Single-Input Functionality against a Dishonest Majority: Practical and Round-Optimal[*]

Zhelei Zhou[†] Bingsheng Zhang[‡] Hong-Sheng Zhou[§] Kui Ren[¶]

February 23, 2024

Abstract

In this work, we focus on Single-Input Functionality (SIF), which can be viewed as a special case of MPC. In a SIF, only one distinguished party called the dealer holds a private input. SIF allows the dealer to perform a computation task with other parties without revealing any additional information about the private input. SIF has diverse applications, including multiple-verifier zero-knowledge, and verifiable relation sharing.

As our main contribution, we propose *the first* 1-round SIF protocol against a dishonest majority in the preprocessing model, which is highly efficient. The only prior work that achieves 1-round online communication assumes an honest majority and is only a feasibility result (Applebaum *et al.*, Crypto 2022). We implement our protocols and conduct extensive experiments to illustrate the practical efficiency of our protocols.

As our side product, we extend the subfield Vector Oblivious Linear Evaluation (sVOLE) into the multiparty setting, and propose a new primitive called multi-verifier sVOLE, which may be of independent interest.

[*]Corresponding authors: Bingsheng Zhang *bingsheng@zju.edu.cn*, and Hong-Sheng Zhou *hszhou@vcu.edu*.
[†]Zhejiang University, and ZJU-Hangzhou Global Scientific and Technological Innovation Center.
[‡]Zhejiang University, and ZJU-Hangzhou Global Scientific and Technological Innovation Center.
[§]Virginia Commonwealth University.
[¶]Zhejiang University, and ZJU-Hangzhou Global Scientific and Technological Innovation Center.

Contents

1 Introduction

MPC vs. SIF. In secure multi-party computation (MPC) [Yao82, GMW87], multiple mutually distrustful players, $(\mathsf{P}_1, \ldots, \mathsf{P}_n)$, are allowed to jointly compute any efficiently computable function f of their private inputs $(x_1, \ldots, x_n)$. Concretely, let circuit $\mathcal{C}$ be the representation of the function f such that $(y_1, \ldots, y_n) \leftarrow \mathcal{C}(x_1, \ldots, x_n)$. After an execution of the MPC protocol for circuit $\mathcal{C}$, each party P_i shall obtain its output y_i. Since its introduction in the early 1980s, secure MPC has been extensively studied and become one of the cornerstones of modern cryptography.

Single-Input Functionality (SIF) is a special case of secure MPC. In SIF, only a distinguished party, called *dealer* D, is allowed to have a private input $\boldsymbol{w}$, while all other parties, called *verifiers* $\mathsf{V}_1, \ldots, \mathsf{V}_n$, have no private inputs. After an execution of the SIF protocol, the dealer D receives no output value while the i-th verifier obtains y_i as its output value. That is, the circuit $\mathcal{C}$ is now specifically defined as follows: $(\emptyset, y_1, \ldots, y_n) \leftarrow \mathcal{C}(\boldsymbol{w}, \emptyset, \ldots, \emptyset)$. For simplicity, we often ignore the empty (input/output) values $\emptyset$'s and write it as $(y_1, \ldots, y_n) \leftarrow \mathcal{C}(\boldsymbol{w})$.

SIF and its applications. As an important cryptographic primitive, SIF was initially studied by Gennaro *et al.* [GIKR02]; this line of research has received lots of attention [AKP20, AKP22b, ZZZR23] very recently. Below, we will give a high-level description of the applications of SIF. More concretely, as already pointed out by Applebaum *et al.* [AKP22b], from SIF, two immediate applications can be obtained: Multi-Verifier Zero-Knowledge (MVZK) and Verifiable Relation Sharing (VRS).

MVZK. In an MVZK protocol, a distinguished party called prover P, who holds a statement-witness pair (x, w), wishes to convince n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$ that $\mathcal{R}(x, w) = 1$ at once for an NP relation $\mathcal{R}$. It is easy to see that SIF implies MVZK directly: let $\mathcal{C}$ be the circuit that evaluates $\mathcal{R}(x, w)$, then the parties can jointly invoke SIF to evaluate $\mathcal{C}$.

MVZK can be used in normal ZK scenarios as long as the identities of the verifiers are known ahead of time. It can also be used in some real life cryptographic systems, e.g., private aggregation system [CB17], as suggested in [YW22]. More concretely, in the private aggregation system like Prio [CB17], a set of servers collect and aggregate the clients' data; and each client need to prove to servers that its data is valid using Secret-shared Non-Interactive Proof (SNIP). Notice that, the SNIP in [CB17] assumes the client (acting as the prover) not to collude with the servers (acting as the verifiers) to ensure soundness; for zero-knowledge property, the SNIP can tolerate all-but-one malicious servers. Hence, if there exists an efficient 1-round MVZK protocol against a dishonest majority (which allows the malicious prover to collude with verifiers), it could be a better alternative technique to SNIP in [CB17].

VRS. In [AKP22b], Applebaum *et al.* introduce a new primitive called VRS, which generalizes MVZK. In a VRS protocol, we consider a distinguished party called dealer D, who holds a secret input s, and n parties called verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$, who have no secret inputs. The dealer D wishes to share the secret s to the verifiers first; for simplicity, we denote by x_i the share received by the i-th verifier. Then the dealer D wishes to prove that the shares satisfying an NP relation $\mathcal{R}$ to the verifiers, i.e., D proves that $\mathcal{R}(x_1, \ldots, x_n, s) = 1$ in a zero-knowledge way. Clearly, SIF also implies VRS: let $(y_1, \ldots, y_n) \leftarrow \mathcal{C}(x_1, \ldots, x_n, s)$ be a circuit such that $y_i = x_i$ for $i \in [n]$ if $\mathcal{R}(x_1, \ldots, x_n, s) = 1$; otherwise, $y_i = \perp$ where $\perp$ is a failure symbol. Then the parties can jointly invoke SIF to evaluate such a circuit $\mathcal{C}$ to realize VRS.

As pointed out by Applebaum *et al.* [AKP22b], VRS has several applications; in particularly, VRS captures a very important primitive, i.e., Verifiable Secret Sharing (VSS) [CGMA85]. VSS ensures a potentially malicious dealer can share a secret such that the verifiers are convinced that the shares they hold are valid; furthermore, in the honest majority setting, the honest verifiers can always recover the secret. In the dishonest majority, the security requirement of VSS is relaxed [NMO$^+$04, DMQO$^+$11]: the honest verifiers may not be able to recover the secret, since the corrupted verifiers can always abort the protocol. As shown in [NMO$^+$04, DMQO$^+$11], VSS can be used to construct MPC protocols in the dishonest majority setting.

SIF with an honest majority. We now introduce several interesting results [AKP22b, YW22, BJO$^+$22] on SIF in the *honest majority* setting. Both the work by Yang and Wang [YW22] and the work by Baum *et al.* [BJO$^+$22] focus on constructing *highly efficient* 2-round protocols in the context of MVZK; the constructions in [YW22] are based on the *random oracle (RO) model*, while the constructions in [BJO$^+$22] are in the *preprocessing model*. On the other hand, the work by Applebaum *et al.* [AKP22b] focuses on the theoretical side and gives a 2-round general SIF protocol in the *plain model*. In particular, as claimed by Applebaum *et al.* [AKP22b], the first round of their protocol is input independent; therefore, their work can be interpreted as a 1-round protocol in the preprocessing model.

SIF against a dishonest majority. Lepinski *et al.* introduce a strengthened version of MVZK protocol against a dishonest majority in the preprocessing model [LMs05]. More precisely, they show how to add fairness among the verifiers, i.e., the malicious verifiers who conclude with the prover learn nothing except the validity of the statement if the honest verifiers accept the proof. However, their work is only a feasibility result and is far from being very practical; furthermore, their work focuses on MVZK and cannot be extended to general SIF directly. As for general SIF, to the best of our knowledge, the only work against a dishonest majority is the work by Zhou *et al.* [ZZZR23]. More concretely, they build 2-round highly efficient SIF protocol in the preprocessing model.

Our main research question. As mentioned above, it is known that 1-round SIF can be constructed in the honest majority setting [AKP22b], assuming a preprocessing model. When turn to the dishonest majority setting, the only prior work [ZZZR23] requires 2-round online communication in the preprocessing model. Therefore, it makes us wonder if it is possible to construct 1-round SIF protocol against a dishonest majority in the preprocessing model? If so, can we build such a protocol with practical efficiency?

We note that constructing such a protocol with practical efficiency is a non-trivial task. The only prior work [AKP22b] that achieves 1-round online communication assumes an honest majority, and it is only a feasibility and is not practical at all. One may suggest using practical MPC protocols against a dishonest majority to realize SIF, for example, the constant-round BMR-style protocol [BMR90]. However, to the best of our knowledge, the BMR-style MPC protocols in the literature require at least 2-round online communication [LSS16, HSS17]. Therefore, naively using MPC protocols to realize SIF is not a solution. Given these difficulties, we ask the following research question:

Is it possible to construct a highly efficient 1-round SIF protocol against a dishonest majority in the preprocessing model?

1.1 Our Contributions

In this work, we will give an affirmative answer to our research question. Our contributions can be summarized as follows.

The first 1-round SIF against a dishonest majority. We present *the first* 1-round highly efficient protocol for SIF against a dishonest majority in the preprocessing model, and our protocol can be proven secure in the Universal Composability (UC) framework [Can01]. Table 1 depicts a comparison between our work and other recent and related work. The full descriptions of our protocol is put in Section 4.

As shown in Table 1, our work is the only one that achieves 1-round online communication in the dishonest majority setting.

Table 1: Comparison of our work and the state-of-the-art work that constructs general SIF.

Ref.	Primitive	#Round[†]	Corruption Threshold	Setup Assumption
[AKP22b]	SIF	1	$t < \frac{n}{2+\epsilon} + 1$[‡]	Preprocessing
[ZZZR23]	SIF	2	$t < n + 1$	Preprocessing
This Work	SIF	1	$t < n + 1$	Preprocessing

[†] Refer to the number of rounds in the online phase.
[‡] Here, ϵ is a small positive constant.

A new form of correlation: mv-sVOLE. We extend the two-party subfield Vector Oblivious Linear Evaluation (sVOLE) [BCG$^+$19a, BCG$^+$19b, WYKW21] correlations into the multi-party setting, which is an essential tool in our SIF construction. More precisely, we propose a new primitive called multiple-verifier sVOLE (mv-sVOLE). In Section 3, we formally define the mv-sVOLE through an ideal functionality, give an efficient construction and prove the security in the UC framework. We note that, there are several works in the literature that also try to extend sVOLE into the multi-party setting, we compare the difference between those works and our mv-sVOLE primitive in Section 3.1.

Implementation and benchmark. To illustrate the efficiency of our protocols, we implement our protocol in C++, conduct comprehensive experiments and report the comparison results with others in Section 5. It turns out that the efficiency of our protocol is very competitive.

Compared to the only prior work that constructs SIF against a dishonest majority [ZZZR23], our protocol outperforms it in both running time and communication. Running on a 1Gbps network with 2ms RTT, for SIF among 3 parties, our protocol takes 37.30ms total running time to evaluate a AES-128 circuit, and our protocol requires 0.25MB total communication. In the same setting, [ZZZR23] requires 317.54ms total running time and 4.35MB total communication. As a result, we make a 8.51$\times$ improvement over [ZZZR23] for total running time and make a 17.40$\times$ improvement for total communication.

Compared to the state-of-the-art SIF (in the context of MVZK) against a honest majority [BJO$^+$22], our protocol is also very competitive. In [BJO$^+$22], two types of 2-round MVZK protocol are proposed: the first protocol tolerates $t < \frac{n}{4}$ malicious verifiers and the second one tolerates $t < \frac{n}{3}$ malicious verifiers. According to [BJO$^+$22], when there are 4 verifiers, the second protocol of [BJO$^+$22] takes 18.25ms total running time to evaluate a AES-128 circuit with 16.24ms online running time, running on a 10Gbps network with roughly 0.6ms RTT. In the same setting, our protocol takes 18.36ms total running time to evaluate the same circuit with only 2.76ms online running time. Our online running time is 5.88$\times$ faster, although our total running time is 0.11ms slower. We also compare the efficiency of our protocol with the first protocol of [BJO$^+$22], and we refer readers to see more details in Section 5.2.

Compared to the state-of-the-art generic MPC protocol against a dishonest majority, our protocol outperforms the state-of-the-art constant-round MPC protocols [WRK17, YWZ20] in both running time and communication. Running on a 10Gbps network with 0.2ms RTT, when there are 3 parties, our protocol takes 14.72ms total running time to evaluate a AES-128 circuit, and our protocol requires 0.25MB total communication. In the same setting, [WRK17] requires 118.39ms total running time and 4.92MB total communication, while [YWZ20] requires 43.85ms total running time and 3.78MB total communication. In this case, our improvement for total running time ranges from 2.98$\times$ to 8.04$\times$, and our improvement for total communication ranges from 15.12$\times$ to 19.68$\times$.

1.2 Our Techniques

Constructing a SIF protocol with only 1-round online phase is a non-trivial task. For better expression, let us take MVZK, a direct application of SIF, as an example. When the online phase is restricted to 1-round, there are no chance for the verifiers to communicate with each other to check whether the prover acts honestly after receiving the prover's messages; as a result, a malicious prover may cause the honest verifiers to output inconsistent results (e.g., some honest verifiers may output acceptance while some may output rejection). Therefore, some "consistency checks" among the verifiers are necessary in the whole protocol construction; otherwise, inconsistency outputs of the verifiers are inevitable. To obtain 1-round online communication, our key observation is that *these consistency checks could be pushed into the preprocessing phase*; in this way, we have the chance to guarantee the security of the protocol. Next, we first talk about the preprocessing phase of our SIF construction; jumping ahead, we propose a new primitive called multiple-verifier sVOLE (mv-sVOLE), which is an essential building block for the preprocessing phase.

Preprocessing phase: using mv-sVOLE as correlations. In our design, we make extensive use of a particular form of correlation, called subfield Vector Oblivious Linear Evaluation (sVOLE) [BCG$^+$19a, BCG$^+$19b, WYKW21]. In the two party setting, sVOLE correlations capture the well-known primitive, i.e., Information-Theoretic Message Authentication Codes (IT-MACs) [BDOZ11, NNOB12]. Let $\mathbb{F}_{p^r}$ be the extension field of a field $\mathbb{F}_p$. In sVOLE, there are two parties involved, i.e., a dealer D and a verifier V, and V holds a global MAC key $\Delta \in \mathbb{F}_{p^r}$. In order to authenticate the vector $\boldsymbol{x} \in \mathbb{F}_p$ held by D to V, we let the dealer D have the MAC tag $\boldsymbol{m} \in \mathbb{F}_{p^r}^{\ell}$ and let the verifier have the local MAC key $\boldsymbol{k} \in \mathbb{F}_{p^r}^{\ell}$ such that $\boldsymbol{m} = \boldsymbol{k} - \Delta \cdot \boldsymbol{x}$. It is easy to see that a malicious D^* who does not know the MAC key, cannot produce another valid $\boldsymbol{m}'$ for $\boldsymbol{x}' \neq \boldsymbol{x}$ except for negligible probability when $|\mathbb{F}_{p^r}|$ is sufficiently large.

In the setting of SIF, we are dealing with $n+1$ parties, i.e., a dealer D and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$, so we have to extend the (two-party) sVOLE correlations into the multi-party setting, which we called multiple-verifier sVOLE (mv-sVOLE). More precisely, we let each verifier V_i privately hold a global MAC key $\Delta^{(i)} \in \mathbb{F}_{p^r}$. For each vector $\boldsymbol{x} \in \mathbb{F}_p^{\ell}$ held by the dealer D, for each $i \in [n]$, we let the dealer D have the MAC tag $\boldsymbol{m}^{(i)} \in \mathbb{F}_{p^r}^{\ell}$ and let the verifier V_i have the local MAC key $\boldsymbol{k}^{(i)} \in \mathbb{F}_{p^r}^{\ell}$ such that $\boldsymbol{k}^{(i)} = \boldsymbol{m}^{(i)} + \Delta^{(i)} \cdot \boldsymbol{x}$. For better expression, we use the notation $[\![\boldsymbol{x}]\!]$ to denote the authenticated vector $\boldsymbol{x}$. In this way, the vector held by the dealer can be authenticated to each verifier. Then, how to generate these mv-sVOLE correlations? One might suggest invoking n instances of sVOLE naively; however, this naive solution is not secure at all: a malicious dealer might use inconsistent values $\boldsymbol{x}' \neq \boldsymbol{x}$ in different instance of sVOLE procedure. To address this security issue, we let the verifiers to pose some lightweight *consistency checks* to detect the malicious behaviors of the dealer. This ensures the verifiers can obtain the correct mv-sVOLE correlations; jumping ahead, it also guarantees the honest verifiers can output the consistent results in the online phase. We defer the details of the solution to generating our mv-sVOLE correlations in Section 3.2.1.

Online phase: checking all the multiplication gates in 1-round. The online phase of our protocol is designed in the "commit-and-prove" paradigm. More concretely, we first let the dealer D commit to his witness $\boldsymbol{w} \in \mathbb{F}_p^m$ using the random mv-sVOLE correlations $[\![\boldsymbol{\mu}]\!]$ generated in the preprocessing phase; that is, D broadcasts $\boldsymbol{\delta} := \boldsymbol{w} - \boldsymbol{\mu} \in \mathbb{F}_p^m$ to verifiers, and all parties compute $[\![\boldsymbol{w}]\!] := [\![\boldsymbol{\mu}]\!] + \boldsymbol{\delta}$. Then we let the dealer "proves" that all the gates of the circuits are processed properly.

