# The Resiliency of MPC with Low Interaction: The Benefit of Making Errors

**(Extended Abstract)**[⋆]

Benny Applebaum[1], Eliran Kachlon[1], and Arpita Patra[2]

[1] Tel-Aviv University, Tel-Aviv, Israel {benny.applebaum,elirn.chalon}@gmail.com
[2] Indian Institute of Science, Bangalore, India arpita@iisc.ac.in

**Abstract.** We study information-theoretic secure multiparty protocols that achieve full security, including guaranteed output delivery, at the presence of an active adversary that corrupts a constant fraction of the parties. It is known that 2 rounds are insufficient for such protocols even when the adversary corrupts only two parties (Gennaro, Ishai, Kushilevitz, and Rabin; Crypto 2002), and that perfect protocols can be implemented in 3 rounds as long as the adversary corrupts less than a quarter of the parties (Applebaum , Brakerski, and Tsabary; Eurocrypt, 2019). Furthermore, it was recently shown that the quarter threshold is tight for any 3-round *perfectly-secure* protocol (Applebaum, Kachlon, and Patra; FOCS 2020). Nevertheless, one may still hope to achieve a better-than-quarter threshold at the expense of allowing some negligible correctness errors and/or statistical deviations in the security.

Our main results show that this is indeed the case. Every function can be computed by 3-round protocols with *statistical* security as long as the adversary corrupts less than third of the parties. Moreover, we show that any better resiliency threshold requires 4 rounds. Our protocol is computationally inefficient and has an exponential dependency in the circuit's depth $d$ and in the number of parties $n$. We show that this overhead can be avoided by relaxing security to computational, assuming the existence of a non-interactive commitment (NICOM). Previous 3-round computational protocols were based on stronger public-key assumptions. When instantiated with statistically-hiding NICOM, our protocol provides *everlasting statistical* security, i.e., it is secure against adversaries that are computationally unlimited *after* the protocol execution.

To prove these results, we introduce a new hybrid model that allows for 2-round protocols with linear resiliency threshold. Here too we prove that, for perfect protocols, the best achievable resiliency is $n/4$, whereas statistical protocols can achieve a threshold of $n/3$. In the plain model, we also construct the first 2-round $n/3$-statistical verifiable secret sharing that supports second-level sharing and prove a matching lower-bound, extending the results of Patra, Choudhary, Rabin, and Rangan (Crypto 2009). Overall, our results refine the differences between statistical and perfect models of security, and show that there are efficiency gaps even for thresholds that are realizable in both models.

**Keywords:** Information-Theoretic Cryptography · Cryptographic Protocols · Secure Computation · Round Complexity.

## 1 Introduction

Interaction is a valuable and expensive resource in cryptography and distributed computation. Consequently, a huge amount of research has been devoted towards characterizing the amount of interaction, typically measured via round complexity, that is needed for various distributed tasks (e.g., Byzantine agreement [44, 27, 29], coin flipping [24, 45], and zero-knowledge proofs [35, 19]) under different security models. In this

⋆ The full version of this paper can be found in [7]. The first two authors are supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under grant agreement no. 639813 ERC-CLC, and the Check Point Institute for Information Security. Arpita Patra would like to acknowledge financial support from SERB MATRICS (Theoretical Sciences) Grant 2020 and Google India AI/ML Research Award 2020.

paper, we focus on two central cryptographic goals: secure-multiparty-computation (MPC) of *general* $n$-party functionalities and *verifiable secret sharing* (VSS) [23]. We strive for full information-theoretic security, including guaranteed output delivery, at the presence of a computationally-unbounded active (aka Byzantine or malicious) rushing adversary that controls up to $t$ of the parties. In this setting, originally presented in the classical works of Ben-Or, Goldwasser, and Wigderson [17] and Chaum, Crépeau and Damgård [21], we assume that each pair of parties is connected by a secure and authenticated point-to-point channel and that all parties have access to a common broadcast channel, which allows each party to send a message to all players and ensures that the received message is identical.

The round complexity of information-theoretic MPC was extensively studied [12, 16, 28, 52, 39, 32, 34, 40, 46, 41, 43, 37, 4, 2, 31, 3, 5, 8]. For passive perfect security, it was recently showed that optimal resiliency of $t = \lfloor (n-1)/2 \rfloor$ and optimal round complexity of two can be simultaneously achieved [4, 31]. For active-security the picture is more complicated, and there seems to be a tradeoff between the number of rounds $r$ and the resiliency threshold $t$. If the adversary is allowed to corrupt a single party ($t = 1$) then 2 rounds are sufficient whenever $n \geq 4$ [37]. Any larger resiliency threshold $t > 1$ requires at least three rounds [32, 34]. For 3-round *error-free perfectly-secure* protocols, it was recently showed that a resiliency threshold of $t = \lfloor (n-1)/4 \rfloor$ is achievable [5] and that no better resiliency can be achieved [8]. The latter paper also shows that, for error-free perfectly-secure protocols, 4 rounds suffice for a threshold of $t_p = \lfloor (n-1)/3 \rfloor$ which is known to be optimal for perfect protocols regardless of their round complexity [17].

In this paper, we will be studying the other extreme point of this tradeoff. We fix a *minimal* model of communication (i.e., a round-complexity bound $r_{\min}$) for which linear resiliency is realizable, and try to characterize the best achievable *resiliency* $t$ within this model. Since 2-round protocols cannot achieve resiliency larger than 1, we ask:

> **Q1:** What is the best resiliency threshold $t$ that can be achieved by a three-round protocol with full information-theoretic active security? Can we beat the $\lfloor (n-1)/4 \rfloor$ perfect-MPC barrier by resorting to statistical security?
> **Q2:** Can we formalize a meaningful two-round model in which a linear resiliency threshold is achievable ?

We provide a complete answer to the first question and show that statistical three-round protocols can achieve $\lfloor (n-1)/3 \rfloor$ resiliency and nothing beyond that! We also answer the second question to the affirmative by presenting a new two-round hybrid model in which linear-resiliency is achievable. This model will serve as a stepping stone towards constructing three-round protocols. Along the way, we reveal new interesting differences between perfectly-secure error-free protocols to protocols that achieve perfect-secrecy but make errors with negligible probability. We continue with a detailed account of our results starting with the two-round hybrid model.

### 1.1 Two-Round Protocols in a Single-Input First-Round Hybrid Model

*Single-Input First-Round Hybrid (SIFR) Model.* We present a new *Single-Input First-Round Hybrid Model* (SIFR). In this model the communication network, which contains the usual peer-to-peer/broadcast channels, is augmented with some ideal $n$-party functionalities $\mathcal{F}$ that are restricted in two ways: (1) Every party $P_i$ is allowed to invoke the functionalities multiple times but only during the *first round*; and (2) The ideal functionalities must be *single-input functionalities*, that is, when $P_i$ invokes a functionality $\mathcal{F}_{\mathsf{si}}^i : \{0,1\}^* \to (\{0,1\}^*)^n$ the functionality delivers an output that depends only on the input of $P_i$. For example, both the authenticated-private channel functionality (that delivers a message from $P_i$ to $P_j$) and the broadcast functionality (that delivers a message from $P_i$ to all other parties) are simple instances of single-input functionalities. A more interesting example is the polynomial-VSS functionality that takes from $P_i$ a degree-$t$ polynomial $Q$ over some finite field $\mathbb{F}$, and delivers to every party $P_j$ an evaluation of $Q$ in some canonical point $\alpha_j \in \mathbb{F}$. We refer to this model as the $\mathcal{F}$-SIFR model or simply as the SIFR model when we wish to keep the oracles $\mathcal{F}$ unspecified.

