynomials for each non-signing party $P_j \notin \mathcal{CS}$. This is the key technique of the VSS protocol. Before going to the next step, we perform an analysis here.

First, consider the secret leakage in this revealing phase. If the network is synchronous, all the parties outside the set $\mathcal{CS}$ must be corrupt (or the dealer is corrupt). Therefore, this phase leaks no more information than the adversary holds. If the network is asynchronous, the adversary who corrupts t_a parties gets more information. However, it will obtain no more than t_s parties' views. As the degree is $(n - t_a - 1, t_s)$, which is designed to tolerate t_s corruptions, the adversary's view is still independent of the secrets.

Second, consider the completeness of the protocol, i.e., why revealing $t_s - t_a$ points is sufficient for the reconstruction of the original polynomial. Recall that only those who are outside the set $\mathcal{CS}$ need to reconstruct the polynomial, and note that the column dimension is of degree t_s . The dealer has confirmed a set of over $n - t_s$ parties who have received the valid row polynomial. The set includes at least $n - t_s - t_s \geq t_a + 1$ honest parties P_i , who will help $P_j \notin \mathcal{CS}$ to reconstruct. Furthermore, the dealer has revealed $t_s - t_a$ points, such that P_j can receive at least $t_s - t_a + t_a + 1 = t_s + 1$ points to reconstruct the column polynomial. In the row dimension, as all honest parties will receive valid column polynomials, they can provide at least $n - t_s$ valid points for reconstruction. Also, the dealer provides $t_s - t_a$ points as supplementary, enabling P_j to reconstruct a polynomial of degree $n - t_s + t_s - t_a - 1 = n - t_a - 1$.

In *Phase 4*, these parties first verify that the broadcasted message is properly signed and that the dealer has revealed the correct $t_s - t_a$ points for all parties outside $\mathcal{CS}$. If any check fails, the honest parties abort. Otherwise, parties proceed as follows. Those parties inside $\mathcal{CS}$ send help points for those $P_j \notin \mathcal{CS}$ and output the shares. Those outside $\mathcal{CS}$ first reconstruct the column of degree t_s using the help message from others and revealed points from the dealer. They further send helping points for the row polynomial, and finally interpolate the row polynomial of degree $n - t_a - 1$.

Protocol Π_{VSS} : Network-Agnostic Verifiable Secret Sharing

Every party is initialized with signature key sk_i , and public key list $\{pk_i\}$ from PKI.

1. **Phase 1: Commit+Share-Dealer:** let $\theta = n - t_s - t_a$, is given θ secrets $\{s_1 \cdots, s_\theta\}$ and does as follows:
 - Randomly sample two $(n - t_a - 1, t_s)$ degree bivariate polynomial $F, R \in \mathbb{F}[X, Y]_{\leq n - t_a - 1, \leq t_s}$, such that $F(-i, 0) = s_i$ for $i \in [\theta]$.

- Commit the polynomial F with key R , as $\mathcal{C} = \text{Commit}(F, R)$.
- Send share_i to P_i , where

$$\text{share}_i = (\mathcal{C}, F(x, i), F(i, y), R(x, i), R(i, y))$$

2. **Phase 2: Check**-Each party P_i : after receiving share_i . If

$$\text{VerifyPoly}(\mathcal{C}, i, F(x, i), F(i, y), R(x, i), R(i, y)) \rightarrow 1$$

Let $\sigma_i = \text{Sign}(sk_i, (sid, \mathcal{C}))$ and send $\text{Sign}_i = (\sigma_i)$ to the dealer.

3. **Phase 3: Collect**-Dealer: after time 2Δ , let $\mathcal{CS} = \emptyset$. Includes P_i into $\mathcal{CS}$ upon receiving valid Sign_i . Upon $|\mathcal{CS}| \geq n - t_s$ inputs $(\mathcal{C}, \mathcal{CS}, \{\sigma_i | P_i \in \mathcal{CS}\}, \Omega)$ into $\mathcal{F}_{RBC}$ where

$$\Omega = \left\{ F(i, j), F(j, i), R(i, j), R(j, i) \mid P_i \notin \mathcal{CS}, j \in [n + 1, n + (t_s - t_a)] \right\} \quad (1)$$

4. **Phase 4: Reconstruct**-Each party P_i : receiving $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \Omega)$ from $\mathcal{F}_{RBC}$, accept Ω if the following holds; otherwise, abort.

- Each $P_i \in \mathcal{CS}$ has signed $(sid, \mathcal{C})$ and $|\mathcal{CS}| \geq n - t_s$.
- Each $P_i \notin \mathcal{CS}$, Ω is constructed as Eq 1
- Every point $(f_{x,y}, r_{x,y}) \in \Omega$, $\text{VerifyPoint}(\mathcal{C}, x, y, f_{x,y}, r_{x,y}) \rightarrow 1$.

Then,

- If $P_i \in \mathcal{CS}$, sends $(F(i, j), F(j, i), R(i, j), R(j, i))$ to P_j , for $j \notin \mathcal{CS}$. **Outputs** $F(x, i), R(x, i)$ and $\mathcal{C}$.
- If $P_i \notin \mathcal{CS}$, accept $t_s - t_a$ points in Ω as points on $F(i, x), F(y, i)$.

Then it works as follows in two steps:

- (a) After receiving $t_a + 1$ valid $F(i, j), R(i, j)$ from $P_j \in \mathcal{CS}$, interpolate $F(i, y), R(i, y)$ from those $t_a + 1 + (t_s - t_a) = t_s + 1$ points. Then, sends $F(i, j), R(i, j)$ to P_j for $P_j \notin \mathcal{CS}$.
- (b) After receiving $n - t_s$ valid $F(j, i), R(j, i)$ from any P_j , interpolate $F(x, i), R(x, i)$ from those $n - t_s + (t_s - t_a) = n - t_a$ points. **Outputs** $F(x, i), R(x, i)$ and $\mathcal{C}$.

Security Analysis This paragraph presents the security analysis of the protocol Π_{VSS} . Formally, it is captured by the following theorem.

Theorem 3. *If hiding and binding properties hold in the commitment scheme, the signature scheme is EUF-CMA secure, and the adversary follows the network-agnostic restriction, within the PKI setup and $\mathcal{F}_{RBC}$, protocol Π_{VSS} satisfies the following properties with overwhelming probability:*

- *Termination*
 1. *If the dealer is honest, all honest parties will complete the protocol.*
 2. *If any honest party completes the protocol, then every honest party will complete the protocol.*

- *Correctness* The combined view of all honest parties at the end of the sharing phase uniquely defines a secret $\mathbf{s}'$. Moreover, if the dealer is honest and inputs $\mathbf{s}$, then $\mathbf{s}' = \mathbf{s}$.
- *Secrecy* If the dealer is honest, the adversary's view throughout the protocol is statistically independent of $\mathbf{s}$.

Proof (Sketch). For readability, we only present the sketch proof here, which has captured the main idea, but without formal security reduction, the detailed proof is given in Appendix B. We consider two network models separately. First, if the network is synchronous:

- *Termination.*
 1. If the dealer is honest, at time 2Δ , the dealer can collect at least $n - t_s$ signatures. It then follows the protocol to broadcast the set of signatures, commitment, the set $\mathcal{CS}$, and revealed points. All honest parties must be included in $\mathcal{CS}$ and terminate.
 2. If some honest party outputs $F(x, i), R(x, i), \mathcal{C}, \mathcal{F}_{RBC}$ delivers a consistent and valid tuple $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \Omega)$ to all parties. Assuming the signature scheme is unforgeable, at least $n - t_s$ (which is $\geq t_a + 1$) honest parties inside $\mathcal{CS}$ have received valid shares. Every party outside $\mathcal{CS}$ can receive $t_s - t_a$ points on $F(x, i)$ and $F(i, y)$. Moreover, the (at least $t_a + 1$) honest parties inside $\mathcal{CS}$ can each send additional $t_a + 1$ points on $F(i, y)$. Then all honest parties can reconstruct the column polynomial $F(i, y)$, and send $F(i, j)$ to $P_j \notin \mathcal{CS}$. Finally, all parties outside $\mathcal{CS}$ can receive $n - t_s$ points on the row polynomial from the honest parties and $t_s - t_a$ points from the Ω .
- *Correctness* Assuming the commitment scheme is binding, all points and polynomials accepted by honest parties with respect to the same commitment C must be mutually consistent, except with negligible probability. After the sharing phase, the accepted evaluations determine bivariate polynomials F and R . If correctness is violated, then for some point (i, j) an honest party must accept a different pair $(F'(i, j), R'(i, j)) \neq (F(i, j), R(i, j))$ under the same commitment C . The original and forged pairs therefore give two distinct valid openings at the same point, which contradicts the binding property of the commitment scheme. Hence, all accepted evaluations are consistent with a unique bivariate polynomial F of degree at most $(n - t_a - 1, t_s)$.
For each party outside $\mathcal{CS}$, the $t_a + 1$ valid points provided by honest parties in $\mathcal{CS}$, together with the $t_s - t_a$ revealed virtual-party points, uniquely determine its degree- t_s column polynomial. The reconstructed columns then provide $n - t_s$ valid points, which together with the $t_s - t_a$ virtual-party points uniquely determine the corresponding degree- $(n - t_a - 1)$ row polynomial. Consequently, for every α , the values $F(-\alpha, j) j \in [n]$ form a degree- t_s Shamir sharing of $F(-\alpha, 0)$. If the dealer is honest and inputs $s_1, \dots, s_\theta$, then $F(-\alpha, 0) = s_\alpha$.
- *Secrecy* Let the commitments be hiding. If the dealer is honest, all parties outside $\mathcal{CS}$ are corrupt. As a result, the adversary can only obtain t_s shares

in two dimensions. However, since the row polynomial is of degree $n - t_a - 1$, this leaves at least $n - t_s - t_a$ degrees for free. Furthermore, each column polynomial is of degree t_s . Since the adversary holds at most t_s points on any such polynomial belonging to an honest party P_i , the constant term $F(i, 0)$ remains independent of the adversary's view.

If the network is asynchronous:

– *Termination*

1. If the dealer is honest, it will eventually gather a set of $n - t_a$ signatures. Even if it only broadcasts a set $\mathcal{CS}$ of size $n - t_s$, the parties outside $\mathcal{CS}$ can still reconstruct the polynomial. This is because there are at least t_a honest parties in the set $\mathcal{CS}$ providing valid points located on the polynomial. Besides, the dealer also reveals $t_s - t_a$ points, such that the honest parties can reconstruct the polynomial of degree t_s .
2. Following the same reasoning as in the synchronous setting.

– *Correctness* Following the same reasoning as in the synchronous setting.

- *Secrecy* Consider an honest dealer who broadcasts $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \Omega)$. This can be interpreted as providing the adversary, who corrupts only t_a real parties, with additional $t_s - t_a$ virtual parties (via the revealed points in Ω). Thus, the adversary effectively controls no more than t_s parties in total, ensuring the secrecy condition is maintained. Formally, if the commitment is perfectly hiding, given that the commitment scheme is hiding, the adversary obtains at most t_s points in the row dimension and at most t_s points in the column dimension. The proof then follows the same logic as in the synchronous case, guaranteeing secrecy.

Furthermore, the protocol Π_{VSS} is UC secure.

Theorem 4. *Assuming access to ideal functionality $\mathcal{F}_{RBC}$ and $\mathcal{F}_{Sig}$, if the DLog assumption holds, in the CRS hybrid model, $\mathcal{F}_{VSS}$ UC-emulates Π_{VSS} .*

Proof (Sketch). The main idea of the proof is that once the dealer is committed to broadcasting the set $\mathcal{CS}$, the simulator can obtain enough information to generate an indistinguishable view. For a corrupt dealer, the adversary must reveal the whole polynomial to the simulator. For an honest dealer, subtly, the simulator is given the total view of virtual party $P_{n+1}, \dots$ in the asynchronous setting, such that it can produce the same view except the commitment. If the communication is hiding, the theorem holds. The detailed proof is given in Appendix B.

Performance Analysis In the following, we analyze the communication complexity of Π_{VSS} . Note that our protocol can share

$$n - t_s - t_a \geq n - \frac{n - t_a - 1}{2} - t_a = \frac{n - t_a + 1}{2} > \frac{n}{3}$$

points, i.e. $O(n)$ points at once. If Π_{VSS} costs $O(n^3)$ bits in one instance, it costs amortized $O(n^2)$ bits per sharing. Formally, we have the following theorem.

Theorem 5. *To share $\theta = n - t_s - t_a$ secrets $\{s_i \mid i \in [\theta]\} \in \mathbb{F}_{2^\kappa}^\theta$, the Π_{VSS} protocol costs $O(n^3\kappa)$ bits in communication complexity.*

Proof. In the *Commit+Share* phase, the dealer transfers $\mathcal{C}$ and 4 univariate polynomials to all the parties, where the commitment is of size $O(n^2\kappa)$ and each polynomial is of size $O(n\kappa)$. This phase costs $O(n^3\kappa)$ bits in communication.

The *Check* phase costs $O(n\kappa)$ bits in communication. The *Collect* phase requires the dealer to broadcast $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \Omega)$, in which Ω includes at most $O(n^2)$ points. As a result, the broadcasted message is of size $O(n^2\kappa)$, and this phase costs $O(n^2\kappa \cdot n + n^3\kappa) = O(n^3\kappa)$ bits in communication. Finally, in the *Reconstruct* phase, each honest party sends $O(n\kappa)$ bits to each party, which incurs $O(n^2\kappa)$ bits in communication.

Robust Reconstruction In our Π_{VSS} protocol, every party will output the tuple $(f, r, \mathcal{C})$, such that each party can verifiably publish its shares for reconstruction. Furthermore, due to the linear homomorphic property of the polynomial commitment, our protocol can support robust reconstruction of any linear operation among shares, just like [2, 25]. Notably, in the reconstruction phase, the parties do not send the commitment but only the share and corresponding proof. Hereafter, we simply use $\llbracket x \rrbracket$. As this is standard, we omit the protocol details.

3.3 Network-Agnostic VSS with Linear Communication

In this subsection, we present a compiler that transforms Π_{VSS} into a protocol Π_{LVSS} with amortized linear communication. The compiler mainly replaces the Pedersen-style commitment and explicit reveal set Ω in Π_{VSS} with KZG/BKZG commitments and a dispute-control mechanism. The dealer maintains a dealer-centric dispute set $\mathcal{DS}$ of parties from whom it does not expect signatures, sets the corresponding evaluations involving virtual parties to zero, and uses [KZG.BZEval](#) to prove $O(n^2)$ zero evaluations with only $O(n\kappa)$ bits. If a party outside $\mathcal{DS}$ fails to sign, it is added to $\mathcal{DS}$ and the sharing is restarted. Finally, we batch $O(n)$ sharing instances into one RBC invocation. Together with packed sharing, these techniques yield amortized linear communication per sharing.

Compiler: Π_{VSS} to Π_{LVSS}

Let the dealer in Π_{VSS} hold a synchronous-dispute set $\mathcal{DS}$ of parties and denote $\mathcal{VS} = \{n+1, n+t_s-t_a\}$ as virtual party set. Besides, given $crs \leftarrow \text{KZG.Setup}(\kappa, n-t_s)$. Then, the dealer and other parties work as in Π_{VSS} with the following modification.

Case 1: if $|\mathcal{DS}| \leq t_s$

1. In phase 1, for the dealer with secrets, for $i \in \mathcal{DS}, j \in \mathcal{VS}$, let $F(i, j) = 0, R(i, j) = 0, F(j, i) = 0, R(j, i) = 0$ and generate a batch

proof

$$\pi_{\mathcal{DS}}^{\text{bz}}[i] = \text{KZG.BZEval}(\text{crs}, F(i, y), R(i, y), \mathcal{VS})$$

$$\pi_{\mathcal{VS}}^{\text{bz}}[j] = \text{KZG.BZEval}(\text{crs}, F(j, y), R(j, y), \mathcal{DS})$$

The dealer replaces the commitment and adds a additional proof for $i \in [n], j \in [t_s + 1]$,

$$\mathcal{C} \leftarrow \text{BKZG.Commit}(\text{crs}, F(), R())$$

$$\dots, \pi'_{j,i} \leftarrow \text{KZG.Eval}(\text{crs}, F(j, y), R(j, y), i)$$

and lets $\text{share}_i = (\mathcal{C}, F(i, y), R(i, y), F(x, i), R(x, i), \pi'_{j,i}, \pi_{\mathcal{DS}}^{\text{bz}}, \pi_{\mathcal{VS}}^{\text{bz}}, \mathcal{DS})$.

