Practical Constructions for Single Input Functionality against a Dishonest Majority[*]

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

September 14, 2023

Abstract

Single Input Functionality (SIF) is a special case of MPC, where only one distinguished party called dealer holds the secret input. SIF allows the dealer to complete a computation task and send to other parties their respective outputs without revealing any additional information about its secret input. SIF has many applications, including multiple-verifier zero-knowledge and verifiable relation sharing, etc. Recently, several works devote to round-efficient realization of SIF, and achieve 2-round communication in the honest majority setting (Applebaum *et al.*, Crypto 2022; Baum *et al.*, CCS 2022; Yang and Wang, Asiacrypt 2022).

In this work, we propose *the first* practical 2-round protocol for SIF against *a dishonest majority* in the preprocessing model; moreover, the online phase is highly efficient as it requires no cryptographic operations and achieves information theoretical security. For SIF among 5 parties, our construction takes 152.34ms (total) to evaluate an AES-128 circuit with 7.36ms online time. Compared to the state-of-the-art (honest majority) solution (Baum *et al.*, CCS 2022), our construction is roughly 2× faster in the online phase, although more preprocessing time is needed. Compared to the state-of-the-art generic MPC against a dishonest majority (Wang *et al.*, CCS 2017; Cramer *et al.*, Crypto 2018), our construction outperforms them w.r.t. both total running time and online running time.

[*] 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

Single-Input Functionality (SIF) has received a lot of attention in the recent years [AKP20, AKP22b]. We can view SIF as a special case of MPC, where only one distinguished party, called the dealer D, is allowed to hold a private input $\boldsymbol{w}$, while all other parties, called the verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$ have no private input. More concretely, let $(y_1, \ldots, y_n) \leftarrow \mathcal{C}(\boldsymbol{w})$ be the SIF they jointly computes; after the execution, the i-th verifier V_i obtains y_i as its private output.

The investigation of SIF can be traced back to the work by Gennaro *et al.* [GIKR02]. Very recently, Applebaum *et al.* [AKP22b] observe that SIF have two direct applications – *Multiple-Verifier Zero-Knowledge (MVZK)* [BJO$^+$22, YW22] and *Verifiable Relation Sharing (VRS)* [AKP20]. Note that, as shown in [AKP22b], MVZK can be viewed as a special case of VRS.

Multiple-Verifier Zero-Knowledge (MVZK). In a MVZK protocol, a distinguished party called prover P, who takes input as the statement x and the witness w, and he wants to convince the n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$ that $\mathcal{R}(x, w) = 1$ for an NP relation $\mathcal{R}$. It is easy to see that MVZK can be implemented via SIF. Namely, let $\mathcal{C}$ be the circuit that evaluates $\mathcal{R}(x, w)$. Then the parties can invoke SIF to jointly evaluate $\mathcal{C}$ such that the verifiers can obtain $\mathcal{C}(x, w)$ as their outputs.

Recently, several 2-round MVZK protocols [AKP22b, BJO$^+$22, YW22] have been constructed in the honest majority setting. More precisely, in these constructions, the adversary is allowed to corrupt the prover and the minority of the verifiers. Among them, Applebaum *et al.* [AKP22b] focus more on the theoretical side and show how to construct a 2-round MVZK protocol in the plain model using non-interactive commitments. On the other hand, Baum *et al.* [BJO$^+$22] and Yang and Wang [YW22] provide highly efficient constructions. More concretely, Yang and Wang [YW22] show how to construct 2-round MVZK protocols in the random oracle (RO) model, where the prover sends a single message to each verifier in the first round, and the verifiers exchange messages among them and make a decision in the second round. Similarly, the MVZK protocols proposed by Baum *et al.* [BJO$^+$22] are also 2-round, but in the preprocessing model[1].

We emphasize again that, all the aforementioned 2-round MVZK protocols are designed in the honest majority setting. *How to construct a 2-round MVZK protocol in the dishonest majority setting (i.e., the prover and the majority of the verifiers are malicious), is still an open problem.*

Verifiable Relation Sharing (VRS). Analogously, in a VRS protocol, we also consider a distinguished prover P who holds a private input s, and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$ who have no private inputs. The prover P shares the secret s to the verifiers; denote the verifier V_i's share as x_i, for $i \in [n]$. In addition, the prover P proves in zero-knowledge that $\mathcal{R}(s, x_1, \ldots, x_n) = 1$ for an NP relation $\mathcal{R}$. Clearly, VRS can also be implemented via SIF. In particular, we define a circuit $(y_1, \ldots, y_n) \leftarrow \mathcal{C}(s, x_1, \ldots, x_n)$ such that $y_i = x_i$ for $i \in [n]$ if $\mathcal{R}(s, x_1, \ldots, x_n) = 1$; otherwise, $y_i = \perp$ for $i \in [n]$, where $\perp$ is a failure symbol. We note that, as shown in [AKP22b], VRS implies many important cryptographic primitives, such as MVZK, Verifiable Secret Sharing (VSS) [CGMA85], Verifiable Function Secret Sharing [BGI16] and Secure Multicast [GIKR01].

Applebaum *et al.* show that a 2-round VRS protocol can be constructed using non-interactive commitment as a building block; their protocol allows the adversary to corrupt the prover and up to $t < \frac{n}{3}$ verifiers [AKP20]. The same authors later improve the corruption threshold without increasing the round complexity [AKP22b]. More precisely, the protocol in [AKP22b] remains secure in the presence of a corrupted prover and up to $t < \frac{n}{2+\epsilon}$ corrupted verifiers, where ϵ is a positive constant.

Again, we stress that, to the best of our knowledge, all the existing 2-round VRS protocols in the literature assume the majority of players are honest, and *constructing a 2-round VRS protocol against a dishonest majority remains an open problem.*

1.1 Problem Statement

2-round SIF with a dishonest majority. In this work, we will investigate the open problem of constructing practical 2-round protocols for SIF in the dishonest majority setting. As mentioned above, a positive answer to this open problem will immediately give us a 2-round MVZK/VRS protocol.

We note that achieving highly efficient 2-round SIF against a dishonest majority is a non-trivial task, even that in the preprocessing model, only the number of communication rounds of the online phase will be counted. One may attempt to directly adapt generic MPC protocols to realize SIF; unfortunately, the round complexity of (the online phase of) a SPDZ-style MPC [DPSZ12, KOS16, KPR18, CDE$^+$18, Kel20] linearly depends on the depth

[1] The preprocessing model is arguably stronger than the RO model.

of the circuit. BMR-style MPC protocols [BMR90, LPSY15, LSS16, HSS17, WRK17b, YWZ20] could be a plausible solution, as they use constant rounds of (online) communication. In particular, the BMR-style protocol by Lindell *et al.* [LSS16] and the BMR-style protocol by Hazay *et al.* [HSS17] use only 2 rounds of online communication. However, these two protocols are far from being extremely practical, and they only support boolean circuits.

Another relevant notation is called *distributed ZK (dZK)* by Boneh *et al.* [BBC+19], which can be viewed as ZK proof over a secretly shared statement. In a dZK, the prover takes the statement-witness pair (x, w) as input; the prover will convince the n verifiers that $(x, w) \in \mathcal{R}$, where the statement x is shared among the verifiers. (Please see more details about dZK in Section 7.) Boneh *et al.* [BBC+19] give a 2-round dZK construction in the RO model; there, the adversary is allowed to corrupt *either* the prover *or* $(n-1)$ verifiers; however, the *collusion* between the malicious prover and the malicious verifiers is *not* allowed. Note that, in SIF, the adversary is allowed to *simultaneously* corrupt the dealer *and* (some of) the verifiers.

2-round SIF with IT security. We also wonder whether it is possible to design a protocol for SIF with Information Theoretic (IT) security in the preprocessing model. Recall that, the successful story of SPDZ-style MPC [DPSZ12, KOS16, KPR18, CDE+18, Kel20] reminds us: if the MPC protocols can be designed with IT security in the preprocessing model, then they (especially, the online phase) are likely to be highly efficient in practice since no heavy cryptographic operations are needed.

Hereby, we raise the research question of this work:

> *Is it possible to construct a highly efficient 2-round SIF for both arithmetic and boolean circuits against a dishonest majority in the preprocessing model with IT security?*

1.2 Our Results

In this work, we will give an affirmative answer to the above question; we present *the first* 2-round highly efficient protocol for SIF against a dishonest majority in the preprocessing model, and our protocol can support both arithmetic and boolean circuits. In addition, our protocol can be proven secure in the Universal Composability (UC) framework [Can01]. We stress that, the online phase of our protocol is information theoretically secure. As mentioned before, both VRS and MVZK can be viewed as special cases of SIF; as a side product, we also obtain *the first* 2-round MVZK and VRS protocols against a dishonest majority in the preprocessing model. In the following, we provide a brief intuition of our construction; the technical overview can be found in Section 3, and construction details including the preprocessing phase and the online phase can be found in Section 4 and Section 5, respectively.

Intuition. In our design, we make extensive use of a particular form of correlated randomness, called *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 order to authenticate the random value $x \in \mathbb{F}_p$, we let the party who holds the MAC key $(\Delta, K) \in \mathbb{F}_{p^r}^2$ compute the MAC tag $m_x := K + \Delta \cdot x \in \mathbb{F}_{p^r}$. It is easy to see that a malicious party who obtains m_x but does not know the MAC key, cannot produce another valid $m_{x'}$ for $x' \neq x$ except for negligible probability when $|\mathbb{F}_{p^r}|$ is sufficiently large. In the dishonest majority setting, IT-MACs are often combined with secret shares [BDOZ11, DPSZ12]. More concretely, random values are shared among all the parties (e.g., for a random x, party P_i obtains x_i such that $x = \sum_{i=1}^{n} x_i$), and the shares are authenticated to each other using IT-MACs. These random values are often used to mask the wire values of the circuit.

Our key observation is that: in SIF, only the dealer holds the private input $\boldsymbol{w}$; revealing the random masked value of $\boldsymbol{w}$ to the dealer does not compromise the security, as the dealer is already aware of $\boldsymbol{w}$. In addition, the random masked value is secretly shared among the verifiers.

Following the above observation, we are able to design a 2-round SIF. We will follow the "gate-by-gate" design paradigm. The dealer first commits to its secret input $\boldsymbol{w}$ by broadcasting $\boldsymbol{\delta} := \boldsymbol{w} - \boldsymbol{x}$, where $\boldsymbol{x}$ is the random value that is held by the dealer and shared among the verifiers. Due to the linearity of shares, the verifiers transform the shares of $\boldsymbol{\delta}$ into the shares of $\boldsymbol{w}$. It is easy to see that all addition gates of the circuit can be processed locally for free. For the multiplication gates of the circuit, we use the "Beaver triples" techniques [Bea92]. More precisely, for each multiplication gate with input wire indices α, β and output wire index γ, we prepare three correlated random values (a, b, c) such that $c = ab$ in advance, and we denote by w_α, w_β the input wire values and denote by w_γ the output wire value. If we set $\eta := w_\alpha - a$ and $\nu := w_\beta - b$, then it is easy to see that

$$\begin{aligned} w_\gamma &= w_\alpha \cdot w_\beta = (w_\alpha - a + a) \cdot (w_\beta - b + b) \\ &= \eta \cdot \nu + \eta \cdot b + \nu \cdot a + c \end{aligned} \tag{1}$$

In MPC protocols in the dishonest majority setting, such as BDOZ-style MPC [BDOZ11] and SPDZ-style MPC [DPSZ12], parties need to publicly open η and ν first; then the parties can obtain the shares of $w_\alpha \cdot w_\beta$ based on Equation 1. To process the multiplication gates, interactions between the protocol parties are required; the overall round complexity of BDOZ-style and SPDZ-style MPC protocols depend on the circuit depth. In contrast, in our setting, we can simply let the dealer broadcast η, ν since w_α, w_β, a, b are known by the dealer. Therefore, all multiplication gates can be processed simultaneously at once! In order to prevent the dealer from cheating in computing η, ν, we let the verifiers to open $\tilde{\eta} := w_\alpha - a, \tilde{\nu} := w_\beta - b$ and check if $\tilde{\eta} = \eta, \tilde{\nu} = \nu$ hold in the following round. For output gates, for instance, the i-th output gate that belongs to V_i. We denote by h_i the output wire value, and let other verifiers open their shares of h_i to V_i, so V_i can recover its output h_i. Clearly, our protocol for SIF can be constructed within 2 rounds.

Comparisons. In Table 1, we compare our result with the related state-of-the-art 2-round protocols for MVZK/dZK/SIF. In addition, in Table 2, we compare our result with related MPC (and 2PC) protocols that achieve practical efficiency against a dishonest majority.

Table 1: Comparison of our work with the relevant state-of-the-art 2-round protocols.