Notice that, for the generic MPC protocols against a dishonest majority [BDOZ11, DPSZ12], the randomness, e.g. a random vector $\boldsymbol{\mu}$, generated in the preprocessing phase are often shared among all the parties and no party can know the entire $\boldsymbol{\mu}$. The reason is that: these randomness is used to mask the wire values of the circuit, so the adversary cannot know other parties' private input. However, the situation is different in the setting of SIF [ZZZR23]: in a SIF, only the dealer holds a private input, so it is safe to let the dealer hold the entire $\boldsymbol{\mu}$.

In the following, we will show how the dealer "proves" that all the gates are processed properly in only 1-round. It is easy to see that addition gates can be processed for free. For multiplication gates, we avoid the use of "beaver triples" techniques [Bea92] to check the correctness of multiplication gates; instead, we extend the techniques in [DIO21, YSWW21], which require sVOLE correlations and are designed for two-party setting, into the multi-party setting. More concretely, for the i-th multiplication gate with input wires α, β and output wire γ, we denote by w_α, w_β the input wire values and denote by w_γ the output wire values. We let D broadcast $d_i := w_\alpha \cdot w_\beta - \eta_i \in \mathbb{F}_p$, where η_i is random and $[\![\eta_i]\!]$ is generated in the preprocessing phase, then all parties can compute $[\![w_\gamma]\!] := [\![\eta_i]\!] + d_i$. In this way, D holds $w_a, m_a^{(j)}$ and V_j holds $\Delta^{(j)}, k_a^{(j)}$ such that $k_a^{(j)} = m_a^{(j)} + w_a \cdot \Delta^{(j)}$ for $a \in \{\alpha, \beta, \gamma\}$ and $j \in [n]$. By the following identity:

$$
\begin{aligned}
B_i^{(j)} &:= k_\alpha^{(j)} \cdot k_\beta^{(j)} - k_\gamma^{(j)} \cdot \Delta^{(j)} \\
&= (m_\alpha^{(j)} + w_\alpha \cdot \Delta^{(j)}) \cdot (m_\beta^{(j)} + w_\beta \cdot \Delta^{(j)}) - (m_\gamma^{(j)} + w_\gamma \cdot \Delta^{(j)}) \cdot \Delta^{(j)} \\
&= \underbrace{m_\alpha^{(j)} \cdot m_\beta^{(j)}}_{\text{Denote by } A_{i,0}^{(j)}} + \underbrace{(m_\beta^{(j)} \cdot w_\alpha + m_\alpha^{(j)} \cdot w_\beta - m_\gamma^{(j)})}_{\text{Denote by } A_{i,1}^{(j)}} \cdot \Delta^{(j)} + (w_\alpha \cdot w_\beta - w_\gamma) \cdot (\Delta^{(j)})^2,
\end{aligned}
\tag{1}
$$

we conclude that if D behaves honestly (i.e., $w_\gamma = w_\alpha \cdot w_\beta$), then we have $B_i^{(j)} = A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)}$. It is easy to see that $B_i^{(j)}$ (resp. $A_{i,0}^{(j)}, A_{i,1}^{(j)}$) can be locally computed by D (resp. V_j); therefore, the correctness of the i-th multiplication gate can be checked by letting D send $A_{i,0}^{(j)}, A_{i,1}^{(j)}$ to V_j and letting V_j check if $B_i^{(j)} = A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)}$ holds for each $j \in [n]$. Notice that, the multiplication gates can be checked together; that is why we can achieve 1-round online communication. We defer the details of improving the efficiency of the above checks in Section 4.2.

2 Preliminaries

2.1 Notations

We use $\lambda \in \mathbb{N}$ to denote the security parameter. We say a function $\mathsf{negl} : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $\mathrm{poly}(\cdot)$ and every sufficiently large λ, $\mathsf{negl}(\lambda) < \frac{1}{\mathrm{poly}(\lambda)}$ holds. We say two distribution ensembles $\mathcal{U} = \{\mathcal{U}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{W} = \{\mathcal{W}_\lambda\}_{\lambda \in \mathbb{N}}$ are statistically (resp. computationally) indistinguishable, which we denote by $\mathcal{U} \overset{s}{\approx} \mathcal{W}$ (resp., $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$), if for any unbounded (resp., PPT) distinguisher $\mathcal{D}$ there exists a negligible function negl s.t. $|\Pr[\mathcal{D}(\mathcal{U}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{W}_\lambda) = 1]| = \mathsf{negl}(\lambda)$. We use $x \leftarrow S$ to denote by the event that sampling a uniformly random x from a finite set S. For $n \in \mathbb{N}$, we to $[n]$ to denote by a set $\{1, \ldots, n\}$. For $a, b \in \mathbb{Z}$ with $a \leq b$, we use $[a, b]$ to denote by a set $\{a, \ldots, b\}$. We use bold lower-case letters, e.g. $\boldsymbol{x}$, to denote by the vectors, and we use x_i to denote by the i-th component of vector $\boldsymbol{x}$.

We consider both arithmetic circuit and boolean circuit. Basing on a finite field $\mathbb{F}_p$ with a prime order p, a circuit $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$ consists of a set of input wires $\mathcal{I}_{\mathsf{in}}$ and a set of output wires $\mathcal{I}_{\mathsf{out}}$, where $|\mathcal{I}_{\mathsf{in}}| = m$ and $|\mathcal{I}_{\mathsf{out}}| = n$. In addition to that, the circuit $\mathcal{C}$ also contains a list of gates of the form $(\alpha, \beta, \gamma, T)$, where α, β (resp. γ) are the indices of the input wires (resp. output wire), and $T \in \{\mathsf{Add}, \mathsf{Mult}\}$ is the gate type. If $p = 2$, then $\mathcal{C}$ is a boolean circuit where $\mathsf{Add} = \oplus$ and $\mathsf{Mult} = \wedge$. If $p > 2$, then $\mathcal{C}$ is an arithmetic circuit where $\mathsf{Add}/\mathsf{Mult}$ corresponds to addition/multiplication in $\mathbb{F}_p$.

We use $\mathbb{F}_{p^r}$ to denote by an extension field of a finite field $\mathbb{F}_p$, where $p \geq 2$ is a prime and $r \geq 1$ is an integer. We can write $\mathbb{F}_{p^r} \cong \mathbb{F}_p[X]/f(X)$, where $f(X)$ is a some monic, irreducible polynomial with degree r. It is easy to see that, every $w \in \mathbb{F}_{p^r}$ can be written uniquely as $w = \sum_{i=1}^{r} v_i \cdot X^{i-1}$ with $v_i \in \mathbb{F}_p$ for all $i \in [r]$. Thus, the elements over $\mathbb{F}_{p^r}$ can be regarded as the vectors in $(\mathbb{F}_p)^r$ equivalently.

2.2 Security Model

We design our protocols and prove their security in the Universal Composability (UC) framework by Canetti [Can01].

In the UC framework, we define a protocol Π to be a computer program (or several programs) which is intended to be executed by multiple parties. Every party has a unique identity pair $(\mathsf{pid}, \mathsf{sid})$, where pid refers to the Party ID (PID) and sid refers to the Session ID (SID). Parties running with the same code and the same SID are viewed to be in the same protocol session. The adversarial behaviors are captured by the adversary $\mathcal{A}$, who is able to control the network and corrupt the parties. When a party is corrupted by $\mathcal{A}$, $\mathcal{A}$ obtains its secret input and internal state.

The UC framework is based on the "simulation paradigm" [GMW87], a.k.a., the ideal/real world paradigm. In the ideal world, the inputs of the parties are sent to an ideal functionality $\mathcal{F}$ who will complete the computation task in a trusted manner and send to each party its respective output. The corrupted parties in the ideal world are controlled by an ideal-world adversary $\mathcal{S}$ (a.k.a., the simulator). In the real world, parties communicate with each other to execute the protocol Π, and the corrupted parties are controlled by the real-world adversary $\mathcal{A}$. There is an additional entity called environment $\mathcal{Z}$, which delivers the inputs to parties and receives the outputs generated by those parties. The environment $\mathcal{Z}$ can communicate with the real-world adversary $\mathcal{A}$ (resp. ideal-world adversary $\mathcal{S}$) and corrupt the parties through the adversary in the real (resp. ideal) world. Roughly speaking, the security of a protocol is argued by comparing the ideal world execution to the real world execution. More precisely, for every PPT adversary $\mathcal{A}$ attacking an execution of Π, there is a PPT simulator $\mathcal{S}$ attacking the ideal process that interacts with $\mathcal{F}$ (by corrupting the same set of parties), such that the executions of Π with $\mathcal{A}$ is indistinguishable from that of $\mathcal{F}$ with $\mathcal{S}$ to $\mathcal{Z}$. We denote by $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ (resp. $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$) the output of $\mathcal{Z}$ in the ideal world (resp. real world) execution. Formally, we have the following definition.

Definition 1. *We say a protocol* Π, UC-realizes *the functionality* $\mathcal{F}$, *if for any PPT environment* $\mathcal{Z}$ *and any PPT adversary* $\mathcal{A}$, *there exists a PPT simulator* $\mathcal{S}$ *s.t.* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

We then describe the *modularity* which is appealing in the UC framework: when a protocol calls subroutines, these subroutines can be treated as separate entities and their security can be analyzed separately by way of realizing an ideal functionality. This makes the protocol design and security analysis much simpler. Therefore, we introduce the notion of "hybrid world". A protocol Π is said to be realized "in the $\mathcal{G}$-hybrid world" if Π invokes the ideal functionality $\mathcal{G}$ as a subroutine. Formally, we have the following definition.

Definition 2. *We say a protocol* Π, UC-realizes *the functionality* $\mathcal{F}$ *in the* $\mathcal{G}$*-hybrid world, if for any PPT environment* $\mathcal{Z}$ *and any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *s.t.* $\mathsf{EXEC}^{\mathcal{G}}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

Adversarial model. As in [AKP22b, YW22, BJO+22, ZZZR23], in this paper, we consider a malicious, static and rushing adversary. We also assume that the adversary is allowed to corrupt the dealer and up to t number of verifiers where $t < n$.

Secure communication model. In this work, we consider simultaneous communication. We also assume the parties are connected by pairwise secure channels and a broadcast channel. We remark that, these secure communication channels are also required in the relevant works [AKP22b, YW22, BJO+22, ZZZR23].

2.3 (Programmable) Subfield VOLE

We first introduce the subfield Vector Oblivious Linear Evaluation (sVOLE) [BCG+19a, BCG+19b, YWL+20, WYKW21, YSWW21], which works over an extension field $\mathbb{F}_{p^r}$. More precisely, in sVOLE, the verifier V holds a global MAC key $\Delta \in \mathbb{F}_{p^r}$ which can be used for multiple times. For a vector $\boldsymbol{x} \in \mathbb{F}_p^\ell$ holds by the dealer D, we let the dealer D have the MAC tag $\boldsymbol{m} \in \mathbb{F}_{p^r}^\ell$ and let the verifier have the local MAC key $\boldsymbol{k} \in \mathbb{F}_{p^r}^\ell$ such that $\boldsymbol{m} = \boldsymbol{k} - \Delta \cdot \boldsymbol{x}$. In this way, the vector $\boldsymbol{x}$ is authenticated to the verifier V. Notice that, the dealer D cannot lie about $\boldsymbol{x}$, because the probability of D computing a valid MAC tag $\boldsymbol{m}'$ for a chosen $\boldsymbol{x}' \neq \boldsymbol{x}$ is at most p^{-r}, which would be negligible if p, r are chosen properly.

We note that, most of the recent and popular approaches for generating subfield VOLE are based on Pseudorandom Correlation Generators (PCGs), e.g., [BCGI18, BCG+19a, WYKW21]. Informally speaking, a PCG allows two parties take a pair of short and correlated seeds, then expand them to produce a much larger amount of correlation randomness. However, typically, the sVOLE correlations generated by PCGs are random, meaning that the dealer D cannot chose the authenticated vector $\boldsymbol{x}$. This is troublesome when the dealer D wants to use the same $\boldsymbol{u}$ to run different instances of sVOLE generation procedures with different verifiers. We note that, given a random sVOLE correlation $(\boldsymbol{x}', \boldsymbol{m}', \Delta, \boldsymbol{k}')$ such that $\boldsymbol{m}' = \boldsymbol{k}' - \Delta \cdot \boldsymbol{x}'$, the dealer D can easily convert it to a sVOLE correlation with chosen $\boldsymbol{x}$ by sending $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}'$ to the verifier and setting $\boldsymbol{m} := \boldsymbol{m}'$, the verifier V then sets $\boldsymbol{k} := \boldsymbol{k}' + \boldsymbol{\delta} \cdot \Delta$; in this way, $\boldsymbol{m} = \boldsymbol{k} - \Delta \cdot \boldsymbol{x}$ holds. However, this approach requires $O(\ell)$ communication cost, where ℓ is the vector length; when a large amount of sVOLE correlations are needed (in other words, ℓ is very large), this approach is not efficient enough.

To address the above issue, Rachuri and Scholl propose the *programmable* subfield VOLE in [RS22]; we model this primitive through an ideal functionality $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$, which is depicted in Figure 1. More precisely, the programmability means that the dealer D can choose a seed s and expand it to a vector of ℓ field elements $\boldsymbol{x} := \mathsf{Expand}(s, \ell)$, where $\mathsf{Expand} : S \times \mathbb{Z} \to \mathbb{F}_p^*$ is a deterministic expansion function that takes a seed s from a seed space S and the output length $\ell \in \mathbb{Z}$ as inputs and outputs a ℓ-length vector $\boldsymbol{x} \in \mathbb{F}_p^\ell$. This allows the dealer to use the same authenticated vector $\boldsymbol{x}$ (by choosing the same seed) in different instances of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$. As noted in [RS22], in practice, the expansion function Expand may correspond to some kind of secure Pseudo Random Generators (PRGs)[1]. Rachuri and Scholl also provide a PCG-style protocol that can efficiently realize $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$, and we refer interesting readers to see that in [RS22].

We note that, the sVOLE correlation satisfies an appealing property, i.e., *additive homomorphism*. More precisely, given authenticated vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{F}_p^\ell$ (i.e., for $i \in [n]$: the dealer D holds $\boldsymbol{x}_i$ and $\boldsymbol{m}_{\boldsymbol{x}_i}$ and the verifier V holds Δ and $\boldsymbol{k}_{\boldsymbol{x}_i}$ such that $\boldsymbol{m}_{\boldsymbol{x}_i} = \boldsymbol{k}_{\boldsymbol{x}_i} - \Delta \cdot \boldsymbol{x}_i$) and the public coefficients $c_1, \ldots, c_n \in \mathbb{F}_p$ and $\boldsymbol{c} \in \mathbb{F}_p^\ell$, the dealer D can locally compute $\boldsymbol{y} := \boldsymbol{c} + \sum_{i=1}^n c_i \cdot \boldsymbol{x}_i$ and the corresponding MAC tag $\boldsymbol{m}_{\boldsymbol{y}} := \sum_{i=1}^n c_i \cdot \boldsymbol{m}_{\boldsymbol{x}_i}$ while the verifier V can locally compute the corresponding local MAC key $\boldsymbol{k}_{\boldsymbol{y}} := \sum_{i=1}^n c_i \cdot \boldsymbol{k}_{\boldsymbol{x}_i} + \Delta \cdot \boldsymbol{c}$ such that $\boldsymbol{m}_{\boldsymbol{y}} = \boldsymbol{k}_{\boldsymbol{y}} - \Delta \cdot \boldsymbol{y}$.

2.4 Single-Input Functionalities

In [AKP22b], Applebaum *et al.* formally define Single-Input Functionalities (SIFs); there the majority of players are assumed to be honest, and the SIFs are defined to capture *full security*. Later, in [ZZZR23], Zhou *et al.* consider a relaxed version of SIF, capturing *security with abort*. In this work, we take the definition from [ZZZR23],

[1] Typically, PRGs are referred as randomized algorithms that can generate pseudorandom strings. However, when the seed (which contains the randomness) and the output length are fixed, we can view a PRG as a deterministic algorithm.

Functionality $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$

The functionality interacts with a dealer D, a verifier V and an adversary $\mathcal{S}$. It is parameterized with a finite field $\mathbb{F}_p$ and its extension field $\mathbb{F}_{p^r}$, and a deterministic expansion function $\mathsf{Expand} : S \times \mathbb{Z} \to \mathbb{F}_p^*$.

Initialization: Upon receiving $(\textsc{Init}, \mathsf{sid})$ from D and V, do:

- If V is honest, sample $\Delta \leftarrow \mathbb{F}_{p^r}$; otherwise, receive $\Delta \in \mathbb{F}_{p^r}$ from the adversary $\mathcal{S}$.
- Store Δ and send $(\textsc{Init}, \mathsf{sid}, \Delta)$ to V. Ignore any subsequent $\textsc{Init}$ commands.