2. In phase 2, P_i obtains $\mathbf{A} \leftarrow \text{BKZG.PartialEval}(\text{crs}, \mathcal{C})$ and replace the verification steps as follows, for $k \in \mathcal{DS}, j \in \mathcal{VS}$

$$\text{BKZG.VerifyPoly}(\text{crs}, \mathcal{C}, F(x, i), F(i, y), R(x, i), R(i, y), \{\pi'_{j,i}\})$$

$$\text{KZG.BZVerify}(\text{crs}, \mathbf{A}[k], \pi_{\mathcal{DS}}^{\text{b}}[k], \mathcal{VS})$$

$$\text{KZG.BZVerify}(\text{crs}, \mathbf{A}[j], \pi_{\mathcal{VS}}^{\text{b}}[j], \mathcal{DS})$$

3. In phase 3, if the dealer collects all the signatures σ_i for any $P_j \notin \mathcal{DS}$, continue with $\Omega = \emptyset$. Otherwise, add P_i who not sign into $\mathcal{DS}$ and restart to the first step. The $\mathcal{DS}$ is also added into the $\mathcal{F}_{RBC}$ along with commitments and signatures.
4. In phase 4, for $P_i \notin \mathcal{DS}, j \in [n + 1, n + t_s - t_a]$, accept $F(i, j) = 0, F(j, i) = 0$ just like it appears in the Ω . Besides, when P_i sends $F(i, j)$ or $F(j, i)$ to P_j , it generates corresponding proof from

$$f_i(j), r_i(j), \pi_{i,j} \leftarrow \text{BKZG.Eval}(\text{crs}_1, f_i(), r_i(), j)$$

$$\{\pi_{j,i}\}_{j \in [n]} \leftarrow \text{BKZG.GetProof}(\{\alpha_k, \pi_{\alpha_k, i}\}_{k \in [t_s + 1]})$$

Case 2: if $|\mathcal{DS}| > t_s$

Follows the Case 1 ($|\mathcal{DS}| \leq t_s$) with the following modifications.

1. In phase 1, the dealer lets $F(i, y) = 0, R(i, y) = 0$ as zero polynomials for $i \in \mathcal{VS}$, and generates a batch proof for $j \in [t_s + 1]$ as

$$\pi_{\mathcal{DS}}^{\text{b}}[j] \leftarrow \text{KZG.BZEval}(\text{crs}, F(j, y), R(j, y), [t_s + 1])$$

2. In phase2, except for BKZG.VerifyPoly , they only check if for $i \in \mathcal{VS}$

$$1 \leftarrow \text{KZG.BZVerify}(\text{crs}, \mathbf{A}[j], \pi_j^{\text{b}}, [t_s + 1])$$

and sign $\sigma_i = \text{Sign}(sk_i, (sid, \mathcal{C}, \text{Case}_2))$ to the dealer.

3. In phase 3, the dealer must collect $n - t_a$ signatures.
4. In phase 4, the parties check if each $P_i \in \mathcal{CS}$ has signed $(sid, \mathcal{C}, \text{Case}_2)$.

Batched Sharing Compiler

In both cases, the dealer shares a batch of $O(n)$ polynomials, in each instance, when the dealer is assigned to broadcast m_i , the dealer holds it and broadcasts all of them in one RBC instance.

Theorem 6. *To share $n \cdot \theta = n \cdot (n - t_s - t_a)$ secrets $\{s_i \mid i \in [n \cdot \theta]\} \in \mathbb{F}_{2^\kappa}^{n \cdot \theta}$, the Π_{LVSS} protocol costs $O(n^3 \kappa)$ bits in communication complexity.*

Proof. First, $share_i$ is now including $(\mathcal{C}, f_i, f'_i, \pi', \pi_{\mathcal{DS}}^{\text{bz}}, \pi_{\mathcal{VS}}^{\text{bz}}, \mathcal{DS})$, each of them is of size $O(n\kappa)$ ($|\mathcal{DS}| \leq n \log(n) \leq n\kappa$ as in the computational settings), the broadcasted message includes $(\mathcal{C}, \sigma, \mathcal{CS}, \mathcal{DS})$, each of them is of size $O(n\kappa)$ or $O(n \log(n))$. Furthermore, the reconstructing message requires $O(n)$ parties to send $O(n\kappa)$ messages in each instance. Finally, using batched secret sharing, Π_{LVSS} costs $O(n^3 \kappa)$ bits in communication.

The Π_{LVSS} follows the same security properties as Π_{VSS} , the detailed security analysis is given in Appendix C .

4 From VSS to MPC

As a proof of concept, we next propose how to construct general MPC protocols built upon our VSS. We follow the asynchronous MPC protocol introduced in [30]. Goyal et al. [45] further optimize the framework in the asynchronous setting, such that it can extract $O(n)$ Beaver triples at once. For simplicity, we just take the simple version in [30]. However, in the network-agnostic setting, current works [18,19] ask that at least $2t_s + 1$ parties complete the verifiable Beaver triple sharing. This is overly optimistic and may never be achieved. We overcome this limitation with the following ideas. Again, based on the key observation: *those who do not complete the sharing can be considered corrupt*, and we claim even $n - t_s$ triples are sufficient to extract at least 1 independent Beaver triples. Formally, we have the following Theorem.

Theorem 7. *In the network-agnostic setting with PKI, there is a computational general MPC protocol that costs $O(n^2|C| + n^2|D| + n^4)$ in communication to evaluate a circuit $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}$ with C gates and depth D .*

High Level Idea Let everyone share a Beaver triple (a, b, c) such that $a \cdot b = c$. It is ensured that at least $n - t_s$ parties will complete it. The interesting part is that those who do not complete must be corrupt in the synchronous setting. Let θ be the number of parties that have completed the Beaver triple sharing and $\mathbf{d} = \lfloor \frac{\theta-1}{2} \rfloor$, the extraction strategy in [18,19] sacrifices $\mathbf{d}$ triples and leaves $\mathbf{d} + 1$ triples. The key observation is: we can eliminate $n - \theta$ corrupt parties in the synchronous setting. In the θ parties, there are at most

$$\max(t_a, (t_s - (n - \theta)))$$

corrupt parties, implying we can extract at least

$$\mathbf{d} + 1 - \max(t_a, (t_s - (n - \theta)))$$

valid Beaver triples. If $t_a \geq (t_s - (n - \theta))$, given $\theta \geq n - t_s$, $n - t_s \geq t_s + t_a + 1$

$$\mathbf{d} + 1 - t_a = \left\lfloor \frac{\theta + 1 - 2t_a}{2} \right\rfloor \geq \left\lfloor \frac{n - t_s + 1 - 2t_a}{2} \right\rfloor \geq \left\lfloor \frac{t_s - t_a + 2}{2} \right\rfloor \geq 1$$

Otherwise, given $n - \theta \geq 0, n - 2t_s \geq 1$, it also satisfies

$$d + 1 - (t_s - (n - \theta)) = \left\lfloor \frac{-\theta + 1}{2} \right\rfloor + n - t_s = \left\lfloor \frac{(n - 2t_s) + n - \theta + 1}{2} \right\rfloor \geq 1$$

As a result, regardless of the network setting, as long as $\theta \geq n - t_s$, the framework in [18,19] allows us to extract at least one independent and fresh Beaver triple. Then, we outline the framework for extracting independent Beaver triples based on [30]. Let $[a]$ be the Shamir secret sharing of a .

1. Let all the parties P_i share random Beaver triples $[a], [b], [c]$ such that $a \cdot b = c$ through three instances of $\Pi_{VSS}^{(i)}$ and further prove that $a \cdot b = c$.
2. Invoke a Π_{ACS} protocol to identify $n - t_s$ parties $\mathcal{CS} = \{P_i\}$ who have completed $\Pi_{VSS}^{(i)}$ and proved the relation.
3. Let $\theta = |\mathcal{CS}|$, $d = \lfloor \frac{\theta-1}{2} \rfloor$, order $P_i \in \mathcal{CS}$ such that all parties in $\mathcal{CS}$ have shared $\{(a_i, b_i, c_i) | i \in [\theta]\}$. All parties define the shared univariate polynomial X, Y, Z of degree $d, d, 2d$ that $[X(j)] = [a_j], [Y(j)] = [b_j], [Z(j)] = [c_j]$ for $j \in [d+1]$.
4. For $j \in [d+2, 2d+1]$, every party aims to get $[Z(j)]$ such that $Z(j) = X(j) \cdot Y(j)$, they use $\{(a_j, b_j, c_j) | j \in [d+2, 2d+1]\}$ as Beaver triples to compute $[Z(j)]$ where each holds $[X(j)], [Y(j)]$. Finally, all honest parties get shared polynomials X, Y, Z .
5. Finally, every party outputs $[X(-1)], [Y(-1)], [Z(-1)]$ as one Beaver triple.

As mentioned, this phase requires opening (sacrificing) d Beaver triples, leaving $d+1$ fresh Beaver triples, and supporting the extraction of at least 1 independent Beaver triple.

4.1 Verifiable Beaver Triple Sharing

In this subsection, we introduce how to verify the consistency of Beaver triples. There are currently many strategies, such as proving on Pedersen commitments [64,55] or distributed zero-knowledge proofs [30,18,19,1]. To avoid unnecessary presentation complexity, we simply use the technique in [30]. The detailed sub-protocol (especially $\Pi_{TripleCheck}$ and $\Pi_{TripTrans}$) is given in Appendix C.

Protocol $\Pi_{VTripleShare}$: Verifiable Beaver Triple Sharing

Let the circuit include C_M multiplication gates, the dealer prepares Beaver triples as follows.

Share Phase: The dealer

1. Let $\theta = \lceil (2C_M + 1)/(n - t_s - t_a) \rceil$, prepare 3θ bivariate polynomials $\{F_x^{(i)}, F_y^{(i)}, F_z^{(i)} | i \in [\theta]\}$ such that $F_x^{(i)}(-j, 0) \cdot F_y^{(i)}(-j, 0) = F_z^{(i)}(-j, 0)$ for $j \in [n - t_s - t_a]$.
2. Input all the $\{(F_x^{(i)}, F_y^{(i)}, F_z^{(i)}) | i \in [\theta]\}$ into 3θ instances of $\Pi_{LVSS}^{(\alpha, i)}$ for $\alpha \in \{x, y, z\}, i \in [\theta]$ in round-by-round case.

Verification Phase

1. Every party P_i : once obtaining all the 3θ shares $\text{share}_{\alpha,i}$ from $\Pi_{LVSS}^{(\alpha,i)}$ for $\alpha \in \{x, y, z\}, i \in [\theta]$, parse the packed sharing into authentic and independent Beaver triples $\text{Trip} = \{\llbracket x_i \rrbracket, \llbracket y_i \rrbracket, \llbracket z_i \rrbracket \mid i \in [2C_M + 1]\}$. Input $(2C_M + 1, \text{Trip})$ into $\Pi_{TripCheck}$.
2. Every party P_i : after obtaining all the 3θ shares from Π_{LVSS} and $\Pi_{TripCheck} \rightarrow 1$, outputs the first C_M triples in Trip .

Security Analysis Here we adopt the security model in [64] to describe our protocol. Specifically, protocol $\Pi_{VTripleShare}$ ensures the Beaver triple relation and consistency among honest parties, as formalized below:

Lemma 1. *In the network-agnostic setting, protocol $\Pi_{VTripleShare}$ satisfies the following properties even when invoked by a corrupt dealer.*

1. *Termination.* If the dealer is honest, all honest parties will eventually output C_M authentic shared triples;
2. *Validity.* Each of the C_M output triples $([x_k], [y_k], [z_k])$ constitutes a correct t_s -degree secret sharing and satisfies $x_k \cdot y_k = z_k$.
3. *Secrecy.* For any output triple $([x], [y], [z])$, the adversary learns no information about the secrets (x, y, z) beyond the multiplicative relation $x \cdot y = z$.

Proof. (Sketch) We give a simplified proof sketch here.

1. *Termination.* First, all instances of $\Pi_{LVSS}^{(i)}$ and $\Pi_{TripCheck}$ for an honest dealer P_i will produce a valid output. Furthermore, if one honest party outputs in $\Pi_{VTripleShare}$, it implies that all $\mathcal{F}_{VSS}$ instances have completed and $\Pi_{TripCheck}$ has delivered a valid proof. Therefore, all honest parties will eventually receive the shares and the proof, enabling them to output as well.
2. *Validity.* First, thanks to Π_{LVSS} all honest parties P_i obtain $F(-1, i)$, which is located on a t_s degree univariate polynomial. Besides, thanks to the $\Pi_{TripCheck}$, the multiplication relation is guaranteed.
3. *Secrecy.* Due to the secrecy of $\Pi_{TripCheck}$, it leaks no information about the first C_M triples. Moreover, Π_{LVSS} ensures that beyond the shares received by the parties, no additional information about the secret is leaked.

Communication Analysis Protocol $\Pi_{VTripleShare}$ invokes 3θ instances of Π_{LVSS} and incurs $\lceil (2C_M + 1)/(n - t_s - t_a) \rceil \cdot n^2\kappa + O(n^3\kappa) = O(nC_M\kappa + n^3\kappa)$ bits of communication. Besides, it also invokes one instance of $\Pi_{TripCheck}$, which includes the C_M Beaver triples and incurs $O(nC_M\kappa + n^2\kappa)$ bits in communication. In summary, the protocol $\Pi_{VTripleShare}$ costs $O(nC_M\kappa + n^3\kappa)$ bits in communication.

4.2 Offline Phase: Triple Extraction

After all honest parties have finished $\Pi_{VTripleShare}$, it is required to extract independent Beaver triples so that the adversary does not know the shared value. This is achieved using the framework in [18,19]. However, we avoid this restriction and point out that $n - t_s$ original triples are sufficient to extract 1 fresh triple. As analyzed above, let θ be the number of parties that have completed the Beaver triple sharing and $d = \lfloor \frac{\theta-1}{2} \rfloor$,

$$d + 1 - \max(t_a, (t_s - (n - \theta))) \geq 1$$

The protocol works as follows:

Protocol $\Pi_{TripExt}$: Beaver Triple Extraction

1. Let each party P_i invoke $\Pi_{VTripleShare}^{(i)}$ to share C_M verified Beaver triples to other parties.
2. After time $T_{VTripleShare}$, all the parties P_i invoke a Π_{ACS} to identify a set $\mathcal{CS}$ ($|\mathcal{CS}| \geq n - t_s$) of parties, who completes $\Pi_{VTripleShare}$.
3. Each P_i , for $j \in [C_M]$, let $\theta = |\mathcal{CS}|$, $d = \lfloor \frac{\theta-1}{2} \rfloor$. Upon receiving all the

$$\text{triples}^{(j)} = \{(\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket) | i \in [C_M]\}$$

from $\Pi_{VTripleShare}^{(j)}$. Let $\text{points}_i = \{\text{triples}^{(j)}[i] | j \in \theta\}$, invoke $\Pi_{TripTrans}^{(j)}(\theta, \text{points}_i)$, which outputs correlated polynomial $\llbracket f_a^{(j)} \rrbracket, \llbracket f_b^{(j)} \rrbracket, \llbracket f_c^{(j)} \rrbracket$ that $f_a \cdot f_b = f_c$.

4. For $k \in [C_M]$, outputs $\llbracket f_a^{(j)}(-1) \rrbracket, \llbracket f_b^{(j)}(-1) \rrbracket, \llbracket f_c^{(j)}(-1) \rrbracket$ as authentic fresh Beaver triples.

Security Analysis We model our protocol like the structure in [30], formally,

Theorem 8. *In the network-agnostic setting, the protocol $\Pi_{TripExt}$ achieves the following properties:*

1. *Termination:* If all honest parties participate in $\Pi_{TripExt}$, $\Pi_{TripExt}$ will terminate.
2. *Correctness:* If $\Pi_{TripExt}$ terminates, every party P_i outputs C_M shared Beaver triples, which satisfy the multiplication relation.
3. *Privacy:* The shared triples are independent of the adversary's view.

Proof. We prove the security properties as follows

1. *Termination:* If all honest parties have invoked $\Pi_{VTripleShare}$, there are at least $n - t_s$ $\Pi_{VTripleShare}$ protocols terminating. Thanks to the Π_{ACS} , everyone will finally obtain a consistency set $\mathcal{CS}$ of $n - t_s$ parties. Also, $\Pi_{TripTrans}$ will terminate for each party.

2. *Correctness*: As the $\Pi_{VTripleShare}$ only outputs Beaver triples, which satisfy the multiplication relation. All the triples inputted into $\Pi_{TripTrans}$ are valid Beaver triples, such that $\Pi_{TripTrans}$ polynomials satisfy $f_x() \cdot f_y() = f_z()$. As a result, $f_x(-1) \cdot f_y(-1) = f_z(-1)$.
3. *Privacy*: The privacy in $\Pi_{VTripleShare}$ holds and $\Pi_{TripTrans}$ only proceeds locally. Besides, as we already mentioned, it allows to extract 1 fresh Beaver triple. Let $\theta = |\mathcal{CS}|$ parties have completed the Beaver triple sharing and $d = \lfloor \frac{\theta-1}{2} \rfloor$. Moreover, in the θ parties, there are at most $\max(t_a, (t_s - (n - \theta)))$ corrupt parties, implying $d + 1 - \max(t_a, (t_s - (n - \theta))) \geq 1$ degree for free.

Communication Analysis In $\Pi_{TripExt}$, each party acts as a dealer in one $\Pi_{VTripleShare}$ instance to distribute C_M triples. Transmitting the shares and authentication tags costs $O(n^2|C_M|\kappa)$ bits, while broadcasting commitments and proofs adds $O(n^4\kappa)$ bits. A single Π_{ACS} invocation to agree on $\mathcal{CS}$ contributes $O(n^3\kappa)$ bits, which is asymptotically dominated. The local transformation $\Pi_{TripTrans}$ requires no extra communication. Thus the total communication complexity is $O(n^2|C_M|\kappa + n^4\kappa)$ bits.

4.3 Online Phase

In this subsection, we present the network-agnostic MPC protocol in [18,19].