Reference	Primitive*	Corruption Threshold	Assumption	UC Security?
[BBC+19]	dZK	$t < \frac{n}{2} + 1^\dagger$	Random Oracle	✗
[HVW22]	dZK‡	$t < \frac{n-2}{6} + 1$	Coin Flipping	✗¶
[YW22]	MVZK	$t < \frac{n}{2} + 1$	Random Oracle	✓
[BJO+22]	MVZK	$t < \frac{n}{3} + 1$	Preprocessing	✗¶
[GIKR02]	SIF	$t < \frac{n}{6} + 1$	-	✗¶
[AKP20]	SIF	$t < \frac{n}{3} + 1$	Non-Interactive Commitment$^\#$	✓
[AKP22b]	SIF	$t < \frac{n}{2+\epsilon} + 1^\S$	Non-Interactive Commitment$^\#$	✓
This Work	SIF	$t < n + 1$	Preprocessing	✓

* dZK: distributed zero-knowledge; MVZK: multiple-verifier zero-knowledge; SIF: single input functionality.

† In [BBC+19], the adversary is allowed to corrupt (1) the prover *or* up to $n-1$ verifiers; or (2) the prover *and* less than $\frac{n}{2}$ verifiers.

‡ In [HVW22], Hazay *et al.* strengthen the formalization of dZK in [BBC+19] by introducing *strong completeness*.

¶ Protocols in [GIKR02, BJO+22, HVW22] are proven to be secure in the standalone model [Can00]; however, we believe that these protocols can be proven to be UC-secure under suitable setup assumptions.

§ Here, ϵ is a positive constant; Protocols in [AKP22b], can be proven secure when $t < \frac{n}{2} + 1$, if n is logarithmic in the security parameter.

$^\#$ In [AKP20, AKP22b], Applebaum *et al.* stress that their protocols are constructed in the *plain model* using non-interactive commitment; their protocols achieve computational (resp. everlasting) security when the underlying non-interactive commitment is perfectly-binding and sub-exponentially hiding (resp. computationally-binding and statistically-hiding). We believe that their protocols could be able to achieve information theoretical security when the underlying non-interactive commitment is replaced by an ideal commitment functionality $\mathcal{F}_{\mathsf{COM}}$ [CF01, CLOS02].

We report our performance results in Section 6.1 and the comparison results with other works in Section 6.2, respectively. Our construction for SIF is highly efficient: for SIF among 5 parties, our construction takes 152.34ms (total) to evaluate an AES-128 circuit with 7.36ms online time. We compare the performance of our protocol with that by Baum *et al.* [BJO+22], called Feta[2]; note that Feta is the only one that has been implemented among the state-of-the-art relevant works [AKP22b, BJO+22, YW22]. When there are 5 parties (1 prover and 4 verifiers), Feta takes 18.25ms to evaluate an AES-128 circuit with 16.24ms online time; roughly, our protocol uses 2× less online time than Feta, although our protocol has a slower pre-processing phase. In addition, we remark that, when the number of parties is 5, Feta only tolerates a single corrupted verifier while our protocol can tolerate 3 corrupted verifiers. Furthermore, Feta is specially designed for MVZK while our protocol is for SIF, supporting MVZK, VRS and more.

To further demonstrate the efficiency of our protocol, we also compare the performance of our SIF with that of several MPC protocols in the dishonest majority setting. When there are 3 parties, our protocol takes 302.43ms to

[2] In Feta [BJO+22], two MVZK protocols have been constructed: 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. In our paper, we only refer to the former one.

Table 2: Comparison of our work with related highly efficient MPC (and 2PC) protocols, against a dishonest majority.

	Reference	Primitive	Number of Rounds (Online Phase)[†]	Corruption Threshold	Circuit Type
2PC (GC-Style)	[WRK17a]	2PC	4	$t = 1$	Boolean
	[KRRW18]	2PC	4	$t = 1$	Boolean
	[DILO22]	2PC	2	$t = 1$	Boolean
	[CWYY23]	2PC	2	$t = 1$	Boolean
MPC (BMR-Style)	[LSS16]	MPC	2	$t < n + 1$	Boolean
	[HSS17]	MPC	2	$t < n + 1$	Boolean
	[WRK17b]	MPC	3	$t < n + 1$	Boolean
	[YWZ20]	MPC	4	$t < n + 1$	Boolean
MPC (SPDZ-Style)	[DPSZ12]	MPC	Circuit Depth	$t < n + 1$	Boolean/Arithmetic[‡]
	[CDE+18]	MPC	Circuit Depth	$t < n + 1$	Boolean/Arithmetic
SIF (BDOZ-Style)	This Work	SIF	2	$t < n + 1$	Boolean/Arithmetic

[†] The listed protocols are all designed in the preprocessing model. Here we only report the rounds for online phase; all these protocols are designed in a modular way, and the rounds for preprocessing phase would be optimized if new techniques are available.

[‡] In [DPSZ12], Damgård *et al.* only claim that their protocols can support arithmetic circuits. However, when extension field is used, we believe their protocols can also support boolean circuits.

evaluate a SHA-256 circuit with only 7.20ms online time. Compared to the state-of-the-art MPC protocols against a dishonest majority [WRK17b, CDE+18], our improvement ranges from 1.2× to 1.7× w.r.t. the total running time, and ranges from 2.2× to 4.4× w.r.t. the online phase running time.

2 Preliminaries

2.1 Notation

We use $\lambda \in \mathbb{N}$ to denote the security parameter. We say that a function $\mathsf{negl} : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $\mathrm{poly}(\cdot)$ and all sufficiently large λ, it holds that $\mathsf{negl}(\lambda) < \frac{1}{\mathrm{poly}(\lambda)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We say that two distribution ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ are statistically (resp. computationally indistinguishable), which we denote by $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$ (resp., $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$), if for any unbounded (resp., PPT) distinguisher $\mathcal{A}$ there exists a negligible function negl s.t. $|\Pr[\mathcal{A}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{A}(\mathcal{Y}_\lambda) = 1]| = \mathsf{negl}(\lambda)$.

We use $x \leftarrow S$ to denote that sampling x uniformly at random from a finite set S. For $n \in \mathbb{N}$, we denote by $[n]$ a set $\{1, \ldots, n\}$. For $a, b \in \mathbb{Z}$ with $a \leq b$, we denote by $[a, b] = \{a, \ldots, b\}$. We use bold lower-case letters like $\boldsymbol{x}$ for vectors, and denote by x_i the i-th element of vector $\boldsymbol{x}$.

We consider an extension field $\mathbb{F}_{p^r}$ of a finite field $\mathbb{F}_p$, where $p \geq 2$ is a prime or a power of a prime and $r \geq 1$ is an integer. We fix some monic, irreducible polynomial $f(X)$ of degree r and write $\mathbb{F}_{p^r} \cong \mathbb{F}_p[X]/f(X)$. Therefore, every $w \in \mathbb{F}_{p^r}$ can be written uniquely as $w = \sum_{i=1}^{r} w_i \cdot X^{i-1}$ with $w_i \in \mathbb{F}_p$ for all $i \in [r]$. Thus, we could view the elements over $\mathbb{F}_{p^r}$ equivalently as the vectors in $(\mathbb{F}_p)^r$.

Based on field $\mathbb{F}_p$, we can define a circuit $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$ as follows: First, the circuit $\mathcal{C}$ 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 $|\mathcal{I}_{\mathsf{out}}| = n$. Second, the circuit $\mathcal{C}$ consists of a list of gates of the form $(\alpha, \beta, \gamma, T)$, where α, β are the indices of the input wires and γ is the index of the output wire, and $T \in \{\mathsf{Add}, \mathsf{Mult}\}$ is the type of the gate. If $p = 2$, then $\mathcal{C}$ is a boolean circuit with $\mathsf{Add} = \oplus$ and $\mathsf{Mult} = \wedge$; note that, one can compute $x \oplus 1$ to negate x in a boolean circuit. If $p \geq 2$ is a prime or a power of a prime, then $\mathcal{C}$ is an arithmetic circuit where $\mathsf{Add}/\mathsf{Mult}$ corresponds to addition/multiplication in $\mathbb{F}_p$.

2.2 Universal Composability

We formalize and analyze the security of our protocols in the Universal Composability (UC) framework by Canetti [Can01]. In the following, we give a high-level description for UC framework, and we refer readers to see more details in [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. In this work, we consider malicious static corruption, i.e., the adversary corrupts the parties at the beginning of the protocol and the corrupted parties may deviate from protocol instructions. We also consider rushing adversaries, who may delay sending messages on behalf of corrupted parties in a given round until the messages sent by all the uncorrupted parties in that round have been received.

Secure communication model. In this work, we consider simultaneous communication, and we assume the parties are connected by pairwise secure channels and a broadcast channel.

2.3 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*. In our paper, the majority of players can be corrupted; we thus consider a relaxed version of their SIFs, capturing *security with abort*.

A formal presentation of (the relaxed version of) the functionality, $\mathcal{F}_{\mathsf{SIF}}$, can be found in Figure 1. More concretely, in a SIF, there are a dealer D and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$. Without loss of generality, we assume that all the parties hold a circuit $\mathcal{C} : \mathbb{F}_p^m \rightarrow \mathbb{F}_p^n$ while the dealer D additionally holds a secret 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 element of $\boldsymbol{y}$.

2.4 Information-Theoretic Message Authentication Codes

Originating from the work by Beaver [Bea92], who shows how to use "Beaver triples" for designing efficient protocols in the dishonest majority setting, many MPC protocols make extensive use of correlated randomness for better efficiency [BDOZ11, DPSZ12, NNOB12, WRK17a, WRK17b, CDE+18, BGIN21, EGP+23]. Among them,

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

The functionality interacts with a dealer D, n verifiers $\mathsf{V}_1,\ldots,\mathsf{V}_n$ and an adversary $\mathcal{S}$. Let $\mathcal{H}_{\mathsf{V}}$ denote the set of honest verifiers. The functionality is parameterized by a circuit $\mathcal{C}$ where $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$.

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 $\mathsf{V}_i^* \notin \mathcal{H}_{\mathsf{V}}$ where y_i is the i-th element of $\boldsymbol{y}$.
- Send $(\textsc{Continue}, \mathsf{sid})$ to the adversary $\mathcal{S}$. For each $\mathsf{V}_i \in \mathcal{H}_{\mathsf{V}}$, 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 1: The Functionality $\mathcal{F}_{\mathsf{SIF}}$

a powerful technique called Information-Theoretic Message Authentication Codes (IT-MACs) has been used to authenticate the values, especially the secret shares [BDOZ11, NNOB12].

In this subsection, we review the IT-MACs [YSWW21] over the extension field $\mathbb{F}_{p^r}$ where $p^r > 2^\lambda$. For simplicity, we assume there are two parties P_1 and P_2. Let $\Delta \in \mathbb{F}_{p^r}$ be the global MAC key held by P_1. A value $x \in \mathbb{F}_p$ known by P_2 is authenticated to P_1 by having P_1 hold a local MAC key K and having P_2 hold the corresponding MAC tag $\mathsf{MAC}_{\Delta,K}(x) := K + \Delta \cdot x \in \mathbb{F}_{p^r}$. It is easy to see that a malicious P_2^* who sees $\mathsf{MAC}_{\Delta,K}(x)$ for a chosen x cannot produce a new valid MAC tag $\mathsf{MAC}_{\Delta,K}(x')$ for $x' \neq x$ except with probability $p^{-r} < 2^{-\lambda}$. Furthermore, the security of IT-MACs holds when an honest party has many MAC keys that share the same Δ but with independently random K, and we call such MAC keys *consistent*.

Another appealing advantage of IT-MACs is *additive homomorphism*. Precisely, for consistent keys $(\Delta, K_1), \ldots, (\Delta, K_n)$, given the public coefficients $c_1, \ldots, c_n, c \in \mathbb{F}_p$, it holds that $\mathsf{MAC}_{\Delta,K}(y) := \sum_{i=1}^n c_i \cdot \mathsf{MAC}_{\Delta,K_i}(x_i) \in \mathbb{F}_{p^r}$, where $y := \sum_{i=1}^n c_i \cdot x_i + c \in \mathbb{F}_p$ and $K := \sum_{i=1}^n c_i \cdot K_i - c \cdot \Delta \in \mathbb{F}_{p^r}$.

3 Technical Overview

Before giving the formal description of our construction, we first provide a technical overview of our design in this section. Full descriptions of our protocols can be found in Section 4 and Section 5, below.

3.1 BDOZ-Style MPC

Our starting point is the BDOZ-style MPC [BDOZ11, NNOB12] which is designed in the preprocessing model; that is, the parties first jointly prepare some correlated randomness in the preprocessing phase, and those correlated randomness will be "consumed" during the online phase to accelerate the online computation. BDOZ-style MPC exploits the merit of IT-MACs (cf. Section 2.4) to achieve malicious security while preserving high performance during the online phase.