We will be interested in two-round protocols in the SIFR model. In such protocols, all the first-round messages depend solely on the input of a single party and the only "mixing" (between different inputs of

different parties) occurs during the second round. Hence, two rounds are indeed essential for computing any non-trivial functionality. As an additional feature, we note that single-input functionalities can be trivially implemented with passive security via a single-round protocol, and so any two-round protocol in the SIFR model immediately translates into a two-round passively-secure protocol in the plain model.

*Limitations of Perfect protocols in SIFR Model.* To get a sense of the model, note that one can perfectly compute any degree-2 functionality over any finite field $\mathbb{F}$ of size larger than $n$ with resiliency of $t = \lfloor (n-1)/4 \rfloor$. Roughly speaking, at the first round each party uses the single-input $\mathcal{F}_{\mathsf{poly}}$ functionality to share each input via Shamir-based secret-sharing with polynomials of degree $t$; then each party locally computes the functionality over the shares (making an arbitrary number of additions and a single multiplication). At the end of this local computation, each party holds a share of the output that lies on a degree-$2t$ polynomial. At the second round, the parties broadcast the output shares and apply Reed-Solomon decoding to overcome the effect of at most $t$ adversarial corruptions.[3] In fact, it was recently showed in [8] (building on [5]) that degree-2 functionalities over any binary extension field are *complete under non-interactive reductions* either with perfect resiliency of $\lfloor (n-1)/3 \rfloor$ or with statistical resiliency of $\lfloor (n-1)/2 \rfloor$. Therefore, the above observation yields an $\lfloor (n-1)/4 \rfloor$-perfect protocol in our model for an arbitrary functionality. In the full version [7], we prove that for perfect protocols this is the best achievable threshold.

**Theorem 1 (perfect 2-round SIFR-protocols).** *General $n$-party functionalities can be perfectly-computed in two rounds in the SIFR Model with resiliency of $t$ if and only if $t \leq \lfloor (n-1)/4 \rfloor$.*

The upper-bound holds in the $\mathcal{F}_{\mathsf{poly}}$-SIFR model. The lower-bound holds relative to any (vector of) computationally-unbounded single-input functionalities and applies even when the adversary is non-rushing. In fact, the negative result shows that even the AND functionality cannot be computed in this model. As a corollary, for any $t \geq n/4$, the theorem rules out the existence of $t$-private secret sharing scheme that is *robustly-multiplicative* in the sense that parties can locally convert shares of $x$ and shares of $y$ to shares of $xy$ that are $t$-*robust*, i.e., they are recoverable even at the presence of $t$-corruptions. (This notion of multiplicative secret-sharing is stronger than the standard variants of multiplicative and strongly-multiplicative secret sharing, see [26].) The negative part of Theorem 1 is proved by turning a two-round $n/4$-perfectly secure protocol for the AND-functionality in the SIFR hybrid model into a two-party protocol in the plain model for AND with perfect security against semi-honest adversaries, contradicting the impossibility result of [22].

*Statistical protocols in $\mathcal{F}_{\mathsf{vsh}}$-SIFR Model.* We show that the $n/4$ lower-bound can be bypassed by allowing the protocol to make negligible correctness errors while preserving perfect secrecy.[4] Our protocol makes use of the bivariate version of the VSS functionality, denoted by $\mathcal{F}_{\mathsf{vsh}}$. Roughly speaking, this single-input functionality receives a symmetric bivariate polynomial $F(x,y)$ of degree less than or equal to $t$ from a dealer and sends the polynomial $f_i(x) = F(x,i)$ to every party $P_i$. (See Fig 2 in Section 3 for a formal definition.)

**Theorem 2 (statistical 2-round SIFR-protocols).** *Any $n$-party functionality $f$ of degree-2 over some finite field $\mathbb{F}$ of cardinality larger than $n$ can be computed by a two-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol with $\lfloor (n-1)/3 \rfloor$-resiliency, perfect-secrecy, statistical-correctness and complexity of* $\mathrm{poly}(S, n, \log|\mathbb{F}|, \log(1/\epsilon))$ *where $S$ is the circuit size of $f$ and $\epsilon$ is the error probability.*

*Moreover, a similar result applies to* any *functionality $f$ except that the complexity is also exponential in the depth of the Boolean circuit that computes $f$. The dependency in the depth can be avoided at the expense of downgrading security to computational and under the assumption that one-way functions exist.*

[3] The above description ignores some technical details such as output randomization which can be easily applied in the $\mathcal{F}_{\mathsf{poly}}$-SIFR model; see for example [5].

[4] Formally, this means that, in addition to standard statistical security, the output distribution of the simulator $\mathcal{S}$ in the ideal world and the output distribution of the adversary $\mathcal{A}$ in the real world are identically distributed. This additional property (which is common to all our positive results) does not seem to be very useful as a feature, but it indicates more accurately what is needed in order to bypass the lower-bounds in the perfect setting.

The "Moreover" part follows from the first part by using the aforementioned completeness of degree-2 functionalities [8, Thm. 5.23] whose overhead is exponential in the circuit's depth in the case of information-theoretic security. This makes the statistical variant of the theorem efficient only for $\mathrm{NC}^1$ functionalities.[5] Similar limitations apply to all known constant-round protocols in the information-theoretic setting even for the case of passively-secure protocols. Let us further mention that even inefficient protocols are non-trivial since security holds against a computationally-unbounded adversary.

*On the proof of Thm. 2: Round Compression via Guards.* The proof of Theorem 2 is based on several novel components. In a nutshell, following a blue-print suggested in [8], we derive a three-round protocol $\pi$ in the SIFR-hybrid model. We then exploit the special structure of the last two-rounds and show how to *compress* them into a single round. In slightly more concrete terms, at the end of the first round, some party, say Alice, holds two values $a$ and $b$ and some other party, say Bob, also has a copy of $b$. (Think of $b$ as a secret-share that was shared by Alice in the first round of $\pi$.) The purpose of the remaining rounds is to release to all parties a value $c = g(a, b)$ that depends on Alice's $a$ and Bob's $b$ while keeping $b$ private. This is done by using two additional rounds: First Alice broadcasts $a$, and then Bob computes the value $c$ based on $(a, b)$ and broadcasts the result. The key observation is that all the relevant information ($a$ and $b$) is known to Alice, and the role of Bob is to make sure that the outcome $c$ is computed properly with respect to his own copy of $b$. (Other consistency mechanisms take care of the "correctness" of $a$). We abstract this notion via a new form of *Secure Computation with a Guard* (SCG) and show that if one is willing to tolerate *statistical errors*, then any function $g$ can be realized (in the plain model) by a single-round protocol that employs correlated randomness. Furthermore, the correlated randomness can be sampled by Bob in a single preprocessing round. This allows us to collapse the last two rounds of $\pi$ into a single round (plus an additional offline preprocessing that is being handled during the first round.) Overall, our single-round SCG's allow us to compress the three-round SIFR-protocol into a two-round SIFR-protocol. The resulting protocol makes use of the $\mathcal{F}_{\mathsf{vsh}}$ functionality and an additional single-input functionality $\mathcal{F}_{\mathsf{tsh}}$ that essentially deals the shares of a random multiplicative triple $(a, b, c = ab)$. In order to remove the $\mathcal{F}_{\mathsf{tsh}}$ oracle, we first implement it in three-rounds in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model, and then compress the last round via an additional use of SCG. (See Section 3 for further details.) Our SCG constructions are based on a combination of message-authentication codes (MACs) and multiparty private-simultaneous-message protocols [28, 38] (also known as fully-decomposable randomized encoding of functions [39, 6]). (See Section 2 for details.)