Protocol $\Pi_{CirEval}$: Circuit Evaluation

- **Preprocess+Input**:
 - Every party invokes an instance of $\Pi_{TripExt}$ which outputs C_M Beaver triples with corresponding auxiliary information.
 - For each input gate held by P_i , P_i invokes an instance of Π_{VSS} to share its input value. Meanwhile, all the parties invoke one Π_{ACS} instance to identify a set $\mathcal{CS}$ of at least $n - t_s$ parties. For $P_i \notin \mathcal{CS}$, all the parties set the input gate held by P_i as 0.
- **Evaluation**: Interpret the circuit as a publicly known ordered gates $(G_1, \dots, G_c)$. Sequentially evaluate $G_k(u, v) \rightarrow w$ as follows:
 - for *addition gate*: let $[w] = [u] + [v]$ locally.
 - for *add-constant gate*: let $[w] = [u] + v$ locally.
 - for *multi-constant-c gate*: let $[w] = c[u]$ locally.
 - for *multiplication gate*: using unused Beaver triples $[x], [y], [z]$, reveal $[u - x], [v - y]$, let $[w] = [u][v] = (u - x)(v - y) + (v - y)[x] + (u - x)[y] + [z]$
- **Output**: Let $[w_i]$ be the output of the wire for P_i . The parties verifiably reconstruct $[w_i]$ for P_i .

Analysis The preprocess phase is secure based on the security of $\Pi_{TripExt}$. For the input phase, it is ensured that at least $n - t_s$ parties will input, and the Π_{ACS} maintains the consistency among all honest parties. Furthermore, based

on the classical MPC protocol, it outputs only the corresponding outputs to the adversary. Focus on the communication complexity, formally:

Lemma 2. *In the network-agnostic setting, let $f : \mathbb{F}_p^n \rightarrow \mathbb{F}_p$ be a publicly-known deterministic function represented by an arithmetic circuit cir over $\mathbb{F}_p$. Suppose cir contains $|C|$ gates with depth $|D|$. Each party P_i holds an input $x^{(i)}$ to f . Then protocol Π_{CirEval} incurs $O(|C|n^2\kappa + n^4\kappa)$ bits in communication and 2 instances of Π_{ACS} .*

Proof (sketch). In Π_{CirEval} , the preprocessing phase invokes one Π_{TripExt} instance to generate $|C_M|$ Beaver triples, which costs $O(n^2|C_M|\kappa + n^4\kappa)$ bits as analyzed above. For input sharing, each input gate requires an instance of Π_{VSS} , incurring $O(|I| \cdot n^2\kappa + n^3\kappa)$ bits in total, where $|I|$ denotes the number of input wires. One Π_{ACS} invocation to agree on $\mathcal{CS}$ adds $O(n^3\kappa)$ bits. During evaluation, addition and constant gates are purely local. Each multiplication gate reveals two masked values via reliable broadcast, consuming $O(n^2\kappa)$ bits per gate. With $|C_M|$ multiplication gates and depth D , this phase costs $O(|C_M| \cdot n^2\kappa + D \cdot n^2\kappa)$ bits. Output reconstruction for 1 gates requires $O(n^2\kappa)$ bits, which is dominated by the preceding terms. Summing over all phases, the total communication complexity is $O((|C_M| + |I| + D) \cdot n^2\kappa + n^4\kappa)$ bits. This simplifies to $O(|C|n^2\kappa + n^4\kappa)$ with 2 instances of Π_{ACS} .

Acknowledgements

We thank the anonymous reviewers for their valuable comments and constructive suggestions, which helped improve the quality and presentation of this paper.

References

1. Abraham, I., Asharov, G., Patil, S., Patra, A.: Perfect asynchronous MPC with linear communication overhead. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024, Part V*. Lecture Notes in Computer Science, vol. 14655, pp. 280–309. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_10
2. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In: Handschuh, H., Lysyanskaya, A. (eds.) *Advances in Cryptology – CRYPTO 2023, Part I*. Lecture Notes in Computer Science, vol. 14081, pp. 39–70. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38557-5_2
3. Aldahiri, A., Khalil, I., Rahman, M.S., Atiquzzaman, M.: A privacy-preserving class imbalance mitigation framework for face recognition. *High-Confidence Computing* **6**(1), 100325 (2026). <https://doi.org/10.1016/j.hcc.2025.100325>
4. Alexandru, A.B., Blum, E., Katz, J., Loss, J.: State machine replication under changing network conditions. In: Agrawal, S., Lin, D. (eds.) *Advances in Cryptology – ASIACRYPT 2022, Part I*. Lecture Notes in Computer Science, vol. 13791, pp. 681–710. Springer, Cham, Switzerland, Taipei, Taiwan (Dec 5–9, 2022). https://doi.org/10.1007/978-3-031-22963-3_23

5. Appan, A., Chandramouli, A., Choudhury, A.: Perfectly-secure synchronous MPC with asynchronous fallback guarantees. In: Milani, A., Woelfel, P. (eds.) 41st ACM Symposium Annual on Principles of Distributed Computing. pp. 92–102. Association for Computing Machinery, Salerno, Italy (Jul 25–29, 2022). <https://doi.org/10.1145/3519270.3538417>
6. Appan, A., Choudhury, A.: Network agnostic MPC with statistical security. In: Rothblum, G.N., Wee, H. (eds.) TCC 2023: 21st Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 14370, pp. 63–93. Springer, Cham, Switzerland, Taipei, Taiwan (Nov 29 – Dec 2, 2023). https://doi.org/10.1007/978-3-031-48618-0_3
7. Asharov, G., Elsheimy, F., Stern, G.: Fast and efficient perfectly secure network-agnostic secure computation. Cryptology ePrint Archive, Paper 2026/655 (2026), <https://eprint.iacr.org/2026/655>
8. Bacho, R., Collins, D., Liu-Zhang, C.D., Loss, J.: Network-agnostic security comes (almost) for free in DKG and MPC. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part I. Lecture Notes in Computer Science, vol. 14081, pp. 71–106. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38557-5_3
9. Bagheri, K.: π : A unified framework for computational verifiable secret sharing. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography – PKC 2025. pp. 110–142. Springer Nature Switzerland, Cham (2025)
10. Bagheri, K., Bardeh, N.G., Khazaei, S., Rahimi, M.: On round-optimal computational VSS. Cryptology ePrint Archive, Paper 2025/1246 (2025), <https://eprint.iacr.org/2025/1246>
11. Bai, G., Damgård, I., Orlandi, C., Xia, Y.: Non-interactive verifiable secret sharing for monotone circuits. In: Pointcheval, D., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 16: 8th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 9646, pp. 225–244. Springer, Cham, Switzerland, Fes, Morocco (Apr 13–15, 2016). https://doi.org/10.1007/978-3-319-31517-1_12
12. Bandrupalli, A., Ji, X., Kate, A., Liu-Zhang, C.D., Song, Y.: Computationally efficient asynchronous MPC with linear communication and low additive overhead. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025, Part IV. Lecture Notes in Computer Science, vol. 16003, pp. 261–294. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2025). https://doi.org/10.1007/978-3-032-01884-7_9
13. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: 25th Annual ACM Symposium on Theory of Computing. pp. 52–61. ACM Press, San Diego, CA, USA (May 16–18, 1993). <https://doi.org/10.1145/167088.167109>
14. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 1–10. ACM Press, Chicago, IL, USA (May 2–4, 1988). <https://doi.org/10.1145/62212.62213>
15. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: Anderson, J., Toueg, S. (eds.) 13th ACM Symposium Annual on Principles of Distributed Computing. pp. 183–192. Association for Computing Machinery, Los Angeles, CA, USA (Aug 14–17, 1994). <https://doi.org/10.1145/197917.198088>
16. Benhamouda, F., Halevi, S., Krawczyk, H., Ma, Y., Rabin, T.: SPRINT: High-throughput robust distributed Schnorr signatures. In: Joye, M., Leander, G. (eds.)

- Advances in Cryptology – EUROCRYPT 2024, Part V. Lecture Notes in Computer Science, vol. 14655, pp. 62–91. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_3
17. Bhimrajka, N., Choudhury, A., Varadarajan, S.: Network-agnostic perfectly secure message transmission revisited. In: Chattopadhyay, A., Bhasin, S., Picek, S., Rebeiro, C. (eds.) Progress in Cryptology - INDOCRYPT 2023: 24th International Conference in Cryptology in India, Part II. Lecture Notes in Computer Science, vol. 14460, pp. 25–44. Springer, Cham, Switzerland, Goa, India (Dec 10–13, 2023). https://doi.org/10.1007/978-3-031-56235-8_2
 18. Bhimrajka, N., Choudhury, A., Varadarajan, S.: Network-agnostic multi-party computation revisited (extended abstract). In: Tang, Q., Teague, V. (eds.) PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 14602, pp. 171–204. Springer, Cham, Switzerland, Sydney, NSW, Australia (Apr 15–17, 2024). https://doi.org/10.1007/978-3-031-57722-2_6
 19. Bhimrajka, N., Choudhury, A., Varadarajan, S.: Network-agnostic verifiable secret sharing with cryptographic security. *IEEE Transactions on Information Theory* **72**(2), 1332–1363 (2026). <https://doi.org/10.1109/TIT.2025.3642814>
 20. Blum, E., Katz, J., Loss, J.: Synchronous consensus with optimal asynchronous fallback guarantees. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019: 17th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 11891, pp. 131–150. Springer, Cham, Switzerland, Nuremberg, Germany (Dec 1–5, 2019). https://doi.org/10.1007/978-3-030-36030-6_6
 21. Blum, E., Katz, J., Loss, J.: Tardigrade: An atomic broadcast protocol for arbitrary network conditions. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 547–572. Springer, Cham, Switzerland, Singapore (Dec 6–10, 2021). https://doi.org/10.1007/978-3-030-92075-3_19
 22. Blum, E., Zhang, C.D.L., Loss, J.: Always have a backup plan: Fully secure synchronous MPC with asynchronous fallback. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 707–731. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_25
 23. Bracha, G.: Asynchronous byzantine agreement protocols. *Information and Computation* **75**(2), 130–143 (1987). [https://doi.org/10.1016/0890-5401\(87\)90054-X](https://doi.org/10.1016/0890-5401(87)90054-X)
 24. Cachin, C., Tessaro, S.: Asynchronous verifiable information dispersal. In: 24th IEEE Symposium on Reliable Distributed Systems (SRDS’05). pp. 191–201 (2005). <https://doi.org/10.1109/RELDIS.2005.9>
 25. Cachin, C., Kursawe, K., Lysyanskaya, A., Stroh, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: Atluri, V. (ed.) ACM CCS 2002: 9th Conference on Computer and Communications Security. pp. 88–97. ACM Press, Washington, DC, USA (Nov 18–22, 2002). <https://doi.org/10.1145/586110.586124>
 26. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 524–541. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_31

27. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In: Neiger, G. (ed.) 19th ACM Symposium Annual on Principles of Distributed Computing. pp. 123–132. Association for Computing Machinery, Portland, OR, USA (Jul 16–19, 2000). <https://doi.org/10.1145/343477.343531>
28. Canetti, R.: Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239 (2003), <https://eprint.iacr.org/2003/239>
29. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 3–22. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-48000-7_1
30. Choudhury, A., Patra, A.: An efficient framework for unconditionally secure multiparty computation. IEEE Transactions on Information Theory **63**(1), 428–468 (2017). <https://doi.org/10.1109/TIT.2016.2614685>
31. Choudhury, A., Patra, A.: On the communication efficiency of statistically-secure asynchronous MPC with optimal resilience. Cryptology ePrint Archive, Report 2022/913 (2022), <https://eprint.iacr.org/2022/913>
32. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: 18th Annual ACM Symposium on Theory of Computing. pp. 364–369. ACM Press, Berkeley, CA, USA (May 28–30, 1986). <https://doi.org/10.1145/12130.12168>
33. Coretti, S., Garay, J.A., Hirt, M., Zikas, V.: Constant-round asynchronous multiparty computation based on one-way functions. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 998–1021. Springer Berlin Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53890-6_33
34. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) Advances in Cryptology – CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622, pp. 572–590. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2007). https://doi.org/10.1007/978-3-540-74143-5_32
35. Das, S., Xiang, Z., Ren, L.: Asynchronous data dissemination and its applications. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021: 28th Conference on Computer and Communications Security. pp. 2705–2721. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484808>
36. Das, S., Xiang, Z., Tomescu, A., Spiegelman, A., Pinkas, B., Ren, L.: Verifiable secret sharing simplified. In: Blanton, M., Enck, W., Nita-Rotaru, C. (eds.) 2025 IEEE Symposium on Security and Privacy. pp. 633–651. IEEE Computer Society Press, San Francisco, CA, USA (May 12–15, 2025). <https://doi.org/10.1109/SP61157.2025.00046>
37. Deligios, G., Erbes, M.M.: Closing the efficiency gap between synchronous and network-agnostic consensus. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024, Part V. Lecture Notes in Computer Science, vol. 14655, pp. 432–461. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_15
38. Deligios, G., Hirt, M., Liu-Zhang, C.D.: Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In: Nissim, K., Waters, B. (eds.) TCC 2021: 19th Theory of Cryptography Conference, Part I.

- Lecture Notes in Computer Science, vol. 13042, pp. 623–653. Springer, Cham, Switzerland, Raleigh, NC, USA (Nov 8–11, 2021). https://doi.org/10.1007/978-3-030-90459-3_21
39. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing* **12**(4), 656–666 (1983). <https://doi.org/10.1137/0212045>
 40. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science. pp. 427–437. IEEE Computer Society Press, Los Angeles, CA, USA (Oct 12–14, 1987). <https://doi.org/10.1109/SFCS.1987.4>
 41. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: 24th Annual ACM Symposium on Theory of Computing. pp. 699–710. ACM Press, Victoria, BC, Canada (May 4–6, 1992). <https://doi.org/10.1145/129712.129780>
 42. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) *Advances in Cryptology – CRYPTO 2018*, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 33–62. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96881-0_2
 43. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology* **20**(1), 51–83 (Jan 2007). <https://doi.org/10.1007/s00145-006-0347-3>
 44. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th Annual ACM Symposium on Theory of Computing. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987). <https://doi.org/10.1145/28395.28420>
 45. Goyal, V., Liu-Zhang, C.D., Song, Y.: Towards achieving asynchronous MPC with linear communication and optimal resilience. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology – CRYPTO 2024*, Part VIII. Lecture Notes in Computer Science, vol. 14927, pp. 170–206. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68397-8_6
 46. Ji, X., Li, J., Song, Y.: Linear-communication asynchronous complete secret sharing with optimal resilience. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology – CRYPTO 2024*, Part VIII. Lecture Notes in Computer Science, vol. 14927, pp. 418–453. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68397-8_13
 47. Ji, X., Liu-Zhang, C.D., Pöllmann, D., Song, Y.: Information-theoretic network-agnostic MPC with polynomial communication. In: Daemen, J., Thomé, E. (eds.) *Advances in Cryptology – EUROCRYPT 2026*, Part III. Lecture Notes in Computer Science, vol. 16543, pp. 307–336. Springer, Cham, Switzerland, Rome, Italy (May 10–14, 2026). https://doi.org/10.1007/978-3-032-25324-8_11
 48. Ji, X., Liu-Zhang, C.D., Song, Y.: Perfectly secure network-agnostic MPC comes for free. In: Daemen, J., Thomé, E. (eds.) *Advances in Cryptology – EUROCRYPT 2026*, Part III. Lecture Notes in Computer Science, vol. 16543, pp. 337–365. Springer, Cham, Switzerland, Rome, Italy (May 10–14, 2026). https://doi.org/10.1007/978-3-032-25324-8_12
 49. Kate, A., Mangipudi, E.V., Mukherjee, P., Saleem, H., Thyagarajan, S.A.K.: Non-interactive VSS using class groups and application to DKG. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) *ACM CCS 2024: 31st Conference on Computer and Communications Security*. pp. 4286–4300. ACM Press, Salt Lake City, UT, USA (Oct 14–18, 2024). <https://doi.org/10.1145/3658644.3670312>

50. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) *Advances in Cryptology – ASIACRYPT 2010*. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer Berlin Heidelberg, Germany, Singapore (Dec 5–9, 2010). https://doi.org/10.1007/978-3-642-17373-8_11
51. Li, F., Yan, J., Zhu, S., Hu, H.: A verifiable multi-secret sharing scheme based on short integer solution. *Chinese Journal of Electronics* **32**(3), 556–563 (2023). <https://doi.org/10.23919/cje.2021.00.062>
52. Li, J., Song, Y.: Constant-round asynchronous MPC with optimal resilience and linear communication. In: Kalai, Y.T., Kamara, S.F. (eds.) *Advances in Cryptology – CRYPTO 2025, Part IV*. Lecture Notes in Computer Science, vol. 16003, pp. 36–69. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2025). https://doi.org/10.1007/978-3-032-01884-7_2
53. Libert, B.: Simulation-extractable KZG polynomial commitments and applications to HyperPlonk. In: Tang, Q., Teague, V. (eds.) *PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Lecture Notes in Computer Science, vol. 14602, pp. 68–98. Springer, Cham, Switzerland, Sydney, NSW, Australia (Apr 15–17, 2024). https://doi.org/10.1007/978-3-031-57722-2_3
54. Liu, Y., Mo, X., Gao, Y., Lu, Y.: Aurelia: Dynamic asynchronous bft with bandwidth-efficient bootstrapping. *High-Confidence Computing* p. 100397 (2026). <https://doi.org/10.1016/j.hcc.2026.100397>
55. Lu, D., Yurek, T., Kulshreshtha, S., Govind, R., Kate, A., Miller, A.K.: HoneyBadgerMPC and AsynchroMix: Practical asynchronous MPC and its application to anonymous communication. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) *ACM CCS 2019: 26th Conference on Computer and Communications Security*. pp. 887–903. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3354238>
56. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of BFT protocols. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) *ACM CCS 2016: 23rd Conference on Computer and Communications Security*. pp. 31–42. ACM Press, Vienna, Austria (Oct 24–28, 2016). <https://doi.org/10.1145/2976749.2978399>
57. Momose, A., Das, S., Ren, L.: On the security of KZG commitment for VSS. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) *ACM CCS 2023: 30th Conference on Computer and Communications Security*. pp. 2561–2575. ACM Press, Copenhagen, Denmark (Nov 26–30, 2023). <https://doi.org/10.1145/3576915.3623127>
58. Momose, A., Ren, L.: Multi-threshold byzantine fault tolerance. In: Vigna, G., Shi, E. (eds.) *ACM CCS 2021: 28th Conference on Computer and Communications Security*. pp. 1686–1699. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484554>
59. Patil, S., Patra, A.: Perfectly-secure network-agnostic MPC with optimal resiliency. In: Balliu, A., Kuhn, F. (eds.) *44th ACM Symposium Annual on Principles of Distributed Computing*. pp. 121–130. Association for Computing Machinery, Huatulco, Mexico (Jun 16–20, 2025). <https://doi.org/10.1145/3732772.3733500>
60. Patra, A., Choudhury, A., Rangan, C.P.: Efficient asynchronous verifiable secret sharing and multiparty computation. *Journal of Cryptology* **28**(1), 49–109 (Jan 2015). <https://doi.org/10.1007/s00145-013-9172-7>

61. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) *Advances in Cryptology – CRYPTO’91*. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1992). https://doi.org/10.1007/3-540-46766-1_9
62. Shoup, V., Smart, N.P.: Lightweight asynchronous verifiable secret sharing with optimal resilience. *Journal of Cryptology* **37**(3), 27 (Jul 2024). <https://doi.org/10.1007/s00145-024-09505-6>
63. Shundong, L., Mengyu, Z., Wenting, X.: Secure scalar product protocols. *Chinese Journal of Electronics* **30**(6), 1059–1068 (2021). <https://doi.org/10.1049/cje.2021.07.018>
64. Su, Y., Lu, Y., Li, J., Wang, Y., Dong, C., Tang, Q.: Dumbo-MPC: Efficient fully asynchronous MPC with optimal resilience. In: Bauer, L., Pellegrino, G. (eds.) *USENIX Security 2025: 34th USENIX Security Symposium*. pp. 1925–1944. USENIX Association, Seattle, WA, USA (Aug 13–15, 2025), <https://www.usenix.org/conference/usenixsecurity25/presentation/su-yuan>
65. Tong, Y., Feng, Q., Luo, M., He, D.: Multi-party privacy-preserving decision tree training with a privileged party. *Science China Information Sciences* **67**(8), 182303 (2024)
66. Tsimos, G., Loss, J., Papamanthou, C.: Gossiping for communication-efficient broadcast. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology – CRYPTO 2022, Part III*. Lecture Notes in Computer Science, vol. 13509, pp. 439–469. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15982-4_15
67. Yao, A.C.C.: Theory and applications of trapdoor functions (extended abstract). In: *23rd Annual Symposium on Foundations of Computer Science*. pp. 80–91. IEEE Computer Society Press, Chicago, Illinois (Nov 3–5, 1982). <https://doi.org/10.1109/SFCS.1982.45>
68. Zhang, F., Zhang, J.: Efficient and information-theoretical secure verifiable secret sharing over bilinear groups. *Chinese Journal of Electronics* **23**(1), 13–17 (2014). <https://doi.org/10.23919/CJE.2014.10848034>

A Building Blocks in MPC

In this section, we propose the building blocks in MPC. First, we present the consensus preliminaries including BA and ACS.

Consensus Tools We introduce two consensus tools, BA and ACS. Blum et al. [20] propose the first BA in the network-agnostic setting, and Deligios et al. [37] further optimize the communication complexity.

Binary Byzantine Agreement (BA) Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of n parties in an agnostic network. Each party P_i starts with an input $b_i \in \{0, 1\}$. A binary Byzantine agreement protocol must satisfy:

- **Termination:** If all honest parties input, every honest party eventually outputs a decision value $d \in \{0, 1\}$. If one honest party outputs, all honest parties eventually output.
- **Agreement:** All honest parties output the same decision value d .
- **Validity:** If all honest parties input value b , then the decision value must be $d = b$. If the decision value $d = b$, at least one honest party must input b .

Asynchronous Common Subset (ACS) The ACS enables parties to get agreement on a common subset of these parties in the asynchronous setting, following the structure in [13, 56]. Recently, [21, 5, 4, 18, 19] proposed the ACS protocol in the network-agnostic setting. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be the set of n parties. An ACS protocol must satisfy the following properties:

- *Agreement:* All honest parties output the same set CS.
- *Validity:* There exists a publicly verifiable property $\mathcal{Q}$ such that every party $P_j \in \text{CS}$ satisfies $\mathcal{Q}$.
- *Liveness:* If at least $n - t$ honest parties satisfy $\mathcal{Q}$, then all honest parties eventually terminate and output a set CS of size at least $n - t$.

To achieve these properties, $\mathcal{Q}$ must be:

1. Every honest party will eventually meet the property $\mathcal{Q}$.
2. If any honest party verifies that a party P_j satisfies $\mathcal{Q}$, then all other honest parties will eventually also verify that P_j meets $\mathcal{Q}$.

Consequently, standard ACS protocols are typically constructed by integrating to a concrete protocol that instantiates $\mathcal{Q}$, such as VSS or an RBC protocol. A modified version of [18] is given in Appendix A.

Here, we propose an ACS protocol and two basic protocols proposed in [30] to support the offline phase of MPC. First, the ACS protocol follows the typical paradigm, which requires $O(n)$ BA instances. A drawback of this structure is its communication cost. The state-of-the-art network-agnostic BA costs $O(n^3\kappa)$ bits in communication, such that ACS costs $O(n^4)$. This structure was introduced in [13]. Many existing network-agnostic ACS [21, 5, 4] only focus on the common set

with an RBC protocol. [18,19] introduced ACS with VSS. We instead model the ACS in a more general case, with any valid prediction $\mathcal{Q}$, which is constructed using the same idea. We illustrate the idea in the following.

Protocol ACS

The protocol is parameterized with a prediction $\mathcal{Q}$ which satisfy:

- Every honest party will eventually meet the property $\mathcal{Q}$.
- If any honest party verifies that a party P_j satisfies $\mathcal{Q}$, then all other honest parties will eventually also verify that P_j meets $\mathcal{Q}$.

The protocol works as follows:

1. Every P_i , upon finding $\mathcal{Q}(P_k) \rightarrow 1$, if has not input into BA_k , input 1 into BA_k .
2. Every P_i , after inputting 1 into over $n - t_s$ BA instances, input 0 into every P_j that P_i has not input.
3. Every party P_i , Waits for every instances of BA has outputs, add P_k into $\mathcal{CS}$ if $\text{BA}_k \rightarrow 1$. Finally, output $\mathcal{CS}$.

Theorem 9. *The ACS protocol ACS satisfies the following properties:*

- *Agreement:* All honest parties output the same set $\mathcal{CS}$.
- *Validity:* There exists a publicly verifiable property $\mathcal{Q}$ such that every party $P_j \in \mathcal{CS}$ satisfies $\mathcal{Q}$.
- *Liveness:* If at least $n - t$ honest parties satisfy $\mathcal{Q}$, then all honest parties eventually terminate and output a set $\mathcal{CS}$ of size at least $n - t$.

Proof. Consider these properties separately:

- *Agreement:* Consider if there are two honest parties P_1, P_2 who output $\mathcal{CS}_1$ and $\mathcal{CS}_2$, respectively, each $\alpha_i \in \mathcal{CS}_1, \beta_i \in \mathcal{CS}_2$ must satisfy $\alpha_i = \beta_i$ thanks to the agreement property of BA. As a result, $\mathcal{CS}_1 = \mathcal{CS}_2$.
- *Validity:* Recall that BA_i outputs 1 on if there is at least one honest party inputs 1. Besides, if any honest party verifies that a party P_j satisfies $\mathcal{Q}$, then all other honest parties will eventually also verify that P_j meets $\mathcal{Q}$. Eventually, all honest parties will find that each $P_i \in \mathcal{CS}$ meets $\mathcal{Q}$.
- *Liveness:* As every honest party will eventually meet the property $\mathcal{Q}$, there are at least $n - t_s$ BA_i instances can output 1. Furthermore, honest parties will not input 0 before finding $n - t_s$ BA_i instances, such that $|\mathcal{Q}| \geq n - t_s$ and ACS terminates.

Then we propose the MPC building blocks. In intuition, the $\Pi_{\text{TripTrans}}$ outputs authentic correlated and shared polynomials $\llbracket f_a \rrbracket, \llbracket f_b \rrbracket, \llbracket f_c \rrbracket$ from correlated Beaver triples $\{\llbracket x \rrbracket, \llbracket y \rrbracket, \llbracket z \rrbracket\}$. First, following [30], we introduce Π_{Rec} first and further introduce the protocols $\Pi_{\text{TripTrans}}$ and $\Pi_{\text{TripCheck}}$. While the communication in Π_{Rec} can be improved from $O(n^2)$ to $O(n)$ with an extra round [30], we only provide the simple version, which costs $O(n^2)$.

Protocol Reconstruct Π_{Rec}

Given a batch of $L = n - t_s$ authenticated sharings, $\llbracket s_1 \rrbracket, \dots, \llbracket s_L \rrbracket$. Reconstruct all secrets for everyone as follows.

1. Define a polynomial $f(x) = s_1 + s_2x + \dots + s_Lx^{L-1}$ of degree $\leq L-1$.
2. For each $i = 1, \dots, n$, every party locally computes $\llbracket f(\alpha_i) \rrbracket = \sum_{k=1}^L \alpha_i^{k-1} \cdot \llbracket s_k \rrbracket$.
3. Every party send its shares (without $\mathcal{F}_{RBC}$) of $\llbracket f(\alpha_1) \rrbracket, \dots, \llbracket f(\alpha_n) \rrbracket$ to all the parties (without the commitment $\mathcal{C}$ in $\llbracket f(\alpha_i) \rrbracket$).
4. For each $i \in [n]$, each party reconstructs $f(\alpha_i)$ and interpolates f from the n points $(\alpha_i, f(\alpha_i))$ and outputs its coefficients $s_1, \dots, s_L$.

Lemma 3. *In the network-agnostic setting, let Π_{Rec} be the reconstruction protocol for authenticated shares $\llbracket x \rrbracket = ([x], [r], \mathcal{C})$, where $[x] = (x_1, \dots, x_n)$ and $[r] = (r_1, \dots, r_n)$ locate polynomials $f, r \in \mathbb{F}_{\leq t_s}[X]$, and $\mathcal{C} = \{C_j\}_{j=0}^{t_s}$ is the polynomial commitment with f, r . Then Π_{Rec} correctly reconstructs $x = f(0)$ for everyone with overwhelming probability. Besides, to reconstruct L secrets, Π_{Rec} costs $O(Ln\kappa + n^2\kappa)$ in communication.*

Proof. Based on the homomorphic property of polynomial commitments for each share (x_i, r_i) . Under the Binding property of polynomial commitment scheme, no corrupt party can forge a pair $(x'_i, r'_i) \neq (f(i), r(i))$ that pass the verification, except with negligible probability. Since at most t_s parties are corrupt, there remain at least $t_s + 1$ honest parties whose shares pass verification and lie on the polynomial f . Interpolating these $t_s + 1$ points uniquely determines f and yields the correct secret $x = f(0)$. Besides, in Π_{Rec} , the parties only broadcasts their shares with a proof. To reconstruct $n(n - t_s)$ secrets, there are n parties multicasting $|m| = O(n\kappa)$ bits, which costs $O(n \cdot n\kappa + n^3\kappa) = O(n^2\kappa)$ in communication. In amortization, Π_{Rec} costs $O(Ln\kappa + n^2\kappa)$ bits per sharing.

Protocol $\Pi_{TripTrans}$

Let each party inputs $(\theta, \text{points} = \{\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket \mid i \in [\theta]\})$, let $\mathbf{d} = \lfloor \frac{\theta-1}{2} \rfloor$.

1. Everyone interpolates univariate polynomials $\llbracket f_a \rrbracket, \llbracket f_b \rrbracket$ of degree $\mathbf{d}$, that $\llbracket f_a(i) \rrbracket = \llbracket a_i \rrbracket$ and $\llbracket f_b(i) \rrbracket = \llbracket b_i \rrbracket$ for $i \in [\mathbf{d} + 1]$.
2. For $i \in [\mathbf{d} + 1, 2\mathbf{d}]$, everyone yields $\llbracket f_a(i) \rrbracket, \llbracket f_b(i) \rrbracket$, and computes $\llbracket \hat{c}_i \rrbracket = \llbracket f_a(i) \rrbracket \cdot \llbracket f_b(i) \rrbracket$ using $\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket$ as Beaver triples. Specifically,
 - (a) Every party P_i reconstructs $\llbracket e \rrbracket = \llbracket f_a(i) \rrbracket - \llbracket a_i \rrbracket, \llbracket d \rrbracket = \llbracket f_b(i) \rrbracket - \llbracket b_i \rrbracket$ through Π_{Rec} .
 - (b) Every party P_i receives e, d from Π_{Rec} , let $\llbracket \hat{c}_i \rrbracket = ed + e\llbracket b \rrbracket + d\llbracket a \rrbracket + \llbracket c_i \rrbracket$.
3. Let $\llbracket \hat{c}_i \rrbracket = \llbracket c_i \rrbracket$ for $i \in [0, \mathbf{d}]$, interpolate $\llbracket f_c(x) \rrbracket =$ of degree $2\mathbf{d}$ from $\{\llbracket \hat{c}_i \rrbracket \mid i \in [0, 2\mathbf{d}]\}$.
4. Output $\llbracket f_a \rrbracket, \llbracket f_b \rrbracket, \llbracket f_c \rrbracket$.

Lemma 4. Let $\{(\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket)\}_{i \in [\theta]}$ be a set of θ authenticated shared triples where $\theta = 2d + 1$ and $d = \lfloor \frac{\theta-1}{2} \rfloor$. Then for every possible adversary, protocol $\Pi_{\text{TripTrans}}$ achieves the following:

1. **Termination:** All honest parties eventually terminate the protocol.
2. **Correctness:** The protocol outputs authenticated polynomial shares (the shared coefficients) $\llbracket f_a \rrbracket, \llbracket f_b \rrbracket, \llbracket f_c \rrbracket$ of degrees d, d , and $2d$ respectively, such that:
 - $f_a(i) = a_i$ and $f_b(i) = b_i$ for $i \in [d + 1]$
 - $f_c(x) = f_a(x) \cdot f_b(x)$ holds as a polynomial identity if and only if all input triples are valid multiplication triples (i.e., $c_i = a_i b_i$ for all $i \in [\theta]$)
3. **Privacy:** If the adversary knows $t' \leq d$ input triples (i.e., corrupts t' parties and learns their shares), then it learns at most t' distinct evaluation points on each of $f_a(\cdot)$, $f_b(\cdot)$, and $f_c(\cdot)$.
4. **Communication:** The communication complexity costs among all parties of $\Pi_{\text{TripTrans}}$ is $O(dn^2\kappa)$.

Note that the communication in $\Pi_{\text{TripTrans}}$ can be further improved to $O(dn\kappa)$, as shown in [30]. As this does not restrict the complexity of our protocol, we just take the simple version. Then we propose the protocol $\Pi_{\text{TripCheck}}$, which requires generating a common coin. While [53] proposed a common coin protocol based on the $\mathcal{F}_{\text{ACS}}$, $\mathcal{F}_{\text{ACS}}$ itself also requires the common coin protocol. We here simply assume the existence of $\mathcal{F}_{\text{Coin}}$, which actually can be realized using threshold signatures [37, 27, 20].

Protocol $\Pi_{\text{TripCheck}}$

Each party inputs $(\theta, \text{Trip} = \{\llbracket x_i \rrbracket, \llbracket y_i \rrbracket, \llbracket z_i \rrbracket \mid i \in [\theta]\})$ Beaver triples.

1. Each party let $d = \frac{\theta-1}{2}$, invoke $\Pi_{\text{TripTrans}}(\theta, \text{Trip})$.
2. Obtain $\llbracket f_x \rrbracket, \llbracket f_y \rrbracket, \llbracket f_z \rrbracket$, invokes $\mathcal{F}_{\text{Coin}}$ to obtain a random value r , publicly reveal $\llbracket f_x(r) \rrbracket, \llbracket f_y(r) \rrbracket, \llbracket f_z(r) \rrbracket$.
3. If $f_x(r) \cdot f_y(r) = f_z(r)$, outputs 1.