Here we provide a high-level description for BDOZ-style MPC. Suppose there are n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. In the preprocessing phase, the parties jointly generate sufficiently many random values. For instance, for random $x \in \mathbb{F}_p$, each party P_i holds an additive share $x_i \in \mathbb{F}_p$ such that $x = \sum_{i=1}^n x_i$. For each ordered pair of parties $(\mathsf{P}_i, \mathsf{P}_j)$, P_i authenticates its own share (namely, x_i) to P_j, i.e., at the end of the preprocessing phase, P_i holds $x_i \in \mathbb{F}_p$ and $M_{x_i}^j \in \mathbb{F}_{p^r}$ and P_j holds $\Delta_j, K_{x_i}^j \in (\mathbb{F}_{p^r})^2$ such that $M_{x_i}^j = \mathsf{MAC}_{\Delta_j, K_{x_i}^j}(x_i) = K_{x_i}^j + \Delta_j \cdot x_i \in \mathbb{F}_{p^r}$. Given these authenticated shares, every party can share its secret input easily. Suppose party P_i wants to share its secret input $w \in \mathbb{F}_p$, other parties simply open their random shares of x to P_i; After recovering x, P_i can simply broadcast $\delta := w - x \in \mathbb{F}_p$ to others. By the additive homomorphism of IT-MACs, all the parties obtain the shares of w.

Then the parties will execute the protocol for online phase in the "gate-by-gate" paradigm. The addition gates can be processed without interactions between the parties. For each multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, one authenticated Beaver triples [Bea92] (i.e., the authenticated shares of a, b, c such that $c = a \cdot b$) should be prepared in the preprocessing phase. To compute $w_\alpha \cdot w_\beta$, each party P_i holds the shares of input wire values $w_{\alpha,i}, w_{\beta,i}$ and the shares of the authenticated Beaver triple (a_i, b_i, c_i); then all the parties can open $\eta := w_\alpha - a$ and $\nu := w_\beta - b$. Since $w_\alpha \cdot w_\beta = (w_\alpha - a + a) \cdot (w_\beta - b + b) = \eta \cdot \nu + \eta \cdot b + \nu \cdot a + c$, all the parties can compute the shares of $w_\alpha \cdot w_\beta$ based on the shared (a, b, c) and the public η and ν. Since the process of multiplication gates involves interactions between the parties, the round complexity of the online phase of BDOZ-style MPC linearly depends on the circuit depth.

3.2 Key Observations

BDOZ-style vs. SPDZ-style. Compared with the BDOZ-style MPC, its follow-up, the SPDZ-style MPC [DPSZ12], is more popular in the community and has been widely studied [KOS16, KPR18, CDE+18, Kel20]. In the SPDZ-style MPC, the shares are formed in a different way: each party holds a share of a single global MAC key; for a secret value x, each party holds a share of x and a share of the MAC tag on x. In most application scenarios, we prefer SPDZ-style shares to BDOZ-style shares, since the size of SPDZ-style shares is smaller; thus, it is more efficient to operate on SPDZ-style shares. However, opening SPDZ-style shares requires a "commit-and-open" procedure, which will increase the round complexity. Since we are pursing for 2-round online communication, BDOZ-style shares are more suitable for our purpose.

The dealer can learn all correlated randomness. In a conventional MPC setting, the BDOZ-style protocol uses the authenticated shares of random values to mask the wire values of the circuit to ensure the privacy, i.e., to make sure that corrupted parties cannot recover the wire values. Our key observation is that in the SIF setting, only the dealer D has a private input $\boldsymbol{w}$ and other parties' inputs are public; therefore, even if all the correlated randomness generated in the preprocessing phase are revealed to the the dealer D, it does not compromise the overall security of the SIF protocol. This is due to the fact that the dealer D already knows her own input.

3.3 Our Techniques

Reducing the round complexity. Enlightened by the observations above, we are able to reduce the round complexity of the online phase of our construction; namely, we can process all the multiplication gates at once within only 2 rounds! More precisely, we first let the dealer D evaluate the entire circuit. Then for each multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, we let the dealer D broadcast $\eta := w_\alpha - a$ and $\nu := w_\beta - b$ to all the verifiers, where (a, b, c) is the Beaver triple; note that, as discussed above, we let D learn (a, b, c) in plaintext. After receiving the messages from the dealer D, the verifiers can locally compute the shares of $w_\alpha \cdot w_\beta$ as it is done in the BDOZ-style MPC. Finally, in order to check whether the dealer D deviates from the protocol while computing the values η and ν for a multiplication gate, the verifiers can jointly check whether η, ν are computed correctly. Namely, the verifiers will publicly open $\tilde{\eta} := w_\alpha - a$ and $\tilde{\nu} := w_\beta - b$ using their authenticated shares of w_α, w_β, a, b, and then the verifiers jointly check if $\tilde{\eta} = \eta$ and $\tilde{\nu} = \nu$. Clearly, the above can be completed within 2 rounds.

Modified preprocessing. Finally, we show how to generate the BDOZ-style correlated randomness in our setting. This can be achieved by the following steps:

- Step 1: Generate the authenticated shares of random values x and Beaver triples (a, b, c) such that the dealer D and the n verifiers jointly holds their shares.
- Step 2: Let the n verifiers open their shares of all the correlated randomness to the dealer D, and then D checks the validity of the shares w.r.t. their MACs and recover them in plaintext.
- Step 3: Let the dealer D send its shares of all the correlated randomness to each verifier; then one verifier, say V_1, will add the dealer's share to its own share; the remaining $(n-1)$ verifiers will locally update the corresponding MAC keys accordingly.

4 SIF against a Dishonest Majority: Preprocessing Phase

As discussed in Section 3, our protocol will be designed in the preprocessing model. In this section, we mainly focus on how to design the protocol for preprocessing phase. We first provide the BDOZ-style preprocessing functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$, then describe our own preprocessing functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}}$. Finally, we give our protocol Π_{Prep} for preprocessing phase which UC-realizes $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}}$ in the $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$-hybrid world.

4.1 BDOZ-Style Preprocessing Functionality

First of all, we provide a quick recap of the BDOZ-style preprocessing phase [BDOZ11, NNOB12, WRK17a, WRK17b], and we make some modifications to adapt to our setting, where there is a dealer D and n verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$. We let the dealer D (resp. each verifier V_i) hold its global MAC keys Δ_0 (resp. Δ_i). To share a value $x \in \mathbb{F}_p$ among $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$, we will randomly select $x_0, x_1, \ldots, x_n \leftarrow \mathbb{F}_p$ such that $x := \sum_{i=0}^{n} x_i$ and give x_0 to D and x_i to V_i for $i \in [n]$. Furthermore, these shares are authenticated to each other using IT-MACs. For example, to authenticate V_i's share (namely, x_i) to V_j, we let V_j hold a local MAC key $K_{x_i}^{j}$ which is uniformly random; meanwhile, we let V_i hold a MAC tag $M_{x_i}^{j}$ such that $M_{x_i}^{j} := \mathsf{MAC}_{\Delta_j, K_{x_i}^{j}}(x_i) = K_{x_i}^{j} + \Delta_j \cdot x_i$. When the parties decide to make x public, the corrupted parties cannot lie about their shares, since the corrupted parties cannot forge a MAC tag except with a negligible probability. For better presentation, we introduce the following notation $[\![x]\!]_{\mathsf{BDOZ}}$ to denote the BDOZ-style shares of value x:

$$[\![x]\!]_{\mathsf{BDOZ}} = \{\{x_i, \{\Delta_i, K_{x_j}^{i}, M_{x_i}^{j}\}_{j \in [0,n] \setminus \{i\}}\}_{i \in [0,n]}\} \ ,$$

where $\{x_0, \{\Delta_0, K_{x_j}^{0}, M_{x_0}^{j}\}_{j \in [n]}\}$ (resp. $\{x_i, \{\Delta_i, K_{x_j}^{i}, M_{x_i}^{j}\}_{j \in [0,n] \setminus \{i\}}\}$) are privately held by the dealer D (resp. each verifier V_i), and we use $[\![x]\!]_{\mathsf{BDOZ}}$ as shorthand when it is not need to explicitly talk about the shares and MAC tags. We call $[\![\cdot]\!]_{\mathsf{BDOZ}}$ the BDOZ-style shares.

Notice that, authenticated Beaver triples (i.e., $[\![a]\!]_{\mathsf{BDOZ}}$, $[\![b]\!]_{\mathsf{BDOZ}}$,$[\![c]\!]_{\mathsf{BDOZ}}$ such that $c = ab$) will also be generated during the BDOZ-style preprocessing phase. Formally, we present the functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$ in Figure 2 which captures the BDOZ-style preprocessing phase, and the functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$ is adapted from [BDOZ11, WRK17b].

Functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$

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

Initial. Upon receiving $(\textsc{Init}, \mathsf{sid})$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, do

1. If $\mathsf{D} \notin \mathcal{H}$, receive $\Delta_0 \in \mathbb{F}_{p^r}$ from the adversary $\mathcal{S}$; otherwise, sample random $\Delta_0 \leftarrow \mathbb{F}_{p^r}$.
2. For $i \in [n]$: If $\mathsf{V}_i \notin \mathcal{H}$, receive $\Delta_i \in \mathbb{F}_{p^r}$ from $\mathcal{S}$; otherwise, sample random $\Delta_i \leftarrow \mathbb{F}_{p^r}$.
3. Return Δ_0 to D and Δ_i to V_i for each $i \in [n]$.
4. Store $\{\Delta_i\}_{i \in [0,n]}$, and ignore the subsequent $\textsc{Init}$ command.

Singles. Upon receiving $(\textsc{Singles}, \mathsf{sid}, u)$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, for $v \in [u]$:

1. Send $(\textsc{Singles}, \mathsf{sid}, v)$ to the adversary $\mathcal{S}$, and wait for an input from $\mathcal{S}$. If it is $\textsc{Abort}$, return $(\textsc{Abort}, \mathsf{sid}, v)$ to every honest party and halt. If it is $\textsc{Continue}$, continue the procedure.
2. Sample random $x \leftarrow \mathbb{F}_p$ and create $[\![x]\!]_{\mathsf{BDOZ}}$ as follows:
   (a) If $\mathsf{D} \notin \mathcal{H}$, receive $x_0 \in \mathbb{F}_p$ and $\{K_{x_j}^{0}, M_{x_0}^{j}\}_{j \in [n]} \in (\mathbb{F}_{p^r})^{2n}$ from $\mathcal{S}$.
   (b) For $i \in [n]$: If $\mathsf{V}_i \notin \mathcal{H}$, receive $x_i \in \mathbb{F}_p$ and $\{K_{x_j}^{i}, M_{x_i}^{j}\}_{j \in [0,n] \setminus \{i\}} \in (\mathbb{F}_{p^r})^{2n}$ from $\mathcal{S}$.
   (c) For each honest $\mathsf{V}_i \in \mathcal{H}$, its share $x_i \in \mathbb{F}_p$ is chosen at random, subject to $x = \sum_{i=0}^{n} x_i$. (Here we can regard D as V_0 for notation convenience.)
   (d) For each honest $\mathsf{V}_i \in \mathcal{H}$ and $j \in [0,n] \setminus \{i\}$, $K_{x_j}^{i}$ is chosen as follows: if $\mathsf{V}_j \in \mathcal{H}$, sample random $K_{x_j}^{i} \leftarrow \mathbb{F}_{p^r}$; otherwise, set $K_{x_j}^{i} := M_{x_j}^{i} - \Delta_i \cdot x_j \in \mathbb{F}_{p^r}$. (Here we can regard D as V_0 for notation convenience.)
   (e) For $i \in [0,n], j \in [0,n] \setminus \{i\}$: Compute $M_{x_i}^{j} := \mathsf{MAC}_{\Delta_j, K_{x_i}^{j}}(x_i) = K_{x_i}^{j} + \Delta_j \cdot x_i \in \mathbb{F}_{p^r}$.
   (f) Send $\{x_0, \{K_{x_j}^{0}, M_{x_0}^{j}\}_{j \in [n]}\}$ to D and $\{x_i, \{K_{x_j}^{i}, M_{x_i}^{j}\}_{j \in [0,n] \setminus \{i\}}\}$ to each V_i for $i \in [n]$.

Triples. Upon receiving $(\textsc{Triples}, \mathsf{sid}, u)$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, for $v \in [u]$:

1. The same as step 1 in **Singles** procedure.
2. For each triple to create, the functionality samples $a, b \leftarrow \mathbb{F}_p$ and sets $c := a \cdot b \in \mathbb{F}_p$, then it creates $[\![a]\!]_{\mathsf{BDOZ}}$, $[\![b]\!]_{\mathsf{BDOZ}}$,$[\![c]\!]_{\mathsf{BDOZ}}$, each as step 2 in **Singles** procedure.