Authentication over subfield: Upon receiving $(\textsc{AuthSub}, \mathsf{sid}, \ell, s)$ from D and $(\textsc{AuthSub}, \mathsf{sid}, \ell)$ from V, where $s \in S$, do:

- Compute $\boldsymbol{x} := \mathsf{Expand}(s, \ell) \in \mathbb{F}_p^\ell$.
- If both parties are honest, sample $\boldsymbol{k} \leftarrow \mathbb{F}_{p^r}^\ell$, then compute $\boldsymbol{m} := \boldsymbol{k} - \Delta \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$.
- If both parties are malicious, halt.
- If D^* is malicious and V is honest, receive $\boldsymbol{m} \in \mathbb{F}_{p^r}^\ell$ from $\mathcal{S}$, then compute $\boldsymbol{k} := \boldsymbol{m} + \Delta \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$.
- If D is honest and V^* is malicious, receive $\boldsymbol{k} \in \mathbb{F}_{p^r}^\ell$ from $\mathcal{S}$, then compute $\boldsymbol{m} := \boldsymbol{k} - \Delta \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$.
- Send $(\textsc{Continue}, \mathsf{sid})$ to $\mathcal{S}$. For each honest party $\mathsf{H} \in \{\mathsf{D}, \mathsf{V}\}$, upon receiving an input from $\mathcal{S}$,
  - If it is $(\textsc{Continue}, \mathsf{sid}, \mathsf{H})$, send the respective output to H. More precisely, if H is the dealer D, send $(\textsc{AuthSub}, \mathsf{sid}, \boldsymbol{m})$ to D; if H is the verifier V, send $(\textsc{AuthSub}, \mathsf{sid}, \boldsymbol{k})$ to V.
  - If it is $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$, send $(\textsc{Abort}, \mathsf{sid})$ to H.

Figure 1: Functionality for programmable subfield VOLE, which is adapted from [RS22]

Functionality $\mathcal{F}_{\mathsf{SIF}}$

The functionality interacts with a dealer D, n verifiers V_1,. . . ,V_n and an adversary $\mathcal{S}$. It is parameterized by a circuit $\mathcal{C}$ where $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$. Let $\mathcal{H}$ denote the set of honest parties.

Upon receiving $(\textsc{Input}, \mathsf{sid}, \boldsymbol{w})$ from D and $(\textsc{Input}, \mathsf{sid})$ from V_i for all $i \in [n]$ where $\boldsymbol{w} \in \mathbb{F}_p^m$, do

- Compute $\boldsymbol{y} := \mathcal{C}(\boldsymbol{w})$, and send $(\textsc{Output}, \mathsf{sid}, y_i)$ to V_i^* for each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$.
- Send $(\textsc{Continue}, \mathsf{sid})$ to the adversary $\mathcal{S}$. For each honest verifier $\mathsf{V}_i \in \mathcal{H}$, upon receiving an input from $\mathcal{S}$,
  - If it is $(\textsc{Continue}, \mathsf{sid}, \mathsf{V}_i)$, send $(\textsc{Output}, \mathsf{sid}, y_i)$ to V_i.
  - If it is $(\textsc{Abort}, \mathsf{sid}, \mathsf{V}_i)$, send $(\textsc{Abort}, \mathsf{sid})$ to V_i.

Figure 2: The Functionality $\mathcal{F}_{\mathsf{SIF}}$

which is depicted in Figure 2, since we focus on the dishonest majority setting. As shown in Figure 2, there are a dealer D and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$. The parties hold a circuit $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$ while the dealer D additionally holds a private input $\boldsymbol{w}$ where $|\boldsymbol{w}| = m$. The functionality $\mathcal{F}_{\mathsf{SIF}}$ takes $\boldsymbol{w}$ from the dealer D, then it computes $\boldsymbol{y} := \mathcal{C}(\boldsymbol{w})$ and delivers y_i to V_i for $i \in [n]$, where y_i is the i-th component of $\boldsymbol{y}$.

2.5 Coin-Tossing

Here we introduce the functionality for coin-tossing, and it allows all parties to receive the same uniformly random string. Throughout the paper, we only consider the security with abort; therefore, here we let the functionality capture the security with abort. Formally, we present the functionality for coin-tossing in Figure 3.

3 Subfield VOLE in the Multi-Party Setting

3.1 Security Definition

Here we extend the sVOLE (which is in the two-party setting) into the multi-party setting, and we call this new form of correlated randomness *multiple-verifier subfield VOLE (mv-sVOLE)*. More precisely, in mv-sVOLE, there

Functionality $\mathcal{F}^{p,r}_{\mathsf{COIN}}$

The functionality interacts with a prover P, n verifier $\mathsf{V}_1, \ldots, \mathsf{V}_n$. It is parameterized with a finite field $\mathbb{F}_p$ and its extension field $\mathbb{F}_{p^r}$. Let $\mathcal{H}$ be the set of the honest parties.

Upon receiving $(\textsc{Toss}, \mathsf{sid}, \ell)$ from P and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, do:

- Sample $\boldsymbol{s} \leftarrow \mathbb{F}^{\ell}_{p^r}$ and send $(\textsc{Toss}, \mathsf{sid}, \boldsymbol{s})$ to all corrupted parties.
- Send $(\textsc{Continue}, \mathsf{sid})$ to the adversary $\mathcal{S}$. For each honest party $\mathsf{H} \in \mathcal{H}$, upon receiving an input from $\mathcal{S}$,
  - If it is $(\textsc{Continue}, \mathsf{sid}, \mathsf{H})$, send $(\textsc{Toss}, \mathsf{sid}, \boldsymbol{s})$ to H.
  - If it is $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$, send $(\textsc{Abort}, \mathsf{sid})$ to H.

Figure 3: Functionality for coin-tossing

are a dealer D and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$, and each verifier V_i privately holds a global MAC key $\Delta^{(i)} \in \mathbb{F}_{p^r}$. For each vector $\boldsymbol{x} \in \mathbb{F}^{\ell}_p$ held by the dealer D, for each $i \in [n]$, we let the dealer D have the MAC tag $\boldsymbol{m}^{(i)} \in \mathbb{F}^{\ell}_{p^r}$ and let the verifier V_i have the local MAC key $\boldsymbol{k}^{(i)} \in \mathbb{F}^{\ell}_{p^r}$ such that $\boldsymbol{k}^{(i)} = \boldsymbol{m}^{(i)} + \Delta^{(i)} \cdot \boldsymbol{x}$. In this way, the vector held by the dealer can be authenticated to each verifier. Formally, we present our mv-sVOLE functionality in Figure 4.

Comparison with other works. Notice that, there are several works in the literature that also try to extend sVOLE into the multi-party setting; in the following, we will describe the difference between those works and ours. In [QYYZ22], Qiu *et al.* also extend sVOLE into the multi-verifiers setting, which is similar to ours; however, they do not consider the consistency of the authenticated values. In other words, their malicious dealer can use inconsistent $\boldsymbol{u}$ for different verifiers. As a result, their multi-verifier sVOLE can be implemented by running two-party sVOLE for n times directly, while our mv-sVOLE functionality cannot be realized through this native approach. In [RS22], Rachuri and Scholl extend sVOLE into the multi-party setting in a different way: they let each party play the role of the dealer in turn, and each parties' private values will be authenticated to all other parties. Therefore, there is no distinguished party in their setting, and their multi-party sVOLE primitive is much more complex than our mv-sVOLE. We conjecture that our mv-sVOLE primitive might be used as a basic building block to realize the multi-party sVOLE in [RS22]. In some constant-round MPC protocols that tailored for bool circuits (e.g., [WRK17,YWZ20]), they make use of a primitive called multi-party authenticated bits. Our mv-sVOLE can be viewed as a generalization of multi-party authenticated bits, since multi-party authenticated bits are specifically designed for the case over binary field (i.e., $p = 2$) while our mv-sVOLE can cover both binary field (i.e., $p = 2$) and large filed (i.e., $p > 2$).

3.2 Efficiently Realizing $\mathcal{F}^{p,r}_{\mathsf{mv\text{-}sVOLE}}$

Here we describe how to efficiently realizes $\mathcal{F}^{p,r}_{\mathsf{mv\text{-}sVOLE}}$. One may attempt to let the dealer D use the same value $\boldsymbol{x}$ to run the two-party sVOLE protocol with each verifier V_i individually. However, this naive approach is not secure since a malicious dealer may use inconsistent $\boldsymbol{x}$ when running different instances of the two-party sVOLE protocol with other parties. To prevent this malicious behavior, we let the parties to perform the additional consistency check to ensure that the dealer D uses the same value $\boldsymbol{x}$. Similar approach has been used in prior work by Wang *et al.* [WRK17] in the context of multi-party authenticated bits, which only deals with case where $p = 2$. Here we generalize their approaches to capture both $p = 2$ and $p > 2$.

In the following, we first give a template construction that efficiently realizes $\mathcal{F}^{p,r}_{\mathsf{mv\text{-}sVOLE}}$. Then we will show that by carefully choosing the parameters, our construction remains secure for both $p = 2$ and large $p > 2$.

3.2.1 A Template Construction

Before formally presenting our protocol, we give a high-level description of our construction. Let ρ_1 and ρ_2 be the parameters. In order to authenticate the same ℓ-length vector to all verifiers respectively, we first let all parties set $\ell' := \ell + \rho_1$ and let the dealer D picks a random seed s from the seed space S. We denote by $\boldsymbol{x} := \mathsf{Expand}(s, \ell') \in \mathbb{F}^{\ell'}_p$. We note that, the last ρ_1 components of the vector $\boldsymbol{x}$ are used to prevent a potentially malicious verifier from learning the first ℓ components of $\boldsymbol{x}$. Then for each $i \in [n]$, we let D and V_i invoke an instance of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$, where D sends s to $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$, and $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ returns $\boldsymbol{x}, \boldsymbol{m}^{(i)}$ to D and returns $\boldsymbol{k}^{(i)}$ to V_i such

Functionality $\mathcal{F}^{p,r}_{\text{mv-sVOLE}}$

The functionality interacts with a dealer D, n verifiers $\mathsf{V}_1, \dots, \mathsf{V}_n$ and an adversary $\mathcal{S}$. It is parameterized with a finite field $\mathbb{F}_p$ and its extension field $\mathbb{F}_{p^r}$. Let $\mathcal{H}$ be the set of the honest parties.

Initialization: Upon receiving $(\text{INIT}, \mathsf{sid})$ from D and $\mathsf{V}_1, \dots, \mathsf{V}_n$, do:

- For each $i \in [n]$, if V_i is honest, sample $\Delta^{(i)} \leftarrow \mathbb{F}_{p^r}$; otherwise, receive $\Delta^{(i)} \in \mathbb{F}_{p^r}$ from the adversary $\mathcal{S}$.
- Store $\{\Delta^{(i)}\}_{i \in [n]}$ and send $(\text{INIT}, \mathsf{sid}, \Delta^{(i)})$ to V_i. Ignore any subsequent INIT commands.

Authentications over subfield: Upon receiving $(\text{AUTHSUB}, \mathsf{sid}, \ell)$ from D and $\mathsf{V}_1, \dots, \mathsf{V}_n$, do:

- If all parties are honest, sample $\boldsymbol{x} \leftarrow \mathbb{F}_p^\ell$. For each $i \in [n]$: sample $\boldsymbol{k}^{(i)} \leftarrow \mathbb{F}_{p^r}^\ell$ and compute $\boldsymbol{m}^{(i)} := \boldsymbol{k}^{(i)} - \Delta^{(i)} \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$.
- If all parties are malicious, halt.
- If D^* is malicious and some of the verifiers are honest, receive $\boldsymbol{x} \in \mathbb{F}_p^\ell$ from the adversary $\mathcal{S}$. For each honest verifier $\mathsf{V}_i \in \mathcal{H}$: receive $\boldsymbol{m}^{(i)} \in \mathbb{F}_{p^r}^\ell$ from the adversary $\mathcal{S}$, and compute $\boldsymbol{k}^{(i)} := \boldsymbol{m}^{(i)} + \Delta^{(i)} \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$.
- If D is honest and some of the verifiers are malicious, sample $\boldsymbol{x} \leftarrow \mathbb{F}_p^\ell$. For each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$: receive $\boldsymbol{k}^{(i)} \in \mathbb{F}_{p^r}^\ell$ from the adversary $\mathcal{S}$; for each honest verifier $\mathsf{V}_i \in \mathcal{H}$: sample $\boldsymbol{k}^{(i)} \leftarrow \mathbb{F}_{p^r}^\ell$. Then compute $\boldsymbol{m}^{(i)} := \boldsymbol{k}^{(i)} - \Delta^{(i)} \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^\ell$ for each $i \in [n]$.
- Send $(\text{CONTINUE}, \mathsf{sid})$ to the adversary $\mathcal{S}$. For each honest party $\mathsf{H} \in \mathcal{H}$, upon receiving an input from $\mathcal{S}$,
  - If it is $(\text{CONTINUE}, \mathsf{sid}, \mathsf{H})$, send the respective output to H. More precisely, if H is the dealer D, send $(\text{AUTHSUB}, \mathsf{sid}, \boldsymbol{x}, \{\boldsymbol{m}^{(j)}\}_{j \in [n]})$ to D; if H is i-th verifier V_i, send $(\text{AUTHSUB}, \mathsf{sid}, \boldsymbol{k}^{(i)})$ to V_i.
  - If it is $(\text{ABORT}, \mathsf{sid}, \mathsf{H})$, send $(\text{ABORT}, \mathsf{sid})$ to H.

Figure 4: Functionality for multiple-verifier subfield VOLE

that $\boldsymbol{k}^{(i)} = \boldsymbol{m}^{(i)} + \boldsymbol{x} \cdot \Delta^{(i)}$. Next, we let the parties perform the following consistency checks for ρ_2 times to ensure that if a potentially malicious dealer D^* uses the *inconsistent* seeds in different instances of $\mathcal{F}^{p,r}_{\text{psVOLE}}$ with different verifiers, D^* will be caught with overwhelming probability. We say the dealer uses the inconsistent seeds, if it uses s_1, s_2 such that $\mathsf{Expand}(s_1, \ell') \neq \mathsf{Expand}(s_2, \ell')$. Notice that, if the dealer uses s_1, s_2 such that $s_1 \neq s_2$ but $\mathsf{Expand}(s_1, \ell') = \mathsf{Expand}(s_2, \ell')$, we still say that the dealer uses the consistent seeds.

Our consistency checks work as follows: We let parties sample a uniformly random $\boldsymbol{s} \leftarrow \mathbb{F}_p^{\ell'}$ and let the dealer D broadcast $u := \boldsymbol{s}^\top \cdot \boldsymbol{x}' \in \mathbb{F}_p$. Then for each $i \in [n]$: the dealer D will send the corresponding MAC tag $w^{(i)} := \boldsymbol{s}^\top \cdot \boldsymbol{m}^{(i)} \in \mathbb{F}_{p^r}$ for u to V_i, and V_i will compute the corresponding local MAC key $v^{(i)} := \boldsymbol{s}^\top \cdot \boldsymbol{k}^{(i)} \in \mathbb{F}_{p^r}$ and checks if $v^{(i)} = w^{(i)} + \Delta^{(i)} \cdot u$ holds. Later, we will show that by carefully choosing parameters, if D uses the inconsistent seeds, then D will be caught with overwhelming probability. Finally, if all the consistency checks pass, all the parties simply output the first ℓ objects. That is, D outputs the first ℓ components of $\boldsymbol{x}, \{\boldsymbol{m}^{(j)}\}_{j \in [n]}$ and V_i outputs the first ℓ components of $\boldsymbol{k}^{(i)}$ for each $i \in [n]$. Formally, we present our protocol construction $\Pi^{\rho_1, \rho_2}_{\text{mv-sVOLE}}$ in Figure 5.

3.2.2 Security Analysis

Case I: for $p = 2$. Here, we are dealing with the case where $p = 2$ and $r = \lambda$, where λ is the security parameter; thus, this can support SIF over boolean circuits, which we will describe in the later sections. In this case (where $p = 2$ and $r = \lambda$), the parameters should be set as $\rho_1 := 2\rho$ and $\rho_2 := \rho$ where $\rho = \Theta(\lambda)$. Notice that, for these parameters, our protocol $\Pi^{2\rho,\rho}_{\text{mv-sVOLE}}$ directly yields the multi-party authenticated bits protocol in [WRK17, Figure 5][2]. In the following, we will explain why the parameters are set in this way.

Let us first consider the case where the dealer D^* is corrupted. We need to ensure that if D^* uses inconsistent seeds, for instance, s_1, s_2 such that $\mathsf{Expand}(s_1, \ell') \neq \mathsf{Expand}(s_2, \ell')$, then D^* would be caught in step 3 of Figure 5 with overwhelming probability. We denote by $\boldsymbol{x}_1 := \mathsf{Expand}(s_1, \ell')$ and $\boldsymbol{x}_2 := \mathsf{Expand}(s_2, \ell')$. Since D^* cannot forge a MAC tag except for a negligible probability, the probability of D^* passing the consistency check is the probability that $\boldsymbol{s}^\top \cdot \boldsymbol{x}_1 = \boldsymbol{s}^\top \cdot \boldsymbol{x}_2$, where $\boldsymbol{s}$ is the random vector returned by $\mathcal{F}^{2,1}_{\text{COIN}}$. If we instantiate Expand

[2]In [WRK17, Figure 5], the authors actually set the parameters as $\rho_1 = \rho_2 := 2\rho$. However, according to their proof, we believe that it is their tiny typo error and the parameters should be set as $\rho_1 := 2\rho$ and $\rho_2 := \rho$.