Lemma 5. Let θ be an odd integer with $\theta = 2d + 1$, and let everyone input points $\text{Trip} = \{(\llbracket x_i \rrbracket, \llbracket y_i \rrbracket, \llbracket z_i \rrbracket)\}_{i \in [\theta]}$ be a set of θ authenticated Beaver triples. Then protocol $\Pi_{\text{TripCheck}}$ satisfies:

1. **Termination:** All honest parties eventually terminate the protocol.
2. **Correctness:**
 - If $z_i = x_i y_i$ for all $i \in [\theta]$, then the protocol outputs 1 with probability 1.
 - If $z_i \neq x_i y_i$ for some $i \in [\theta]$, then the protocol outputs 0 with probability at least $1 - \frac{2d}{|\mathbb{F}|}$.
3. **Privacy:** The protocol reveals only the three values $f_x(r)$, $f_y(r)$, $f_z(r)$ for a single random point r , where f_x, f_y, f_z are the polynomials obtained from $\Pi_{\text{TripTrans}}$.
4. **Communication Complexity:** The protocol requires $O(\theta n^2 \kappa)$ bits of communication and one call to $\mathcal{F}_{\text{Coin}}$.

B Security Proofs on Π_{VSS}

We build our proof of protocols and sub-protocols as follows. We redefine the Ideal functionality and use the Ideal-Real hybrid prove framework as follows.

B.1 Proof of VSS Properties

First, we propose the formal definition of the bivariate polynomial Pedersen commitment scheme.

Definition 6 (Binding). *The bivariate polynomial commitment scheme $\Pi = (\text{Setup}, \text{Commit}, \text{VerifyPoly}, \text{VerifyPoint})$ is computationally binding if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{Binding}}(\kappa) = \Pr \left[\begin{array}{l} (\mathbb{G}, p, g, h) \leftarrow \text{Setup}(1^\kappa); \\ (\mathcal{C}, i, m, \alpha, \alpha', \beta, \beta') \leftarrow \mathcal{A}(\mathbb{G}, p, g, h) : \\ (\alpha, \alpha') \neq (\beta, \beta') \wedge \\ \text{VerifyPoint}(\mathcal{C}, i, m, \alpha, \alpha') = 1 \wedge \\ \text{VerifyPoint}(\mathcal{C}, i, m, \beta, \beta') = 1 \end{array} \right] \leq \text{negl}(\kappa),$$

where Setup outputs a cyclic group $\mathbb{G}$ of prime order p with independent generators $g, h \in \mathbb{G}$, and VerifyPoint is as defined in the scheme.

Definition 7 (Hiding). *The bivariate polynomial commitment scheme $\Pi = (\text{Setup}, \text{Commit}, \text{VerifyPoly}, \text{VerifyPoint})$ is statistically hiding if for each F and F'*

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{Hiding}}(\kappa) = \left| \Pr \left[\begin{array}{l} (\mathbb{G}, p, g, h) \leftarrow \text{Setup}(1^\kappa); \\ (F_0, F_1, \{\alpha_i\}_{i \in [\min(d_1, d_2)]}) \leftarrow \mathcal{A}(\mathbb{G}, p, g, h); \\ b \xleftarrow{\$} \{0, 1\}; R \xleftarrow{\$} \mathcal{R}; \mathcal{C} \leftarrow \text{Commit}(F_b, R) : \\ \mathcal{A}(\mathcal{C}, \{R(x, \beta), R(\beta, y)\}_{\beta \in \{\alpha\}}) = b \end{array} \right] - \frac{1}{2} \right| = 0,$$

where F_0 and F_1 are bivariate polynomials of degree at most (d_1, d_2) over $\mathbb{Z}_p$, $\mathcal{R}$ is the set of all bivariate polynomials of degree at most (d_1, d_2) over $\mathbb{Z}_p$, and $\text{Commit}(F, R)$ outputs a commitment matrix $\mathcal{C}$ as defined in the scheme.

Lemma 6 ([25]). *Assume the DLog problem, the Pederson bivariate polynomial commitment scheme $\Pi = (\text{Setup}, \text{Commit}, \text{VerifyPoly}, \text{VerifyPoint})$ is computationally binding and statistically hiding in the CRS hybrid model, .*

Proof. As proved in [25], we present why Lemma 6 holds in the CRS hybrid model. First, consider the *computationally binding* property.

In the CRS hybrid model, the simulator $\mathcal{S}$ can assign the system parameters. Let the simulator $\mathcal{S}$ break the DLog problem, such that $\mathcal{S}$ is given $\mathcal{S}(\mathbb{G}, p, g, g^x)$, $\mathcal{S}$ assign the CRS as $\text{crs} = (\mathbb{G}, p, g, h = g^x)$ and gives it to an adversary $\mathcal{A}$ who

breaks the *Binding* property, i.e. $\mathcal{A}(\mathbb{G}, p, g, h) \rightarrow (\mathcal{C}, i, m, \alpha, \alpha', \beta, \beta')$. Then, once find that

$$(\alpha, \alpha') \neq (\beta, \beta') \wedge \text{VerifyPoint}(\mathcal{C}, i, m, \alpha, \alpha') = 1 \wedge \text{VerifyPoint}(\mathcal{C}, i, m, \beta, \beta') = 1$$

The $\mathcal{S}$ gets $g^\alpha h^\beta = g^{\alpha'} h^{\beta'}$, i.e. $\alpha + x\beta = \alpha' + x\beta'$, and solves that $x = (\alpha - \alpha')(\beta' - \beta)^{-1}$. Formally, it holds

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{Binding}}(\kappa) \leq \text{Adv}_{DL}(\kappa)$$

Then, consider the *statistically hiding* property. This holds simply, as the commitment key $R \leftarrow \mathcal{R}$ is chosen randomly, the commitment $\mathcal{C} = g^{F_{i,j} + xR_{i,j}}$ follows randomly and uniform distribution, regardless of F . Besides, opening $\min(d_1, d_2)$ leaves at least one random mask element unknown in both dimensions. As a result, the *statistically hiding* property holds.

Redefine Theroem 3 If hiding and binding properties hold in the commitment scheme, the signature scheme is EUF-CMA secure, and the adversary follows the network-agnostic restriction, within the PKI setup, protocol Π_{VSS} satisfies the following properties with overwhelming probability:

- *Termination*
 1. If the dealer is honest, all honest parties will complete the protocol.
 2. If any honest party completes the protocol, then every honest party will complete the protocol.
- *Correctness* The combined view of all honest parties at the end of the sharing phase uniquely defines a secret $\mathbf{s}'$. Moreover, if the dealer is honest and inputs $\mathbf{s}$, then $\mathbf{s}' = \mathbf{s}$.
- *Secrecy* If the dealer is honest, the adversary's view throughout the protocol is statistically independent of $\mathbf{s}$.

Proof. Define the adversary's advantage in breaking the Termination, Correctness properties of Π_{VSS} as $\text{Adv}_{\Pi_{VSS}, \mathcal{A}}^{\text{Termination}}$, $\text{Adv}_{\Pi_{VSS}, \mathcal{A}}^{\text{Correctness}}$ respectively.

First, consider the Termination property, given a simulator $\mathcal{S}$ working to break the EUF-CMA security of signatures with (pk) input. $\mathcal{S}$ first works like a set of honest parties in the protocol Π_{VSS} with adversary $\mathcal{A}$, except for the signature scheme. At the start, the $\mathcal{S}$ randomly chooses one party from the honest parties, i.e., $P_i \xleftarrow{\$} n$. Once the adversary decides to corrupt P_i , $\mathcal{S}$ aborts. Other parties' keys are generated honestly. Once P_i is asked to sign m , $\mathcal{S}$ queries the signing oracle, as in a chosen message attack. After the adversary broadcasts the $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \Omega)$, if P_i has not sign $\mathcal{C}$ but $\{\sigma_i\}$ includes a valid signature that P_i signed. $\mathcal{S}$ outputs the $m = \mathcal{C}, \sigma_i$ and breaks the EUF-CMA security. If the $\{\sigma_i\}$ includes a signature that $P_j \neq P_i$ has not signed, abort. Otherwise, i.e., each party $P_i \in \mathcal{CS}$ has honestly signed the commitment $\mathcal{C}$, implying that each honest party in $\mathcal{CS}$ has received a valid commitment and can help others to reconstruct, as illustrated above. Formally, we have

$$\text{Adv}_{\Pi_{VSS}, \mathcal{A}}^{\text{Termination}} \leq n \text{Adv}_{\Pi, \mathcal{A}}^{\text{EUF-CMA}}$$

Second, consider the Correctness property, given a simulator $\mathcal{S}$ working to break the Binding of a bivariate polynomial commitment scheme. Once $\mathcal{S}$ is given $\mathcal{A}(\mathbb{G}, p, g, h)$, it sets $crs = (\mathbb{G}, p, g, h)$ as the commitment parameter. $\mathcal{S}$ works like a set of honest parties with the adversary $\mathcal{A}$ and collect each points $f_{i,j} = F(i, j), r_{i,j} = R(i, j)$ that $\mathcal{A}$ claims $g^{f_{i,j}} h^{r_{i,j}} = \prod_{\alpha}^{d_1} \prod_{\beta}^{d_2} C_{i,j}^{\alpha j^{\beta}}$, where $\mathcal{C} = \{C_{i,j}\}$ is the commitment broadcasted by $\mathcal{A}$. After the simulated protocol terminates, the simulator check is all the shares in honest parties interpolate the distinct bivariate polynomial F, R . If not, interpolate both F, R and F', R' , evaluate at a different point, for example $F(k, \ell) \neq F'(k, \ell)$ and $R(k, \ell) \neq R'(k, \ell)$, it must satisfy $g^{F(k, \ell)} h^{R(k, \ell)} = g^{F'(k, \ell)} h^{R'(k, \ell)}$. As a result, $\mathcal{S}$ outputs $(\mathcal{C}, k, \ell, F(k, \ell), F'(k, \ell), R(k, \ell), R'(k, \ell))$ to break the Binding property. As a result, formally,

$$\text{Adv}_{\Pi_{VSS}, \mathcal{A}}^{\text{Correctness}} \leq \text{Adv}_{\Pi, \mathcal{A}}^{\text{Binding}}$$

Finally, we prove that the view of the adversary is statistically independent of the secret. As mentioned, regardless of whether in synchronous or asynchronous settings, the adversary's view is not beyond t_s parties, including the commitment. The simulator $\mathcal{S}$ is given the view for corrupted parties, $\{P_i\}_{i \in \text{Corr}}$. Note that, if in the asynchronous setting, $\{P_i\}_{i \in \text{Corr}}$ includes the virtual corrupt parties, $P_{n+1}, \dots, P_{n+t_s-t_a}$. As shown in the protocol, the adversary's view is not beyond $(\mathcal{C}, \{R(x, i), R(i, y), F(x, i), F(i, y)\}_{i \in \text{Corr}})$. This is obviously independent of the secrets $F(-k, 0)_{k \in [1, n-t_s-t_a]}$. In otherwords, assume $\mathcal{S}$ is assigned to break the binding of the commitment, no matter how $\mathcal{S}$ sets the secret and interpolates the polynomial F, F' , the results it received followed the same distribution. Otherwise, the commitment is independent and uniform. Besides, as the row polynomial $f(x)$ is of degree $n - t_a - 1$, and the adversary only gets at most t_s linearly independent combination points, these points are independent from the distribution of the $n - t_s - t_a$ secrets.

B.2 Proofs of Signature in UC

The signature scheme does not directly depend on the network, and the signature functionality does not require a router machine. We build the functionality based on [28], which proves a PKI-based signature scheme is proven secure in the UC framework. Our functionality and proof idea comes from [28]. Focus on the **SigGen** entry, the functionality allows the adversary to assign signatures for all honest parties, such that the simulator can bind among parties, corresponding messages and signatures, even without secret keys.

$\mathcal{F}_{\text{Sig}}$

The functionality is parameterized by a set of parties $\{P_i\}_{i \in [n]}$ and honest parties $H \subseteq \{P_i\}_{i \in [n]}$ selected by $\mathcal{A}$.

KeyGen Upon receiving a value (KeyGen) from some party P_i that $P_i \in H$ for the first time, send (KeyGen, P_i) to the $\mathcal{A}$. Upon re-

ceiving $(\text{VerificationKey}, P_i, v_i)$ from the $\mathcal{A}$ for the first time, record (VK, P_i, v_i) and send (VK, P_i, v_i) to the P_i .

SigGen Upon receiving (Sign, m) from P_i , if the entry (VK, P_i, v_i) is recorded, send (Sign, m, P_i) to $\mathcal{A}$. Upon receiving (Sign, m, σ) from the $\mathcal{A}$, and no entry $(\text{Signature}, m, \sigma, v_i, 0)$ has been recorded, record $(\text{Signature}, m, \sigma, v_i, 1)$ and send $(\text{Signature}, m, \sigma, v_i, 1)$ to P_i directly.

SigVer Upon receiving a value $(\text{Verify}, m, \sigma, P_i)$ from every party P_j , if not record (VK, P_i, v_i) , send (Abort) to P_j . Otherwise, send $(\text{Verify}, m, \sigma, P_i)$ to the $\mathcal{A}$ and receive ϕ , output $(\text{Verified}, P_i, m, f)$ to the P_j where

- if $(\text{Signature}, m, \sigma, v_i, f')$ is recorded, set $f = f'$.
- Else, if the signer $P_i \in H$, set $f = 0$, otherwise set $f = \phi$, record $(\text{Signature}, m, \sigma, v_i, \phi)$.

Lemma 7 ([28]). *Assuming PKI, $\mathcal{F}_{\text{Sig}}$ UC-emulates any sequential execution of a signature scheme that is EUF-CMA secure.*

Proof. Note that $\mathcal{F}_{\text{Sig}}$ sends and receives messages directly, but not using the router. In initialization, the simulator is given the not corrupted set H , a list of random tapes $\{rp_i\}_{i \in [n]}$ of IMI P_i , and an adversary oracle $\mathcal{A}'$. In the ideal world, the view of any party will be $\text{IDEAL}_{\mathcal{S}, \mathcal{F}, \mathcal{Z}} = (rp_i, sk_i, vk_i, \{\sigma\}, \{\phi\})$, where σ is the signature and ϕ is the verified result. We let $\mathcal{S}$ simulate the view in different hybrids.

Hybrid₀: In this hybrid, the simulator $\mathcal{S}$ is given the secret key sk_i that every honest party holds. Then the simulator follows the protocol and acts just like a set of honest parties.

Hybrid₁: In this hybrid, the simulator has no idea about the private key of honest parties, it instead generated a forged key for every party $P_i \in H$, the simulator generates a list $\{(sk_i, vk_i)\}_{i \in [n]}$, that $(sk_i, pk_i) \leftarrow \text{MS.G}(1^\kappa, rp_i)$. Then the simulator let $\mathcal{A}$ accept the forged pk_i as the public key of honest P_i . $\mathcal{S}$ follows the protocol and acts just like a set of honest parties. **Hybrid₁** and **Hybrid₀** are not distinguishable as the simulated view follows the same distribution, just with a different random tape.

Hybrid₂: In this hybrid, the simulator interacts with the functionality instead of the protocol. Besides, the simulator notifies every time once the $\mathcal{A}$ signs a message. $\mathcal{S}$ generated forged keys list $\{(sk_i, vk_i)\}_{i \in [n]}$, that $(sk_i, pk_i) \leftarrow \text{MS.G}(1^\kappa, rp_i)$, then:

- Let $\mathcal{A}$ accept the forged pk_i as public key of honest P_i .
- Once any corrupt signer regist on the PKI, $\mathcal{S}$ invokes the functionality entry **KeyGen**, once receiving (KeyGen, p_i) , pass $(\text{VerificationKey}, P_i, pk_i)$ to the $\mathcal{F}_{\text{Sig}}$.
- Once any honest party invokes **SigGen** to sign m , $\mathcal{S}$ receives (Sign, m, P_i) , and assign signature $\sigma_i = \text{Sign}(sk_i, m)$ to be the signature of m .
- Once adversary or environment generates a valid signature that can pass **Verify**, record by invoking **SigGen** entry of $\mathcal{F}_{\text{Sig}}$.

Note that the honest party will register his own public key in the functionality, while the $\mathcal{S}$ do not know the corresponding private key sk_i . Instead, $\mathcal{S}$ forged a private key and generated a forged signature, but the functionality allows the $\mathcal{S}$ to assign the forged signature to be a "real" signature. Besides, all the signatures (even generated by $\mathcal{A}$) will be registered on $\mathcal{F}_{multisig}$. **Hybrid₂** and **Hybrid₁** are not distinguishable; in both hybrids, the public key, private key, and signatures are all forged by the $\mathcal{S}$, such that follows the same distribution. Besides, for the honest party, the signatures work just like **Hybrid₁**. The only exception is the **verified result**. In the **Hybrid₁**, the signature is verified through an algorithm, but in the **Hybrid₂**, the signature is verified through a recording in the functionality.