Figure 2: The Functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$ for BODZ-Style Preprocessing

4.2 Our Preprocessing Functionality

Unlike the BDOZ-style preprocessing phase where secrets are shared among all the parties, in our preprocessing phase, we share secrets among the verifiers $\mathsf{V}_1, \ldots, \mathsf{V}_n$ and let the dealer D hold the entire secrets. The shares' authentication among $\mathsf{V}_1, \ldots, \mathsf{V}_n$ follows the same method as in Section 4.1. More precisely, for a secret x, we let each verifier V_i hold a share x_i and $\{K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}}$; meanwhile, we let the dealer D hold the verifiers' shares $x_1, \ldots, x_n$. Notice that, the verifiers' MAC keys and MAC tags are hidden from the dealer. In this way, the dealer D can announce the secret $x := \sum_{i=1}^n x_i$ by itself, later the verifiers can open their shares to check if the announcement is correct. Even if the malicious dealer D^* colludes with some verifiers, D^* cannot make a false announcement without being detected, since the corrupted verifiers cannot lie about their shares. For notation convenience, we use the following way of representing x:

$$[\![x]\!] = \{\{x_i\}_{i\in[n]}, \{x_i, \{\Delta_i, K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]}\} \ ,$$

where $\{x_i\}_{i\in[n]}$ (resp. $\{x_i, \{\Delta_i, K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}}\}$) is privately held by the prover P (resp. the verifier V_i), and we use $[\![x]\!]$ as shorthand when it is not need to explicitly talk about the shares and MAC tags.

For modularity, we assume that there is an ideal functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ providing us with the above, and we present the functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ in Figure 3. Similar to $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$ depicted in Figure 2, our $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ also generates the authenticated Beaver triples, i.e., $[\![a]\!], [\![b]\!], [\![c]\!]$ such that $c = ab$.

Functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$

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

Initial. Upon receiving $(\textsc{Init}, \mathsf{sid})$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, do

1. For $i \in [n]$: If $\mathsf{V}_i \notin \mathcal{H}$, receive $\Delta_i \in \mathbb{F}_{p^r}$ from $\mathcal{S}$; otherwise, sample random $\Delta_i \leftarrow \mathbb{F}_{p^r}$.
2. Return Δ_i to V_i for each $i \in [n]$.
3. Store $\{\Delta_i\}_{i\in[n]}$, and ignore the subsequent $\textsc{Init}$ command.

Singles. Upon receiving $(\textsc{Singles}, \mathsf{sid}, u)$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, for $v \in [u]$:

1. Send $(\textsc{Singles}, \mathsf{sid}, v)$ to the adversary $\mathcal{S}$, and wait for an input from $\mathcal{S}$. If it is $\textsc{Abort}$, return $(\textsc{Abort}, \mathsf{sid}, v)$ to every honest party and halt. If it is $\textsc{Continue}$, continue the procedure.
2. Sample random $x \leftarrow \mathbb{F}_p$ and create $[\![x]\!]$ as follows:
   (a) For $i \in [n]$: If $\mathsf{V}_i \notin \mathcal{H}$, receive $x_i \in \mathbb{F}_p$ and $\{K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}} \in (\mathbb{F}_{p^r})^{2n}$ from $\mathcal{S}$.
   (b) For each honest $\mathsf{V}_i \in \mathcal{H}$, its share $x_i \in \mathbb{F}_p$ is chosen at random, subject to $x = \sum_{i=1}^n x_i$.
   (c) For each honest $\mathsf{V}_i \in \mathcal{H}$ and $j \in [n] \setminus \{i\}$, $K^i_{x_j}$ is chosen as follows: if $\mathsf{V}_j \in \mathcal{H}$, sample random $K^i_{x_j} \leftarrow \mathbb{F}_{p^r}$; otherwise, set $K^i_{x_j} := M^i_{x_j} - \Delta_j \cdot x_j \in \mathbb{F}_{p^r}$.
   (d) For $i \in [n], j \in [n] \setminus \{i\}$: Compute $M^i_{x_j} := \mathsf{MAC}_{\Delta_j, K^j_{x_i}}(x_i) = K^j_{x_i} + \Delta_j \cdot x_i \in \mathbb{F}_{p^r}$.
   (e) Send $\{x_i\}_{i\in[n]}$ to D and $\{x_i, \{K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}}\}$ to each verifier V_i.

Triples. Upon receiving $(\textsc{Triples}, \mathsf{sid}, u)$ from D and $\mathsf{V}_1, \ldots, \mathsf{V}_n$, for $v \in [u]$:

1. The same as step 1 in **Singles** procedure.
2. For each triple to create, the functionality samples $a, b \leftarrow \mathbb{F}_p$ and sets $c := ab \in \mathbb{F}_p$, then it creates $[\![a]\!], [\![b]\!], [\![c]\!]$, each as step 2 in **Singles** procedure.

Figure 3: The Functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ for Preprocessing

Operations on our $[\![\cdot]\!]$ shares. By additive homomorphism of IT-MACs described in Section 2.4, when the MAC keys are consistent (i.e., each party holds its own single global MAC key and many independently random local MAC keys), linear operations on our $[\![\cdot]\!]$ shares can be performed locally. For completeness, we present it in Figure 4.

4.3 Our Protocol for Preprocessing Phase

In this subsection, we show how to efficiently realize our preprocessing functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$. Our key idea is to convert BDOZ-style shares $[\![\cdot]\!]_{\mathsf{BDOZ}}$ into our shares $[\![\cdot]\!]$. We achieve this by letting the verifiers send their shares

Local Operations

Addition of shares: Given two consistent $[\![x]\!]$ and $[\![y]\!]$ (i.e., both $[\![x]\!]$ and $[\![y]\!]$ are generated under the same global MAC keys), $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ obtain the consistent shares of $z := x + y \in \mathbb{F}_p$ by locally performing the followings:

- D computes $z_i := x_i + y_i \in \mathbb{F}_p$ for $i \in [n]$.
- For $i \in [n]$: Each V_i computes $z_i := x_i + y_i \in \mathbb{F}_p, K^i_{z_j} := K^i_{x_j} + K^i_{y_j} \in \mathbb{F}_{p^r}, M^j_{z_i} := M^j_{x_i} + M^j_{y_i} \in \mathbb{F}_{p^r}$ for $j \in [n] \setminus \{i\}$.

Multiplication by constants: Given $[\![x]\!]$ and a constant $c \in \mathbb{F}_p$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ obtain the consistent shares of $z := c \cdot x \in \mathbb{F}_p$ by locally performing the followings:

- D computes $z_i := c \cdot x_i \in \mathbb{F}_p$ for $i \in [n]$.
- For $i \in [n]$: Each V_i computes $z_i := c \cdot x_i \in \mathbb{F}_p, K^i_{z_j} := c \cdot K^i_{x_j} \in \mathbb{F}_{p^r}, M^j_{z_i} := c \cdot M^j_{x_i} \in \mathbb{F}_{p^r}$ for $j \in [n] \setminus \{i\}$.

Addition of constants: Given $[\![x]\!]$ and a constant $c \in \mathbb{F}_p$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ obtain the consistent shares of $z := c + x \in \mathbb{F}_p$ by locally performing the followings:

- D computes $z_1 := c + x_1 \in \mathbb{F}_p$ and $z_i := x_i \in \mathbb{F}_p$ for $i \in [2, n]$.
- V_1 computes $z_1 := c + x_1 \in \mathbb{F}_p, K^1_{z_j} := K^1_{x_j} \in \mathbb{F}_{p^r}, M^j_{z_1} := M^j_{x_1} \in \mathbb{F}_{p^r}$ for $j \in [2, n]$. For $i \in [2, n]$: Each V_i sets $z_i := x_i \in \mathbb{F}_p, K^i_{z_j} := K^i_{x_j} \in \mathbb{F}_{p^r}, M^j_{z_i} := M^j_{x_i} \in \mathbb{F}_{p^r}$ for $j \in [2, n] \setminus \{i\}$ and computes $K^i_{z_1} := K^i_{x_1} - c \cdot \Delta_i \in \mathbb{F}_{p^r}$, $M^1_{z_i} := M^1_{x_i} \in \mathbb{F}_{p^r}$.

Figure 4: Local Operations on our $[\![\cdot]\!]$ Shares

to the dealer privately, so the dealer can obtain the entire random values. After that, we let the dealer open its (original) shares to the verifiers, so the verifiers can update their shares locally. We present our protocol $\Pi^{\mathsf{Ours}}_{\mathsf{Prep}}$ for preprocessing phase in Figure 5 and prove the security through Theorem 1.

Protocol Π_{Prep}

Initial. On input $(\textsc{Init}, \mathsf{sid})$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ work as follows:

1. $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{Init}, \mathsf{sid})$ to $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$, which returns $\{\Delta^0_j\}_{j \in [n]}$ to P and $\{\Delta^i_j\}_{j \in [0,n] \setminus \{i\}}$ to V_i for each $i \in [n]$.
2. V_i outputs $\{\Delta^i_j\}_{j \in [n] \setminus \{i\}}$ for each $i \in [n]$.

Singles. On input $(\textsc{Singles}, \mathsf{sid}, u)$, for each $v \in [n]$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ work as follows:

1. $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{Singles}, \mathsf{sid}, 1)$ to $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$, which returns either $\textsc{Abort}$ or $[x]_{\mathsf{BDOZ}}$ to them. If it is the first case, they simply abort. If it is the second case, D receives $\{x_0, \{K^0_{x_j}, m_j(x_0)\}_{j \in [n]}\}$ and each V_i receives $\{x_i, \{K^i_{x_j}, m_j(x_i)\}_{j \in [0,n] \setminus \{i\}}\}$. Notice that, $x = \sum_{i=0}^n x_i$.
2. For $i \in [n]$: V_i opens its share to D by sending $x_i, m_0(x_i)$ to P over a private channel. Then D checks if $m_0(x_i) = K^0_{x_i} + \Delta^0_i \cdot x_i$ holds. If not, D aborts.
3. D opens its share to $\mathsf{V}_1, \ldots, \mathsf{V}_n$ by sending $x_0, m_i(x_0)$ to each V_i over a private channel. Then V_i checks if $m_i(x_0) = K^i_{x_0} + \Delta^i_0 \cdot x_0$ holds. If not, V_i aborts.
4. V_1 updates its share and MAC tag by setting $x'_1 := x_1 + x_0$ and $m_i(x'_1) := m_i(x_1)$ for $i \in [2, n]$. For $i \in [2, n]$: V_i updates its local MAC keys by setting $K^i_{x'_1} := K^i_{x_1} - x_0 \cdot \Delta^i_1$. In this way, we create new authenticated shares of x among the verifiers.
5. For notation convenience, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ set $x'_i := x_i, m_j(x'_i) := m_j(x_i), K^j_{x'_i} := K^j_{x_i}$ for $i \in [2, n], j \in [n] \setminus i$. Notice that, now $x = \sum_{i=1}^n x'_i$
6. D outputs $\{x'_i\}_{i \in [n]}$ and V_i outputs $\{x'_i, \{K^i_{x'_j}, m_j(x'_i)\}_{j \in [n] \setminus \{i\}}\}$ for each $i \in [n]$.

Triples. On input $(\textsc{Triples}, \mathsf{sid}, u)$, for each $v \in [n]$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ work as follows:

1. $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ send $(\textsc{Triples}, \mathsf{sid}, 1)$ to $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$, which returns either $\textsc{Abort}$ or $[a]_{\mathsf{BDOZ}}, [b]_{\mathsf{BDOZ}}, [c]_{\mathsf{BDOZ}}$ to them. If it is the first case, they simply abort. If it is the second case, they receive $[a]_{\mathsf{BDOZ}}$, $[b]_{\mathsf{BDOZ}}$, $[c]_{\mathsf{BDOZ}}$ such that $c = ab$.
2. For each $t \in \{a, b, c\}$, $\mathsf{D}, \mathsf{V}_1, \ldots, \mathsf{V}_n$ convert $[t]_{\mathsf{BDOZ}}$ to $[t]$ as step 2-6 in **Singles** procedure.

Figure 5: Our Protocol Π_{Prep} for Preprocessing Phase in the $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$-Hybrid World

Theorem 1. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. The protocol Π_{Prep} depicted in Figure 5 UC-realizes the functionality $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ depicted in Figure 3 with information theoretic security in the $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $(n-1)$ verifiers.*

Proof. We leave the formal proof in Appendix A.1. □

5 SIF against a Dishonest Majority: Main Protocol

In this section, we aim to provide the main protocol for SIF against a dishonest majority. Since we have described how to realize the preprocessing phase in Section 4, here we mainly focus on the online phase. The intuition of our protocol for online phase can be found in Section 3. We give a high-level description of our protocol for online phase in the following.

Protocol Π_{SIF}

Inputs: The dealer D and the verifiers $\mathsf{V}_1, \dots, \mathsf{V}_n$ hold a circuit $\mathcal{C} : \mathbb{F}_p^m \to \mathbb{F}_p^n$. We denote by $\mathcal{I}_{\mathsf{in}}$ the input wires of $\mathcal{C}$ and denote by $\mathcal{I}_{\mathsf{out}}$ the output wires of $\mathcal{C}$. We assume the circuit $\mathcal{C}$ has s multiplication gates. The dealer D also holds a secret input $\boldsymbol{w}$ in $(\mathbb{F}_p)^m$.