### 1.2 Two-Round Verifiable Secret Sharing

Motivated by Theorem 2, our next goal is to realize the $\mathcal{F}_{\mathsf{vsh}}$ functionality in the standard model within a minimal number of rounds. The round complexity of VSS was extensively studied in the literature [32, 46, 30, 42, 43, 10, 1, 37, 49]. In the perfect setting, we have a complete answer: In order to achieve a linear resiliency $t$, one must use a two-round protocol, and within this "budget" the best achievable resiliency is $t = \lfloor (n-1)/4 \rfloor$ [32]. Patra et al. [46] were the first to suggest that this bound may be bypassed by allowing negligible statistical errors. Specifically, they view VSS as a stand-alone two-phase primitive, and showed that the sharing phase of VSS with statistical error and perfect secrecy can be realized in two rounds if and only if $t \leq \lfloor (n-1)/3 \rfloor$.

Unfortunately, the resulting protocol does not implement the polynomial-based $\mathcal{F}_{\mathsf{vsh}}$-functionality and so we cannot plug it into Theorem 2. Indeed, the existing protocol suffer from several caveats that make it less suitable for MPC applications. Specifically, after the sharing phase some of the honest parties may not hold a valid share, let alone a "second-level share". In addition, the sub-protocol needed for the "reconstruction" phase is relatively complicated and requires two rounds. In contrast, existing *perfect* VSS protocols [32, 42] realize the $\mathcal{F}_{\mathsf{vsh}}$ functionality, and correspondingly enable a trivial single-round reconstruction in which the parties broadcast their views. The possibility of an analogous statistical realization of $\mathcal{F}_{\mathsf{vsh}}$ in two rounds and resiliency threshold of $\lfloor (n-1)/3 \rfloor$ was left open by previous works. We answer this question in the affirmative. (See Section 4 for further details.)

[5] As usual in such settings, the exponential dependency in the depth can be replaced by an exponential dependency in the (non-deterministic) branching-program complexity of $f$.

**Theorem 3 (2-round statistical protocols for $\mathcal{F}_{\mathsf{vsh}}$ ).** *There exists a 2-round protocol that $\lfloor (n-1)/3 \rfloor$-securely realizes the $n$-party functionality $\mathcal{F}_{\mathsf{vsh}}$ over an arbitrary finite field $\mathbb{F}$ with perfect-secrecy and statistical-correctness. The communication complexity is polynomial in $n, \log |\mathbb{F}|$ and $\log(1/\epsilon)$ where $\epsilon$ is the error-probability. The computational complexity is polynomial in $\log |\mathbb{F}|, \log(1/\epsilon)$ and exponential in the number of parties.*

The exponential dependency in the number of parties is due to the use of a clique finding algorithm over an "agreement graph" of size $n$. While this dependency is unfortunate, the protocol is still meaningful since it provides security against unbounded adversaries. The existence of a similar protocol with polynomial dependency in $n$ is left as an interesting open question.

*Resiliency Lower-bounds.* We further strengthen the lower-bounds of [46] and show that any resiliency of $t \geq n/3$ cannot be achieved by a VSS with a two-round sharing phase even if both secrecy and correctness are statistical, and even if the adversary is non-rushing. This result applies to the more general setting where the VSS is viewed as a two-phase primitive, as opposed to an MPC functionality. (See the full version [7] for further details.) We also reveal an additional qualitative difference for the $t \geq n/3$ regime: No matter how many rounds are used in the sharing phase, the reconstruction phase cannot be implemented by letting the parties broadcast their local view. That is, even during the reconstruction some secrecy must be maintained. (See the full version [7] for further details.) Indeed, existing constructions in this regime [51, 43], employ information-theoretic MACs or signatures and keep some of the secret-key information private even during reconstruction. Our lower-bound shows that this is inherent.

### 1.3 Three-Round MPC in the Standard Model

We can now get back to the case of three-round plain-model protocols for general functionalities. Recall that in **Q1** we asked what is the best resiliency that can be achieved by 3 rounds protocols. This question was recently resolved in the perfect setting. Specifically, it was shown that 3 rounds can achieve a resiliency of $t = \lfloor (n-1)/4 \rfloor$ [5][6], and that even a slightly better resiliency threshold of $t = \lfloor (n-1)/4 \rfloor + 1$ requires at least four rounds [8].[7]

Again, we show that a small statistical error allows us to bypass the lower-bound. Specifically, by taking the two-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol from Theorem 2 and instantiating the $\mathcal{F}_{\mathsf{vsh}}$ oracle with the two-round implementation from Theorem 3, we derive a three-round statistical protocol that remains secure as long as at most $\lfloor (n-1)/3 \rfloor$ of the parties are being corrupted. We further prove a matching lower bound on the resiliency of three-round statistical protocols by showing that a 3-round protocol with $(\lfloor (n-1)/3 \rfloor + 1)$-resiliency for an authenticated-VSS functionality can be collapsed into a VSS with a 2-round sharing phase, contradicting our VSS negative results. (See full version [7] for further details.) Overall we derive the following theorem.

**Theorem 4 (3-round protocols with optimal resiliency).** *Every $n$-party functionality can be computed in three-rounds with statistical security against an active rushing computationally-unbounded adversary that corrupts at most $\lfloor (n-1)/3 \rfloor$ of the parties. The communication complexity of the protocol is polynomial in $n, 2^D$ and $S$ and the computational complexity is polynomial in $2^n, 2^D$ and $S$ where $S$ and $D$ are the size and depth of the Boolean circuit that computes $f$.*

[6] The positive result can now be obtained by combining the simple 2-round VSS-hybrid protocol for quadratic functions (Thm 1) with the 2-round perfect-VSS of [32] and with the completeness of degree-2 arithmetic functionalities [8]. The original proof from [5] was significantly more complicated since it relied on a weaker degree-2 completeness result that was applicable only over the binary field.

[7] The impossibility of three-round plain-model perfect protocols with resiliency $t \geq \lfloor (n-1)/4 \rfloor + 1$ seems to be incomparable to the impossibility of two-round perfect SIFR-model protocols (Theorem 1). One could deduce the latter result from the former with the aid of two-round protocols for single-input functionalities with perfect resiliency of $t \geq \lfloor (n-1)/4 \rfloor + 1$. However, such protocols do not exist even for the special case of the VSS functionality [32].

*Furthermore, the security threshold is tight for three-round protocols. That is, there is a finite functionality that cannot be computed in three rounds at the presence of an active (non-rushing) computationally-unbounded adversary that corrupts $\lfloor (n-1)/3 \rfloor + 1$ of the parties.*

Theorem 4 fully characterizes the feasible security threshold of three-round protocols with information-theoretic active security. As already mentioned the exponential dependency in the depth is expected, and seems to be unavoidable given the current state of the art. The exponential dependency in $n$ is derived from our VSS construction (Theorem 3), and we hope that future works will be able to improve it and get a polynomial overhead.

*Downgrading to computational security.* One way to bypass the exponential blow-up in $n$ is to replace the two-round $\lfloor (n-1)/3 \rfloor$-statistical VSS with the cryptographic VSS of [10]. The latter achieves the same $\lfloor (n-1)/3 \rfloor$-resiliency against computationally-bounded adversaries assuming the existence of a non-interactive commitment (NICOM). Specifically, by plugging this VSS into the computational part of Theorem 2, we get the following theorem. (See full version [7] for further details.)