Protocol $\Pi^{\rho_1,\rho_2}_{\mathsf{mv\text{-}sVOLE}}$

Parameter: ρ_1, ρ_2.

Initialization: On input $(\text{INIT}, \mathsf{sid})$, for each $i \in [n]$, D and V_i send $(\text{INIT}, \mathsf{sid})$ to the i-th instance of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$, which returns $\Delta^{(i)} \in \mathbb{F}_{p^r}$ to V_i.

Authentications over subfield: On input $(\text{AUTHSUB}, \mathsf{sid}, \ell)$, D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ do the followings:

1. All parties set $\ell' := \ell + \rho_1$. Then D picks a random seed $s \leftarrow S$, where S is the seed space of the expansion function Expand.
2. For each $i \in [n]$, D sends $(\text{AUTHSUB}, \mathsf{sid}, \ell', s)$ to the i-th instance of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ while V_i sends $(\text{AUTHSUB}, \mathsf{sid}, \ell')$ to the same instance. Then $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ returns $\boldsymbol{x} \in \mathbb{F}_p^{\ell'}, \boldsymbol{m}^{(i)} \in \mathbb{F}_{p^r}^{\ell'}$ to D, where $\boldsymbol{x} := \mathsf{Expand}(s, \ell')$, and returns $\boldsymbol{k}^{(i)}$ to V_i such that $\boldsymbol{k}^{(i)} = \boldsymbol{m}^{(i)} + \boldsymbol{x}' \cdot \Delta^{(i)}$.
3. For each $i \in [\rho_2]$, all parties perform the following consistency check:
   (a) D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\text{TOSS}, \mathsf{sid}, \ell')$ to $\mathcal{F}^{p,1}_{\mathsf{COIN}}$, which returns $\boldsymbol{s}_i \in \mathbb{F}_p^{\ell'}$ to all parties.
   (b) D broadcasts $u_i := \boldsymbol{s}_i^\top \cdot \boldsymbol{x} \in \mathbb{F}_p$ to all verifiers. Then for each $j \in [n]$: D sends $w_i^{(j)} := \boldsymbol{s}_i^\top \cdot \boldsymbol{m}^{(j)} \in \mathbb{F}_{p^r}$ to V_j privately.
   (c) For each $j \in [n]$: V_j computes $v_i^{(j)} := \boldsymbol{s}_i^\top \cdot \boldsymbol{k}^{(j)} \in \mathbb{F}_{p^r}$. Then V_j checks if $v_i^{(j)} = w_i^{(j)} + \Delta^{(j)} \cdot u_i$ holds. If not, V_j simply aborts.
4. D outputs the first ℓ components of $\boldsymbol{x}$, $\{\boldsymbol{m}^{(j)}\}_{j\in[n]}$ and V_i outputs the first ℓ components of $\boldsymbol{k}^{(i)}$ for each $i \in [n]$.

Figure 5: Protocol for multiple-verifier subfield VOLE in the $\{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}\}$-hybrid world

with a secure PRG and we denote by $\mathcal{I}$ the set of indices where $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, then it is easy to see that $\Pr[\boldsymbol{s}^\top \cdot \boldsymbol{x}'_1 = \boldsymbol{s}^\top \cdot \boldsymbol{x}'_2] = \Pr[\oplus_{i\in\mathcal{I}} s_i = 0] = \frac{1}{2} + \epsilon(\lambda)$, where $\epsilon(\lambda)$ is the negligible distance between the pseudorandom random strings that generated by PRGs and the uniformly random strings. In other words, in each consistency check, a cheating D^* can pass the check with probability $\frac{1}{2} + \epsilon(\lambda)$. Thus, we need to let the parties perform $\rho = \Theta(\lambda)$ consistency checks so that a cheating D^* can pass the check with probability $O(2^{-\rho})$. That is why we set $\rho_2 := \rho$ where $\rho = \Theta(\lambda)$.

Then we consider the case where the dealer is honest and some verifiers are corrupted. We need to ensure that the malicious verifiers cannot learn any information about the dealer's output, i.e., the first ℓ components of $\boldsymbol{x}$. In the i-th consistency check, for each random $\boldsymbol{s}_i \in \mathbb{F}_2^{\ell'}$ returned by $\mathcal{F}^{2,1}_{\mathsf{COIN}}$, we denote by $\boldsymbol{a}_i$ the first ℓ components of $\boldsymbol{s}_i$ and denote by $\boldsymbol{b}_i$ the last ρ_1 components of $\boldsymbol{s}_i$. We also denote by $\tilde{\boldsymbol{x}}$ the first ℓ components of $\boldsymbol{x}$ and denote by $\boldsymbol{y}$ the last ρ_1 components of $\boldsymbol{x}$. Then we have the equation $u_i = \boldsymbol{a}_i^\top \cdot \tilde{\boldsymbol{x}} + \boldsymbol{b}_i^\top \cdot \boldsymbol{y}$. Notice that, there are ρ_2 such equations since we need to perform ρ_2 consistency checks. Therefore, we have to prove that $\{\boldsymbol{b}_i\}_{i\in[\rho_2]}$ are linearly independent so that $\boldsymbol{b}_i^\top \cdot \boldsymbol{y}$ can act as "one-time pad" to $\boldsymbol{a}_i^\top \cdot \tilde{\boldsymbol{x}}$; otherwise, the malicious verifiers have the chance to learn the linear combination of $\tilde{\boldsymbol{x}}$, which is the dealer's output. By [WRK17, Lemma A.4], Wang *et al.* prove that the probability of $\{\boldsymbol{b}_i\}_{i\in[\rho_2]}$ being linearly dependent is at most $2^{-(\rho_1-\rho_2)}$. In order to make this probability negligible, we have to set $\rho_1 := 2\rho$ since ρ_2 is already as set as $\rho_2 := \rho$, where $\rho = \Theta(\lambda)$. Formally, we have the following theorem, and we refer interesting readers to see the proof in [WRK17, Theorem A.3].

Theorem 1 (Adapted from [WRK17])**.** *Let λ be the security parameter. Let $\mathbb{F}_{2^\lambda}$ be the extension field. Set $\rho_1 := 2\rho$ and $\rho_2 := \rho$ where $\rho = \Theta(\lambda)$. Let* Expand *be a secure PRG. Then the protocol $\Pi^{2\rho,\rho}_{\mathsf{mv\text{-}sVOLE}}$ depicted in Figure 5 UC-realizes $\mathcal{F}^{2,\lambda}_{\mathsf{mv\text{-}sVOLE}}$ depicted in Figure 4 in the $\{\mathcal{F}^{2,\lambda}_{\mathsf{sVOLE}}, \mathcal{F}^{2,1}_{\mathsf{COIN}}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Case II: for large $p > 2$. When $p = 2$ and $r = \lambda$, we have explained the reason why the parameters should be set as $\rho_1 := 2\rho$ and $\rho_2 := \rho$, where $\rho = \Theta(\lambda)$. It is easy to see that the efficiency of our protocol $\Pi^{\rho_1,\rho_2}_{\mathsf{mv\text{-}sVOLE}}$ would be improved, if the parameters ρ_1, ρ_2 could be set smaller (meanwhile, the selected parameters must be able to ensure the security of our protocol). Jumping ahead, we find that, when $p^{-1} = \mathsf{negl}(\lambda)$ and $r = 1$, the parameters can be set as minimum, i.e., $\rho_1 = \rho_2 := 1$.

Let us first focus on ρ_2, which is the number of consistency checks. Recall that, when $p = 2$, the probability of a malicious D^* passing each consistency check is $\frac{1}{2} + \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a negligible error that caused by PRGs; therefore, $\rho = \Theta(\lambda)$ consistency checks are needed to ensure a malicious D^* cannot pass the consistency check except for a negligible probability. We observe that, if we could lower the probability of a malicious D^* passing each consistency check, then the parameter ρ_2 could be set smaller. By Theorem 3, we can prove that the probability of a malicious D^* passing each consistency check can be reduced to $p^{-1} + \epsilon(\lambda)$. Thus, if p is a sufficiently large prime such that $p^{-1} = \mathsf{negl}(\lambda)$, we only need to perform the consistency check once. In other words, the parameter ρ_2 can be set as $\rho_2 := 1$.

Now let us focus on ρ_1, which is the length of the random mask vector $\boldsymbol{y}$. For the random vector $\boldsymbol{s} \in \mathbb{F}_p^{\ell'}$ returned by $\mathcal{F}_{\mathsf{COIN}}^{p,1}$, we denote by $\boldsymbol{a}$ the first ℓ components of $\boldsymbol{s}$ and denote by $\boldsymbol{b}$ the last ρ_1 components of $\boldsymbol{s}$. We also denote by $\tilde{\boldsymbol{x}}$ the first ℓ components of $\boldsymbol{x}$ and denote by $\boldsymbol{y}$ the last ρ_1 components of $\boldsymbol{x}$. Then we have the equation $u = \boldsymbol{a}^\top \cdot \tilde{\boldsymbol{x}} + \boldsymbol{b}^\top \cdot \boldsymbol{y}$. Unlike the previous case where $p = 2$ and there are ρ such equations, here we only have one such equation. Thus, we observe that $\rho_1 = 1$ is sufficient to mask $\boldsymbol{a}^\top \cdot \tilde{\boldsymbol{x}}$ with $\boldsymbol{b}^\top \cdot \boldsymbol{y}$, since the probability of $\boldsymbol{b}^\top \cdot \boldsymbol{y}$ being zero is negligible. That is why we can set the parameter ρ_1 as $\rho_1 := 1$. Formally, we prove the security through the following theorems.

Theorem 2. *Let λ be the security parameter. Let $\mathbb{F}_{p^r}$ be the extension field where p is a sufficiently large prime order such that $p^{-1} = \mathsf{negl}(\lambda)$ and $r = 1$. Set $\rho_1 := 1$ and $\rho_2 := 1$. Let* Expand *be a secure PRG. Then the protocol $\Pi_{\mathsf{mv\text{-}sVOLE}}^{1,1}$ depicted in Figure 5 UC-realizes the functionality $\mathcal{F}_{\mathsf{mv\text{-}sVOLE}}^{p,1}$ depicted in Figure 4 in the $\{\mathcal{F}_{\mathsf{psVOLE}}^{p,1}, \mathcal{F}_{\mathsf{COIN}}^{p,1}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Proof. The proof can be found in Appendix A.1. □

Theorem 3. *Let $\mathbb{F}_p$ be the field with prime order p. Let $\boldsymbol{s}$ be the column vector over field $\mathbb{F}_p^k$ whose elements are all non-zero, Let $\boldsymbol{t}$ be the column vector that is uniformly sampled from $\mathbb{F}_p^k$. Then we have $\Pr[\boldsymbol{s}^\top \cdot \boldsymbol{t} = 0] = \frac{1}{p}$.*

Proof. The proof can be found in Appendix A.2. □

3.2.3 Instantiating $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$

Notice that, our protocol $\Pi_{\mathsf{mv\text{-}sVOLE}}^{p,r}$ makes block box use of $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$. Here we describe two approaches to instantiate $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$ in the following.

Approach I: PCG-style. Recently, many works (e.g., [BCGI18, BCG⁺19a, WYKW21]) employ Pseudorandom Correlation Generators (PCGs) to generate sVOLE correlations, i.e., they let two parties take a pair of short seeds, then expand them to a large amount of sVOLE correlations. One of the most appealing features of the PCG-style approach is that: it only requires *sublinear* communication cost. However, typically, the correlations generated by PCGs are random; therefore, traditional PCGs cannot be used to realize $\Pi_{\mathsf{mv\text{-}sVOLE}}^{p,r}$ directly.

Basing on the PCG construction in [WYKW21], Rachuri and Scholl give a PCG-style protocol that can efficiently realize $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$ in [RS22]; their protocol can cover both $p = 2$ and $p > 2$. More precisely, the main building block in [WYKW21] is a primitive called *single-input* sVOLE (spsVOLE), where only one component of the authenticated vector $\boldsymbol{x}$ is non-zero while other components are zero. Rachuri and Scholl modify the spsVOLE protocol in [WYKW21] to support programmable inputs, i.e., the authenticated vector $\boldsymbol{x}$ can be expanded from a chosen seed; they also show that the modified spsVOLE can be used to realize $\mathcal{F}_{\mathsf{psVOLE}}^{p,r}$ with essentially the same steps as [WYKW21]. We refer interesting readers to see that in [RS22].

Approach II: IKNP-style. For binary field where $p = 2$, it is known that sVOLE is equivalent to a primitive called Correlated Oblivious Transfer (COT) [ALSZ13]. More precisely, at the end of a COT protocol, the sender obtains ℓ pairs of messages $\{\boldsymbol{m}_0^{(i)}, \boldsymbol{m}_1^{(i)}\}_{i\in[n]} \in \mathbb{F}_2^r$ such that $\boldsymbol{m}_0^{(i)} \oplus \boldsymbol{m}_1^{(i)} = \Delta$, where $\Delta \in \mathbb{F}_2^r$ is chosen by the sender and $\boldsymbol{m}_0^{(i)}, \boldsymbol{m}_1^{(i)}, \Delta$ can be also viewed as elements in the extension field $\mathbb{F}_{2^r}$; meanwhile, the receiver obtains $\{b^{(i)}\}_{i\in[\ell]} \in \mathbb{F}_2$ and $\{\boldsymbol{m}_{b^{(i)}}^{(i)}\}_{i\in[n]} \in \mathbb{F}_2^r$. If we set $\boldsymbol{u} := (b^{(1)}, \ldots, b^{(\ell)}) \in \mathbb{F}_2^\ell$, $\boldsymbol{m} := (\boldsymbol{m}_{b^{(1)}}^{(1)}, \ldots, \boldsymbol{m}_{b^{(\ell)}}^{(\ell)}) \in \mathbb{F}_{2^r}^\ell$ and $\boldsymbol{k} := (\boldsymbol{m}_0^{(1)}, \ldots, \boldsymbol{m}_0^{(\ell)}) \in \mathbb{F}_{2^r}^\ell$, it is easy to see that the sender holds $\Delta, \boldsymbol{k}$ and the receiver holds $\boldsymbol{u}, \boldsymbol{m}$ such that $\boldsymbol{k} = \boldsymbol{m} \oplus \boldsymbol{u} \cdot \Delta$, which is in the form of sVOLE correlations.

One approach for generating a large amount of COTs is to employ the Oblivious Transfer Extension (OTE) techniques by Ishai, Kilian, Nissim and Petrank (hereafter, IKNP) [IKNP03], i.e., given a small number of

OTs (which requires expensive public-key operations), then extend them to a large number of OTs using only symmetric-key operations. Compared to PCG-style approach, IKNP-style approach is computation-efficient, although IKNP-style approach requires much more communication cost. When only a middle number of COTs (e.g., thousands of COTs) are needed or a local area network (e.g., a network with 10Gbps bandwidth) is employed, it turns out that IKNP-style approach may outperform PCG-style approach with respect to total end-to-end time, since in both case the communication cost is no longer the performance bottleneck. For this reason, sometimes, one may prefer to choose the IKNP-style approaches. We note that, the receiver's choice bits $\{b^{(i)}\}_{i\in[\ell]}$ (a.k.a, the authenticated vector $\boldsymbol{u}$ as explained previously) are chosen all by itself; therefore, we can easily instantiate $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ with the maliciously secure IKNP-style OTE protocols [KOS15, Roy22] by letting the receiver sample a random seed s and expand it to $\{b^{(i)}\}_{i\in[\ell]}$ through PRGs.

4 SIF against a Dishonest Majority

4.1 Preprocessing Phase

4.1.1 Functionality for Preprocessing Phase

Here we describe the functionality for preprocessing phase, which is denoted by $\mathcal{F}^{p,r}_{\mathsf{Prep}}$. Our $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ is very similar to our previously defined mv-sVOLE $\mathcal{F}^{p,r}_{\mathsf{mv\text{-}sVOLE}}$, except that $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ additionally allows the dealer D to authenticate his secret values over *extension field* to each verifier respectively. Note that, for authentications over extension field, the dealer D is allowed to use *inconsistent* values to generate correlations. More precisely, given n vectors over the extension field $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(n)} \in \mathbb{F}^d_{p^r}$ held by the dealer D, for each $i \in [n]$, we let D have the MAC tag $\boldsymbol{m}^{(i)} \in \mathbb{F}^d_{p^r}$ and let each V_i have the local MAC key $\boldsymbol{k}^{(i)} \in \mathbb{F}^d_{p^r}$ such that $\boldsymbol{k}^{(i)} = \boldsymbol{m}^{(i)} + \Delta^{(i)} \cdot \boldsymbol{u}^{(i)}$. Formally, we present the functionality for preprocessing phase $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ in Figure 6.