Preprocessing Phase. Both the concrete structure of the circuit and the witness are unknown.

1. $\mathsf{D}, \mathsf{V}_1, \dots, \mathsf{V}_n$ send $(\textsc{Init}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Prep}}$, which returns $\Delta_i \in \mathbb{F}_{p^r}$ to V_i for each $i \in [n]$.
2. $\mathsf{D}, \mathsf{V}_1, \dots, \mathsf{V}_n$ send $(\textsc{Singles}, \mathsf{sid}, m)$ to $\mathcal{F}_{\mathsf{Prep}}$, which returns $\{\llbracket \mu_i \rrbracket\}_{i \in [m]}$ to them. More precisely, for each $i \in [m]$, D holds $\{\mu_{i,1}, \dots, \mu_{i,n}\}$ such that $\mu_i = \sum_{j=1}^n \mu_{i,j}$; meanwhile, each V_j holds $\{\mu_{i,j}, \{K^j_{\mu_{i,k}}, M^k_{\mu_{i,j}}\}_{k \in [n] \setminus \{j\}}\}$. D computes $\mu_i := \sum_{j=1}^n \mu_{i,j} \in \mathbb{F}_p$ for each $i \in [m]$.
3. $\mathsf{D}, \mathsf{V}_1, \dots, \mathsf{V}_n$ send $(\textsc{Triples}, \mathsf{sid}, s)$ to $\mathcal{F}_{\mathsf{Prep}}$, which returns $\{\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket\}_{i \in [t]}$ to them where $c_i = a_i b_i$. More precisely, for each $i \in [s]$ and $\rho_i \in \{a_i, b_i, c_i\}$, D holds $\{\rho_{i,1}, \dots, \rho_{i,n}\}$ such that $\rho_i = \sum_{j=1}^n \rho_{i,j}$; meanwhile, each V_j holds $\{\rho_{i,j}, \{K^j_{\rho_{i,k}}, M^k_{\rho_{i,j}}\}_{k \in [n] \setminus \{j\}}\}$. D computes $\rho_i := \sum_{j=1}^n \rho_{i,j} \in \mathbb{F}_p$ for each $i \in [s]$ and $\rho_i \in \{a_i, b_i, c_i\}$.

Online Phase. Now the concrete structure of the circuit and the witness are known by the parties.

Round 1: The dealer D works as follows:

1. For $i \in \mathcal{I}_{\mathsf{in}}$: D broadcasts $\delta_i := w_i - \mu_i \in \mathbb{F}_p$ to all verifiers, where w_i is i-th element in $\boldsymbol{w}$.
2. For each gate $(\alpha, \beta, \gamma, T)$, D evaluates the circuit $\mathcal{C}$ in a predefined topological order:
   (a) If $T = \mathsf{Add}$, D computes $w_\gamma := w_\alpha + w_\beta \in \mathbb{F}_p$, where $w_\alpha, w_\beta, w_\gamma$ are the wire values correspond to the wire indices α, β, γ of this gate.
   (b) If $T = \mathsf{Mult}$ and it is the i-th multiplication gate, D computes $w_\gamma := w_\alpha \cdot w_\beta \in \mathbb{F}_p$ first, then broadcasts $\eta_i := w_\alpha - a_i \in \mathbb{F}_p$ and $\nu_i := w_\beta - b_i \in \mathbb{F}_p$ to all verifiers.

Round 2: The verifiers $\mathsf{V}_1, \dots, \mathsf{V}_n$ work as follows:

3. For $i \in \mathcal{I}_{\mathsf{in}}$: the verifiers compute $\llbracket w_i \rrbracket := \llbracket \mu_i \rrbracket + \delta_i$ using the received $\delta_i \in \mathbb{F}_p$.
4. For each gate $(\alpha, \beta, \gamma, T)$, the verifiers evaluate the circuit $\mathcal{C}$ in a predefined topological order:
   (a) If $T = \mathsf{Add}$, the verifiers compute $\llbracket w_\gamma \rrbracket := \llbracket w_\alpha \rrbracket + \llbracket w_\beta \rrbracket$.
   (b) If $T = \mathsf{Mult}$ and it is the i-th multiplication gate, the verifiers compute $\llbracket w_\gamma \rrbracket := \llbracket c_i \rrbracket + \eta_i \llbracket b_i \rrbracket + \nu_i \llbracket a_i \rrbracket + \eta_i \cdot \nu_i$ using the received $\eta_i, \nu_i \in \mathbb{F}_p$.
5. The verifiers perform the followings to check the multiplication gates: For i-th multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, the verifiers open $\llbracket \tilde{\eta}_i \rrbracket := \llbracket w_\alpha \rrbracket - \llbracket a_i \rrbracket$ and $\llbracket \tilde{\nu}_i \rrbracket := \llbracket w_\beta \rrbracket - \llbracket b_i \rrbracket$ to each other. Then the verifiers check if $\tilde{\eta}_i = \eta_i$ and $\tilde{\nu}_i = \nu_i$ hold. The verifiers will abort if any verifier fails to open its share or any check fails
6. The verifiers perform the followings to obtain their output: For $i \in \mathcal{I}_{\mathsf{out}}$ with authenticated output wire value $\llbracket h_i \rrbracket$, the verifiers open $\llbracket h_i \rrbracket$ to V_i. If any verifier fails to open its share, V_i aborts; otherwise, V_i outputs h_i.

Figure 6: Our Main Protocol Π_{SIF} in the $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$-Hybrid World

We design our protocol for online phase in the "gate-by-gate" paradigm. The dealer D who holds the secret input $\boldsymbol{w} \in \mathbb{F}_p^m$ first commits to all input wire values to the verifiers $\mathsf{V}_1, \dots, \mathsf{V}_n$ by consuming m random values $\{\llbracket \mu_i \rrbracket\}_{i \in [m]}$ produced by $\mathcal{F}_{\mathsf{Prep}}$. More precisely, for each $i \in \mathcal{I}_{\mathsf{in}}$, D broadcasts the masked input wire value $\delta_i := w_i - \mu_i$ to all verifiers. Then $\mathsf{V}_1, \dots, \mathsf{V}_n$ obtain the shares of input wire value by computing $\llbracket w_i \rrbracket := \llbracket \mu_i \rrbracket + \delta_i$. As discussed in Section 4.2, $\llbracket \cdot \rrbracket$ is additively homomorphic; therefore, addition gates can be processed for free. For each multiplication gate, one authenticated Beaver triple $(\llbracket a_i \rrbracket, \llbracket b_i \rrbracket, \llbracket c_i \rrbracket)$ such that $c_i = a_i \cdot b_i$ will be consumed to ensure the multiplication gate will be processed properly. This technique is originated from the work by Beaver [Bea92]. More precisely, for each multiplication gate $(\alpha, \beta, \gamma, \mathsf{Mult})$, D broadcasts $\eta_i := w_\alpha - a_i$ and $\nu_i := w_\beta - b_i$ to the verifiers, where w_α and w_β are the input wires values of this gate. By the following eqution $w_\alpha \cdot w_\beta = (w_\alpha - a_i + a_i) \cdot (w_\beta - b_i + b_i) = (\eta_i + a_i) \cdot (\nu_i + b_i) = \eta_i \cdot \nu_i + \eta_i \cdot b_i + \nu_i \cdot a_i + c_i$, it is clear that if D acts honestly, the verifiers are able to reconstruct the shares of output wire value $\llbracket w_\gamma \rrbracket$ by locally computing $\eta_i \cdot \nu_i + \eta_i \llbracket b_i \rrbracket + \nu_i \llbracket a_i \rrbracket + \llbracket c_i \rrbracket$. If D acts maliciously, i.e., D broadcasts the false η_i or ν_i, the verifiers are able to detect this malicious behavior by opening $\llbracket \tilde{\eta}_i \rrbracket := \llbracket w_\alpha \rrbracket - \llbracket a_i \rrbracket$ and $\llbracket \tilde{\nu}_i \rrbracket := \llbracket w_\beta \rrbracket - \llbracket b_i \rrbracket$ to each other, and checking if $\tilde{\eta}_i = \eta_i$

and $\tilde{\nu}_i = \nu_i$ hold. Finally, the verifiers hold the shares of output wire values $\{[\![h_i]\!]\}_{i\in[n]}$. In order to let V_i obtain its own output, other verifiers simply open $[\![h_i]\!]$ to V_i. Notice that, during the protocol execution, the honest verifiers would abort if any check fails or any verifier fails to open its share.

Formally, we present our main protocol Π_{SIF}, which captures both preprocessing phase and online phase, in Figure 6 and prove the security through Theorem 2.

Theorem 2. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. The protocol Π_{SIF} depicted in Figure 6 UC-realizes the functionality $\mathcal{F}_{\mathsf{SIF}}$ depicted in Figure 1 with information theoretic security in the $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$-hybrid world, in the presence of a static malicious adversary corrupting up to the dealer and $(n-1)$ verifiers.*

Proof. We leave the formal proof in Appendix A.2. □

6 Implementation and Evaluation

We implement a prototype of our protocols in C++, and conduct a benchmark on various circuit evaluations. The performance of our protocols is reported in Section 6.1. Three state-of-the-art works in the literature [AKP22b, YW22, BJO+22] are closely related to our protocol. Among them, only Feta is implemented by the authors and reported the performance in their paper; therefore, we compare the efficiency of our protocols with Feta [BJO+22] in Section 6.2. We then compare the performance of our SIF protocols with the state-of-the-art generic MPC protocols in the dishonest majority setting (cf. Section 6.2).

We present experimental validation of the efficiency of our protocols over well-known boolean circuits. Note that, our protocols can support both arithmetic and boolean circuits. Hereby, we report the benchmark results over boolean circuits (AES-128 and SHA-256) in order to have a fair comparison between our work and Feta [BJO+22]. (They only provide the benchmark results on boolean circuit evaluation in their paper. We note that at the time of submission, the code of Feta for threshold $t < \frac{n}{3} + 1$ is not publicly available.)

We instantiate the BDOZ-style preprocessing functionality $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$ over extension field $\mathbb{F}_{p^r}$ for boolean circuits with the offline protocol in [WRK17b] and the code is available in the EMP-toolkit framework Github repository [WMK16]. We set $p = 2$ and $r = 128$, which provides at least 40-bit statistical security. All experiments are executed on a machine with Intel Xeon Silver 4214 CPU at 2.20GHz and 128 GB Memory, running Ubuntu 20.04.5 LTS. The network setting is exactly the same as in [BJO+22], i.e. at delay of 0.6ms and bandwidth of 10Gbit/s. Each experiment is run 40 times and the median is taken.

6.1 Performance of Our Protocols

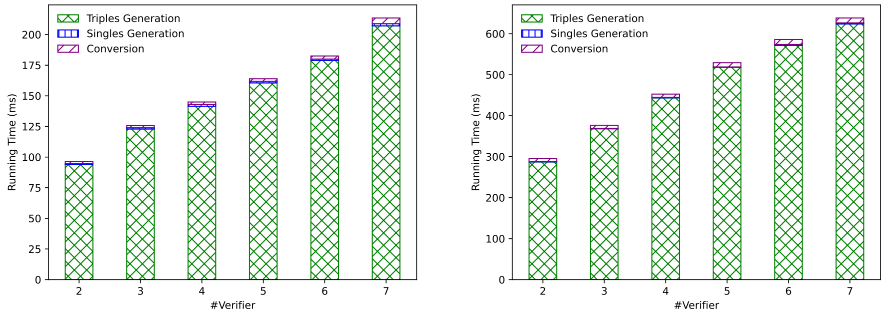
Table 3 illustrates the running time of our protocol w.r.t. AES-128 and SHA-256 evaluation. The numbers of verifiers are 2, 4, 7, respectively. We report the evaluation results in 4 dimensions: preprocessing time, dealer time, verifier time and proof size. The numbers of running time consist of both computation and communication time. The proof size refers to the size of the message that the dealer sends to each verifier. The running time is reported in millisecond (ms) and the proof size is reported in KiloByte (KB).

Table 3: The Performance of our protocols.

		#Verifiers		
		2	4	7
AES-128	Preprocessing Time (ms)	96.28	144.98	213.55
	Dealer Time (ms)	0.22	0.27	0.41
	Verifier Time (ms)	2.04	7.09	20.91
	Proof Sizes (KB)	12.75	12.75	12.75
SHA-256	Preprocessing Time (ms)	295.23	452.75	623.86
	Dealer Time (ms)	0.75	0.86	1.04
	Verifier Time (ms)	6.45	22.01	66.89
	Proof Sizes (KB)	44.84	44.84	44.84

As shown in Table 3, the performance of our protocols is highly efficient: when there are single prover and 4 verifiers, it takes 152.34ms to evaluate an AES-128 circuit, in which online running time (the sum of dealer time and verifier time) is merely 7.36ms. Notice that, the dealer running time of our protocol is extremely fast, since in addition to evaluating the entire circuit in plaintext, the extra computation cost for dealer is 1 addition (resp. 2 additions) per input wire (resp. AND gate), which is almost for free.