Let ϕ_1 denote the verified result in **Hybrid₁**, ϕ_2 denote the verified result in **Hybrid₂**. Note that the corrupt signatures are directly exposed to the $\mathcal{S}$, s.t. the corrupt verification result must be the same distribution. If $\phi_1 = no, \phi_2 = yes$, it means the recorded signature is not verified through the **Verify**. This violates the correctness of the signature scheme. If $\phi_1 = yes, \phi_2 = no$, this means the adversary, without sk , generated a signature which can pass the **Verify**. In other words, if **Hybrid₁** and **Hybrid₂** are distinguishable, there exists a simulator that can win the EUF-CMA signature game. A formal EUF-CMA game is defined as follows:

$$\begin{aligned}
& G_{\mathcal{A}}^{EUF-CMA}(1^\kappa) : \\
& pk, sk \leftarrow \text{Sign}(1^\kappa) \\
& \{m_i\} \leftarrow \mathcal{S}'(pk_i) \\
& \{\sigma_i = \text{Sign}(sk_i, m_i)\} \\
& (m_f, \sigma_f) \leftarrow \mathcal{S}'(\{\sigma_i\}) \\
& \text{If } m_f \neq m_i \text{ and } 1 \leftarrow \text{Verify}(pk, m_f, \sigma_f) \\
& \text{returns 1; Otherwise return 0}
\end{aligned}$$

Then $\mathcal{S}'$ participates in the above game and works just like $\mathcal{S}$ in **Hybrid₂** except let $vk_i = pk_i$ for some honest party P_i . Upon receiving (Sign, m, P_i) from $\mathcal{F}_{Sig}$, ask for signature from the game $G_{\mathcal{A}}^{EUF-CMA}(1^\kappa) : \{m_i\} \leftarrow \mathcal{S}(pk_i)$. Upon $\mathcal{F}_{Sig}$ sends a signature $(\text{Verify}, m, \sigma_i, P_i)$ in the **SigVer** entry, and $m \notin \{m_i\}$, $1 \leftarrow \text{Verify}(m)$, output m, σ_i , which support $\mathcal{S}'$ to win the game $G_{\mathcal{A}}^{EUF-CMA}(1^\kappa)$.

B.3 Proofs of RBC

Notations In the following, as we model the network as a router $\mathcal{R}$, we denote $(P_i \rightarrow P_j : m)$ as a message sent from P_i to P_j with message body m . Besides, we denote $(\{P_i\} \rightarrow \{P_j\} : m_{i,j})$ ($i, j \in [n]$) as a set of messages (n^2 entries) sent from a set of parties $\{P_i\}$ to a set of parties $\{P_j\}$. Moreover, in the synchronous setting, we allow $\hat{\mathcal{R}}$ to give a longer time window $\hat{\Delta}$ than $\mathcal{R}$, that is, Δ , i.e., $\hat{\Delta}$ depends on the round complexity of the protocol.

Security Analysis First, we focus on the *Input Phase* and *Check Phase*, and note that our modification will not affect the agreement on the Merkle root. Formally, we have

Lemma 8. *In the protocol Π_{RBC} , assuming PKI for an EUF-CMA secure signature scheme, let the adversary follow the network-agnostic restriction. If one honest party sets $\text{out} = h$, all honest parties will finally set $\text{out} = h$.*

Proof. This lemma follows the same idea as in [58]. First, we have the claim:

Claim(1): In the synchronous setting, the lemma 8 holds

In the synchronous setting, it is ensured that the message will arrive within a time window Δ . Consider the case where two honest parties have broadcast inconsistent $(\text{propose}, h)_S$ and $(\text{propose}, h')_S$. Then, during the waiting period, at least one party (the one that sent later) will see a conflicting $(\text{propose}, h^*)_S$ from the party that sent earlier and will not vote for either h or h' . In summary, all honest parties that vote will vote for the same value h . Once an honest party sets $\text{out} = h$, it has received $n - t_s \geq t_s + 1$ (vote, h) messages and broadcast the quorum certificate. As a result, there cannot be another quorum-certificate for a different value with $n - t_s$ parties, and all honest parties will eventually receive this quorum and set $\text{out} = h$.

Claim(2): In the asynchronous setting, the lemma 8 holds

In the asynchronous setting, if there is an honest party who outputs set $\text{out} = h$, it must receive a quorum-certificate (vote, h) of $n - t_s$ parties. As the adversary corrupts at most t_a parties, there cannot be another quorum-certificate (vote, h') of $n - t_s$ parties, as $2(n - t_s) = n + (n - 2t_s) \geq n + t_a + 1$. This happens only if one honest party has voted for both h and h' , which is impossible. Besides, as the honest who set $\text{out} = h$ will also broadcast the quorum-certificate (vote, h) , all honest parties will set $\text{out} = h$.

Based on *Claim(1)* and *Claim(2)*, the lemma holds for both network models, which completes the proof.

Then, after reaching agreement on the Merkle tree root h , every honest party can verify whether a codeword symbol $\mathbf{c}[i]$ is committed to by the tree via an authentication path. The parties then disseminate the codeword symbols, enabling the reconstruction of the message. Formally, we have:

Theorem 10. *In the protocol Π_{RBC} , assuming the adversary follows the network-agnostic restriction, if one honest party sets $\text{out} = h$, then all honest parties will eventually output a consistent message m that is bound to h .*

Proof. First, we claim that if one honest party sets $\text{out} = h$, then at least one honest party has received the valid codeword $\mathbf{c}$ (encoding m). This is straightforward because at least one (vote, h) message is sent by an honest party, which implies that party has received the valid $\mathbf{c}$.

Furthermore, assuming the Merkle tree is perfectly binding, honest parties will only accept a valid $\mathbf{c}$ (i.e., the correct codeword of m). Also, any honest party that has received $\mathbf{c}$ will transfer $\mathbf{c}[k]$ to P_k with a proof, so that all P_k will accept a valid $\mathbf{c}[k]$. Finally, these parties P_k will each disseminate their own $\mathbf{c}[k]$ symbol. Since there are at least $n - t_s$ honest parties, everyone will eventually receive $n - t_s$ valid codeword symbols and reconstruct m , which is encoded using a Reed-Solomon code of degree $n - t_s$.

Theorem 11. In Π_{RBC} , assuming PKI for an EUF-CMA secure signature scheme, let the adversary follow the network-agnostic restriction, the protocol Π_{RBC} satisfies the following properties:

- *Consistency:* If any pair of honest parties output m and m' , then $m = m'$.
- *Integrity:* If the dealer is honest and inputs m , then no honest party will output m' for $m' \neq m$.
- *Validity:* If the dealer is honest, every honest party eventually terminates.
- *Totality:* If any honest party finishes the protocol, all honest parties eventually terminate.

Proof. First, if an honest party outputs m , it must set $\text{out} = h$. Besides, based on Theorem 10, all honest parties will output m . In other words, *Consistency* and *Totality* properties hold in the protocol Π_{RBC} .

Moreover, consider the *Integrity* and *Validity*, if the dealer is honest, it will sign and only sign m (specifically, the Merkle root h binding to m). Assume the signature scheme is unforgeable, there cannot be an inconsistent $(\text{propose}, h')$, and all honest parties will vote for (vote, h) . Finally, all the parties will set $\text{out} = h$. Based on Theorem 10, all honest parties output m . In other words, the *Consistency* and *Totality* properties hold.

Theorem 12. Assuming PKI for an EUF-CMA secure signature scheme, $\mathcal{F}_{RBC}$ UC-emulates protocol Π_{RBC} in the network-agnostic setting.

$\mathcal{F}_{RBC}$

The functionality is parameterized by a set of honest parties $H \subseteq [n]$.

Input(m) this operation is invoked once by the sender S , who inputs a message m . In response, $\mathcal{F}_{RBC}$ sends the message $\text{NotifyInput}(m)$ to the ideal-world adversary and the message $\text{Output}(m)$ to P_i for $i \in [n]$ through the router $\hat{\mathcal{R}}$.

Theorem 13. Given an ideal functionality $\mathcal{F}_{Sig}$, $\mathcal{F}_{RBC}$ UC-emulates protocol Π_{RBC} in the agnostic network.

Proof. Like lemma 7, in Π_{RBC} , we model the signature scheme as functionality $\mathcal{F}_{Sig}$. Then, let the real-world router be $\mathcal{R}$ and the ideal-world router be $\hat{\mathcal{R}}$, the simulator $\mathcal{S}$ works as follows. The $\mathcal{S}$ simulates follows the following rules:

1. Upon receiving any message that from $\mathcal{Z}$ to the adversary, pass it to the $\mathcal{A}$,
Upon receiving any message that from $\mathcal{A}$ to the environment, pass it to $\mathcal{Z}$.
2. All the signatures in the following are comes from the $\mathcal{F}_{Sig}$, i.e., let $\mathcal{A}$ assign signatures signed by honest party and decide verified result for corrupt signatures (regardless of if it is signed before), s.t. the simulator $\mathcal{S}$ will get corresponding signatures and verified results.

3. If any message sent from the corrupt parties arrived at the router $\hat{\mathcal{R}}$, transfer it to the adversary $\mathcal{A}$.

Hybrid₀: In this hybrid, the simulator $\mathcal{S}$ follows the protocol and acts just like a set of honest parties. Note that in the $\mathcal{F}_{sig}$, let the adversary decide all the signatures in $\mathcal{F}_{sig}$ and check the signatures from it.

Hybrid₁: In this hybrid, the simulator $\mathcal{S}$ interacts with the $\mathcal{F}_{RBC}$, considering two cases: the sender is honest or corrupt:

1. **Honest sender**: Upon receiving $\text{NotifyInput}(m)$, utilize m to perform as an honest sender, just acts like the honest parties in the protocol, and simulate the view for the adversary.
2. **Corrupt sender**: $\mathcal{S}$ follows the protocol, acts as a set of honest parties. For all $P_i \in H$, upon receiving $(\text{propose}, h, \sigma_S)$ check σ_S through $\mathcal{F}_{Sign}$, i.e. let the adversary decide the result. If the check passed and $\mathcal{A}$ sends valid $\mathbf{c}$ to P_i , simulate P_i to broadcast $(\text{vote}, h)_{P_i}$. Upon any P_i is designated to receive $n - t_s$ $(\text{vote}, h)_{P_k}$, decode m from $\mathbf{c}$, which is decided by h , and send m into $\text{Input}(\mathbf{m})$ entry of $\mathcal{F}_{RBC}$. Then, $\mathcal{S}$ follows the protocol with $\mathcal{A}$, after any honest P_i who receives $n - t_s$ different $(\text{rec}, \mathbf{c}[k], \pi_k)$, allows $\hat{\mathcal{R}}$ to transfer $\text{Output}(m)$ to P_i .

In this hybrid, $\mathcal{S}$ is given the full Merkle tree with root h (as at least one honest party receives), if the adversary gives any authentic path that is not located on the original Merkle tree. The simulator can identify two different messages that map to the same hash value and break the collision resilience of a hash function.

Hybrid₀ is exactly the protocol, except the $\mathcal{F}_{sig}$ replaces all the parts of signature. **Hybrid₁** and **Hybrid₀** are not distinguishable. As proved in the lemma 8, if one honest party sets $\text{out} = h$, all honest parties will finally set $\text{out} = h$. As a result, in both **Hybrid₀** and **Hybrid₁**, upon any P_i is designated to receive $n - t_s$ $(\text{vote}, h)_{P_k}$, i.e. set $\text{out} = h$ all honest parties will only output m binding on h . In the **Hybrid₁**, after $\mathcal{S}$ sends m into $\text{Input}(\mathbf{m})$, everyone finally obtains m . This also works in **Hybrid₀**, follows the *Consistency* and *Totality*, after any party sets $\text{out} = h$, all honest parties receive consistent m .

B.4 Proofs of VSS in UC

Functionality: Packed VSS $\mathcal{F}_{VSS}$

The functionality is parameterized by a set of honest parties $H \subset [n]$ of totally n parties

Share($\text{Input}, F(x, y)$). Upon being invoked by the sender, if the sender is corrupt, it also receives $R(x, y)$ from the adversary; otherwise, it randomly samples R . Finally, let $\mathcal{C} \leftarrow \text{Commit}(F(x, y), R(x, y))$, and

- send $(\text{share}, F(x, i), F(i, y), R(x, i), R(i, y), \mathcal{C})$ to $\mathcal{A}$ for $i \in [n] \setminus H$. If $n - H \leq t_a$, further reveal $(\text{share}, F(x, i), F(i, y), R(x, i), R(i, y))$ to $\mathcal{A}$ for $i \in [n + 1, n + t_s - t_a]$.

– send (**output**, $F(x, i)$, $R(x, i)$, $\mathcal{C}$) to P_i for $i \in [n]$, through the router.

Redefine theorem 4 Assuming access to ideal functionality $\mathcal{F}_{RBC}$ and $\mathcal{F}_{Sig}$, $\mathcal{F}_{VSS}$ UC-emulates Π_{VSS} .

Proof. Given access to $\mathcal{F}_{RBC}$ and $\mathcal{F}_{Sig}$, the simulator $\mathcal{S}$ works as follows. Let the real-world router be $\mathcal{R}$ and the ideal-world router be $\hat{\mathcal{R}}$; if the network is asynchronous, the $\mathcal{S}$ can arbitrarily delay messages at $\hat{\mathcal{R}}$; if the network is synchronous, the $\mathcal{S}$ can delay messages at $\hat{\mathcal{R}}$ at most $2T_{BC} + 3\Delta$. For simplicity, when describing the network model, we abuse the **$\mathcal{A}$ -allowed** to refer to both cases: (1) the time over Δ in synchronous networks, or (2) the adversary allows $\mathcal{R}$ to pass the message. The $\mathcal{S}$ simulates follows the following rules:

1. Upon receiving any message that from $\mathcal{Z}$ to the adversary, pass it to the $\mathcal{A}$. Upon receiving any message that from $\mathcal{A}$ to the environment, pass it to $\mathcal{Z}$.
2. All the signatures in the following are comes from the $\mathcal{F}_{Sig}$, i.e., let the adversary $\mathcal{A}$ to assign signatures signed by honest party and decide verified result corrupt signatures (regardless of if it is signed before), s.t. the simulator $\mathcal{S}$ will get corresponding signature and verified result.
3. If any message sent from the corrupt parties arrived at the router $\hat{\mathcal{R}}$, inform the adversary $\mathcal{A}$ with the header.

In the following, we let $\mathcal{S}$ work through a sequence of hybrids. For $i \in \{0, 1, 2, 3\}$, let View_i denote the output distribution of the environment $\mathcal{Z}$ in Hybrid_i . We show that every two adjacent hybrids are computationally indistinguishable.

Hybrid₀: In this hybrid, $\mathcal{S}$ works in the real world with a protocol. Specifically, if the dealer is honest, $\mathcal{S}$ is given the shared secrets F, R message that the honest dealer holds; otherwise, $\mathcal{S}$ is given nothing about the input message. Then it follows the protocol and acts just like a set of honest parties. Hybrid_0 has no difference with the original protocol. Based on the theorem, the protocol finally terminates, and further satisfies *Correctness* and *Secrecy*.

Hybrid₁: In this hybrid, only for an honest dealer, $\mathcal{S}$ interacts with $\mathcal{F}_{VSS}$ and receives the shared secrets. $\mathcal{S}$ can simply share the secrets with $\mathcal{A}$. Specifically, for $i \in [n] \setminus H$, $\mathcal{S}$ receives

$$(\text{share}, F(x, i), F(i, y), R(x, i), R(i, y), \mathcal{C})$$

If $n - H \leq t_a$, it further receives $(\text{share}, F(x, i), F(i, y), R(x, i), R(i, y))$ for $i \in [n + 1, n + t_s - t_a]$.

Besides, let the $\mathcal{A}$ decide the set of $\mathcal{CS}$. If in the synchronous setting, all honest parties will be included in $\mathcal{CS}$, while in the asynchronous setting, the $\mathcal{A}$ totally decides the $\mathcal{CS}$, as long as $|\mathcal{CS}| \geq n - t_s$. In the synchronous setting, any P_i outside $\mathcal{CS}$ is corrupt, such that P_i has already received the shares and can simulate the broadcast phase. In the asynchronous setting, the $\mathcal{S}$ is able to

reveal $t_s - t_a$ points for honest P_i as it holds $(F(x, i), F(i, y), R(x, i), R(i, y))$ for $i \in [n + 1, n + t_s - t_a]$.

For an honest dealer, $\mathcal{F}_{VSS}$ samples the masking polynomial with the same distribution as in the real execution and provides exactly the views exposed by the protocol; for a corrupt dealer, Hybrid_1 is unchanged from Hybrid_0 . Therefore, it holds for $\text{View}_0 \equiv \text{View}_1$.

Hybrid₂: In this hybrid, for both honest or corrupt dealers, $\mathcal{S}$ only interacts with $\mathcal{F}_{VSS}$; all the points and polynomials revealed by corrupted parties must be located on the polynomials F, R . The difference between Hybrid_2 and Hybrid_1 is in the case of a corrupt dealer. Focus on this case, in the *Share* and *Check* phase, the simulator works as a set of honest parties in the protocol. Once the $\mathcal{A}$ broadcast a message through $\mathcal{F}_{RBC}$, parse the message as $(\mathcal{C}, \mathcal{CS}, \mathbf{\Omega}, \{\sigma_i | P_i \in \mathcal{CS}\})$. If any check in the following fails, abort.