<u>Notation $[\![\cdot]\!]$.</u> For better expression, for a vector $\boldsymbol{u}$ over the subfield $\mathbb{F}^\ell_p$ or the extension field $\mathbb{F}^\ell_{p^r}$, we introduce the following notation $[\![\boldsymbol{u}]\!]$ to denote the values held by parties:

$$[\![\boldsymbol{u}]\!] := \{\{\boldsymbol{u}, \{\boldsymbol{m}^{(i)}\}_{i\in[n]}\}, \{\Delta^{(i)}, \boldsymbol{k}^{(i)}\}_{i\in[n]}\} \ ,$$

where $\boldsymbol{u}, \{\boldsymbol{m}^{(i)}\}_{i\in[n]}$ (resp. $\Delta^{(i)}, \boldsymbol{k}^{(i)}$) are the private information held by the dealer D (resp. the i-th verifier V_i). We use $[\![\boldsymbol{u}]\!]$ as shorthand when there is need to explicitly talk about the MAC tags and MAC keys. We also note that, $[\![\cdot]\!]$ is *additively homomorphic*. More precisely, given $[\![\boldsymbol{u}_1]\!], \ldots, [\![\boldsymbol{u}_n]\!]$ and the public coefficients $c_1, \ldots, c_n$ and $\boldsymbol{c}$, the parties can locally compute $[\![\boldsymbol{y}]\!] := \boldsymbol{c} + \sum_{i=1}^n c_i \cdot [\![\boldsymbol{u}_i]\!]$. This property is inherited from the additive homomorphism of sVOLE which is described in Section 2.3.

4.1.2 Efficiently Realizing $\mathcal{F}^{p,r}_{\mathsf{Prep}}$

Here we show how to construct a protocol that efficiently realizes the functionality for preprocessing phase $\mathcal{F}^{p,r}_{\mathsf{Prep}}$. Since we have already described how to generate mv-sVOLE correlations in Section 3.2.1, here we focus on the authentication for values over extension field.

By the characteristic of extension field $\mathbb{F}_{p^r} \cong \mathbb{F}_p[X]/f(X)$, i.e., for every value over extension field $u \in \mathbb{F}_{p^r}$, it can be written uniquely as $u = \sum_{i=1}^r v_i \cdot X^{i-1}$ where $v_i \in \mathbb{F}_p$ for all $i \in [r]$. Inspired by [YSWW21], we find that we can pack some authenticated values over subfield $\mathbb{F}_p$ into the desired authenticated values over extension field $\mathbb{F}_{p^r}$. More precisely, D and V_i first invoke the programmable sVOLE functionality $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ to generate r copies of random sVOLE correlations, i.e., D obtains $v^{(i)}_j, m^{(i)}_j$ and V_i obtains $\Delta^{(i)}, k^{(i)}_j$ such that $k^{(i)}_j = m^{(i)}_j + u^{(i)}_j \cdot \Delta^{(i)}$ for each $j \in [r]$. Then, the dealer D locally computes $u^{(i)} := \sum_{j=1}^r v^{(i)}_j \cdot X^{j-1}$, $M^{(i)} := \sum_{j=1}^r m^{(i)}_j \cdot X^{j-1}$ and V_i locally computes $K^{(i)} := \sum_{j=1}^r k^{(i)}_j \cdot X^{j-1}$. It is easy to see that $K^{(i)} = M^{(i)} + u^{(i)} \cdot \Delta^{(i)}$ holds.

Formally, we present our protocol Π_{Prep} for preprocessing phase in Figure 7 and prove the security through Theorem 4.

Theorem 4. *Let λ be the security parameter. Let $\mathbb{F}_{p^r}$ be the extension field such that $p^{-r} = \mathsf{negl}(\lambda)$. Let* Expand *be a secure PRG. Then the protocol Π_{Prep} depicted in Figure 7 UC-realizes the functionality $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ depicted in Figure 6 in the $\{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Functionality $\mathcal{F}^{p,r}_{\mathsf{Prep}}$

The functionality interacts with a prover D, n verifiers $\mathsf{V}_1, \dots, \mathsf{V}_n$ and an adversary $\mathcal{S}$. Let $\mathcal{H}$ be the set of the honest parties.

Initialization/Authentications over subfield: The same as in Figure 4.

Authentications over extension field: Upon receiving $(\textsc{AuthExt}, \mathsf{sid}, d)$ from D and $\mathsf{V}_1, \dots, \mathsf{V}_n$, do:

1. If all parties are honest, sample $\boldsymbol{u}^{(1)}, \dots, \boldsymbol{u}^{(n)} \leftarrow \mathbb{F}^d_{p^r}$. For each $i \in [n]$: sample $\boldsymbol{k}^{(i)} \leftarrow \mathbb{F}^d_{p^r}$ and compute $\boldsymbol{m}^{(i)} := \boldsymbol{k}^{(i)} - \Delta^{(i)} \cdot \boldsymbol{u}^{(i)} \in \mathbb{F}^d_{p^r}$.
2. If all parties are malicious, halt.
3. If D^* is malicious and some of the verifiers are honest, for each honest verifier $\mathsf{V}_i \in \mathcal{H}$: receive $\boldsymbol{u}^{(i)}, \boldsymbol{m}^{(i)} \in \mathbb{F}^d_{p^r}$ from the adversary $\mathcal{S}$, and compute $\boldsymbol{k}^{(i)} := \boldsymbol{m}^{(i)} + \Delta^{(i)} \cdot \boldsymbol{u}^{(i)} \in \mathbb{F}^d_{p^r}$.
4. If D is honest and some of the verifiers are malicious, sample $\boldsymbol{u}^{(1)}, \dots, \boldsymbol{u}^{(n)} \leftarrow \mathbb{F}^d_{p^r}$. For each malicious verifier $\mathsf{V}^*_i \notin \mathcal{H}$: receive $\boldsymbol{k}^{(i)} \in \mathbb{F}^d_{p^r}$ from the adversary $\mathcal{S}$; for each honest verifier $\mathsf{V}_i \in \mathcal{H}$: sample $\boldsymbol{k}^{(i)} \leftarrow \mathbb{F}^d_{p^r}$. Then compute $\boldsymbol{m}^{(i)} := \boldsymbol{k}^{(i)} - \Delta^{(i)} \cdot \boldsymbol{u}^{(i)} \in \mathbb{F}^d_{p^r}$ for each $i \in [n]$.
5. Send $(\textsc{Continue}, \mathsf{sid})$ to the adversary $\mathcal{S}$. For each honest party $\mathsf{H} \in \mathcal{H}$, upon receiving an input from $\mathcal{S}$,
   - If it is $(\textsc{Continue}, \mathsf{sid}, \mathsf{H})$, send the respective output to H. More precisely, if H is the dealer D, send $(\textsc{AuthSub}, \mathsf{sid}, \{\boldsymbol{u}^{(j)}, \boldsymbol{m}^{(j)}\}_{j \in [n]})$ to D; if H is i-th verifier V_i, send $(\textsc{AuthSub}, \mathsf{sid}, \boldsymbol{k}^{(i)})$ to V_i.
   - If it is $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$, send $(\textsc{Abort}, \mathsf{sid})$ to H.

Figure 6: Functionality for preprocessing phase

Protocol Π_{Prep}

Initialization/Authentications over subfield: The same as in Figure 5.

Authentications over extension field: On input $(\textsc{AuthExt}, \mathsf{sid}, d)$, D and $\mathsf{V}_1, \dots, \mathsf{V}_n$ do the followings:

1. For each $i \in [d]$ and $h \in [n]$, D and V_h do the followings:
   (a) D picks a random seed $s \leftarrow S$, where S is the seed space of the expansion function Expand. Then D sends $(\textsc{AuthSub}, \mathsf{sid}, r, s)$ to the h-th instance of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ while V_h send $(\textsc{AuthSub}, \mathsf{sid}, r)$ to the same instance. Finally, $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ returns $\{v^{(h)}_{i,j}, m^{(h)}_{i,j}\}_{j \in [r]}$ to D, where $(v^{(h)}_{i,1}, \dots, v^{(h)}_{i,r}) := \mathsf{Expand}(s, r)$, and returns $\{k^{(h)}_{i,j}\}_{j \in [r]}$ to V_h such that $k^{(h)}_{i,j} = m^{(h)}_{i,j} + v^{(h)}_{i,j} \cdot \Delta^{(h)}$ for each $j \in [r]$.
   (b) For each $h \in [n]$: D computes $u^{(h)}_i := \sum_{j=1}^r v^{(h)}_{i,j} \cdot X^{j-1} \in \mathbb{F}_{p^r}$, $M^{(h)}_i := \sum_{j=1}^r m^{(h)}_{i,j} \cdot X^{j-1} \in \mathbb{F}_{p^r}$ and each verifier V_h computes $K^{(h)}_i := \sum_{j=1}^r k^{(h)}_{i,j} \cdot X^{j-1} \in \mathbb{F}_{p^r}$. Note that, $K^{(h)}_i = M^{(h)}_i + u^{(h)}_i \cdot \Delta^{(h)}$ holds.
2. D outputs $\{u^{(j)}_i, M^{(j)}_i\}_{i \in [d], j \in [n]}$ and V_j outputs $\{K^{(j)}_i\}_{i \in [d]}$ for each $j \in [n]$.

Figure 7: Protocol for preprocessing phase in the $\{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}\}$-hybrid world

Protocol Π_{SIF}

Inputs: D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ hold a circuit $\mathcal{C}$ over a field $\mathbb{F}_p$. The circuit $\mathcal{C}$ has m input wires and t multiplication gates. D additionally holds a private input $\boldsymbol{w} \in \mathbb{F}_p^m$.

Preprocessing Phase: The circuit and the private input are unknown.

1. D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{Init}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Prep}}^{p,r}$, which returns $\Delta^{(i)} \in \mathbb{F}_{p^r}$ to V_i for each $i \in [n]$.
2. D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{AuthSub}, \mathsf{sid}, m + t)$ to $\mathcal{F}_{\mathsf{Prep}}^{p,r}$, which returns $[\![\boldsymbol{\mu}]\!]$ and $[\![\boldsymbol{\eta}]\!]$ to the parties.
3. D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{AuthExt}, \mathsf{sid}, 1)$ to $\mathcal{F}_{\mathsf{Prep}}^{p,r}$, which returns $\{u^{(j)}, v^{(j)}\}_{j\in[n]}$ to D and returns $z^{(j)}$ to each verifier V_j such that $z^{(j)} = v^{(j)} + u^{(j)} \cdot \Delta^{(j)}$.

Online Phase: The circuit and the private input are known by the parties.

1. For each $i \in \mathcal{I}_{\mathsf{in}}$: D broadcasts $\delta_i := w_i - \mu_i \in \mathbb{F}_p$. All the parties locally computes $[\![w_i]\!] := [\![\mu_i]\!] + \delta_i$.
2. For each gate $(\alpha, \beta, \gamma, T)$ in a pre-defined topology order:
   (a) If $T = \mathsf{Add}$, all the parties locally compute $[\![w_\gamma]\!] := [\![w_\alpha]\!] + [\![w_\beta]\!]$.
   (b) If $T = \mathsf{Mult}$ and it is the i-th multiplication gate, D broadcasts $d_i := w_\alpha \cdot w_\beta - \eta_i \in \mathbb{F}_p$. All parties compute $[\![w_\gamma]\!] = [\![\eta_i]\!] + d_i$.
3. D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ perform the followings to ensure the multiplication gates are processed correctly:
   (a) For i-th multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, the parties holds $[\![w_\alpha]\!], [\![w_\beta]\!], [\![w_\gamma]\!]$; more precisely, for $a \in \{\alpha, \beta, \gamma\}$ and $j \in [n]$, D holds $w_a, m_a^{(j)}$ while V_j holds $k_a^{(j)}, \Delta^{(j)}$ such that $k_a^{(j)} = m_a^{(j)} + w_a \cdot \Delta^{(j)}$. Then for each $j \in [n]$: D locally computes $A_{i,0}^{(j)} := m_\alpha^{(j)} \cdot m_\beta^{(j)} \in \mathbb{F}_{p^r}$ and $A_{i,1}^{(j)} := m_\beta^{(j)} \cdot w_\alpha + m_\alpha^{(j)} \cdot w_\beta - m_\gamma^{(j)} \in \mathbb{F}_{p^r}$ while V_j locally computes $B_i^{(j)} := k_\alpha^{(j)} \cdot k_\beta^{(j)} - k_\gamma^{(j)} \cdot \Delta^{(j)} \in \mathbb{F}_{p^r}$.
   (b) D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{Toss}, \mathsf{sid}, 1)$ to $\mathcal{F}_{\mathsf{COIN}}^{p,r}$, which returns $\chi \in \mathbb{F}_{p^r}$ to all parties.
   (c) For each $j \in [n]$: D computes $V^{(j)} := \sum_{i=1}^{t} A_{i,0}^{(j)} \cdot \chi^i + v^{(j)} \in \mathbb{F}_{p^r}$ and $U^{(j)} := \sum_{i=1}^{t} A_{i,1}^{(j)} \cdot \chi^i + u^{(j)} \in \mathbb{F}_{p^r}$, and sends $V^{(j)}, U^{(j)}$ to V_j privately.
   (d) For each $j \in [n]$: V_j computes $Z^{(j)} := \sum_{i=1}^{t} B_i^{(j)} \cdot \chi^i + z^{(j)} \in \mathbb{F}_{p^r}$ and checks if $Z^{(j)} = V^{(j)} + U^{(j)} \cdot \Delta^{(j)}$ holds. If not, V_j simply aborts.
4. For each $i \in \mathcal{I}_{\mathsf{out}}$ (without loss of generality, we assume this output wire belongs to V_i), D sends the output wire value y_i and its corresponding MAC tag m_{y_i} to V_i who holds the local MAC key k_{y_i}. Then V_i simply checks if $k_{y_i} = m_{y_i} + y_i \cdot \Delta^{(i)}$ holds. If not, V_i aborts.

Figure 8: Main Protocol for SIF in the $\{\mathcal{F}_{\mathsf{Prep}}^{p,r}, \mathcal{F}_{\mathsf{COIN}}^{p,r}\}$-hybrid world

Proof. The proof can be found in Appendix A.3. □

4.2 Main Protocol

In this subsection, we will provide the main protocol for SIF, which consists of a preprocessing phase and a online phase. Since we have already described how to realize the preprocessing phase in Section 4.1, here we focus on the online phase. In the following, we will give a high-level description of our online phase protocol.

We first let the dealer D commit to his witness $\boldsymbol{w} \in \mathbb{F}_p^m$ using the random mv-sVOLE correlations $[\![\boldsymbol{\mu}]\!]$ generated by $\mathcal{F}_{\mathsf{Prep}}^{p,r}$ in the preprocessing phase; that is, D broadcasts $\boldsymbol{\delta} := \boldsymbol{w} - \boldsymbol{\mu} \in \mathbb{F}_p^m$ to verifiers, and all parties compute $[\![\boldsymbol{w}]\!] := [\![\boldsymbol{\mu}]\!] + \boldsymbol{\delta}$. It is easy to see that the addition gates of the circuit can be processed locally for free, due to the additive homomorphism of $[\![\cdot]\!]$. For multiplication gates, we extend the techniques in [DIO21, YSWW21] which are designed for (s)VOLE correlations to our mv-sVOLE correlations. More precisely, for the i-th multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, given the random $[\![\eta_i]\!]$ generated by $\mathcal{F}_{\mathsf{Prep}}^{p,r}$ in the preprocessing phase, D broadcasts $d_i := w_\alpha \cdot w_\beta - \eta_i \in \mathbb{F}_p$ to verifiers, then all parties compute $[\![w_\gamma]\!] := [\![\eta_i]\!] + d_i$. As a result, D holds $w_a, m_a^{(j)}$ and V_j holds $\Delta^{(j)}, k_a^{(j)}$ such that $k_a^{(j)} = m_a^{(j)} + w_a \cdot \Delta^{(j)}$ for $a \in \{\alpha, \beta, \gamma\}$ and $j \in [n]$. By Equation 1, we conclude that if D behaves honestly (i.e., $w_\gamma = w_\alpha \cdot w_\beta$), then we have $B_i^{(j)} = A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)}$. It is easy to see that $B_i^{(j)}$ (resp. $A_{i,0}^{(j)}, A_{i,1}^{(j)}$) can be locally computed by D (resp. V_j); therefore, the correctness of the i-th multiplication gate can be checked by letting D send $A_{i,0}^{(j)}, A_{i,1}^{(j)}$ to V_j and letting V_j check if $B_i^{(j)} = A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)}$ holds for each $j \in [n]$. We can check t multiplication gates in a batch to reduce the communication cost, using the random linear combination technique [YSWW21]. That is, we let the parties sample a uniformly random $\chi \leftarrow \mathbb{F}_{p^r}$, then we let D send $A_0^{(j)} := \sum_{i=1}^t A_{i,0}^{(j)} \cdot \chi^i$ and $A_1^{(j)} := \sum_{i=1}^t A_{i,1}^{(j)} \cdot \chi^i$ to V_j and let V_j check if $B^{(j)} = A_0^{(j)} + A_1^{(j)} \cdot \Delta^{(j)}$ holds for $j \in [n]$, where $B^{(j)} := \sum_{i=1}^t B_i^{(j)} \cdot \chi^i$. Notice that, $A_0^{(j)}, A_1^{(j)}$ may leak some information about the wire values; thus, we use random $u^{(j)}, v^{(j)}, z^{(j)}$ such that $z^{(j)} = v^{(j)} + u^{(j)} \cdot \Delta^{(j)}$ to mask $A_0^{(j)}, A_1^{(j)}$. Formally, we present our main protocol Π_{SIF} in Figure 7 and prove the security through Theorem 5.