Microbenchmark. From Table 3, we found that the most time-consuming step of our protocols lies in the preprocessing phase. As discussed in Section 4, our preprocessing phase can be divided into two parts: the first part involves generating the BDOZ-style shares $[\![\cdot]\!]_{\mathsf{BDOZ}}$ and the second part involves converting $[\![\cdot]\!]_{\mathsf{BDOZ}}$ into the shares $[\![\cdot]\!]$ as required by our SIF protocol. The generation of BDOZ-style shares can also be divided in two components: triples generation (i.e., the generation of the Beaver triples in $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$) and singles generation (i.e., the generation of the single random values in $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$). To figure out which part is the most time-consuming, we conduct a microbenchmark of our preprocessing protocol and plot the results in Figure 7. As shown in Figure 7, the cost of the preprocessing procedure mainly comes from the triples generations.



(a) Evaluation on AES-128

(b) Evaluation on SHA-256

Figure 7: Microbenchmark of our preprocessing protocol. “Triples Generation” refers to the time taken to generate all the Beaver triples in $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$; “Singles Generation” refers to the time taken to generate all the single random values in $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$; “Conversion” refers to the time taken to convert $[\![\cdot]\!]_{\mathsf{BDOZ}}$ into the shares $[\![\cdot]\!]$ that we need.

In this work, we instantiate the BDOZ-style shares generation protocol with the realization proposed by Wang *et al.* [WRK17b]. We notice that the recent work by Yang *et al.* [YWZ20] mainly focuses on speeding up the triples generation in BDOZ-style shares. Compared to [WRK17b], their improvement ranges from roughly 4× to 6×. Moreover, they also reduce the communication cost for the rest of the preprocessing protocol by roughly 1.3× without increasing the cost for the online phase; thus, they obtain a more efficient BMR-style MPC than [WRK17b]. Since our preprocessing protocol is designed in a modular fashion, if we instantiated the BDOZ-style shares generation protocol with that by Yang *et al.* [YWZ20], our preprocessing time could be benefited by the same magnitude of improvement.

We make an estimation of the running time according to the improvement reported in [YWZ20] and put the results in Table 4. As shown in Table 4, for SIF among 3 parties, our protocol (combined with [YWZ20]) is expected to take only 25.83ms preprocessing time for AES-128 evaluation and 80.95ms for SHA-256 evaluation. Compared to the running time of our SIF protocol using preprocessing protocol realized by [WRK17b], this is a roughly 3.7× improvement.

It is worth to mention that since our SIF protocol uses BDOZ-type preprocessing in a blackbox fashion, should there be any faster and better ideas to generate the BDOZ-type shares in the coming future, the preprocessing time of our SIF protocol can also be benefited.

6.2 Comparison with Relevant Works

Comparison with Feta **[BJO$^+$22].** Here we compare the performance of our protocols with Feta. We stress that, Feta is specifically designed for MVZK, while our SIF protocol can be applied not only to MVZK, but also to other applications, such as VRS. The comparison results are depicted in Table 5. The numbers for Feta reported in Table 5 are taken in their published paper. Our protocol is evaluated on the same hardware and network configuration as in [BJO$^+$22].

As shown in Table 5, our protocol is roughly 2× faster in the online phase compared to Feta, when there are single prover and 4 verifiers. The main drawback of our protocol lies in the time-consuming preprocessing phase, compared to Feta. However, our protocol is in the dishonest majority setting (the corruption threshold of our protocol is $t < n + 1$), while Feta assumes an honest majority (the corruption threshold of Feta is $t < \frac{n}{3} + 1$);

Table 4: The performance comparison of preprocessing phase among WRK [WRK17b], YWZ [YWZ20] and ours. The number of parties is set as $n = 3$.

	Protocol	Triples Generation Time (ms)	Total Preprocesing Time (ms)
AES-128	WRK [WRK17b]	100.34	119.19
	YWZ [YWZ20]†	25.21	39.72
	Ours + [WRK17b]§	94.09	96.28
	Ours + [YWZ20]†,§	**23.64**	**25.83**
SHA-256	WRK [WRK17b]	290.01	345.61
	YWZ [YWZ20]†	72.87	115.64
	Ours + [WRK17b]§	286.18	295.23
	Ours + [YWZ20]†,§	**71.90**	**80.95**

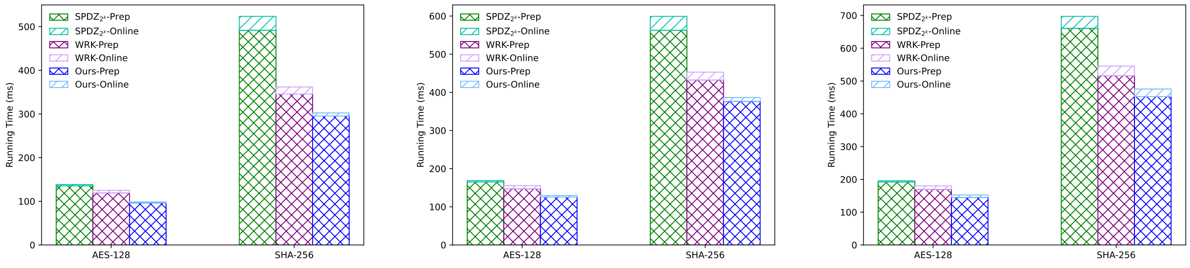
† The numbers in "YWZ [YWZ20]" and "Ours + [YWZ20]" rows are estimated according to the improvement that reported in [YWZ20].
§ "Ours + [WRK17b]" (resp. "Ours + [YWZ20]") refers to the combination of our protocol and [WRK17b] (resp. [YWZ20]).

Table 5: Performance comparison between ours and Feta [BJO+22]. We set the number of verifiers $n = 4$, in this case, Feta only tolerates a single corrupted verifier while our protocol can tolerate 3 corrupted verifiers.

	Protocol	Corruption Threshold	Prep. Time(ms)	Online Time(ms)	Proof Size(KB)
AES-128	Feta [BJO+22]	t<$\frac{n}{3}$+1	**2.01**	16.24	**2.75**
	This Work	**t<n+1**	144.98	**7.36**	12.75
SHA-256	Feta [BJO+22]	t<$\frac{n}{3}$+1	**3.41**	47.07	**8.60**
	This Work	**t<n+1**	452.75	**22.87**	44.84

typically, the protocols against a dishonest majority are less efficient than the protocols that assume an honest majority. Furthermore, in the following paragraph, we will show that our preprocessing phase is faster than some state-of-the-art generic MPC protocols against a dishonest majority.

Comparison with generic MPC against a dishonest majority. To further demonstrate the efficiency of our protocol, we compare our SIF protocols with the state-of-the-art (SOTA) generic MPC protocols over boolean circuits in the dishonest majority setting, i.e., the WRK protocol by Wang *et al.* [WRK17b] and the SPD$\mathbb{Z}_{2^k}$ protocol by Cramer *et al.* [CDE+18]. The code of WRK can be found in EMP toolkit [WMK16] and the code of SPD$\mathbb{Z}_{2^k}$ is provided in [Kel20]. The WRK protocol [WRK17b] represents the SOTA of BMR-style MPC, and the SPD$\mathbb{Z}_{2^k}$ protocol [CDE+18] represents the SOTA of SPDZ-style MPC.



(a) Number of Parties = 3 (b) Number of Parties = 4 (c) Number of Parties = 5

Figure 8: Performance comparison among WRK [WRK17b], SPD$\mathbb{Z}_{2^k}$ [CDE+18] and ours.

As shown in Figure 8, our protocol outperforms WRK and SPD$\mathbb{Z}_{2^k}$ in many aspects. For SIF among three parties, our protocol takes 302.43ms to evaluate a SHA-256 circuit with 7.20ms online running time; while WRK (resp. SPD$\mathbb{Z}_{2^k}$) takes 361.75ms (resp. 523.49ms) to evaluate the same circuit with 16.13ms (resp. 31.85ms) online running time. In this case, our improvement for total running time ranges from 1.2× to 1.7× and our improvement for online running time ranges from 2.2× to 4.4×.

7 Related Work

Multiple-Verifier Zero-Knowledge (MVZK). In a MVZK protocol, the prover who holds $(x, w) \in \mathcal{R}$ can convince n verifiers that x is true at once. The notion of MVZK was first introduced by Burmester and Desmedt [BD91]. Later, Abe *et al.* propose a 2-round MVZK protocol for circuit satisfiability in the presence of a malicious adversary corrupting a prover and up to $t < \frac{n}{3}$ verifiers [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 protocol by Groth and Ostrovsky [GO07, GO14] can be transformed in a 2-round MVZK protocol, and its corruption threshold is $t < \frac{n}{2} + 1$. However, these works [ACF02, GO07, GO14] require heavy public-key operations and are not concretely efficient.

Very recently, there are three papers [AKP22b, YW22, BJO$^+$22] studying 2-round MVZK protocols. Among them, the protocol by Applebaum *et al.* [AKP22b] is the only that provides the full security, i.e., the honest parties are guaranteed to receive the output. More precisely, the protocol by Applebaum *et al.* [AKP22b] assumes non-interactive commitment and its corruption threshold is $t < \frac{n}{2+\epsilon} + 1$, where ϵ is a small positive constant. However, Applebaum *et al.* [AKP22b] focus on a theoretical perspective, and their protocol is not practical. In contrast, the protocols by Yang and Wang [YW22] and Baum *et al.* [BJO$^+$22] are designed to achieve practical efficiency, but their protocols provide weaker security guarantees than [AKP22b]. Yang and Wang [YW22] propose 2-round MVZK protocols assuming a random oracle in the corruption threshold of $t < \frac{n}{2} + 1$; but their protocols only achieve security with abort. Baum *et al.* [BJO$^+$22] employ a stronger assumption (i.e., the preprocessing model) to construct a 2-round MVZK protocol in the corruption threshold of $t < \frac{n}{3} + 1$, and their protocol achieves security with identifiable abort (i.e., when the honest parties do not obtain their output, they can identify the cheaters) which is stronger than [YW22].

The 2-round MVZK protocols above are all proven to be secure in the presence of an honest majority. In contrast, in this work, we remove the honest majority assumption and construct the first 2-round MVZK protocol with IT security against a dishonest majority in the preprocessing model.

Distributed Zero-Knowledge (dZK). The concept of dZK 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. In dZK, the prover is allowed 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 distributed between 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] give two 2-round dZK constructions under RO model in two different settings: (i) in their first construction, the adversary can corrupt the prover *and* up to $t < \frac{n}{2}$ verifiers; (ii) in their second construction, the adversary can corrupt the prover *or* up to $t < n$ verifiers. Several follow-up works [BGIN19, BGIN20, BGIN21] demonstrate that dZKs are quite useful in the context of MPC. Concretely, these works showed how to compile semi-honest MPC protocols into malicious ones using dZKs. In recent work by Hazay *et al.* [HVW22], they strengthen the formalization of [BBC$^+$19] by adding *strong completeness*, which can prevent corrupted verifiers from framing the honest prover, i.e., causing the proof of a correct claim to fail. They call their new formalization *strong-complete dZK*. Hazay *et al.* construct their strong-complete dZK in the corruption threshold of $t < \frac{n-2}{6} + 1$, assuming an ideal coin-flipping. While the constructions in [BBC$^+$19] only achieve security with abort, the construction by Hazay *et al.* can achieve full security.

Verifiable Relation Sharing (VRS). VRS allows the prover to share its input x to multiple verifiers; at the same time, the prover needs to prove in zero-knowledge that the shared data satisfies some properties. The main difference between VRS and dZK is that: in VRS, the prover is allowed to choose the statement and the verifiers' shares; while in dZK, the prover has no control over the statement and verifiers' shares.

To our knowledge, the first VRS was implicitly studied by Gennaro *et al.* [GIKR02] in the context of SIF; their 2-round protocol achieves perfect security and full security in the plain model, and its corruption threshold is $t < \frac{n}{6} + 1$. Applebaum *et al.* improve the corruption threshold to $t < \frac{n}{3} + 1$ at the cost of degrading the perfect security to computational security [AKP20]. Later, the same authors improve the corruption threshold to $t < \frac{n}{2+\epsilon} + 1$, where ϵ is a small positive constant [AKP22b]. Although dZK and VRS are quite different, Hazay *et al.* show a connection between these two primitives [HVW22]. More precisely, under a certain restricted condition (i.e., the relations are robust, and we refer readers to see the definition of robust relations in [HVW22]), Hazay *et al.* show a construction of VRS from dZK as well as a construction of dZK from VRS without further computational assumptions; and both constructions are at the cost of one additional round.

The primitives we discussed above (i.e., MVZK, dZK and VRS) have various applications in both theory and practice, including round-optimal MPC [ACGJ18, AKP22a], private data aggregation [CB17], and anonymous messaging system [CBM15].