**Theorem 5 (3-round computational MPC).** *Assuming the existence of NICOM, every $n$-party functionality $f$ admits a three-round protocol with computational security against a computationally-bounded adversary that actively corrupts up to $t \leq \lfloor (n-1)/3 \rfloor$ of the parties. The complexity is polynomial in $n$ and in the circuit's size of $f$. Moreover, if $f$ is a single-input functionality the round complexity can be reduced to 2.*

The optimality of three rounds for any $t > 1$ is owing to the two-round impossibility result of [34] that remains valid even in the cryptographic setting. For the special case of $t = 1$ and $n = 4$, [37] shows a two-round construction from any one-way function. Other existing round-optimal constructions [2, 11] work with $t < n/2$, albeit rely on public-key encryption schemes and two-round witness indistinguishable proofs (ZAPs). These assumptions are believed to be strictly stronger than NICOM that can be based on injective one-way functions [18, 55, 36] or even on general one-way functions assuming standard complexity-theoretic de-randomization assumptions [13].

We further mention that if one employs a perfectly-hiding NICOM, then our protocol achieves *everlasting security*, i.e., it is secure against adversaries that are computationally unlimited *after* the protocol execution [54]. For this result one has to invoke the statistical variant of Theorem 2, and so the protocol is efficient only for $\mathrm{NC}^1$ functionalities or general single-input functionalities. Perfectly-hiding NICOM can be based on collision-resistance hash functions at the CRS model. Even in this model, the round-complexity lower-bounds of [34] hold, and one cannot hope for two-round protocols.

The "moreover" part of the theorem covers an interesting family of "single-input" functionalities including important tasks such as distributed ZK, multiplication triple generation (modellled via $\mathcal{F}_{\mathsf{tsh}}$) and VSS. Our two-round protocol complements the incomparable result of [34] that achieves a similar round-complexity with perfect-security, but with a smaller resiliency threshold of $t < n/6$. The proof of Theorem 5 of appears in the full version [7].

### 1.4 Discussion: The benefit of errors

Since the works of Rabin and Ben-Or [51] and Beaver [15], it is known that *statistical* protocols can achieve a resiliency threshold $t_s = \lfloor (n-1)/2 \rfloor$ that is strictly larger than the best resiliency threshold $t_p = \lfloor (n-1)/3 \rfloor$ that is achievable by *perfect* protocols [50, 17]. Patra et al. [46] were the first to suggest that the statistical setting may lead to better round complexity even for thresholds of $t \leq t_p$ which are *perfectly realizable* (i.e., realizable with perfect security). Specifically, they showed that the sharing phase of statistical VSS with $t = \lfloor (n-1)/3 \rfloor$ can be carried in two rounds, bypassing a three-round lower-bound of [34]. Another indication for a possible advantage was given by [37] who showed that 4-party linear functions can be statistically computed in two rounds with threshold of $t = 1$ which is impossible in the perfect setting as shown by [33, Thm 8].[8] However, to the best of our knowledge, so far we did not have a single example of

[8] We thank Yuval Ishai for pointing this out.

an infinite MPC functionality whose statistical round complexity is strictly smaller than its perfect round complexity under a perfectly-realizable threshold $t \leq t_p$. Theorem 4 settles this question in a strong way showing that, for any $n/4 \leq t \leq \lfloor (n-1)/3 \rfloor$, statistical $t$-security can be achieved for *all* functions in three rounds, whereas perfect $t$-security cannot be achieved in three rounds even for simple finite functionalities [8].

The separation proved in the SIFR model (Thm 1 vs. Thm 2) should be taken with more care. An immediate corollary of Thm 1 asserts that for any perfect resiliency-threshold $t$ that is larger than $\lfloor (n-1)/4 \rfloor$, one cannot transform an $r$-round perfect-VSS (modeled as some ideal sharing functionality) into an $r+1$-round general MPC in a "black-box" way. Furthermore, since it is known that for $t_p = \lfloor (n-1)/3 \rfloor$ perfect VSS takes exactly 3 rounds, one can naively conclude that for such resiliency general perfectly-secure MPC cannot be implemented in less than $3+2=5$ rounds. Nevertheless, [8] constructed a 4-round perfectly-secure $t_p$-resilient MPC protocol in the plain model. This construction is based on a 3-round implementation of the $\mathcal{F}_{\mathsf{vsh}}$ functionality in a fairly complicated way that exploits the concrete properties of the underlying $\mathcal{F}_{\mathsf{vsh}}$-*protocol*. Specifically, the transformation makes use of intermediate values that are available before the $\mathcal{F}_{\mathsf{vsh}}$-protocol terminates. The impossibility of perfect two-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol for general functionalities (Thm 1) should therefore be interpreted as saying that such a complication is *inherent*! In contrast, the statistical relaxation allows us to obtain a significantly simpler reduction (i.e., two-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR) as shown in Thm 2.

We end up the introduction, by depicting in Figure 1 the resiliency-vs-round landscape of MPC in various models.

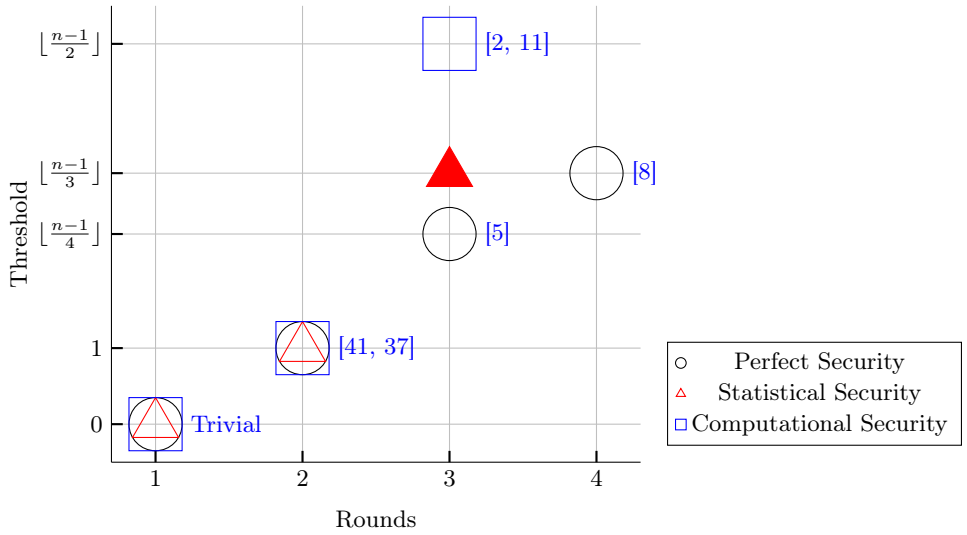


**Figure 1**: The best trade-offs known between the thresholds $t$ and the number of rounds $r$ in the plain model. Circles, triangles and squares indicate perfect, statistical and computational security, respectively. Our results are marked with solid shapes. Each of the marked points is optimal in the sense that it cannot be moved up. That is, no better resiliency can be achieved under the corresponding model with the permitted round complexity.

*Organization.* In the extended abstract version, we present a succinct version of the upper bounds. Section 2 presents the high-level idea of the Secure-Computation-with-Guard primitive, which is being employed in Section 3 towards the construction of 2-round statistical $\mathcal{F}_{\mathsf{vsh}}$-SIFR Protocols. In Section 4 we construct 2-round VSS protocols.