Theorem 5. *Let λ be the security parameter. Let $\mathbb{F}_{p^r}$ be the extension field such that $p^{-r} = \mathsf{negl}(\lambda)$. Let C be the circuit with t multiplication gates. Then the protocol Π_{SIF} depicted in Figure 8 UC-realizes $\mathcal{F}_{\mathsf{SIF}}$ depicted in Figure 2 with statistical security in the $\{\mathcal{F}_{\mathsf{Prep}}^{p,r}, \mathcal{F}_{\mathsf{COIN}}^{p,r}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Proof. The proof can be found in Appendix A.4. □

Towards one-round online communication. During the online phase of our protocol Π_{SIF}, the only interaction between the parties is the coin-tossing procedure. In order to achieve one-round online communication, we can replace the coin-tossing with a Random Oracle (RO) to generate the random element χ. More precisely, given a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{F}_{p^r}$ which is modeled as a RO, we let D compute $\chi := \mathsf{H}(\{\delta_i\}_{i\in[m]}, \{d_i\}_{i\in[t]})$. Since $\{\delta_i\}_{i\in[m]}, \{d_i\}_{i\in[t]}$ are broadcasted by D, verifiers can locally compute χ.

Towards better efficiency. In Step 3 of our online phase protocol, the parties need to compute χ^i for $i \in [t]$. When p is a large prime, the computation of χ^i for $i \in [t]$ can be very expensive. To obtain better computational efficiency, it was suggested in prior work [YSWW21] that we can replace χ^i with independent uniform coefficients χ_i for $i \in [t]$. More concretely, instead of querying RO to obtain χ and then computing χ^i for $i \in [t]$, we can query RO to directly obtain $\chi_1, \ldots, \chi_t$ and use χ_i to replace χ^i for $i \in [t]$. Notice that, this approach will slightly increase the soundness error, but the resulting soundness error is still negligible. We refer interested readers to see [YSWW21] for more details.

5 Implementation and Evaluation

We implement a prototype of our protocols in C++ using EMP toolkip [WMK16], and conduct a benchmark on various circuit evaluations and various network configurations. Our code is available at `https://github.com/ZheleiZhou/SIF-Implmentation`. All experiments are executed on a machine with Intel(R) Core(TM) i7-12700 at 2.10 GHz and 512 GB Memory, running Ubuntu 22.04.3 LTS. Each experiment is run 10 times and the median is taken. For arithmetic circuits, we instantiate the sVOLE protocol over a 61-bit field (i.e., $p = 2^{61} - 1$ and $r = 1$) using the recent protocols [WYKW21]. For boolean circuits, we instantiate the sVOLE over a binary field (i.e., $p = 2$ and $r = 128$) with two choices: (i) for large boolean circuits (e.g., a circuit with 10^7 gates), we employ the PCG-style COT protocol [YWL+20]; (ii) for small or median boolean circuits (e.g., the AES-128 circuits), we emply the IKNP-style COT protocol [KOS15]. Notice that, sVOLE is equivalent to COT over binary field, as we discussed in Section 3.2.3. All implementations achieve at least 40-bit statistical security.

5.1 Our Performance

Here we report the performance of our protocols over both arithmetic circuits and boolean circuits. Table 2 illustrate the running time of our protocol with respect to a randomly generated boolean (resp. arithmetic) circuit with 10^6 AND (resp. multiplication) gates. The number of running time consists of both running time and communication time. As shown in Table 2, our protocol is highly efficient: when there are 3 parties in total and run over a LAN network (RTT:2ms, bandwidth:1Gbps), it takes 13.58s (resp. 4.54s) to evaluate a randomly generated boolean (resp. arithmetic) circuit with 10^7 AND (resp. multiplication) gates, where online time is merely 3.64s (resp. 2.22s). Notice that, when switch to a poorer network (i.e., the WAN network), the running time of our protocol only increases slightly, since the communication cost of our protocol is relatively small. More concretely, when there are 3 parties in total, the total communication is 23.85MB (resp. 164.14MB) for boolean (resp. arithmetic) circuits.

Table 2: The Performance of our protocols. The results are evaluated on a randomly generated boolean (resp. arithmetic) circuit with 10^7 AND (resp. multiplication) gates.

Network[†]	#Party	Boolean Circuit			Arithmetic Circuit		
		Preprocessing Time (s)	Online Time (s)	Total Time (s)	Preprocessing Time (s)	Online Time (s)	Total Time (s)
LAN	3	9.94	3.64	13.58	2.32	2.22	4.54
	5	16.42	6.48	22.90	3.18	3.67	6.85
	7	22.59	8.87	31.73	4.57	5.19	9.76
WAN	3	9.95	3.91	13.86	2.43	3.76	6.19
	5	16.48	6.85	23.33	3.29	6.73	10.02
	7	22.86	9.50	32.36	4.59	9.72	14.31

[†] LAN (RTT: 2ms, bandwidth: 1Gbps); WAN (RTT: 12ms, bandwidth: 100Mbps).

5.2 Comparison with Recent Work

Comparison with SIF against a dishonest majority. To the best of our knowledge, the only work in the literature that constructs SIF against a dishonest majority is [ZZZR23]. Both [ZZZR23] and our work can tolerate up to one malicious dealer and $t < n$ malicious verifiers. We implement their protocols, conduct experiments and report the comparison results in Table 3.

Table 3: Comparison between [ZZZR23] and our protocol. The results are evaluated on a AES-128 circuit.

Network[†]	#Party	Protocol	Running Time (ms)		Comm. (MB)[‡]	
			Prep.	Online	Prep.	Online
LAN	3	[ZZZR23]	123.45	4.74	4.13	0.22
		Ours	26.07	4.59	0.24	0.013
		Improv.	**4.74×**	**1.03×**	**17.21×**	**16.92×**
	5	[ZZZR23]	190.12	8.93	8.26	0.66
		Ours	41.49	7.01	0.47	0.026
		Improv.	**4.58×**	**1.27×**	**17.57×**	**25.38×**
WAN	3	[ZZZR23]	446.06	22.10	4.13	0.22
		Ours	86.48	5.32	0.24	0.013
		Improv.	**5.16×**	**4.15×**	**17.21×**	**16.92×**
	5	[ZZZR23]	735.77	50.92	8.26	0.66
		Ours	114.76	8.75	0.24	0.026
		Improv.	**6.41×**	**5.82×**	**17.57×**	**25.38×**

[†] LAN (RTT: 2ms, bandwidth: 1Gbps); WAN (RTT: 12ms, bandwidth: 100Mbps).
[‡] Refer to the maximum amount of data sent from one party.

As shown in Table 3, our protocol outperforms [ZZZR23] in both running time and communication. In particular, both computation and communication cost of our preprocessing phase are much cheaper than that of [ZZZR23]. The reason is that the preprocessing phase in [ZZZR23] requires the generation of beaver triples which is very heavy, while our protocol does not. Our online phase also requires less running time and communication than [ZZZR23], since our online phase only requires 1 round while [ZZZR23] needs two rounds.

Comparison with SIF against a honest majority. Among three recent and related work in the honest majority [AKP22b, BJO+22, YW22], Feta [BJO+22] is the only one that achieves practical efficiency and implements their protocols; hence, here we compare the efficiency of our protocol with Feta. Notice that, in [BJO+22], the authors construct two MVZK protocols: in the first protocol, up to $t < \frac{n}{3}$ verifiers can be corrupted, while in the second protocol, up to $t < \frac{n}{4}$ verifiers are corrupted. Here we compare with the first one, since it is more efficient[3]. We report the comparison result in Table 4.

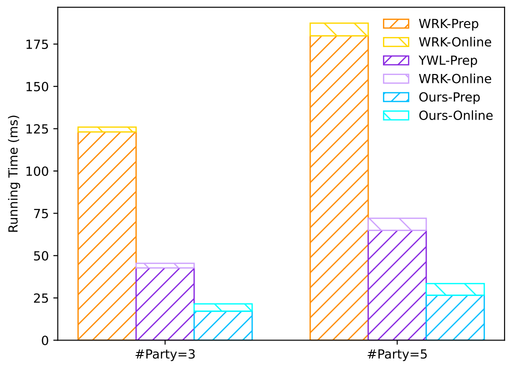
Table 4: Comparison between Feta [BJO+22] and our protocol. The results are evaluated on a AES-128 circuit. The number of verifiers is set as $n = 5$, in this case, Feta only tolerates one corrupted verifier while ours can tolerate 4 corrupted verifiers.

Network†	Protocol	Threshold	Prep. Time(ms)	Online Time(ms)	Proof Size(KB) ‡
LAN	Feta [BJO+22]	$t<\frac{n}{4}+1$	**11.85**	8.04	**2.50**
	This Work	**t<n+1**	43.76	**7.97**	6.69
WAN	Feta [BJO+22]	$t<\frac{n}{4}+1$	**38.47**	23.85	**2.50**
	This Work	**t<n+1**	138.97	**8.38**	6.69

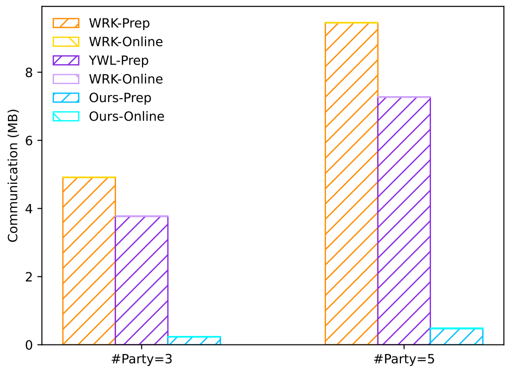
† LAN (RTT: 2ms, bandwidth: 1Gbps); WAN (RTT: 12ms, bandwidth: 100Mbps).
‡ Refer to the amount of data that the dealer sends to each verifier.

As shown in Table 4, the efficiency of our protocol is competitive, even compared with the Feta protocol that assumes an honest majority (more concretely, the corruption threshold of Feta is $t < \frac{n}{4} + 1$). In particular, the online phase of our protocol is 2.85× faster than that of Feta in the WAN network setting.

Comparison with generic MPC against a dishonest majority. To further demonstrate the efficiency of our protocols, we compare our protocol with the state-of-the-art constant-round BMR-style MPC protocols that achieves practical efficiency in the dishonest majority setting, i.e., the WRK protocol by Wang *et al.* [WRK17] and the YWL protocol by Yang *et al.* [YWZ20]. We conduct comprehensive experiments on a network configuration where RTT is as 0.2ms and bandwidth is set as 10Gbps. We plot the results in Figure 9. The numbers of YWL protocol are estimated according to the improvements over WRK protocol that reported in [YWZ20]. As shown in Figure 9, our protocol outperforms the WRK protocol and the YWL protocol in both running time and communication. In particular, our improvement for total running time ranges from 2.11× to 5.60× and our improvement for total communication ranges from 14.86× to 19.68×.



(a) Comparison of Running Time



(b) Comparison of Communication

Figure 9: Comparison among WRK [WRK17], YWL [YWZ20] and our protocol. Results are evaluated on a AES-128 circuit. Experiments are conducted on a 10Gbps network with 0.2ms RTT.

[3]Typically, if a protocol has smaller corruption threshold, the protocol would be more efficient.

6 Related Work

Here we provide a comprehensive literature overview on the related work in both honest majority and dishonest majority settings.

In the honest majority setting. The study of SIF was initialized by Gennaro *et al.* [GIKR02]. More precisely, they proposed a 2-round SIF protocol in the plain model with corruption threshold of $t < \frac{n}{6} + 1$, their protocol achieves perfect security. Applebaum *et al.* improved the corruption threshold to $t < \frac{n}{3} + 1$ while keep the same round complexity, at the cost of degrading the perfect security to computational security [AKP20]. Later, the same authors further improved the corruption threshold to $t < \frac{n}{2+\epsilon} + 1$, where ϵ is a small positive constant [AKP22b]. In [AKP22b], Applebaum *et al.* also pointed out that SIF has lots of applications, e.g., round-optimal MPC [ACGJ18, AKP22a], private data aggregation [CB17], anonymous messaging system [CBM15], and verifiable secret sharing [CGMA85].

As mentioned before, MVZK is a direct application of SIF, and the notion of MVZK can be traced back to the work by Burmester and Desmedt [BD91]. Abe *et al.* proposed a 2-round MVZK protocol for circuit satisfiability with corruption threshold of $t < \frac{n}{3} + 1$ [ACF02]; the corruption threshold of their protocol can be improved to $t < \frac{n}{2} + 1$ at the cost of increasing round complexity. The ZK protocols by Groth and Ostrovsky [GO07, GO14] can be transformed into the 2-round MVZK protocols with corruption threshold of $t < \frac{n}{2} + 1$. These works [ACF02, GO07, GO14] require heavy public-key operations and are not concretely efficient. Very recently, there are two papers [YW22, BJO+22] studying 2-round MVZK protocols in the honest majority setting, and they avoided the use of public-key operations. Yang and Wang [YW22] proposed 2-round MVZK protocols in the RO model with corruption threshold of $t < \frac{n}{2} + 1$. Baum *et al.* [BJO+22] employed a stronger assumption (i.e., the preprocessing model) to construct two types of the 2-round MVZK protocols: the first protocol tolerates $\frac{n}{3}$ malicious verifiers and the second protocol tolerates $\frac{n}{4}$ malicious verifiers.

Distributed Zero-Knowledge (dZK) is a related cryptographic primitive, and it was proposed by Boneh *et al.* [BBC+19]. In dZK, there is a distinguished prover holding $(x, w) \in \mathcal{R}$ and the statement x is shared among the verifiers; the prover wishes to convince the verifiers that x is correct in zero-knowledge even if the verifiers do not know the entire x. The main difference between dZK and MVZK is that: in dZK, the statement x is shared among the verifiers and no verifier knows the entire statement x; in contrast, in MVZK, each verifier knows the entire statement x. Boneh *et al.* [BBC+19] gave a 2-round dZK construction in the RO model with corruption threshold of $t < \frac{n}{2} + 1$. Very recently, Hazay *et al.* strengthened the formalization of [BBC+19] by adding *strong completeness* [HVW23], which prevents the malicious verifiers from framing the honest prover, i.e., causing the proof of a correct claim to fail. They constructed their dZK in the corruption threshold of $t < \frac{n-2}{6} + 1$, by assuming an ideal coin-flipping.

In the dishonest majority setting. In [LMs05], Lepinski *et al.* propose a notion called fair ZK, which can be viewed as a strengthened version of MVZK against a dishonest majority. Fair ZK ensures that the malicious verifiers can learn nothing beyond the validity of the statement if the honest verifiers accept the proof. However, their work cannot be extended to general SIF directly and is far from being practical. To the best of our knowledge, the only prior work that focuses on constructing practical SIF protocols against a dishonest majority is the work by Zhou *et al.* [ZZZR23]. More precisely, they build highly efficient 2-round SIF protocol in the preprocessing model.

In terms of dZK, Boneh *et al.* give a 2-round dZK construction in the RO model [BBC+19]; however, they assume the adversary can corrupt the prover *or* up to $t < n$ verifiers. In other words, they do not allow the malicous prover to collude with the malicious verifiers.

As for VSS, another direct application of SIF, the authors of [NMO+04, DMQO+11] proposed the 1-round VSS protocols against a dishonest majority in the commodity based model. They also showed how to construct MPC against a dishonest majority using VSS.

References

[ACF02] Masayuki Abe, Ronald Cramer, and Serge Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 206–223. Springer, Heidelberg, December 2002.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424. Springer, Heidelberg, August 2018.

[AKP20] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 562–594. Springer, Heidelberg, November 2020.

[AKP22a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Round-optimal honest-majority MPC in minicrypt and with everlasting security - (extended abstract). In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 103–120. Springer, Heidelberg, November 2022.

[AKP22b] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading NIZKs with honest majority - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 33–56. Springer, Heidelberg, August 2022.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 535–548. ACM Press, November 2013.

[BBC$^+$19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, August 2019.

[BCG$^+$19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Heidelberg, August 2019.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[BD91] Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 81–95. Springer, Heidelberg, April 1991.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, Heidelberg, May 2011.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992.

[BJO$^+$22] Carsten Baum, Robin Jadoul, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. Feta: Efficient threshold designated-verifier zero-knowledge proofs. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 293–306. ACM Press, November 2022.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 259–282, 2017.

[CBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE Computer Society Press, May 2015.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th FOCS*, pages 383–395. IEEE Computer Society Press, October 1985.

[DIO21] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. *ITC 2021*, 2021.

[DMQO$^+$11] Rafael Dowsley, Jorn MULLER-QUADE, Akira Otsuka, Goichiro Hanaoka, Hideki Imai, and Anderson CA Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(2):725–734, 2011.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

[FIS14] Stephen H Friedberg, Arnold J Insel, and Lawrence E Spence. *Linear algebra*, volume 4. Pearson Essex, 2014.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 178–193. Springer, Heidelberg, August 2002.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341. Springer, Heidelberg, August 2007.

[GO14] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *Journal of Cryptology*, 27(3):506–543, July 2014.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 598–628. Springer, Heidelberg, December 2017.

[HVW23] Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. Your reputation's safe with me: Framing-free distributed zero-knowledge proofs. In Guy N. Rothblum and Hoeteck Wee, editors, *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part I*, volume 14369 of *Lecture Notes in Computer Science*, pages 34–64. Springer, 2023. https://eprint.iacr.org/2022/1523.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

[LMs05] Matt Lepinski, Silvio Micali, and abhi shelat. Fair-zero knowledge. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 245–263. Springer, Heidelberg, February 2005.

[LSS16] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More efficient constant-round multi-party computation from BMR and SHE. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 554–581. Springer, Heidelberg, October / November 2016.