- Each $P_i \in \mathcal{CS}$ has signed $(\text{sid}, \mathcal{C})$ and $|\mathcal{CS}| \geq n - t_s$.
- Each $P_i \notin \mathcal{CS}$, $\mathbf{\Omega}$ is constructed as Eq 1
- Every point $(f_{x,y}, r_{x,y}) \in \mathbf{\Omega}$, $\text{VerifyPoint}(\mathcal{C}, f_{x,y}, r_{x,y}) \rightarrow 1$.

For any honest parties $P_i \in \mathcal{CS}$, the adversary $\mathcal{A}$ must share

$$(F(x, i), F(i, y), R(x, i), R(i, y))$$

to the simulator $\mathcal{S}$. As $|\mathcal{CS}| \geq n - t_s$, there at least $t_a + 1$ shares. Furthermore, for any $P_j \notin \mathcal{CS}$, the simulator $\mathcal{S}$ also receives $t_s - t_a$ points on the polynomials. Then the simulator $\mathcal{S}$ can interpolate $F(x, y), R(x, y)$ and send $(\text{Input}, F(x, y), R(x, y))$ to the $\mathcal{F}_{VSS}$. Finally, $\mathcal{S}$ let $\hat{\mathcal{R}}$ deliver the shares to those $P_i \in \mathcal{CS}$. Once any $P_i \notin \mathcal{CS}$ is allowed to receive enough shares to reconstruct the row polynomial of degree $n - t_a - 1$, $\mathcal{S}$ let $\hat{\mathcal{R}}$ deliver the shares to those P_i .

If the dealer is corrupt, we do not care about the secrecy. Upon a valid message $(\mathcal{C}, \mathcal{CS}, \mathbf{\Omega}, \{\sigma_i | P_i \in \mathcal{CS}\})$ has been broadcasted by the dealer, the protocol hybrid_1 will terminate just like hybrid_2 . Otherwise, both hybrid will abort. As a result, hybrid_1 and hybrid_2 are not distinguishable.

The two hybrids differ only if a corrupt dealer produces two inconsistent openings that are nevertheless accepted with respect to the same commitment C . This event yields an adversary against the binding property of the commitment scheme. Therefore, for every PPT environment $\mathcal{Z}$,

$$|\Pr[\text{View}_1 = 1] - \Pr[\text{View}_2 = 1]| \leq \text{Adv}_{H, B}^{\text{Binding}}(\kappa) = \text{negl}(\kappa).$$

Hybrid₃: $\mathcal{S}$ interacts with $\mathcal{F}_{VSS}$, and we do not ask the adversary to reveal points located on any polynomial, but the revealed univariate polynomials must lie on the bivariate polynomials F, R . In this hybrid, $\mathcal{S}$ is assigned to break the DLog Assumption. First, upon given the $(\mathbb{G}, p, g, g^x)$, $\mathcal{S}$ sets $g, h = g^x$ as the *crs* of the Pedersen Commitment scheme. Then, the simulator $\mathcal{S}$ works just like Hybrid_2 to simulate the protocol. If the final interpolation phase fails, i.e., the interpolated F, R is over degree $(n - t_a - 1, t_s)$, the simulated protocol aborts.

Once the simulated protocol terminates or aborts, $\mathcal{S}$ collects every row polynomials (f_i, r_i) and column polynomials (f'_i, r'_i) for honest parties, along with

each $g^{f_i(j)}h^{r_i(j)}$ and $g^{f'_i(j)}h^{r'_i(j)}$ for $j \in [n]$ (can be evaluated). $\mathcal{S}$ also collects the commitment $\mathcal{C}$ and common set $\mathcal{CS}$ broadcasted by the dealer.

The basic proof idea is that, as the $\mathcal{S}$ has collected the commitment $\mathcal{C}$, it can perform homomorphic evaluation to obtain the commitment on any point. The honest parties accept a point only if $g^{F(u,v)}h^{R(u,v)} = \prod_{i=0}^{d_1} \prod_{j=0}^{d_2} C_{i,j}^{u^i v^j}$. Once the simulator finds two different $F(u,v)_1$ and $F(u,v)_2$ that are accepted, it breaks the binding property of the bivariate polynomial Pedersen commitment.

If the simulated protocol aborts, w.l.o.g., assume $F(i,y)$ is of degree over $t_s + 1$, where $\mathcal{S}$ collects at least $t_s + 2$ points (as $\mathcal{S}$ holds all the view from honest parties). $\mathcal{S}$ utilizes two sets of $t_s + 1$ points to interpolate and evaluate two different $[F(i,0)]_1$ and $[F(i,0)]_2$ and wins the binding game. Otherwise, $\mathcal{S}$ commits $\bar{\mathcal{C}} = \text{Commit}(F, R)$ where F, R is the interpolated polynomial. We claim that if the simulated protocol does not abort, it is ensured that $\bar{\mathcal{C}} = \mathcal{C}$. Intuitively, at least $(n - t_a) \cdot (n - t_s)$ points on F, R have been checked and satisfy $g^{F(u,v)}h^{R(u,v)} = \prod_{i=0}^{d_1} \prod_{j=0}^{d_2} C_{i,j}^{u^i v^j}$, which totally determine $F, R, \mathcal{C}$. Specifically, for $u \in H \cup \{n + 1, \dots, n + t_s - t_a\}$, $v \in (\mathcal{CS} \cap H) \cup \{n + 1, \dots, n + t_s - t_a\}$,

$$\prod_{i=0}^{d_1} \prod_{j=0}^{d_2} \bar{C}_{i,j}^{u^i v^j} = g^{F(u,v)}h^{R(u,v)} = \prod_{i=0}^{d_1} \prod_{j=0}^{d_2} C_{i,j}^{u^i v^j}$$

The first equation comes from the fact that $\bar{\mathcal{C}}$ is a commitment of F, R , while the second equation comes from the checking phase in the simulated protocol. As a result, $\bar{\mathcal{C}} = \mathcal{C}$. Therefore, if $\bar{\mathcal{C}} \neq \mathcal{C}$ or the extracted polynomials are inconsistent with the accepted openings, $\mathcal{S}$ obtains two different valid openings under the same commitment and breaks its binding property. By Lemma 6, this further yields an algorithm for solving the DLog problem. Hence,

$$|\Pr[\text{View}_2 = 1] - \Pr[\text{View}_3 = 1]| \leq \text{Adv}_{\Pi, \mathcal{B}}^{\text{Binding}}(\kappa) \leq \text{Adv}_{\mathcal{B}'}^{\text{DL}}(\kappa) = \text{negl}(\kappa).$$

Finally, Hybrid_0 is the real execution and Hybrid_3 is the ideal execution with simulator $\mathcal{S}$. By the triangle inequality,

$$\begin{aligned} & |\Pr[\text{REAL}_{\mathcal{A}, \Pi_{VSS}, \mathcal{Z}} = 1] - \Pr[\text{IDEAL}_{\mathcal{S}, \mathcal{F}_{VSS}, \mathcal{Z}} = 1]| \\ & \leq \sum_{i=0}^2 |\Pr[\text{View}_i = 1] - \Pr[\text{View}_{i+1} = 1]| = \text{negl}(\kappa). \end{aligned}$$

Therefore, Π_{VSS} UC-emulates $\mathcal{F}_{VSS}$.

C Building Blocks and Proofs in Π_{LVSS}

In the following, we analyze the security of LVSS of amortized $O(n)$ bits per sharing. This is mainly inspired by [2], we also utilize the KZG commitment scheme. We first introduce the building blocks and further propose the construction of Π_{LVSS} .

C.1 Building Blocks

First, we introduce the main building blocks in this Π_{LVSS} .

Definition 8 (q-Strong Diffie-Hellman (q-SDH) Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g . Given $(g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^q})$ for a random $\alpha \in \mathbb{Z}_p$, the q-SDH problem is to output a pair $(c, g^{1/(\alpha+c)})$ for some $c \in \mathbb{Z}_p$. We say the q-SDH assumption holds if for all polynomial-time adversaries $\mathcal{A}$ and any polynomial $q = q(\kappa)$,*

$$\Pr \left[(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\kappa); \alpha \leftarrow \mathbb{Z}_p; : h = g^{1/(\alpha+c)} \right] \leq \text{negl}(\kappa).$$

Definition 9 (Algebraic Group Model (AGM) Adversary[42]). *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g . An adversary $\mathcal{A}$ is said to be algebraic if for every group element $Z \in \mathbb{G}$ that $\mathcal{A}$ outputs, it also provides an algebraic representation (or witness) of Z with respect to all previously received group elements.*

Formally, let $\mathbf{L} = (L_1, \dots, L_m) \in \mathbb{G}^m$ be the list of group elements given to $\mathcal{A}$ during the game. Whenever $\mathcal{A}$ outputs a new group element $Z \in \mathbb{G}$, it must also output a vector $\mathbf{A} = (a_1, \dots, a_m) \in \mathbb{Z}_p^m$ such that:

$$Z = \prod_{i=1}^m L_i^{a_i}.$$

The representation $\mathbf{A}$ may depend on $\mathcal{A}$'s internal randomness and the inputs received. In the AGM, we assume all adversaries are algebraic. Security proofs in this model hold against any polynomial-time algebraic adversary.

Bivariate Polynomial Commitment [50] proposes the KZG commitment scheme that only costs one field element. Then, **Bingo**[2] (Crypto 2023) extends KZG to support committing bivariate polynomials. We denote the KZG commitment scheme as KZG, and the bivariate KZG commitment in **Bingo** as BKZG. Given groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and a bilinear map $e(\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$, and given a polynomial $f \in \mathbb{F}_{<d}(X)$, KZG has four algorithms as follows:

- **KZG.Setup** $(\kappa, d) \rightarrow \mathbf{crs}$: Let $\tau, a \xleftarrow{\$} \mathbb{F}_{2^\kappa}$, let g, h be random generator of $\mathbb{G}_1, \mathbb{G}_2$, $\hat{g} \leftarrow g^a$. Return $\mathbf{crs} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, h^\tau, \{g^{\tau^i}, \hat{g}^{\tau^i} \mid 0 \leq i \leq d\})$.
- **KZG.Commit** $(\mathbf{crs}, f(x); f'(x) \xleftarrow{\$} \mathbb{F}_{2^\kappa}^d) \rightarrow \mathcal{C}$: Return $\mathcal{C} \leftarrow g^{f(\tau)} \hat{g}^{f'(\tau)}$.
- **KZG.Eval** $(\mathbf{crs}, f(x), f'(x), i) \rightarrow (f(i), f'(i), \pi)$: Let $m \leftarrow f(i)$, $\hat{m} \leftarrow f'(i)$. Let

$$q(x) \leftarrow \frac{f(x) - m}{x - \omega_i}, \hat{q}(x) \leftarrow \frac{f'(x) - \hat{m}}{x - \omega_i}, \pi \leftarrow g^{q(\tau)} \hat{g}^{\hat{q}(\tau)}$$

Return $(m, \hat{m}, \pi)$.

- **KZG.Verify** $(\mathbf{crs}, \mathcal{C}, \omega_i, m, \hat{m}, \pi) \rightarrow 0/1$: If $e(\mathcal{C} g^{-m} \hat{g}^{-\hat{m}}, h) = e(\pi, h^{\tau - \omega_i})$, return 1. Else, return 0.

Besides, we slightly modify the batch proof technique in [50] and propose two new algorithm for batch zero proof as follows.

- **KZG.BZEval**($crs, f(x), f'(x), \mathcal{S}$) $\rightarrow \pi_{\mathcal{S}}$: Let $\mathcal{S} = \{\omega_i\}_{i \in I}$ be a set of points. Compute the zero polynomial $Z_{\mathcal{S}}(x) = \prod_{i \in I} (x - \omega_i)$. Define the quotient polynomials

$$q(x) = \frac{f(x)}{Z_{\mathcal{S}}(x)}, \quad \hat{q}(x) = \frac{f'(x)}{Z_{\mathcal{S}}(x)}.$$

Output the batch proof

$$\pi_{\mathcal{S}} \leftarrow g^{q(\tau)} \cdot \hat{g}^{\hat{q}(\tau)}$$

- **KZG.BZVerify**($crs, \mathcal{C}, \mathcal{S}, \pi_{\mathcal{S}}$) $\rightarrow \{0, 1\}$: Let $Z_{\mathcal{S}}(x) = \prod_{i \in I} (x - \omega_i)$, Compute $Z_{\mathcal{S}}(\tau)$ in the exponent of h , i.e. $h^{Z_{\mathcal{S}}(\tau)}$, obtain $h^{Z_{\mathcal{S}}(\tau)}$. If

$$e(\mathcal{C}, h) = e(\pi_{\mathcal{S}}, h^{Z_{\mathcal{S}}(\tau)}),$$

return 1; otherwise return 0.

Bingo[2] proposed an interesting method to let KZG constrain the degree in the column dimensions. Let *InterpolateExp* denote an algorithm which inputs points $(i, g^{f(i)})$, and outputs an algebraic representation of the coefficients of f . BKZG works as follows:

- **BKZG.Setup**(κ, d_1, d_2) $\rightarrow \mathbf{crs}$: Return $\mathbf{crs} \leftarrow \mathbf{KZG.Setup}(\kappa, d_1)$.
- **BKZG.Commit**($\mathbf{crs}, F(x, y); F'(x, y)$) $\rightarrow \mathcal{C}$: Extract coefficients of F and get row polynomials $f_i(x)$, such that $f_i(x) \cdot y^i = F(x, y)$. Return row commitment vector $\mathcal{C}$ on f , i.e.

$$\sum_{i=0}^{d_2} f_i(x) y^i \leftarrow \mathbf{parse}(F(x, y)), \sum_{i=0}^{d_2} \hat{f}_i(x) y^i \leftarrow \mathbf{parse}(\hat{F}(x, y))$$

$$\mathcal{C} \leftarrow \{\mathcal{C}_i \mid \mathcal{C}_i = g^{f_i(\tau)} \hat{g}^{\hat{f}_i(\tau)}, 0 \leq i \leq d_2\}$$

- **BKZG.PartialEval**($\mathbf{crs}, \mathcal{C}, n$) $\rightarrow \mathbf{A}$: Given row commitment vector $\mathcal{C}$, parse $\mathcal{C} = \{\mathcal{C}_i\}$, return the full commitment $\mathbf{A} \leftarrow \{\prod_{j=0}^{d_1} \mathcal{C}_j^{i^j} \mid 1 \leq i \leq n\}$.
- **BKZG.Eval**($\mathbf{crs}, f(\cdot), f'(\cdot), \omega_i$) $\rightarrow (m, \hat{m}, \pi)$:
 $(m, \hat{m}, \pi) \leftarrow \mathbf{KZG.Eval}(\mathbf{crs}, f(\cdot), f'(\cdot), \omega_i)$, return $(m, \hat{m}, \pi)$.
- **BKZG.GetProof**($\{\omega_j, \pi_j \mid 1 \leq j \leq d_2 + 1\}$) $\rightarrow \{\omega_i, \pi_i \mid 1 \leq i \leq n\}$: Given a vector of $d_2 + 1$ proof π_j (at one column point) that evaluated at different point ω_j , generate the full proof π_i evaluated at points $\omega_i \in [1, n]$ as follows. Return $\{\omega_i, \pi_i \mid 1 \leq i \leq n\}$,
- **BKZG.Verify**($\mathbf{crs}, (i, j), \mathbf{A}, m, \hat{m}, \pi$) $\rightarrow 0/1$:
Return 1 if **KZG.Verify**($\mathbf{crs}, \mathbf{A}(i), \omega_j, m, \hat{m}, \pi$), otherwise returns 0.
- **BKZG.VerifyPoly**($crs, \mathcal{C}, F(x, i), F(i, y), R(x, i), R(i, y), \{\pi'_{j,i}\}_{j \in \mathcal{S}}$) $\rightarrow \{0, 1\}$:
Compute the full commitment $\mathbf{A} \leftarrow \mathbf{BKZG.PartialEval}(crs, \mathcal{C}, n)$. If $A[i] \neq \mathbf{KZG.Commit}(crs, F(x, i), R(x, i))$, return 0. Further verify if

$$\mathbf{KZG.Verify}(crs, \mathbf{A}[j], f(j, i), r(j, i), \pi'_{j,i}) \rightarrow 1 \text{ for } j \in [t_s + 1]$$

, if any instance fails, returns 0. Otherwise, return 1.