## 2 Secure Computation with a Guard

In this section we present a new *Secure Computation with a Guard* (SCG) primitive that will be employed later in our constructions. In SCG, there are two *senders*, Alice and Bob, with asymmetric roles: Bob knows a single input $b \in B$ whereas Alice knows both inputs $a \in A$ and $b \in B$. The goal is to release the value of $f(a, b)$ to a receiver Carol who holds no input. (This can be formalized by the 3-party functionality $F((a, b), b, \perp) = (\perp, \perp, f(a, b))$.) Syntactically, the protocol consists of an offline phase, denoted $\mathsf{scg.off}$, and an online phase, $\mathsf{scg.on}$. In the offline phase, Bob sends a single message to Carol and a single message to Alice, these messages depend only on the randomness of Bob and do not depend on his input. In the online phase, both Alice and Bob send a single message to Carol based on the offline messages and on their inputs $a, b$. At the end Carol should output the value $f(a, b)$ or a special abort symbol. As in the setting of private simultaneous message (PSM) protocols [28], we require security against an adversary that corrupts the receiver. In addition, if Alice (resp., Bob) is malicious, the receiver must abort or terminate with an output of the form $f(a', b)$ for some $a' \in A$ (resp., $f(a', b)$). In this sense, "Bob guards the computation" against corrupted Alice and "Alice guards the computation" against a corrupted Bob.

Roughly speaking, in our construction Bob samples, in the offline phase, randomness $r$ for a 2-party $f$-PSM protocol and sends it to Alice. In addition, Bob signs all the possible messages that Alice may send in the PSM protocol via an information-theoretic message authentication code, sends the tags to Alice and delivers to Carol a permuted version of all the keys. In the online phase, Bob sends the PSM message $s_B(b; r)$ that corresponds to his input $b$, whereas Alice sends the messages $s_A(a; r), s_B(b; r)$ together with their tags. Carol aborts if the tags do not match or if the $B$-part of the messages is inconsistent, otherwise it used the PSM decoder to recover the output. A naive implementation of this idea yields an overhead which is linear in the domain size, however, by using multiparty PSM, we can reduce the overhead to be poly-logarithmic in the domain-size. Overall, we prove the following theorem. (See full version [7] for a proof.)

**Lemma 1 (Polynomial-time SCG Protocols).** *Let $A = \mathbb{F}_2^{m_1}$, $B = \mathbb{F}_2^{m_2}$ and $C = \mathbb{F}_2^{p}$. Let $m = m_1 + m_2$ and let $f : A \times B \to C$ be a Boolean circuit with depth logarithmic in $m$, size polynomial in $m$ and bounded fan-in and fan-out. Then, for every statistical parameter $\epsilon$, there exists an SCG protocol with complexity* $\mathrm{poly}(m) \cdot \log(1/\epsilon)$.

## 3 A Two-Round Statistically-Secure $\mathcal{F}_{\mathsf{vsh}}$-SIFR Protocol

In this section, we prove Theorem 2. For an integer $x$, we use $\|x\|$ to denote the set $\{1, \ldots, x\}$. Let us denote the set of $n$ parties by $\mathsf{P}$.

### 3.1 Definitions

The following definitions are parameterized by a resiliency threshold $t$, a finite field $\mathbb{F}$ of size $q > n$, and a tuple of $n$ non-zero elements in $\mathbb{F}$, one for each party in $\mathsf{P}$, which are denoted (with a slight abuse of notation) by $1, \ldots, n$. Throughout this section, we fix $t$ to $\lfloor (n-1)/3 \rfloor$.

**Definition 1 ($\lfloor\cdot\rceil$-sharing).** *A value $s$ is said to be* committed *amongst $\mathsf{P}$, denoted as $\lfloor s \rceil$, if there exists a polynomial $f(x)$ of degree at most $t$ with $f(0) = s$ such that every honest party $P_i$ either holds $f(i)$ or $\perp$ and at least $t + 1$ honest parties hold non-$\perp$ values.*

**Definition 2 ($[\cdot]$-sharing).** *A value $s$ is said to be* $t$-shared *amongst $\mathsf{P}$, denoted as $[s]$, if there exists a polynomial $f(x)$ of degree at most $t$ with $f(0) = s$ such that every honest party $P_i$ holds $f(i)$.*

**Definition 3 ($[[\cdot]]$-sharing).** *A value $s$ is said to be* doubly $t$-shared *amongst $\mathsf{P}$, denoted as $[[s]]$, if there exist polynomials $f(x), \{f_i(x)\}_{i \in \{1,\ldots,n\}}$, all of degree at most $t$ with $f(0) = s$ and $f(i) = f_i(0)$ for $i \in \{1, \ldots, n\}$ such that $f(0), \{f_i(0)\}_{i \in \{1,\ldots,n\}}$ are $t$-shared via polynomials $f(x), \{f_i(x)\}_{i \in \{1,\ldots,n\}}$ and every honest $P_i$ holds $f_i(x)$.*

**Definition 4 ($\langle\cdot\rangle$-sharing).** *A value $s$ is said to be* doubly $2t$-shared *amongst* $\mathsf{P}$*, denoted as $\langle s\rangle$, if there exist a degree-$2t$ polynomial $f(x)$ and degree-$t$ polynomials $\{f_i(x)\}_{i\in\{1,\ldots,n\}}$ with $f(0)=s$ and $f(i)=f_i(0)$ for every* honest *party $P_i$ such that $\{f_i(0)\}_{i\in\{1,\ldots,n\}}$ are $t$-shared via polynomials $\{f_i(x)\}_{i\in\{1,\ldots,n\}}$ and every honest $P_i$ holds $f(i)$ and $f_i(x)$.*

**Definition 5 (First-level and Second-level sharing, Shares and Share-shares).** *In the double secret sharing definitions ($[[\cdot]]$ and $\langle\cdot\rangle$), the sharings done for the shares of the secret are referred as* second-level *sharings, while the sharing for the actual secret is termed as* first-level *sharing and the shares of the shares are termed as share-shares. The $i$th share of $s$ is denoted as $s_i$ (the context will make it clear whether the shares correspond to $t$ or $2t$ sharing). The $j$th share-share of the $i$th share $s_i$ of $s$ is denoted as $s_{ij}$.*

The sharings $[\cdot], [[\cdot]]$ and $\langle\cdot\rangle$ are linear i.e. local addition of the shares of $[a]$ and $[b]$ results in $[a+b]$ (similarly for the other types of sharing). Furthermore, addition of $\langle a\rangle$ and $[[b]]$ results in $\langle a+b\rangle$.

### 3.2 The High-level Idea

Our goal is to build a 2-round statistical protocol in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model, that can evaluate any $n$-party degree-2 functionality (over a field larger than $n$).

*Prologue.* Our starting point is the following completeness theorem from [8, Prop. 4.5 and Thm. 5.23].