[NMO$^+$04] Anderson C. A. Nascimento, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, and Hideki Imai. Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity based model. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04*, volume 3089 of *LNCS*, pages 355–368. Springer, Heidelberg, June 2004.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

[QYYZ22] Zhi Qiu, Kang Yang, Yu Yu, and Lijing Zhou. Maliciously secure multi-party PSI with lower bandwidth and faster computation. In Cristina Alcaraz, Liqun Chen, Shujun Li, and Pierangela Samarati, editors, *ICICS 22*, volume 13407 of *LNCS*, pages 69–88. Springer, Heidelberg, September 2022.

[Roy22] Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Heidelberg, August 2022.

[RS22] Rahul Rachuri and Peter Scholl. Le mans: Dynamic and fluid MPC for dishonest majority. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 719–749. Springer, Heidelberg, August 2022.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit, 2016.

[WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 39–56. ACM Press, October / November 2017.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

[YSWW21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

[YW22] Kang Yang and Xiao Wang. Non-interactive zero-knowledge proofs to multiple verifiers. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 517–546. Springer, Heidelberg, December 2022.

[YWL+20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

[YWZ20] Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1627–1646. ACM Press, November 2020.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.

[ZZZR23] Zhelei Zhou, Bingsheng Zhang, Hong-Sheng Zhou, and Kui Ren. Practical constructions for single input functionality against a dishonest majority. Cryptology ePrint Archive, Paper 2023/1367, 2023. https://eprint.iacr.org/2023/1367.

A Security Proofs

A.1 Proof of Theorem 2

Theorem 2. *Let λ be the security parameter.Let $\mathbb{F}_{p^r}$ be the extension field where p is a sufficiently large prime order such that $p^{-1} = \mathsf{negl}(\lambda)$ and $r = 1$. Set $\rho_1 := 1$ and $\rho_2 := 1$. Let* Expand *be a secure PRG. Then the protocol $\Pi^{1,1}_{\textsf{mv-sVOLE}}$ depicted in Figure 5 UC-realizes the functionality $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$ depicted in Figure 4 in the $\{\mathcal{F}^{p,1}_{\textsf{psVOLE}}, \mathcal{F}^{p,1}_{\textsf{COIN}}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Proof. We prove the security of the protocol $\Pi^{1,1}_{\textsf{mv-sVOLE}}$ by showing it is a UC-secure realization of $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$. We will first describe the workflow of the simulator $\mathcal{S}$ in the ideal-world with $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$, the dummy dealer $\tilde{\mathsf{D}}$ and the dummy verifiers $\tilde{\mathsf{V}}_1, \ldots, \tilde{\mathsf{V}}_n$, then give a proof that for any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$, the simulation in the ideal-world $\mathsf{EXEC}_{\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable from the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,1}_{\textsf{psVOLE}},\mathcal{F}^{p,1}_{\textsf{COIN}}}_{\Pi_{\textsf{mv-sVOLE}},\mathcal{A},\mathcal{Z}}$.

When the dealer is honest. In this case, up to $n-1$ verifiers are corrupted and the malicious verifiers attempt to learn the information about the dealer's output, i.e., the first ℓ components of the vector $\boldsymbol{x} := \mathsf{Expand}(s, \ell')$. We denote by $\mathcal{H}$ the set of honest parties. We describe the simulation strategy of $\mathcal{S}$ in the following:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,1}_{\textsf{psVOLE}}, \mathcal{F}^{p,1}_{\textsf{COIN}}$ honestly for the adversary $\mathcal{A}$. Therefore, $\mathcal{S}$ knows $\Delta^{(i)}$ and $\boldsymbol{k}^{(i)}$ for each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$. Then $\mathcal{S}$ sends $\Delta^{(i)}$ and $\boldsymbol{k}^{(i)}$ to $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$ on behalf of each malicious dummy verifier $\tilde{\mathsf{V}}_i$.

2. $\mathcal{S}$ picks a uniformly random $\tilde{\boldsymbol{x}} \leftarrow \mathbb{F}_p^{\ell'}$. Then for each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$, $\mathcal{S}$ computes $\tilde{\boldsymbol{m}}^{(i)} := \boldsymbol{k}^{(i)} - \tilde{\boldsymbol{x}} \cdot \Delta^{(i)} \in \mathbb{F}_p^{\ell'}$; notice that, $\mathcal{S}$ is able to compute $\tilde{\boldsymbol{m}}^{(i)}$ since $\mathcal{S}$ knows $\Delta^{(i)}$ and $\boldsymbol{k}^{(i)}$.

3. $\mathcal{S}$ executes the step 3 in Figure 5 honestly on behalf of the honest parties.

4. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply stops simulating this party and returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$ when $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$ sends $(\textsc{Continue}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,1}_{\textsf{psVOLE}},\mathcal{F}^{p,1}_{\textsf{COIN}}}_{\Pi_{\textsf{mv-sVOLE}},\mathcal{A},\mathcal{Z}}$.

- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}^{p,1}_{\textsf{psVOLE}}, \mathcal{F}^{p,1}_{\textsf{COIN}}$ honestly for the adversary $\mathcal{A}$, sends the corrupted dummy parties' respective output to $\mathcal{F}^{p,1}_{\textsf{mv-sVOLE}}$, picks $\tilde{\boldsymbol{x}} \leftarrow \mathbb{F}_p^{\ell'}$ and uses $\tilde{\boldsymbol{x}}$ to complete the protocol execution with $\mathcal{A}$.

  Lemma 1. *Let $\mathbb{F}_{p^r}$ be the extension field where p is a sufficiently large prime order such that $p^{-1} = \mathsf{negl}(\lambda)$ and $r = 1$. Let* Expand *be a secure PRG. Then hybrid* Hyb_1 *is computationally indistinguishable from hybrid* Hyb_0.

Proof. Here we will show that any adversary $\mathcal{A}$ cannot know any information about $\boldsymbol{x}$ in the real-world execution (i.e., Hyb_0), so that $\mathcal{A}$ will not distinguish Hyb_0 from Hyb_1 when $\mathcal{S}$ uses a randomly selected $\tilde{\boldsymbol{x}}$ in Hyb_1.

We denote by $\tilde{\boldsymbol{s}}$ the value returned by $\mathcal{F}^{p,1}_{\mathsf{COIN}}$ in hybrid Hyb_1 and we set $\tilde{u} := \sum_{i=1}^{\ell'} \tilde{s}_i \cdot \tilde{x}'_i$. We denote by y (resp., $\tilde{y}$) the last component of $\boldsymbol{x}$ (resp. $\tilde{\boldsymbol{x}}$). With the above notations, it is easy to observe that $u = (\sum_{i=1}^{\ell} s_i \cdot x_i) + s_{\ell+1} \cdot y$ and $\tilde{u} = (\sum_{i=1}^{\ell} \tilde{s}_i \cdot \tilde{x}_i) + \tilde{s}_{\ell+1} \cdot \tilde{y}$. Notice that, since Expand is a secure PRG, y and $\tilde{y}$ are computationally indistinguishable and the probability of y (or $\tilde{y}$) being non-zero is $1 - p^{-r}$, which is overwhelming. We also note that, since $\boldsymbol{s}$ are sampled by $\mathcal{F}^{p,1}_{\mathsf{COIN}}$ and $\tilde{\boldsymbol{s}}$ are uniformly sampled, the probability of $s_{\ell+1}$ (or $\tilde{s_{\ell+1}}$) being non-zero is also $1-p^{-1}$. Therefore, the probability of $s_{\ell+1} \cdot y$ (or $\tilde{s}_{\ell+1} \cdot \tilde{y}$) being non-zero is $(1-p^{-1})$, which is overwhelming. When $s_{\ell+1} \cdot y$ (resp. $\tilde{s}_{\ell+1} \cdot \tilde{y}$) is non-zero, it serves as a "one-time pad" to $\sum_{i=1}^{\ell} s_i \cdot x_i$ (resp. $\sum_{i=1}^{\ell} \tilde{s}_i \cdot \tilde{x}_i$). In other words, if $s_{\ell+1} \cdot y$ and $\tilde{s}_{\ell+1} \cdot \tilde{y}$ are non-zero, then u and $\tilde{u}$ are perfectly indistinguishable. In conclusion, hybrid Hyb_1 is computationally indistinguishable from hybird Hyb_0. □

- Hybrid Hyb_2: Same as Hyb_1, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply stops simulating this party and returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ when $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ sends $(\textsc{Continue}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_2 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}, \mathcal{S}, \mathcal{Z}}$. In conclusion, when the dealer is honest, $\mathsf{EXEC}_{\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}, \mathcal{S}, \mathcal{Z}}$ is computationally indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^{p,1}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{mv\text{-}sVOLE}}, \mathcal{A}, \mathcal{Z}}$.

When the dealer is malicious. In this case, the malicious dealer D^* may use inconsistent s (therefore, this will result in inconsistent $\boldsymbol{x}$) when running different instances of $\mathcal{F}^{p,1}_{\mathsf{psVOLE}}$ with different honest verifiers. We need to prove that if the malicious dealer D^* cheats, D^* would be caught with overwhelming probability. The simulation strategy of the simulator $\mathcal{S}$ is straightforward: $\mathcal{S}$ simply acts as honest verifiers and follows the protocol honestly. For completeness, we describe the simulation strategy of $\mathcal{S}$ in the following:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,1}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}$ honestly for the adversary $\mathcal{A}$. In this way, $\mathcal{S}$ receives s from malicious D^*, and $\mathcal{S}$ knows whether D^* uses the inconsistent s.
2. $\mathcal{S}$ completes the protocol execution honestly on behalf of the honest verifiers.
3. If the malicious D^* uses the inconsistent s and passes the consistency check, $\mathcal{S}$ would abort.
4. If the malicious D^* uses the consistent s and passes the consistency check (i.e., D^* sends the correct $\boldsymbol{m}^{(i)}$ to each honest verifier V_i), $\mathcal{S}$ sends s and $\{\boldsymbol{m}^{(i)}\}_{i \text{ s.t. } \mathsf{V}_i \in \mathcal{H}}$ to $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ on behalf of the malicious dummy $\tilde{\mathsf{D}}^*$.
5. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply stops simulating this party and returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ when $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ sends $(\textsc{Continue}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,1}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{mv\text{-}sVOLE}}, \mathcal{A}, \mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}^{p,1}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}$ honestly for the adversary $\mathcal{A}$, follows the protocol honestly on behalf of the honest verifiers, and $\mathcal{S}$ aborts if the malicious D^* uses the inconsistent s and D^* passes the consistency check.

  Lemma 2. *Let $\mathbb{F}_{p^r}$ be the extension field where p is a sufficiently large prime order such that $p^{-1} = \mathsf{negl}(\lambda)$ and $r = 1$. Let* Expand *be a secure PRG. Then hybrid* Hyb_1 *is computationally indistinguishable from hybird* Hyb_0.

  Proof. It is easy to see that the adversary $\mathcal{A}$ would distinguish Hyb_0 from Hyb_1 if $\mathcal{S}$ aborts. Here we will show that the probability of a cheating D^* passing the consistency check is negligible, so the probability of $\mathcal{S}$ aborting is also negligible.

  If D^* uses inconsistent s, for instance, s_1, s_2 such that $s_1 \neq s_2$. We denote by $\boldsymbol{x}_1 := \mathsf{Expand}(s_1, \ell')$ and $\boldsymbol{x}_2 := \mathsf{Expand}(s_2, \ell')$. We also denote by $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2$ the uniformly sampled vectors from $\mathbb{F}_p^{\ell'}$. Since Expand is a secure PRG, $\boldsymbol{x}_1$ (resp. $\boldsymbol{x}_2$) is computationally indistinguishable from $\tilde{\boldsymbol{x}}_1$ (resp. $\tilde{\boldsymbol{x}}_2$). By Theorem 3, we know that $\Pr[\boldsymbol{s}^\top \cdot \tilde{\boldsymbol{x}}_1 = \boldsymbol{s}^\top \cdot \tilde{\boldsymbol{x}}_2] = \Pr[\boldsymbol{s}^\top \cdot (\tilde{\boldsymbol{x}}_1 - \tilde{\boldsymbol{x}}_2)] = p^{-1} = 0$, which is negligible. By the union bound, we conclude that the probability of $\boldsymbol{s}^\top \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$ is also negligible. In other words, unless D^* is able to forge a MAC tag which happens with probability p^{-1}, the cheating D^* can pass the consistency check with negligible probability. In conclusion, hybrid Hyb_1 is computationally indistinguishable from hybird Hyb_0. $\square$

- Hybrid Hyb_2: Same as Hyb_1, except that if the malicious D^* uses the consistent s and passes the consistency check (i.e., D^* sends the correct $\boldsymbol{m}^{(i)}$ to each honest verifier V_i), $\mathcal{S}$ sends s and $\{\boldsymbol{m}^{(i)}\}_{i \text{ s.t. } \mathsf{V}_i \in \mathcal{H}}$ to $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ on behalf of the malicious dummy $\tilde{\mathsf{D}}^*$. Perfect indistinguishability is trivial.
- Hybrid Hyb_3: Same as Hyb_2, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply stops simulating this party and returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ when $\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}}$ sends $(\textsc{Continue}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_3 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is malicious, $\mathsf{EXEC}_{\mathcal{F}^{p,1}_{\mathsf{mv\text{-}sVOLE}},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^{p,1}_{\mathsf{psVOLE}},\mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{mv\text{-}sVOLE}},\mathcal{A},\mathcal{Z}}$. $\square$

A.2 Proof of Theorem 3

Theorem 3. *Let $\mathbb{F}_p$ be the field with prime order p. Let $\boldsymbol{s}$ be the column vector over field $\mathbb{F}_p^k$ whose elements are all non-zero, Let $\boldsymbol{t}$ be the column vector that is uniformly sampled from $\mathbb{F}_p^k$. Then we have $\Pr[\boldsymbol{s}^\top \cdot \boldsymbol{t} = 0] = \frac{1}{p}$.*

Proof. We will use some knowledge of linear algebra to prove this lemma. Let $\boldsymbol{s}^\top \cdot \boldsymbol{x} = 0$ be a linear equation, that is, we let $\boldsymbol{s}^\top$ be the coefficients and let $\boldsymbol{x}$ be the variables. The null space of $\boldsymbol{s}^\top$ is defined as a set $\{\boldsymbol{y} \in \mathbb{F}_p^k : \boldsymbol{s}^\top \cdot \boldsymbol{y} = 0\}$ and we denote by $\mathcal{N}(\boldsymbol{s}^\top)$ the null space of $\boldsymbol{s}^\top$ for better expression. The dimension of $\mathcal{N}(\boldsymbol{s}^\top)$ is also called the nullity of $\boldsymbol{s}^\top$. It is easy to see that the rank of $\boldsymbol{s}^\top$ is 1. Due to the rank-nullity theorem [FIS14] which states that given any coefficient matrix A, the rank of A plus the nullity of A is equal to the total number of columns in A, we can easily conclude that the nullity of $\boldsymbol{s}^\top$ is $k-1$. In other words, given the first $k-1$ components of $\boldsymbol{x}$, there exists a unique x_k such that $x_k = -\sum_{i=1}^{k-1} s_k^{-1} \cdot s_i \cdot x_i$.

Now let us look back to equation that we want to prove. Notice that, $\boldsymbol{s}^\top \cdot \boldsymbol{t} = 0$ if and only if $\boldsymbol{t} \in \mathcal{N}(\boldsymbol{s}^\top)$. Therefore, we have

$$\begin{aligned}
&\Pr[\boldsymbol{s}^\top \cdot \boldsymbol{t} = 0] = \Pr[\boldsymbol{t} \in \mathcal{N}(\boldsymbol{s}^\top)] \\
&= \sum_{(v_1, v_2, \ldots, v_{k-1}) \in \mathbb{F}_p^{k-1}} \Pr[t_k = -\sum_{i=1}^{k-1} s_k^{-1} \cdot s_i \cdot v_i \mid t_1 = v_1, t_2 = v_2, \ldots, t_{k-1} = v_{k-1}] \cdot \\
&\quad \Pr[t_1 = v_1, t_2 = v_2, \ldots, t_{k-1} = v_{k-1}] \\
&= p^{k-1} \cdot (\frac{1}{p} \cdot \frac{1}{p^{k-1}}) = \frac{1}{p} \quad .
\end{aligned}$$

The penultimate equation holds because $\boldsymbol{t}$ is uniformly sampled from $\mathbb{F}_p^k$. This completes the proof. $\square$

A.3 Proof of Theorem 4

Theorem 4. *Let λ be the security parameter. Let $\mathbb{F}_{p^r}$ be the extension field such that $p^{-r} = \mathsf{negl}(\lambda)$. Let* Expand *be a secure PRG. Then the protocol Π_{Prep} depicted in Figure 7 UC-realizes the functionality $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ depicted in Figure 6 in the $\{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Proof. The security of **Initialization** and **Authentications over subfield** is trivial; thus, here we only focus on **Authentications over extension field**. We will first describe the workflow of $\mathcal{S}$, then give an proof to show that the ideal-world $\mathsf{EXEC}_{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{S},\mathcal{Z}}$ is perfectly indistinguishable from the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{psVOLE}},\mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}}$; notice that, the perfect security only holds for the **Authentications over extension field** part.