8 Conclusion

In this paper, we construct *the first* 2-round highly efficient protocols for SIF against a dishonest majority in the preprocessing model. Our constructions can naturally support both boolean and arithmetic circuits. As a side product, we also obtain *the first* 2-round MVZK and VRS protocols against a dishonest majority in the preprocessing model. To demonstrate the practicality of our constructions, we implement our protocols and conduct extensive experiments. The performance of our protocol is competitive, compared to the state-of-the-art relevant work [BJO+22] in the honest majority setting and the MPC protocols [WRK17b, CDE+18] in the dishonest majority setting. Finally, we leave *whether it is possible or not to construct 1-round SIF protocols (in the preprocessing model) against a dishonest majority* as an open problem. If possible, then how will we construct such protocols?

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.

[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.

[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.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[BGIN19] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 869–886. ACM Press, November 2019.

[BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero-knowledge proofs. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 244–276. Springer, Heidelberg, December 2020.

[BGIN21] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Sublinear GMW-style compiler for MPC with preprocessing. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 457–485, Virtual Event, August 2021. Springer, Heidelberg.

[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.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[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.

[CDE$^+$18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 769–798. Springer, Heidelberg, August 2018.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

[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.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multiparty secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CWYY23] Hongrui Cui, Xiao Wang, Kang Yang, and Yu Yu. Actively secure half-gates with minimum overhead under duplex networks. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 35–67, Cham, 2023. Springer Nature Switzerland.

[DILO22] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Authenticated garbling from simple correlations. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 57–87. Springer, Heidelberg, August 2022.

[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.

[EGP$^+$23] Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, Yifan Song, and Chenkai Weng. Superpack: Dishonest majority mpc with constant online communication. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 220–250, Cham, 2023. Springer Nature Switzerland.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589. ACM Press, July 2001.

[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.

[HVW22] Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. Your reputation's safe with me: Framing-free distributed zero-knowledge proofs. Cryptology ePrint Archive, Report 2022/1523, 2022. https://eprint.iacr.org/2022/1523.

[Kel20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1575–1590. ACM Press, November 2020.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 158–189. Springer, Heidelberg, April / May 2018.

[KRRW18] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 365–391. Springer, Heidelberg, August 2018.

[LPSY15] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 319–338. Springer, Heidelberg, August 2015.

[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.

[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.

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

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 21–37. ACM Press, October / November 2017.

[WRK17b] 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.

[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.

[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.

A Security Proofs

A.1 Proof of Theorem 1

Theorem 1. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. The protocol Π_{Prep} depicted in Figure 5 UC-realizes the functionality $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}}$ depicted in Figure 3 with information-theoretic security in the $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$-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 Π_{Prep} by showing it is a UC-secure realization of $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}}$. We describe the workflow of the simulator $\mathcal{S}$ in the ideal-world with $\mathcal{F}_{\mathsf{Prep}}$, the dummy dealer $\tilde{\mathsf{D}}$ and the dummy verifiers $\tilde{\mathsf{V}}_1, \ldots, \tilde{\mathsf{V}}_n$, and give a proof that for any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$, the simulation in the ideal-world $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}}, \mathcal{S}, \mathcal{Z}}$ is statistically indistinguishable from the real-world execution $\mathsf{EXEC}_{\Pi_{\mathsf{Prep}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}}$. Notice that, here we focus on **Singles** procedure, since **Initial** procedure is trivial and **Triples** procedure fully relies on **Singles** procedure.

When dealer is honest. In this case, we denote by $\mathcal{H}_{\mathsf{V}}$ the set of the honest verifiers in the real-world execution and $|\mathcal{H}_{\mathsf{V}}| \geq 1$. The simulator $\mathcal{S}$ needs to simulate the honest dealer and the honest verifiers. We describe the simulation strategy of $\mathcal{S}$ as follows:

1. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$ and waits for $\mathcal{A}$ to send its input. If $\mathcal{A}$ sends ABORT, $\mathcal{S}$ simply halts; otherwise, $\mathcal{S}$ receives $\{x_i, \Delta_i, \{K_{x_j}^i, M_{x_i}^j\}_{j\in[0,n]\setminus\{i\}}\}$ from $\mathcal{A}$ for each malicious $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$. Then $\mathcal{S}$ sends $\{x_i, \Delta_i, \{K_{x_j}^i, M_{x_i}^j\}_{j\in[0,n]\setminus\{i\}}\}$ to $\mathcal{F}_{\mathsf{Prep}}$ for each dummy $\tilde{\mathsf{V}}_i^*$. After that, $\mathcal{S}$ picks a random $x \leftarrow \mathbb{F}_p$ and honestly generates the rest of $[\![x]\!]_{\mathsf{BDOZ}}$ for the honest dummy parties.
2. On behalf of the honest dealer, $\mathcal{S}$ waits for each malicious $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$ to send $x_i^*, M_{x_i^*}^0$. Then $\mathcal{S}$ checks if $M_{x_i^*}^0 = K_{x_i}^0 + \Delta_0 \cdot x_i^*$ holds where $K_{x_i}^0, \Delta_0$ are the private information held by $\mathcal{S}$. If not, $\mathcal{S}$ aborts.
3. On behalf of the honest dealer, $\mathcal{S}$ sends $x_0, M_{x_0}^i$ to each $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$ privately.
4. On behalf of the honest parties, $\mathcal{S}$ honestly updates their shares, local MAC keys and MAC tags.

We prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: Real world execution $\mathsf{EXEC}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ executes the step 1 in the simulation strategy above. Perfect indistinguishability holds since $\mathcal{S}$ simply imitates the adversary's behavior by emulating $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$.
- Hybrid Hyb_2: Same as Hyb_1, except that $\mathcal{S}$ executes the step 2 in the simulation strategy above.

  Lemma 1. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. Hybrid Hyb_2 is statistically indistinguishable from Hyb_1 with adversarial advantage at most p^{-r}.*

  Proof. If a malicious $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$ is able to find a pair of $(x_i^*, M_{x_i^*}^0)$ such that $M_{x_i^*}^0 = K_{x_i}^0 + \Delta_0 \cdot x_i^*$ but $x_i^* \neq x_i$, then the adversary will find the distinction. The reason is that: in hybrid Hyb_2, the honest dealer will output x_i^*; while in the ideal world, the honest dealer will output x_i. By the properties of IT-MACs which is described in Section 2.4, we know that any malicious $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$ can forge such a valid pair $(x_i^*, M_{x_i^*}^0)$ with probability $\frac{1}{|\mathbb{F}_{p^r}|}$. Therefore, Hyb_2 is statistically indistinguishable from Hyb_1 with adversarial advantage at most p^{-r}. □

- Hybrid Hyb_3: Same as Hyb_2, except that $\mathcal{S}$ executes the step 3-4 in the simulation strategy above.

  Lemma 2. *Hybrid Hyb_3 is perfectly indistinguishable from Hyb_2.*

  Proof. Here we argue that the outputs of the honest parties in both ideal world and hybrid Hyb_3 are perfectly indistinguishable. First of all, we talk about the honest parties' shares. Since the secret x is randomly picked in both ideal world and hybrid Hyb_3, even if the adversary can choose its own share, the honest parties' shares are still uniformly random conditioned on that x is uniformly random. Secondly, since the global MAC keys Δ_i of each honest $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$ are chosen randomly in both ideal world and hybrid $\mathcal{H}_3$, they are also uniformly random. Thirdly, for each honest $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, its local MAC keys $K_{x_j}^i$ is computed as follows: if $\mathsf{V}_j \in \mathcal{H}_\mathsf{V}$, sample random $K_{x_j}^i \leftarrow \mathbb{F}_p$; otherwise, set $K_{x_j}^i := M_{x_j}^i - \Delta_j \cdot x_j \in \mathbb{F}_{p^r}$. Since Δ_j is uniformly random, in both case, honest parties' local MAC keys are perfectly indistinguishable in both ideal world and hybrid Hyb_3. Finally, since the MAC tags are deterministic conditioned on the shares, global MAC keys and local MAC keys, the honest parties' MAC tags are also perfectly indistinguishable in both ideal world and hybrid $\mathcal{H}_3$. In conclusion, Hyb_3 is perfectly indistinguishable from Hyb_3. □

Hybrid Hyb_3 is identical to the ideal world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is honest, $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{Ours}},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from $\mathsf{EXEC}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}}$ with adversarial advantage at most p^{-r}.

When dealer is malicious. In this case, we also denote by $\mathcal{H}_\mathsf{V}$ the set of the malicious verifiers in the real-world execution and $|\mathcal{H}_\mathsf{V}| \geq 1$. The simulator $\mathcal{S}$ needs to simulate the honest verifiers. We describe the simulation strategy of $\mathcal{S}$ as follows:

1. $\mathcal{S}$ emulates $\mathcal{F}_{\mathsf{Prep}}^{\mathsf{BDOZ}}$ using the same strategy as in Step 1 in the previous case.
2. On behalf of each honest verifier $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, $\mathcal{S}$ sends $x_i, M_{x_i}^0$ to malicious D^*.
3. On behalf of each honest verifier $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, $\mathcal{S}$ waits for D^* to send $x_0^*, M_{x_0^*}^i$. Then $\mathcal{S}$ checks if $M_{x_0^*}^i = K_{x_0}^i + \Delta_i \cdot x_0^*$ holds where $K_{x_0}^i, \Delta_i$ are the private information held by $\mathcal{S}$. If not, $\mathcal{S}$ aborts.

4. On behalf of the honest parties, $\mathcal{S}$ honestly updates their shares, local MAC keys and MAC tags.

We prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: Real world execution $\mathsf{EXEC}^{\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ executes the step 1 in the simulation strategy above. Perfect indistinguishability holds since $\mathcal{S}$ simply imitates the adversary's behavior by emulating $\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}$.
- Hybrid Hyb_2: Same as Hyb_1, except that $\mathcal{S}$ executes the step 2-3 in the simulation strategy above.

  Lemma 3. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. Hybrid Hyb_2 is statistically indistinguishable from Hyb_1 with adversarial advantage at most $2^{-\lambda}$.*

  Proof. If the malicious D^* is able to find a pair of $(x_0^*, M^i_{x_0^*})$ such that $M^i_{x_0^*} = K^i_{x_0} + \Delta_i \cdot x_0^*$ but $x_0^* \neq x_0$ for any $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, then the adversary will find the distinction. The reason is that: in hybrid $\mathcal{H}_2$, the malicious dealer will cause inconsistent output of the honest verifiers (i.e., make the honest V_i update its representations with x_0^* while make other honest ones update their representations with x_0); while in the ideal world, the malicious dealer cannot do that. By the properties of IT-MACs which is described in Section 2.4, we know that any malicious D^* can forge such a valid pair $(x_0^*, M^i_{x_0^*})$ with probability $\frac{1}{|\mathbb{F}_{p^r}|}$. Since the adversary can only corrupt the dealer and attempt to forge such a valid pair for n honest verifiers, the overall adversarial probability is at most $\frac{n}{|\mathbb{F}_{p^r}|} < 2^{-\lambda}$. □

- Hybrid Hyb_3: Same as Hyb_2, except that $\mathcal{S}$ executes the step 4 in the simulation strategy above. Perfect indistinguishability holds due to the similar argument as in Lemma 2.

Hybrid Hyb_3 is identical to the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}},\mathcal{S},\mathcal{Z}}$. In conclusion, when the prover is malicious, $\mathsf{EXEC}_{\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^{\mathsf{BDOZ}}_{\mathsf{Prep}}}_{\Pi_{\mathsf{Prep}},\mathcal{A},\mathcal{Z}}$ with adversarial advantage at most $2^{-\lambda}$. □

A.2 Proof of Theorem 2

Theorem 2. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. The protocol Π_{SIF} depicted in Figure 6 UC-realizes the functionality $\mathcal{F}_{\mathsf{SIF}}$ depicted in Figure 1 with information-theoretic security in the $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$-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 Π_{SIF} by showing it is a UC-secure realization of $\mathcal{F}_{\mathsf{SIF}}$. We describe the workflow of the simulator $\mathcal{S}$ in the ideal-world with $\mathcal{F}_{\mathsf{SIF}}$, the dummy dealer $\tilde{\mathsf{D}}$ and the dummy verifiers $\tilde{\mathsf{V}}_1, \ldots, \tilde{\mathsf{V}}_n$, and give a proof that for any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$, the simulation in the ideal-world $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{SIF}},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from the real-world execution $\mathsf{EXEC}^{\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}}_{\Pi_{\mathsf{SIF}},\mathcal{A},\mathcal{Z}}$.