**Proposition 1 ([8]).** *Let $\mathcal{F}$ be an $n$-party functionality that can be computed by a Boolean circuit of size $S$ and depth $D$ and let $\mathbb{F}$ be an arbitrary extension field of the binary field $\mathbb{F}_2$. Then, the task of securely-computing $\mathcal{F}$ non-interactively reduces to the task of securely-computing the degree-2 $n$-party functionality $f$ over $\mathbb{F}$ that each of its outputs is of the form*

$$x^{\alpha}x^{\beta}+\sum_{j=1}^{n}r^{j}, \tag{1}$$

*where $x^{\alpha}$ and $x^{\beta}$ are the inputs of party $P_{\alpha}$ and $P_{\beta}$ respectively and $r^{j}$ is an input of party $P_j$ for $j\in\{1,\ldots,n\}$.*

*The reduction preserves active perfect-security (resp., statistical-security) with resiliency threshold of $\lfloor(n-1)/3\rfloor$ (resp., $\lfloor(n-1)/2\rfloor$) and the complexity of the function $f$ and the overhead of the reduction is $\mathrm{poly}(n,S,2^D,\log|\mathbb{F}|)$. Furthermore, assuming one-way functions, one can get a similar reduction that preserves computational-security with resiliency threshold of $\lfloor(n-1)/2\rfloor$ and complexity/security-loss of $\mathrm{poly}(n,S,\log|\mathbb{F}|)$.*

Throughout this section we fix $\mathbb{F}$ to an $\mathbb{F}_2$-extension field of size larger than $n$, and assume that all the sharing functionalities are defined with respect to $\mathbb{F}$. (Specifically, we can take the smallest such field.) By Proposition 1, it suffices to focus on functionalities whose output can be written as (1). From now on, we focus on such a functionality $f$ and construct a 2-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol whose complexity is polynomial $n$ and in the description of $f$. For simplicity, we will discuss computation of one degree-2 term as above. The extension, guaranteeing that the same $x$ values are used across different degree-2 terms, will follow easily.

*2-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol.* Given access to an ideal VSS functionality, denoted $\mathcal{F}_{\mathsf{vsh}}$, that can generate a $[[\cdot]]$-sharing of a party's secret, we show how to construct a 2-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol for degree-2 computation. The $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol is efficient and statistically-secure for threshold $t<n/3$. Therefore, when the protocol is instantiated with a realisation of VSS, the security (statistical vs. cryptographic) and efficiency of the final MPC protocol reduce to that of underlying realisation of VSS.

Building on $\mathcal{F}_{\mathsf{vsh}}$, we first design a 2-round triple secret sharing (TSS) protocol in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model, that verifiably generates $[\cdot]$-sharing of a party's triple secrets $a,b,c$ satisfying the product relation $c=ab$. The TSS completes the sharing in the first round, and the verification of the product relation is done in the second round. Subsequently, we use both the VSS functionality $\mathcal{F}_{\mathsf{vsh}}$ and the TSS protocol, in order to obtain a protocol for degree-2 computation in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model, which is both efficient and statistically secure.

*Partial Degree-reduction.* Traditionally, evaluating a degree-2 function involves secret-sharing the values and multiplying them distributively. The secret sharing takes the form of $t$-sharing and the share-wise multiplication results in a non-random $2t$-sharing of the product. The latter is transformed to a $t$-sharing via degree-reduction and randomization, and lastly the $t$-shared product is reconstructed robustly to complete degree-2 function evaluation. The degree-reduction in each step of multiplication seems necessary to keep the degree inflation in check when a sequence of multiplications needs to be performed. With degree-two functions as the end goal, we ditch full-fledged round-expensive degree-reduction. Rather we settle for generating a randomized *double* $2t$-sharing of the product which enables robust reconstruction via the second-level $t$-sharings. That is, we perform one-time degree reduction for the second-level sharings alone. This idea is borrowed from [8]. As we demonstrate in this work, the degree reduction is an easier task than the degree-reduction of the first-level sharing. The key idea is to have $P_i$ monitor the degree reduction for the $i$th second-level sharings. We elaborate more below, starting with the description of a 3-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol, and then showing how to shave a round in order to obtain a 2-round protocol.

*A 3-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol.* Our aim is to compute $\langle x^\alpha x^\beta + \sum_{k=1}^{n} r^k \rangle$ and reconstruct the output via robust reconstruction of its second-level $t$-sharings. For simplicity, we ignore the additive terms and focus on producing $\langle y \rangle = \langle x^\alpha x^\beta \rangle$ and reconstructing the product. First, the $\mathcal{F}_{\mathsf{vsh}}$ functionality is used to generate $[[x^\alpha]]$ and $[[x^\beta]]$ in the first round. A local multiplication over the shares generates a non-random $2t$-sharing of the product $x^\alpha x^\beta$. Generating $\langle y \rangle$ is then done in two steps– randomization of the sharing and degree-reduction of the second-level sharing. The former requires generating a $\langle 0 \rangle$ and adding to the non-random second-level degree-reduced sharing of $y$. Generating a $\langle 0 \rangle$ requires producing $t$ $[[\cdot]]$-sharing via the $\mathcal{F}_{\mathsf{vsh}}$ functionality and can be concluded in the first round. Next, the degree reduction for the $i$th second-level sharing is conducted under the supervision of $P_i$ that produces an independent triple sharing $([a^i], [b^i], [c^i])$ which is then used to turn the $t$-sharing of the $i$th shares of $x^\alpha$ and $x^\beta$ (respectively, $x_i^\alpha$ and $x_i^\beta$) to a $t$-sharing of their product via Beaver's circuit randomization technique [14]. The TSS generates the triple sharings in the first round via $\mathcal{F}_{\mathsf{vsh}}$ and completes the verification of product relation in the second round. Having all the material ready by the first round (except the verification of the product-relation), Beaver's technique can be initiated in the second round, and it requires the reconstruction of $u_i = (x_i^\alpha - a^i)$, $v_i = (x^\beta - b^i)$ to compute $[y_i] = [x_i^\alpha x_i^\beta]$ as $u_i v_i + u_i[b^i] + v_i[a^i] + [c^i]$. Subsequently, degree-2 computation requires reconstruction of $y_i$ (the randomized version of it) which, if correct, is a share of the first-level $2t$-sharing of the product $x^\alpha x^\beta$. The above approach leads to a 3-round protocol. We compress the two sequential reconstructions, each of which typically achieved via a single round communication followed by error correction, into a single-round affair.

*Shaving a Round using 3-party secure computation with a guard (SCG).* Our approach takes note that the $j$th share-share $y_{ij}$ is expressed as $u_i v_i + u_i b_j^i + v_i a_j^i + c_j^i$ (where $a_j^i, b_j^i$ and $c_j^i$ are the $j$-th shares of $a^i, b^i$ and $c^i$), and the parties $P_i, P_j$ jointly hold all the inputs before the start of round 2. Specifically, the two values $u_i, v_i$ that can be publicly reconstructed earliest at round 2 are already available to $P_i$ in the end of round 1 (as she herself generated the triples and $\mathcal{F}_{\mathsf{vsh}}$ instances for $x^\alpha, x^\beta$ conclude in round 1). The shares of $a, b, c$, on the other hand, is known to both $P_i, P_j$ in the end of round 1 as soon as the relevant $\mathcal{F}_{\mathsf{vsh}}$ conclude. This perfectly creates a vacuum for a 3-party primitive between Alice, Bob and Carol. Alice holds inputs $\mathsf{x}, \mathsf{y}$ respectively from sets $X$ and $Y$ and Bob holds $\mathsf{y}$. Together, they would like to enable Carol to compute $f(\mathsf{x}, \mathsf{y})$ (and nothing else) with a one-shot communication. While Alice alone can do this, having Bob allows us to conduct a 'secure computation with a guard' (SCG). Between Alice and Bob, the honest party guards the computation ensuring certain level of correctness. Specifically, Carol outputs either $f(\mathsf{x}', \mathsf{y})$ or $\perp$ when Alice is corrupt, whereas $f(\mathsf{x}, \mathsf{y})$ or $\perp$ when Bob is corrupt. Using a SCG for a slightly tweaked function $g(\mathsf{x}, \mathsf{y}) = (\mathsf{x}, f(\mathsf{x}, \mathsf{y}))$ and ensuring that correct $\mathsf{x}$ is made available to Carol in round 2, it is possible to make Carol output either $f(\mathsf{x}, \mathsf{y})$ or $\perp$ in the Alice-corrupt case. We plug in an SCG, for every triple $(i, j, k)$, with $P_i$ in the shoes of Alice with inputs $\mathsf{x} = (u_i, v_i)$, $\mathsf{y} = (a_j^i, b_j^i, c_j^i)$, $P_j$ in the shoes of Bob with input $\mathsf{y}$, and the function $g$ outputting $(u_i, v_i, y_{ij})$ to $P_k$ (Carol). We reconstruct $u_i, v_i$ from their $t$-sharing in round 2 to make them available with every $P_k$, to make sure that either $f(\mathsf{x}, \mathsf{y})$ or $\perp$ are extracted from the reconstruction of the SCG of $g$. For an honest $P_i$, SCG with every honest $P_j$ will disclose $y_{ij}$, while with that of a corrupt $P_j$ is guaranteed to output either $y_{ij}$ or $\perp$. Denoting a SCG leading to $\perp$ as a *silent* one, the reconstruction