When the dealer is honest. In this case, up to $n-1$ verifiers are corrupted and the malicious verifiers want to learn some information about the dealer's input. We denote by $\mathcal{H}$ the set of honest parties. We describe the simulation strategy of $\mathcal{S}$ in the following:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ honestly for $\mathcal{A}$.
2. For each $i \in [d]$ and for each malicious verifier $\mathsf{V}^*_h \notin \mathcal{H}$:
   (a) $\mathcal{S}$ receives $\boldsymbol{k}^{(h)}_i \in \mathbb{F}^r_{p^r}$ from V^*_h by emulating $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$.
   (b) $\mathcal{S}$ computes $K^{(h)}_i := \sum_{j=1}^r k^{(h)}_{i,j} \cdot X^{j-1} \in \mathbb{F}_{p^r}$.
3. $\mathcal{S}$ sends $\boldsymbol{K}^{(h)} := (K^{(h)}_1, \ldots, K^{(h)}_d) \in \mathbb{F}^d_{p^r}$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ on behalf of each malicious dummy $\tilde{\mathsf{V}}_h$.
4. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\text{ABORT}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ when $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ sends $(\text{CONTINUE}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{Prep}}, \mathcal{A}, \mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ honestly for $\mathcal{A}$, receives $\{\boldsymbol{k}^{(h)}_i\}_{i \in [d]}$ from each malicious verifier $\mathsf{V}^*_h \notin \mathcal{H}$, and sends $\boldsymbol{K}^{(h)} := (K^{(h)}_1, \ldots, K^{(h)}_d) \in \mathbb{F}^d_{p^r}$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ on behalf of each malicious dummy $\tilde{\mathsf{V}}_i$, where $K^{(h)}_i := \sum_{j=1}^r k^{(h)}_{i,j} \cdot X^{j-1}$ for all $i \in [d]$. Perfect indistinguishability holds, because there is no communication among the parties and $\mathcal{A}$ cannot learn anything about the dealer's input due to the security of $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$.
- Hybrid Hyb_2: Same as Hyb_1, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ returns $(\text{ABORT}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ when $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ sends $(\text{CONTINUE}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_2 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{p,r}_{\mathsf{Prep}}, \mathcal{S}, \mathcal{Z}}$. In conclusion, when the dealer is honest, $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{Prep}}, \mathcal{A}, \mathcal{Z}} \equiv \mathsf{EXEC}_{\mathcal{F}^{p,r}_{\mathsf{Prep}}, \mathcal{S}, \mathcal{Z}}$ holds.

When the dealer is malicious. In this case, some of the verifiers are honest. Denote by $\mathcal{H}$ the set of honest parties. We describe the simulation strategy in the following:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ honestly for $\mathcal{A}$.
2. For $i \in [d]$:
   (a) $\mathcal{S}$ receives $s^{(h)}_i \in S$ and $\{\boldsymbol{m}^{(h)}_i\}_{h \text{ s.t. } \mathsf{V}_h \in \mathcal{H}}$ from the malicious D^* by emulating $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$.
   (b) $\mathcal{S}$ computes $(v^{(h)}_{i,1}, \ldots, v^{(h)}_{i,r}) := \mathsf{Expand}(s^{(h)}_i, r)$, $u^{(h)}_i := \sum_{j=1}^r v^{(h)}_{i,j} \cdot X^{j-1}$ and $M^{(h)}_i := \sum_{j=1}^r m_{i,j} \cdot X^{j-1}$ for each honest verifier $\mathsf{V}_h \in \mathcal{H}$.
3. $\mathcal{S}$ sets $\boldsymbol{u}^{(h)} := (u^{(h)}_1, \ldots, u^{(h)}_d)$, $\boldsymbol{M}^{(h)} := (M^{(h)}_1, \ldots, M^{(h)}_d)$ for each honest $\mathsf{V}_h \in \mathcal{H}$.
4. $\mathcal{S}$ sends $\{\boldsymbol{u}^{(h)}, \boldsymbol{M}^{(h)}\}_{h \text{ s.t. } \mathsf{V}_h \in \mathcal{H}}$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ on behalf of the corrupted dummy $\tilde{\mathsf{D}}^*$.
5. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\text{ABORT}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ when $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ sends $(\text{CONTINUE}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{psVOLE}}, \mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{Prep}}, \mathcal{A}, \mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{psVOLE}}$ honestly for $\mathcal{A}$, receives $\{s^{(h)}_i, \boldsymbol{m}^{(h)}_i\}_{h \text{ s.t. } \mathsf{V}_h \in \mathcal{H}}$ from the malicious D^*, computes $(v^{(h)}_{i,1}, \ldots, v^{(h)}_{i,r}) := \mathsf{Expand}(s^{(h)}_i, r)$, $u^{(h)}_i := \sum_{j=1}^r v^{(h)}_{i,j} \cdot X^{j-1}$ and $M^{(h)}_i := \sum_{j=1}^r m_{i,j} \cdot X^{j-1}$ for each honest verifier $\mathsf{V}_h \in \mathcal{H}$. Then $\mathcal{S}$ sets $\boldsymbol{u}^{(h)} := (u^{(h)}_1, \ldots, u^{(h)}_d)$, $\boldsymbol{M}^{(h)} := (M^{(h)}_1, \ldots, M^{(h)}_d)$ for each honest $\mathsf{V}_h \in \mathcal{H}$, and sends $\{\boldsymbol{u}^{(h)}, \boldsymbol{M}^{(h)}\}_{h \text{ s.t. } \mathsf{V}_h \in \mathcal{H}}$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ on behalf of the corrupted dummy $\tilde{\mathsf{D}}^*$. Perfect indistinguishability is trivial.

- Hybrid Hyb_2: Same as Hyb_1, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ when $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ sends $(\textsc{Continue}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_2 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is malicious, $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{sVOLE}},\mathcal{F}^{p,1}_{\mathsf{COIN}}}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}} \equiv \mathsf{EXEC}_{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{S},\mathcal{Z}}$ holds. □

A.4 Proof of Theorem 5

Theorem 5. *Let λ be the security parameter. Let $\mathbb{F}_{p^r}$ be the extension field such that $p^{-r} = \mathsf{negl}(\lambda)$. Let $\mathcal{C}$ be the circuit with t multiplication gates. Then the protocol Π_{SIF} depicted in Figure 8 UC-realizes the functionality $\mathcal{F}_{\mathsf{SIF}}$ depicted in Figure 2 with statistical security in the $\{\mathcal{F}^{p,r}_{\mathsf{Prep}}, \mathcal{F}^{p,r}_{\mathsf{COIN}}\}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $n-1$ verifiers.*

Proof. Similar to the proof of Theorem 2, we will first describe the workflow of $\mathcal{S}$, then give an proof to show that the ideal-world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{F}^{p,r}_{\mathsf{COIN}}}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}}$.

When the dealer is honest. In this case, up to $n-1$ verifiers are malicious, and we need to ensure that the malicious verifiers cannot learn the dealer's input $\boldsymbol{w}$. We prove this by constructing a simulator $\mathcal{S}$ who does not hold $\boldsymbol{w}$, but is able to generate the "fake proof" that would make a honest verifier accept. We denote by $\mathcal{H}$ the set of honest parties. We describe the simulation strategy of $\mathcal{S}$ as follows:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ for the adversary $\mathcal{A}$.
2. $\mathcal{S}$ picks a random $\tilde{\boldsymbol{w}} \leftarrow \mathbb{F}_p^m$ and uses $\tilde{\boldsymbol{w}}$ to execute the step 1-3 in the online phase of Π_{SIF} honestly on behalf of the honest dealer D.
3. In the final step of the online phase, for each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$, if $\mathcal{S}$ receives $(\textsc{Output}, \mathsf{sid}, y_i)$ from $\mathcal{F}_{\mathsf{SIF}}$, then $\mathcal{S}$ sends $y_i, m_{y_i} := k_{y_i} - y_i \cdot \Delta^{(i)}$ to V_i^*; notice that, $\mathcal{S}$ knows k_{y_i} and $\Delta^{(i)}$ by emulating $\mathcal{F}^{p,r}_{\mathsf{Prep}}$.
4. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}_{\mathsf{SIF}}$ when $\mathcal{F}_{\mathsf{SIF}}$ sends $(\textsc{Continue}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{F}^{p,r}_{\mathsf{COIN}}}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{Prep}}$ honestly for $\mathcal{A}$, picks a random $\tilde{\boldsymbol{w}} \leftarrow \mathbb{F}_p^m$ and uses $\tilde{\boldsymbol{w}}$ to execute the step 1-3 in the online phase of Π_{SIF} honestly on behalf of the honest dealer D, and in the final step of the online phase, for each malicious verifier $\mathsf{V}_i^* \notin \mathcal{H}$, if $\mathcal{S}$ receives $(\textsc{Output}, \mathsf{sid}, y_i)$ from $\mathcal{F}_{\mathsf{SIF}}$, then $\mathcal{S}$ sends $y_i, m_{y_i} := k_{y_i} - y_i \cdot \Delta^{(i)}$ to V_i^*.

  Lemma 3. *Hybrid Hyb_1 is perfectly indistinguishable from hybrid Hyb_0.*

  Proof. Due to the security of $\mathcal{F}^{p,r}_{\mathsf{Prep}}$, $\mathcal{A}$ cannot know anything about random vectors $\boldsymbol{\mu}, \boldsymbol{\eta}$ that are used to mask the wire values. Therefore, even if $\mathcal{S}$ uses a randomly selected $\tilde{\boldsymbol{w}}$ as the dealer's input, $\mathcal{A}$ cannot be aware of that in the step 1-3 in the online phase of Π_{SIF}. In the final step of the online phase, since $\mathcal{S}$ sends y_i, m_{y_i} such that $m_{y_i} = k_{y_i} - y_i \cdot \Delta^{(i)}$ to V_i^* if $\mathcal{S}$ receives $(\textsc{Output}, \mathsf{sid}, y_i)$ from $\mathcal{F}_{\mathsf{SIF}}$, V_i^* is convinced to output acceptance. In conclusion, hybrid Hyb_1 is perfectly indistinguishable from hybrid Hyb_0. □

- Hybrid Hyb_2: Same as Hyb_1, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}_{\mathsf{SIF}}$ when $\mathcal{F}_{\mathsf{SIF}}$ sends $(\textsc{Continue}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_2 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is honest, $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{F}^{p,r}_{\mathsf{COIN}}}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}} \equiv \mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$ holds.

When the dealer is honest. In this case, $\mathcal{S}$ has to extract the malicious dealer's input. We describe the simulation strategy of $\mathcal{S}$ in the following:

1. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Prep}}^{p,r}$ for the adversary $\mathcal{A}$.
2. $\mathcal{S}$ acts as honest verifiers to interact with D^* and completes the protocol execution.
3. If D^* makes at least one of the honest verifiers output y_i, then $\mathcal{S}$ computes $\boldsymbol{w} := \boldsymbol{\mu} + \boldsymbol{\delta}$; notice that, $\mathcal{S}$ knows $\boldsymbol{\mu}$ since $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Prep}}^{p,r}$ for the adversary $\mathcal{A}$. Then $\mathcal{S}$ uses the extracted $\boldsymbol{w}$ to compute $\tilde{\boldsymbol{y}} := \mathcal{C}(\boldsymbol{w})$. If $\tilde{y}_i = y_i$ holds for all honest verifiers V_i who outputs y_i, $\mathcal{S}$ sends $\boldsymbol{w}$ to $\mathcal{F}_{\mathsf{SIF}}$ on behalf of the malicious dummy $\tilde{\mathsf{D}}^*$; otherwise, $\mathcal{S}$ aborts.
4. Whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}_{\mathsf{SIF}}$ when $\mathcal{F}_{\mathsf{SIF}}$ sends $(\textsc{Continue}, \mathsf{sid})$.

   We then prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: This is the real-world execution $\mathsf{EXEC}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{Prep}}^{p,r},\mathcal{F}_{\mathsf{COIN}}^{p,r}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Prep}}^{p,r}$ honestly for $\mathcal{A}$, extracts the dealer's input $\boldsymbol{w}$, and computes $\tilde{\boldsymbol{y}} := \mathcal{C}(\boldsymbol{w})$. If $\tilde{y}_i = y_i$ holds for all honest verifiers V_i who outputs y_i, $\mathcal{S}$ sends $\boldsymbol{w}$ to $\mathcal{F}_{\mathsf{SIF}}$ on behalf of the malicious dummy $\tilde{\mathsf{D}}^*$; otherwise, $\mathcal{S}$ aborts.

Lemma 4. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^{-r} = \mathsf{negl}(\lambda)$. Let $\mathcal{C}$ be the circuit with t multiplication gates. Then hybrid Hyb_1 is statistically indistinguishable from hybrid Hyb_0 with adversarial advantage at most $(t+3) \cdot p^{-r}$.*

Proof. Here we prove that the probability of a cheating D^* using $\boldsymbol{w}$ and convincing a honest verifier V_i that y_i is the correct output, where $\tilde{\boldsymbol{y}} := \mathcal{C}(\boldsymbol{w})$ and $\tilde{y}_i \neq y_i$, is negligible. It is easy to see that the cheating D^* is able to do that if D^* can forge a MAC tag, which happens at probability p^{-r}.

Now let us focus on the case where D^* cannot forge the MAC tag. We will prove that if D^* commits to $\boldsymbol{w}$ using $\boldsymbol{\mu}$, then D^* cannot convince a honest verifier a false y_i is the correct output, except with negligible probability. It is easy to see that the wire values that are associated with addition gates must be computed correctly. For the i-th multiplication gates, we assume that the output wire values of the former $i-1$ multiplication gates are always correct. For the i-th multiplication gates, the parties holds $[\![w_\alpha]\!], [\![w_\beta]\!], [\![w_\gamma]\!]$ with $w_\gamma = w_\alpha \cdot w_\beta + e_i$, where $e_i \in \mathbb{F}_p$ is an error chosen by D^*. Then for each honest $\mathsf{V}_j \in \mathcal{H}$, we have

$$\begin{aligned}
B_i^{(j)} &= k_\alpha^{(j)} \cdot k_\beta^{(j)} - k_\gamma^{(j)} \cdot \Delta^{(j)} \\
&= (m_\alpha^{(j)} + w_\alpha \cdot \Delta^{(j)}) \cdot (m_\beta^{(j)} + w_\beta \cdot \Delta^{(j)}) - (m_\gamma^{(j)} + (w_\gamma + e_i) \cdot \Delta^{(j)}) \cdot \Delta^{(j)} \\
&= A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)} - e_i \cdot (\Delta^{(j)})^2 \ .
\end{aligned}$$

Then in the step 3 of online phase, D^* sends $\hat{U}^{(j)} := U^{(j)} + e_U^{(j)}$ and $\hat{V}^{(j)} := V^{(j)} + e_V^{(j)}$, where $e_U^{(j)}, e_v^{(j)}$ are the errors chosen by D^*. Furthermore, for each honest V_j, we have

$$\begin{aligned}
Z^{(j)} &= \sum_{i=1}^{t} B_i^{(j)} \cdot \chi^i + z^{(j)} \\
&= \sum_{i=1}^{t} (A_{i,0}^{(j)} + A_{i,1}^{(j)} \cdot \Delta^{(j)} - e_i \cdot (\Delta^{(j)})^2) \cdot \chi^i + v^{(j)} + u^{(j)} \cdot \Delta^{(j)} \\
&= U^{(j)} + V^{(j)} \cdot \Delta^{(j)} - (\sum_{i=1}^{t} e_i \cdot \chi^i) \cdot (\Delta^{(j)})^2 \\
&= (\hat{U}^{(j)} - e_U^{(j)}) + (\hat{V}^{(j)} - e_V^{(j)}) \cdot \Delta^{(j)} - (\sum_{i=1}^{t} e_i \cdot \chi^i) \cdot (\Delta^{(j)})^2 \ .
\end{aligned}$$

If the check passes, then we have $Z^{(j)} = \hat{U}^{(j)} + \hat{V}^{(j)} \cdot \Delta^{(j)}$. In this case, we have the following equation:

$$e_U^{(j)} + e_V^{(j)} \cdot \Delta^{(j)} + (\sum_{i=1}^{t} e_i \cdot \chi^i) \cdot (\Delta^{(j)})^2 = 0 \ .$$

By the famous Schwartz-Zippel lemma [Sch80,Zip79], we know that if $\sum_{i=1}^{t} e_i \cdot \chi^i \neq 0$, then the probability of the above equation holds is at most $2 \cdot p^{-r}$, and the probability of $\sum_{i=1}^{t} e_i \cdot \chi^i = 0$ holds is at most $t \cdot p^{-r}$.

By the union bound of the probability, we conclude that Hyb_1 is statistically indistinguishable from hybrid Hyb_0 with adversarial advantage at most $(t+3) \cdot p^{-r}$. $\square$

- Hybrid Hyb_2: Same as Hyb_1, except that whenever $\mathcal{A}$ wants to make a honest party $\mathsf{H} \in \mathcal{H}$ abort, $\mathcal{S}$ simply returns $(\textsc{Abort}, \mathsf{sid}, \mathsf{H})$ to $\mathcal{F}_{\mathsf{SIF}}$ when $\mathcal{F}_{\mathsf{SIF}}$ sends $(\textsc{Continue}, \mathsf{sid})$. Perfect indistinguishability is trivial.

Hybrid Hyb_2 is the ideal world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is malicious, $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^{p,r}_{\mathsf{Prep}},\mathcal{F}^{p,r}_{\mathsf{COIN}}}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}}$ with adversarial advantage at most $(t+3) \cdot p^{-r}$. $\square$