When the dealer is honest. In this case, we denote by $\mathcal{H}_\mathsf{V}$ the set of the honest verifiers in the real-world execution and $|\mathcal{H}_\mathsf{V}| \geq 1$. The simulator $\mathcal{S}$ simulates the honest dealer and the honest verifiers, emulates $\mathcal{F}_{\mathsf{Prep}}$ for $\mathcal{A}$, and needs to simulate the view of $\mathcal{A}$ without knowing the secret input $\boldsymbol{w}$ of the dealer. We describe the strategy of $\mathcal{S}$ as follows:

1. $\mathcal{S}$ receives $(\textsc{Output}, \mathsf{sid}, y_i)$ from $\mathcal{F}_{\mathsf{SIF}}$ for each corrupted dummy verifier $\tilde{\mathsf{V}}_i^*$.
2. In the preprocessing phase: For a randomly picked $x \in \{(\mu_i)_{i\in[m]}, (a_i, b_i, c_i)_{i\in[s]}\}$, $\mathcal{S}$ emulates $\mathcal{F}^{\mathsf{Ours}}_{\mathsf{Prep}}$ for $\mathcal{A}$ and waits for $\mathcal{A}$ to send its input. If $\mathcal{A}$ sends $\textsc{Abort}$, $\mathcal{S}$ simply halts; otherwise, $\mathcal{S}$ receives x_i, Δ_i and $\{K^i_{x_j}, M^j_{x_i}\}_{j\in[n]\setminus\{i\}}$ from $\mathcal{A}$ for each $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$. After that, $\mathcal{S}$ honestly generates the rest of $[\![x]\!]$ for the honest parties.
3. In Round 1 of the online phase, for i-th input wire where $i \in [m]$, $\mathcal{S}$ picks $w_i' \leftarrow \mathbb{F}_p$ as input wire. Then $\mathcal{S}$ acts as the honest dealer to execute the round 1 protocol using the randomly picked $\{w_i'\}_{i\in[m]}$.
4. In Round 2 of the online phase, $\mathcal{S}$ acts as the honest verifiers to execute the round 2 protocol honestly, except that when the malicious $\mathsf{V}_i^* \notin \mathcal{H}_\mathsf{V}$ want to obtain its output h_i, $\mathcal{S}$ performs the followings tricks to make V_i^* believe $h_i = y_i$, where y_i is the received output of the dummy $\tilde{\mathsf{V}}_i^*$: Since $\mathcal{S}$ acts as the honest dealer

previously, $\mathcal{S}$ knows each output share $h_{i,j}$ held by each malicious verifier $\mathsf{V}_j^* \notin \mathcal{H}_\mathsf{V}$. Then $\mathcal{S}$ picks $h'_{i,j}$ for each honest verifier $\mathsf{V}_j \in \mathcal{H}_\mathsf{V}$, such that $\sum_{j \text{ s.t. } \mathsf{V}_j \notin \mathcal{H}_\mathsf{V}} h_{i,j} + \sum_{j \text{ s.t. } \mathsf{V}_j \in \mathcal{H}_\mathsf{V}} h'_{i,j} = y_i$. After that, $\mathcal{S}$ generates the new valid MAC tags $\{M^k_{h'_{i,j}}\}_{k \in [n] \setminus \{j\}}$ for each honest verifier $\mathsf{V}_j \in \mathcal{H}_\mathsf{V}$ ($\mathcal{S}$ is able to do so since $\mathcal{S}$ emulates $\mathcal{F}_\mathsf{Prep}$ and knows the global MAC keys for each party). In this way, at the time of opening the output $[h_i]$, $\mathcal{S}$ can make V_i^* believe $h_i = y_i$. Notice that, we are dealing with the rushing adversary $\mathcal{A}$, which means that the adversary $\mathcal{A}$ can delay its messages until it receives the messages from the honest parties. In other words, the adversary $\mathcal{A}$ can send its messages after $\mathcal{S}$ sending its simulated messages described above. Notice that, $\mathcal{S}$ would abort if the adversary $\mathcal{A}$ forges its MAC tags for its new maliciously generated shares that would make the honest verifier output a false result.

5. During the simulation, for each honest verifier $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$ in the real-world execution, if the adversary $\mathcal{A}$ attempts to cause it abort, $\mathcal{S}$ will send $(\textsc{Abort}, \mathsf{sid}, \tilde{\mathsf{V}}_i)$ to $\mathcal{F}_\mathsf{SIF}$ to make the dummy $\tilde{\mathsf{V}}_i$; otherwise, $\mathcal{S}$ will send $(\textsc{Continue}, \mathsf{sid}, \tilde{\mathsf{V}}_i)$ to $\mathcal{F}_\mathsf{SIF}$.

We prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: Real-world execution $\mathsf{EXEC}^{\mathcal{F}^\mathsf{Ours}_\mathsf{Prep}}_{\Pi_\mathsf{SIF}, \mathcal{A}, \mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ executes step 1-3 in the simulation above.

  Lemma 4. *Hybrid* Hyb_1 *is perfectly indistinguishable from* Hyb_0.

  Proof. Since $\mathcal{S}$ emulates $\mathcal{F}_\mathsf{Prep}$ for $\mathcal{A}$ honestly, the adversary cannot distinguish between hybrid $\mathcal{H}_1$ and the ideal world. As for the round 1 of the protocol, since μ_i is uniformly random picked, it perfectly hides the input value w_i. Therefore, the adversary $\mathcal{A}$ cannot compute w'_i to compare it with the real input value that the environment $\mathcal{Z}$ feeds to the honest dummy dealer $\tilde{\mathsf{D}}$. In a word, hybrid Hyb_1 is perfectly indistinguishable from Hyb_0. □

- Hybrid Hyb_2: Same as Hyb_1, except that $\mathcal{S}$ executes step 4-5 in the simulation above.

  Lemma 5. *Let* $\mathbb{F}_{p^r}$ *be the underlying extension field with* $p^r > n \cdot 2^\lambda$. *Hybrid* Hyb_2 *is statistically indistinguishable from* Hyb_1 *with adversarial advantage at most* p^{-r}.

  Proof. The adversary $\mathcal{A}$ will find the distinction when the simulator $\mathcal{S}$ aborts, which occurs when $\mathcal{A}$ forges its MAC tags for its new maliciously generated shares that would make any honest verifier V_i outputs h_i that is deviated from the received output y_i of the dummy $\tilde{\mathsf{V}}_i$, i.e. $h_i \neq y_i$. By the properties of IT-MACs which is described in Section 2.4, we know that the malicious verifiers cannot forge the MAC tags, unless the malicious verifiers know the global MAC keys of the honest verifiers, which occurs with probability at most p^{-r}. In conclusion, Hyb_2 is statistically indistinguishable from Hyb_1 with adversarial advantage at most p^{-r}. □

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, $\mathsf{EXEC}_{\mathcal{F}_\mathsf{SIF}, \mathcal{S}, \mathcal{Z}}$ is statistically indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^\mathsf{Ours}_\mathsf{Prep}}_{\Pi_\mathsf{SIF}, \mathcal{A}, \mathcal{Z}}$ with adversarial advantage at most p^{-r}.

When the dealer is malicious. In this case, we also denote by $\mathcal{H}_\mathsf{V}$ the set of the honest verifiers in the real-world execution and $|\mathcal{H}_\mathsf{V}| \geq 1$. The simulator $\mathcal{S}$ simulates the honest verifiers, emulates $\mathcal{F}_\mathsf{Prep}$ for $\mathcal{A}$, and needs to extract the secret input $\boldsymbol{w}$ from adversary's messages. We describe the simulation strategy of $\mathcal{S}$ as follows:

1. In the preprocessing phase: $\mathcal{S}$ prepares the correlated randomness using the same strategy as in Step 2 in the previous case.
2. In the online phase, $\mathcal{S}$ simply acts as the honest verifiers to execute the protocol Π_SIF and obtains the result h_i for each honest $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$. $\mathcal{S}$ extracts the witness $\boldsymbol{w}$ as follows. For each input value mask δ_i that $\mathcal{A}$ sends, $\mathcal{S}$ recovers the input value $w_i := \delta_i + \mu_i \in \mathbb{F}_p$; note that, $\mathcal{S}$ emulates $\mathcal{F}_\mathsf{Prep}$ for $\mathcal{A}$ previously, so $\mathcal{S}$ knows μ_i. In this way, $\mathcal{S}$ obtains the whole witness $\boldsymbol{w}$. Then $\mathcal{S}$ computes $\boldsymbol{y} := \mathcal{C}(\boldsymbol{w})$ and aborts if $y_i \neq h_i$ for any $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$. If $y_i = h_i$ holds for any $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, $\mathcal{S}$ sends $(\textsc{Prove}, \mathsf{sid}, \mathcal{C}, \boldsymbol{w})$ to $\mathcal{F}_\mathsf{SIF}$ on behalf of the corrupted dummy dealer $\tilde{\mathsf{D}}^*$.

3. During the simulation, for each honest verifier $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$ in the real-world execution, if the adversary $\mathcal{A}$ attempts to cause it abort, $\mathcal{S}$ will send $(\textsc{Abort}, \mathsf{sid}, \tilde{\mathsf{V}}_i)$ to $\mathcal{F}_\mathsf{SIF}$ to make the dummy $\tilde{\mathsf{V}}_i$; otherwise, $\mathcal{S}$ will send $(\textsc{Continue}, \mathsf{sid}, \tilde{\mathsf{V}}_i)$ to $\mathcal{F}_\mathsf{SIF}$.

We prove the indistinguishability through the following hybrids.

- Hybrid Hyb_0: Real-world execution $\mathsf{EXEC}^{\mathcal{F}^\mathsf{Ours}_\mathsf{Prep}}_{\Pi_\mathsf{SIF},\mathcal{A},\mathcal{Z}}$.
- Hybrid Hyb_1: Same as Hyb_0, except that $\mathcal{S}$ executes step 1-3 in the simulation above.

Lemma 6. *Let $\mathbb{F}_{p^r}$ be the underlying extension field with $p^r > n \cdot 2^\lambda$. Let C be the circuit with s multiplication gates. Let n be the number of verifiers. Hybrid $\mathcal{H}_2$ is statistically indistinguishable from $\mathcal{H}_1$ with adversarial advantage at most $\frac{2s+1}{n \cdot 2^\lambda}$.*

Proof. The adversary $\mathcal{A}$ will find the distinction when the simulator $\mathcal{S}$ aborts. Note that, $\mathcal{S}$ aborts when $y_i \neq h_i$ for any $\mathsf{V}_i \in \mathcal{H}_\mathsf{V}$, where h_i is the output of the honest V_i in the real-world execution while y_i is the output of the dummy honest $\tilde{\mathsf{V}}_i$ in the ideal-world execution. In the following, we will show that the probability of $\mathcal{S}$ aborting is at most $\frac{2s+1}{n \cdot 2^\lambda}$.

First of all, we prove that all the values on the wires in the circuit are correct when the verification checks pass. It is trivial that the values associated with the input wires and the output wires of the addition gates are computed correctly. Therefore, we focus the multiplication gates. Note that, when the malicious dealer D^* cheats in the i-th multiplication gate, i.e., produce false η_i and ν_i. It will be detected due to the checks performed in step 5 of Π_SIF, unless the malicious dealer *collude* with some malicious verifiers and the malicious verifiers succeed to forge new MAC tags. By the properties of IT-MACs which is described in Section 2.4, we know that the malicious verifiers cannot forge the MAC tags, unless the malicious verifiers know the global MAC keys of the honest verifiers, which occurs with probability at most $\frac{1}{|\mathbb{F}_{p^r}|}$. Since there are two new MAC tags that the malicious verifiers have to forge in each multiplication gates, and there are total s multiplication gates in the circuit, the probability of the adversary $\mathcal{A}$ cheating in the multiplication gates without being detected is $\frac{2s}{|\mathbb{F}_{p^r}|}$.

Now, we assume that all the values on the wires in the circuit are correct. If $C(\boldsymbol{w}) = \boldsymbol{y}$ but any honest V_i in the real-world execution output $h_i \neq y_i$, then the adversary $\mathcal{A}$ must corrupt some verifiers and forge their MAC tags when it is the time to open $[h_i]$ to V_i, so the adversary $\mathcal{A}$ is able to put an influence the output value h_i such that $h_i \neq y_i$. This event would occur with probability at most $\frac{1}{|\mathbb{F}_{p^r}|}$. In conclusion, hybrid Hyb_1 is statistically indistinguishable from Hyb_0 with adversarial advantage at most $\frac{2s+1}{|\mathbb{F}_{p^r}|} < \frac{2s+1}{n \cdot 2^\lambda}$. □

Hybrid Hyb_1 is the ideal-world execution $\mathsf{EXEC}_{\mathcal{F}_\mathsf{SIF},\mathcal{S},\mathcal{Z}}$. In conclusion, when the dealer is honest, $\mathsf{EXEC}_{\mathcal{F}_\mathsf{SIF},\mathcal{S},\mathcal{Z}}$ is statistically indistinguishable from $\mathsf{EXEC}^{\mathcal{F}^\mathsf{Ours}_\mathsf{Prep}}_{\Pi_\mathsf{SIF},\mathcal{A},\mathcal{Z}}$ with adversarial advantage at most $\frac{2s+1}{n \cdot 2^\lambda}$. □