of $y_i$ for an honest $P_i$ reduces to fitting the unique $t$-degree polynomial over the disclosed $y_{ij}$. On the other hand, a corrupt $P_i$ needs to keep at least $n-t$ SCGs non-silent (which is the case for an honest $P_i$), and consequently it must agree to the inputs fed by the $n-2t$ honest parties. Furthermore, the SCG, together with the reconstructed $u_i$ and $v_i$, ensure that nothing but $y_{ij}$ or $\perp$ can make way to the output. Thus, if some value is reconstructed in this case, it will be $y_i$. Lastly, $P_i$ can cheat by not ensuring $c^i = a^i b^i$ for its triple. However, TSS offers a mechanism to detect this mischief in round 2. Therefore, the reconstruction, if at all successful, results in the correct $y_i$ for a corrupt $P_i$. Leveraging super-honest majority, we will always have enough $y_i$ ($n - t \geq 2t + 1$) for the reconstruction of $y$.

*Employing the SCGs.* Recall that apart from the single-round communication, SCGs need an offline input-independent communication round. In our protocols, the offline can be run in round 1. Furthermore, we apply the SCG's only to functions whose formula size is polynomial in $n$, and our construct is polynomial-time. SCG also plays a key role in our TSS protocol.

*Epilogue.* For the degree-2 completeness, we need every party to output different $y$ (yet with the same form). To ensure that the same $x$ inputs are used for computation of all the $y$ values, the same secret sharing of the $x$ values needs to be used for computation of $\langle y \rangle$ as above for all $y$ values. With the above high-level idea, we first present the notion of secure computation with a guard, and then use this notion to derive a 2-round statistically-secure degree-2 protocol in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model.

### 3.3 $\mathcal{F}_{\mathsf{vsh}}$-SIFR Protocol for Degree-2 Computation

For the $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol, we use an idealized version of VSS given in Fig 2, which is used in the first round of the $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol.

**Functionality $\mathcal{F}_{\mathsf{vsh}}$**

$\mathcal{F}_{\mathsf{vsh}}$ receives $F(x, y)$ from $D \in \mathsf{P}$. If $F(x, y)$ is not a symmetric bivariate polynomial of degree less than or equal to $t$ in both $x$ and $y$, then it replaces $F(x, y)$ with a default choice of such polynomial. Lastly, it sends $f_i(x) = F(x, i)$ to every $P_i$.

**Figure 2**: Functionality $\mathcal{F}_{\mathsf{vsh}}$

We further require a reconstruction protocol for $[s]$. We define two variants of reconstruction– $\mathsf{rec}$ for public reconstruction and $\mathsf{rec}_j$ for private reconstruction to party $P_j$. $\mathsf{rec}_j$ is given below and $\mathsf{rec}$ can be realized by running $n$ copies of $\mathsf{rec}_j$ for every $P_j$. Both require one round. In $\mathsf{rec}_j$, on holding $s_i$ ($i$th share of $s$), $P_i$ sends $s_i$ to $P_j$ who applies RS error correction to correct $t$ errors and reconstruct the underlying polynomial $f(x)$ and output $s = f(0)$.

We now move on to present a TSS protocol and then building on it, a degree-2 computation protocol, both in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model.

**Triple Secret Sharing** The goal of this protocol is to allow a dealer to share three values $(a, b, c)$ via VSS such that $c = ab$ holds. Given access to an ideal VSS in the first round, we achieve our goal in 2 rounds. We abstract out the need in a functionality $\mathcal{F}_{\mathsf{tsh}}$ given in Fig 3 and present our protocol subsequently.

**Functionality $\mathcal{F}_{\mathsf{tsh}}$**

$\mathcal{F}_{\mathsf{tsh}}$ receives $f^a(x), f^b(x), f^c(x)$ from $D \in \mathsf{P}$. If any of the polynomials is not a polynomial of degree less than or equal to $t$ or $f^c(0) \neq f^a(0) f^b(0)$, then it sends $\perp$ to every $P_i$ and $f^a(i), f^b(i), f^c(i)$ otherwise.

**Figure 3**: Functionality $\mathcal{F}_{\mathsf{tsh}}$

Following the idea proposed in [17] and recalled in [9], the dealer chooses two polynomials of degree at most $t$, $f^a(x)$ and $f^b(x)$ with $f^a(0) = a$ and $f^b(0) = b$. It then picks a sequence of $t$ polynomials $f^1(x), \ldots, f^t(x)$, all of degree at most $t$ such that $f^c(x)$ which is equal to $f^a(x)f^b(x) - \sum_{\alpha=1}^{t} x^\alpha f^\alpha(x)$ is a random polynomial of degree at most $t$ with the constant term equalling $ab$. Both [17, 9] elucidate the idea of choosing the coefficients of $f^1(x), \ldots, f^t(x)$ in a way that simultaneously cancels out the higher order coefficients and randomizes the remaining coefficients of the product polynomial $f^a(x)f^b(x)$. The dealer hides these $t+3$ polynomials in symmetric bivariate polynomials and invokes $t+3$ instances of $\mathcal{F}_{\mathsf{vsh}}$. At the end of the first round the sharings are returned by the $\mathcal{F}_{\mathsf{vsh}}$ functionalities, and the check for the product relation $c = ab$ is enabled by letting every party $P_i$ verify if $f^c(i) = f^a(i)f^b(i) - \sum_{\alpha=1}^{t} x^\alpha f^\alpha(i)$. Therefore, a complaint can be raised in round 2 and reconstruction of the shares of the complainant can be done in round 3 to enable public verification. To conclude the verification in round 2, we need to shave a round, or compress the rounds 2 and 3 into a single one. Given that, the complaint bit (indicating whether $P_i$'s verification succeeds or not) is known to $P_j$ and the $j$th share-shares are known to both $P_i$ and $P_j$ at the end of round 1, round 2 can be used for running (the online phase) of an SCG protocol to reveal the $j$th share-shares when the complaint bit is true. So the function of our interest is $f : \mathbb{F}_2 \times \mathbb{F}^{t+3} \to \mathbb{F}^{t+4}$ defined by