In this scheme, the verifier obtains the pre-commitment $\mathcal{C} = \{C_i \mid 0 \leq i \leq d_2\}$ of length $d_2 + 1$, and partially evaluates it to a full commitment $\mathbf{A}$ of size n , s.t. naturally proves the column polynomials are of degree at most d_2 . To handle the column dimension, the BKZG provides a **GetProof** algorithm to derive the proofs on the full column polynomial from $d_2 + 1$ points' proofs $\{\pi_i\}$ on this column dimension. Abraham et al. [2] have proved lemma 9 :

Lemma 9. *If the dlog and q -SDH assumptions hold, then in the AGM model, BKZG scheme satisfies properties: Binding, Correctness, Hiding.*

- **Binding:** *The scheme is **Binding** if for any polynomial adversary $\mathcal{A}$, the game $\mathbf{G}_{\mathcal{A}}^{\text{binding}}(1^\kappa)$ outputs 1 in a negligible probability, i.e., there exists negligible polynomial $v(\cdot)$, $\Pr[\mathbf{G}_{\mathcal{A}}^{\text{binding}}(1^\kappa)] < v(\kappa)$*

$\mathbf{G}_{\mathcal{A}}^{\text{binding}}(1^\kappa)$:

$\mathbf{crs}_1 \leftarrow \text{Setup}(1^\kappa, d_1)$

$(\mathcal{C}, \{\omega_i, m_i, \hat{m}_i, \pi_i\}_{i \in [d_1+1]}) \leftarrow \mathcal{A}(\mathbf{crs}_1)$

$p() \leftarrow \text{Interpolate}(\{\omega_i, m_i\}); \hat{p}() \leftarrow \text{Interpolate}(\{\omega_i, \hat{m}_i\})$

check $\text{Verify}(\mathbf{crs}_1, \mathcal{C}, \omega_i, m_i, \hat{m}_i, \pi_i) = 1$ and $\mathcal{C} \neq \text{Commit}(\mathbf{crs}_1, p, \hat{p})$:

if above equation holds, return 1; otherwise, return 0

- **Correctness:** *The scheme is **Correctness** if the following two conditions hold:*
 - *Let $\mathcal{C} = \text{Commit}(\mathbf{crs}_1, F, R)$, and $A = \text{PartialEval}(\mathcal{C})$, then for any evaluated result with proof, $(f_j(i), r_j(i), \pi_{i,j}) \leftarrow \text{Eval}(i, F(i, j), R(i, j))$, the generated proof and corresponding points must satisfy*

$$\text{Verify}(A, (i, j), F(i, j), R(i, j), \pi_{i,j}) = 1$$

- *Given a set of valid $\{F(i, \omega_j, R(i, \omega_j), \pi_{i,j})\}_{j \in [d]}$, which is generated from $\text{Eval}(F(i, \cdot), R(i, \cdot), \omega_j)$ for $j \in [d]$, if*

$$\{\pi_j\}_{j \in [n]} = \text{GetProof}(\{\omega_j, \pi_{i,j}\})$$

Then it must satisfy, $\text{Verify}(A, (i, j), F(i, j), R(i, j), \pi_{i,j}) = 1$ for $j \in [n]$

- **Hiding:** *For all the polynomial adversary $\mathcal{A}$ interacting with a real protocol, there exists a polynomial simulator $\mathcal{S}$, s.t. the game $\mathbf{G}_{\mathcal{A}}^{\text{hiding}}(1^\kappa)$ outputs 1 in a negligible probability, i.e., there exists negligible polynomial $v(\cdot)$,*

$$\Pr[\mathbf{G}_{\mathcal{A}}^{binding}(1^\kappa) - \frac{1}{2}] < v(\kappa).$$

$$\mathbf{G}_{\mathcal{A}}^{hiding}(1^\kappa) :$$

$\mathbf{crs}_1 \leftarrow \text{Setup}(1^\kappa, d_1; td); b \xleftarrow{\$} \{0, 1\}, r(x) \xleftarrow{\$} \mathbb{F}_{\leq d_1}(X)$
 $f(x) \leftarrow \mathcal{A}(\mathbf{crs}_1); C_1 = \text{KZG.Commit}(\mathbf{crs}_1, f, r), C_2 = \mathcal{S}(\mathbf{crs}_1, td)$
 $\mathcal{O}_{Eval:0}(\omega) : \text{Return: } \text{KZG.Eval}(\mathbf{crs}_1, f, r, \omega); \mathcal{O}_{Open:0}() : \text{Return: } f, r$
 $\mathcal{O}_{Eval:1}(\omega) : \text{Return: } \mathcal{S}(\text{Eval}, \mathbf{crs}_1, td, C_2, f(\omega), r(\omega), \omega)$
 $\mathcal{O}_{Open:1}() : \text{Return: } \mathcal{S}(\text{Open}, \mathbf{crs}_1, td, C_2, f, r)$
 $b' \in \{0, 1\} \leftarrow \mathcal{A}_{\mathcal{O}_{Eval:b}, \mathcal{O}_{Open:b}}(C_b)$
 if $b = b'$ return 1; otherwise, return 0

We define a binding game for the batch zero proof extension. The adversary outputs a commitment $\mathcal{C}$, a set $\mathcal{S}$ of points together with a batch proof $\pi_{\mathcal{S}}$, and a set of $d+1$ points with individual openings $\{\omega_i, m_i, \hat{m}_i, \pi_i\}_{i=1}^{d+1}$, such that $\mathcal{S} \subseteq \{\omega_i\}_{i=1}^{d+1}$. The adversary wins if all individual openings verify, the batch proof verifies, yet there exists some $\omega^* \in \mathcal{S}$ with $(m^*, \hat{m}^*) \neq (0, 0)$. Formally, define the game $\mathbf{G}_{\mathcal{A}}^{bzb}(1^\kappa)$ as follows.

$$\mathbf{G}_{\mathcal{A}}^{bzb}(1^\kappa) :$$

$\mathbf{crs} \leftarrow \text{KZG.Setup}(1^\kappa, d)$
 $(\mathcal{C}, \mathcal{S}, \pi_{\mathcal{S}}, \{\omega_i, m_i, \hat{m}_i, \pi_i\}_{i=1}^{d+1}) \leftarrow \mathcal{A}(\mathbf{crs})$
 Check $\forall i \in [d+1] : \text{KZG.Verify}(\mathbf{crs}, \mathcal{C}, \omega_i, m_i, \hat{m}_i, \pi_i) = 1$
 and $\text{KZG.BZVerify}(\mathbf{crs}, \mathcal{C}, \mathcal{S}, \pi_{\mathcal{S}}) = 1$
 and $\exists \omega^* \in \mathcal{S} : (m^*, \hat{m}^*) \neq (0, 0) :$
 return 1; otherwise return 0

Besides, we show that batch zero proof binding reduces to the interpolation binding of the underlying KZG scheme.

Lemma 10. *If KZG satisfies interpolation binding, then the batch zero proof extension is binding, i.e., for all PPT $\mathcal{A}$,*

$$\Pr[\mathbf{G}_{\mathcal{A}}^{bzb}(1^\kappa) = 1] \leq \Pr[\mathbf{G}_{\mathcal{A}}^{binding}(1^\kappa) = 1] + \frac{d+1}{\mathbb{F}}$$

Proof. Assume there exists an adversary $\mathcal{A}$ that wins $\mathbf{G}_{\mathcal{A}}^{bzb}$ with non-negligible probability $\varepsilon(\kappa)$. We construct an adversary $\mathcal{B}$ against the interpolation binding game $\mathbf{G}_{\mathcal{B}}^{ib}$ as follows.

- $\mathcal{B}$ receives crs from its challenger.
- $\mathcal{B}$ runs $\mathcal{A}$ on input crs and obtains the tuple $(\mathcal{C}, \mathcal{S}, \pi_{\mathcal{S}}, \{\omega_i, m_i, \hat{m}_i, \pi_i\}_{i=1}^{d+1})$.
- $\mathcal{B}$ outputs $(\mathcal{C}, \{\omega_i, m_i, \hat{m}_i, \pi_i\}_{i=1}^{d+1})$ to its challenger.

Now analyze $\mathcal{B}$'s success probability. Since $\mathcal{A}$ wins $\mathbf{G}_{\mathcal{A}}^{bzb}$, we have:

- For all i , $\text{KZG.Verify}(crs, \mathcal{C}, \omega_i, m_i, \hat{m}_i, \pi_i) = 1$;
- $\text{KZG.BZVerify}(crs, \mathcal{C}, \mathcal{S}, \pi_{\mathcal{S}}) = 1$;
- There exists $\omega^* \in \mathcal{S}$ with $(m^*, \hat{m}^*) \neq (0, 0)$.

Let $p(x) = \text{Interpolate}(\{(\omega_i, m_i)\})$ and $\hat{p}(x) = \text{Interpolate}(\{(\omega_i, \hat{m}_i)\})$ be the unique degree- $< d + 1$ polynomials passing through the $d + 1$ points. By the correctness of KZG, the verification of individual openings implies that $\mathcal{C}$ is a commitment to $(p, \hat{p})$; otherwise $\mathcal{A}$ would have broken the standard binding property. More precisely, if $\mathcal{C} \neq \text{KZG.Commit}(crs, p, \hat{p})$, then $\mathcal{B}$ already wins the interpolation binding game directly (since all single openings verify and the commitment differs from the interpolated commitment). Thus assume for contradiction that $\mathcal{C} = \text{KZG.Commit}(crs, p, \hat{p}) = g^{p(\tau)} \hat{g}^{\hat{p}(\tau)}$.

Now consider the batch zero proof verification:

$$e(\mathcal{C}, h) = e(\pi_{\mathcal{S}}, h^{Z_{\mathcal{S}}(\tau)})$$

where $Z_{\mathcal{S}}(x) = \prod_{\omega_i \in \mathcal{S}} (x - \omega_i)$ and $\pi_{\mathcal{S}} = g^{\phi_{\mathcal{A}}(\tau)}$ chosen arbitrarily by $\mathcal{A}$. In other words, $\mathcal{A}$ provides $\phi_{\mathcal{A}}(x)$ that $\phi_{\mathcal{A}}(\tau) \cdot Z_{\mathcal{S}}(\tau) - p(\tau) - a\hat{p}(\tau) = 0$. This happens with probability as most $\frac{d+1}{\mathbb{F}}$.

C.2 The Security Analysis in Π_{LVSS}

The protocol Π_{LVSS} follows the same security guarantee as Π_{VSS} .

Theorem 14. *If the BKZG commitment scheme satisfies binding, hiding and correctness as defined in Lemma 9, the signature scheme is EUF-CMA secure, and the adversary respects the network-agnostic corruption bounds, then protocol Π_{LVSS} realizes the following properties with overwhelming probability:*

- *Termination*
 1. *If the dealer is honest, all honest parties complete the protocol.*
 2. *If any honest party completes the protocol, then every honest party completes the protocol.*
- *Correctness* *The joint view of all honest parties at the end of the sharing phase defines a unique vector of secrets $\mathbf{s}'$ with a t_s -degree polynomial. If the dealer is honest and shares $\mathbf{s}$, then $\mathbf{s}' = \mathbf{s}$.*
- *Secrecy* *If the dealer is honest, the adversary's view throughout the protocol is statistically (resp. computationally) independent of $\mathbf{s}$ provided the commitment scheme is statistically (resp. computationally) hiding.*

Proof. We analyze the three properties separately, distinguishing Case 1 ($|\mathcal{DS}| \leq t_s$) and Case 2 ($|\mathcal{DS}| > t_s$) when necessary, and further considering synchronous vs. asynchronous networks. The proof is a formal reduction to the properties of the KZG commitment scheme (Lemma 9).

Termination

1. *Honest dealer* In Case 1, the dealer starts with $|\mathcal{DS}| \leq t_s$. It sends shares and proofs to all parties. By the network assumptions, all honest parties eventually receive their shares and, after successful verification, return a valid signature. Since at most t_s parties (those in $\mathcal{DS}$) may refuse to sign or respond, the dealer collects at least $n - t_s$ signatures from parties not in $\mathcal{DS}$. The condition of Phase 3 is therefore satisfied, $\Omega = \emptyset$, and the dealer broadcasts $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \pi'_{j,i}, \pi_{\mathcal{DS}}^{\text{bz}}, \pi_{\mathcal{VS}}^{\text{bz}})$ via RBC. Because RBC guarantees agreement and totality, every honest party eventually delivers this tuple. Now consider an honest party $P_j \notin \mathcal{CS}$. Since $\mathcal{CS}$ contains at least $n - t_s$ parties, at least $t_a + 1$ of them are honest (as $n - t_s \geq t_a + 1$). Moreover, the $t_s - t_a$ virtual parties $\{n + 1, \dots, n + t_s - t_a\}$ have fixed zero evaluations on the column polynomials, and all proofs for these zero values are verified via the batch proof $\pi_{\mathcal{VS}}^{\text{bz}}$. Hence P_j can obtain $t_a + 1$ correct column points from the honest parties in $\mathcal{CS}$ together with the $t_s - t_a$ virtual zero points, which gives exactly $t_s + 1$ points to reconstruct its column polynomial $F(j, y)$. After reconstructing the column polynomial, P_j uses the BKZG evaluation proofs to provide valid row points to any $P_k \notin \mathcal{CS}$, and symmetrically receives enough row points to reconstruct its own row polynomial. Thus every honest party completes the protocol.
 In Case 2, $|\mathcal{DS}| > t_s$ the honest dealer knows the network is asynchronous. The dealer collects at least $n - t_a$ signatures (as in the asynchrony) and broadcasts $\mathcal{CS}$ of size at least $n - t_a$. By the same counting argument, honest parties outside $\mathcal{CS}$ can reconstruct their column polynomials using points from honest parties inside $\mathcal{CS}$. Phase 3 continues without restart, and Phase 4 provides the necessary evaluations. Hence termination holds in all network modes.
2. *Some honest party completes* If an honest party completes, it has accepted a valid RBC message $(\mathcal{C}, \mathcal{CS}, \{\sigma_i\}, \pi'_{j,i}, \pi_{\mathcal{DS}}^{\text{bz}}, \pi_{\mathcal{VS}}^{\text{bz}})$. The agreement property of RBC ensures every honest party eventually receives the same message. The signature unforgeability guarantees that the σ_i originate from the corresponding parties. The same reconstruction procedure as above then guarantees that every honest party can compute its output and terminate.

Correctness Define an extractor that, upon the first honest completion of the sharing phase, collects every point (i, j) together with its value $(f_{i,j}, r_{i,j})$ that was accepted by that party, interpolates a bivariate polynomial $F(x, y), R(x, y)$ of degree $(n - t_a - 1, t_s)$ from these points, and outputs $r_k = F(-k, 0)$ for $k \in [\theta]$ with $\theta = n - t_s - t_a$. Correctness requires that (i) for an honest dealer with secret $\mathbf{s}$, $r_k = s_k$; and (ii) any honest party that later completes reconstruction for coordinate k outputs r_k .

We prove correctness through two hybrids. The argument is written for Case 2 ($|\mathcal{DS}| > t_s$); Case 1 is analogous.

Hybrid₀: The real execution of Π_{LVSS} .

Hybrid₁: As Hybrid₀, except that every batch zero proof π_S for a set S is replaced by the corresponding collection of single-point zero proofs $\{\pi_i\}_{i \in S}$. If for any $i \in S$ the derived π_i does not satisfy $\text{KZG.Verify}(\text{crs}, C, i, 0, 0, \pi_i) = 1$, this hybrid aborts.

If an adversary distinguishes Hybrid₀ from Hybrid₁ with non-negligible advantage, we obtain a winning instance for the batch-zero-proof binding game $\mathbf{G}_A^{\text{bzb}}$ (Lemma 10). By that lemma, this advantage is bounded by the advantage against the interpolation-binding game plus a negligible term. Hence the two hybrids are computationally indistinguishable.

Hybrid₂: As Hybrid₁, but if interpolation fails, i.e. two distinct accepted values for the same coordinate, or degree larger than $(n - t_a - 1, t_s + 1)$, this hybrid aborts. Assume an adversary causes an abort in Hybrid₂ with non-negligible probability. Then there exists a commitment C^* and $d + 1$ points with valid single-point proofs that are inconsistent with each other. This directly breaks the interpolation-binding game $\mathbf{G}^{\text{binding}}$ (Lemma 9). Thus Hybrid₂ aborts only with negligible probability.

Secrecy Assume an honest dealer. In the synchronous setting the adversary corrupts at most t_s real parties; in the asynchronous setting it corrupts at most t_a real parties and additionally learns the $t_s - t_a$ virtual parties' fixed zero values (which are publicly known). In both cases the adversary's view comprises:

- the commitment C ,
- at most t_s evaluations of row polynomials $\{F(x, i), R(x, i)\}$ for corrupted parties,
- at most t_s evaluations of column polynomials $\{F(i, y), R(i, y)\}$,
- all batch and individual proofs $\pi_{\mathcal{D}S}^{\text{bz}}, \pi_{\mathcal{V}S}^{\text{bz}}, \pi'_{j,i}$,
- the $t_s - t_a$ virtual zero points (already included in the t_s bound on column evaluations).

Let $\mathcal{S}_{\text{hide}}$ be the simulator guaranteed by the hiding property of BKZG (Lemma 9). Our overall simulator $\mathcal{S}$ works as follows:

1. Receive the secrets $\mathbf{s}$ and choose an arbitrary bivariate polynomial F' of the appropriate degree that agrees with $\mathbf{s}$ on the secret positions and matches the t_s evaluations that the adversary is entitled to see on the corrupted parties and virtual parties. Such an F' exists because the row polynomial degree is $n - t_a - 1$, leaving at least $n - t_s - t_a = \theta$ degrees of freedom.
2. Run $\mathcal{S}_{\text{hide}}$ to generate a simulated commitment C' and simulated proofs for all points the adversary obtains. More precisely, for every required evaluation, query the hiding oracle $\mathcal{O}_{\text{Eval}}$; for the opening, use $\mathcal{O}_{\text{Open}}$ if needed.
3. Follow the protocol with these simulated values, and output whatever the adversary outputs.

If the adversary can distinguish this simulated execution from the real protocol with non-negligible probability, we obtain a distinguisher that breaks the hiding game $\mathbf{G}_A^{\text{hiding}}$. Consequently, the adversary's view reveals no information about the secrets beyond what is implied by the t_s points, and secrecy follows.