$$f(\mathsf{x}, \{\mathsf{x}^\alpha\}_{\alpha \in \|t+3\|}) = \begin{cases} (\mathsf{x}, 0, \ldots, 0) & \text{if } \mathsf{x} = 0, \\ (\mathsf{x}, \{\mathsf{x}^\alpha\}_{\alpha \in \|t+3\|}) & \text{otherwise,} \end{cases} \tag{2}$$

with $A := \mathbb{F}_2$, $B := \mathbb{F}^{t+3}$ and $C := \mathbb{F}^{t+4}$ (as per Lemma 1). To conduct $P_i$'s verification, we plug in SCGs between every triple $(i, j, k)$ varying over all $j, k$, with $P_i$ as Alice $P_j$ as Bob and $P_k$ as carol. $P_i$ and $P_j$ together allow $P_k$ to compute the $j$th share-shares of all the $t+3$ $[[\cdot]]$-sharing if $P_i$'s complaint bit is on. Precisely, $P_i$'s inputs are the complaint bit and the share-shares, whereas $P_j$'s input is just the share-shares. The offline of the SCGs are run during the first round, and the online in round 2. An SCG instance that leads to $\perp$ for a Carol, is labelled as *silent*.

For an honest $P_i$ with genuine complaint, $n - t$ SCGs corresponding to the honest $P_j$s will spit out the correct share-shares (via correctness), while the rest will either be silent or spit out correct share-shares (via correctness with a guard for honest-Alice case). This enables public reconstruction of the $i$th shares of all the $t+3$ $[[\cdot]]$-sharing and so subsequent public verification will instate the compliant publicly. Thus an honest party can always convince others about its complaint and can ensure $D$'s disqualification. A corrupt $P_i$, on the other hand, can only force the SCGs to output $f$ on either $\mathsf{x} = 0$ or 1, apart from turning them silent. A corrupt $P_i$ needs to keep at least $n - t$ SCGs non-silent (which is the case for an honest $P_i$) for not to be disqualified. Among these, it must allow every $P_k$ to receive at least $n - 2t$ correct share-shares corresponding to the SCGs with honest $P_j$s. Therefore, the reconstruction, if at all successful, results in reconstructing the correct shares, ensuring a successful public verification and absolution of $D$. Therefore, a corrupt $P_i$ cannot make a false allegation against an honest $D$. Protocol $\mathsf{tsh}$ is described in Fig. 4 and is proven to realize functionality $\mathcal{F}_{\mathsf{tsh}}$ (Lemma 2) in the full version [7]. Finally, note that the function, that is computed using SCG is a formula of constant multiplicative depth and therefore has an efficient realization.

**Protocol** tsh

**Inputs:** $D$ has inputs $(a, b, c)$ such that $c = ab$.

**Output:** By the end of **R1**, the parties output $[[a]], [[b]], [[c]]$ or discards $D$. If $D$ is not discarded, then by the end of **R2**, the parties output $[a], [b], [c]$ where $c = ab$ holds or output $\perp$.

**R1** $D$ and the parties do the following

- *(VSS calls)* $D$ chooses $t+3$ random polynomials $f^a(x), f^b(x), f^c(x), f^1(x), \ldots, f^t(x)$, each of degree $t$ such that (a) $f^a(0) = a$, $f^b(0) = b$, $f^c(0) = c$ and (b) $f^c(x) = f^a(x)f^b(x) - \sum_{\alpha=1}^{t} x^\alpha f^\alpha(x)$ as discussed in [17, 9]. $D$

picks $t+3$ symmetric bivariate polynomials $F^a(x,y), F^b(x,y), F^c(x,y), F^1(x,y), \ldots, F^t(x,y)$ with $F^a(x,0) = F^a(0,y) = f^a(x)$, $F^b(x,0) = F^b(0,y) = f^b(x)$, $F^c(x,0) = F^c(0,y) = f^c(x)$ and $F^\alpha(x,0) = F^\alpha(0,y) = f^\alpha(x)$ for every $\alpha \in \{1,\ldots,t\}$ and invokes $t+3$ instances of $\mathcal{F}_{\mathsf{vsh}}$ with these bivariate polynomials.

- *(SCG offline calls)* For every triple $(i,j,k)$, $P_i$ in the role of Alice, $P_j$ in the role of Bob, and $P_k$ in the role of Carol, run $\mathsf{scg.off}^{ijk}$, an execution of the offline phase of an SCG instance $\mathsf{scg}^{ijk}$ for function $f$ as given in Equation 2.
- *(Local Computation)* Upon conclusion of the instances of $\mathcal{F}_{\mathsf{vsh}}$, let the parties hold $t+3$ $[[\cdot]]$-sharing $\{[[z^\alpha]]\}_{\alpha \in \|t+3\|}$ ready where $z^1 = f^a(0) = a$, $z^2 = f^b(0) = b$, $z^3 = f^c(0) = c$ and $z^\alpha = f^\alpha(0)$. Every $P_i$ sets $\mathtt{flag}_i = 0$, if $z_i^3 = z_i^1 z_i^2 - \sum_{\alpha=1}^{t} i^\alpha z_i^{\alpha+3}$ and 1 otherwise.

**R2** The parties do the following.

- *(SCG online calls)* For every triple $(i,j,k)$, $P_i$, as Alice, inputs $\mathsf{x} = \mathtt{flag}_i$ and $\mathsf{x}^\alpha = z_{ij}^\alpha$ (the $j$th share-share of $z_i^\alpha$) and $P_j$, as Bob, inputs $\{\mathsf{x}^\alpha\}_{\alpha \in \|t+3\|}$ to the execution of $\mathsf{scg.on}^{ijk}$ (the matching online of $\mathsf{scg.off}^{ijk}$).
- *(Local Computation)* Every $P_k$ acts as follows. For every $i$ and $j$, execution $\mathsf{scg}^{ijk}$ is considered to be *silent* if the output of $P_k$ is $\perp$. Let for a non-silent $\mathsf{scg}^{ijk}$, $P_k$ outputs $(\mathtt{flag}_{ij}, \{z_{ij}^\alpha\}_{\alpha \in \{1,\ldots,t+3\}})$. $P_k$ discards $D$ and outputs $\perp$, if there exists a party $P_i$ such that all the following are true.
  - At least $n-t$ executions of $\{\mathsf{scg}^{ijk}\}_j$ are non-silent. Let $\mathsf{L}_i$ denote the set of all such $j$s.
  - All $\{\mathtt{flag}_{ij}\}_{j \in \mathsf{L}_i}$ from non-silent executions are 1.
  - For every $\alpha \in \{1,\ldots,t+3\}$, there is a unique polynomial of degree at most $t$ that passes through $\{z_{ij}^\alpha\}_{j \in \mathsf{L}_i}$. Let $z_i^\alpha$ denote the constant term of the polynomial. The condition $z_i^3 = z_i^1 z_i^2 - \sum_{\alpha=1}^{t} i^\alpha z_i^{\alpha+3}$ is *not* satisfied.

  Otherwise, $P_k$ outputs $[a], [b], [c]$ ignoring the second-level sharing.

**Figure 4**: Protocol $\mathsf{tsh}$

**Lemma 2 (Security).** *Protocol* $\mathsf{tsh}$ *realises functionality* $\mathcal{F}_{\mathsf{tsh}}$*, except with probability* $\epsilon$*, tolerating a static, active adversary* $\mathcal{A}$ *corrupting* $t$ *parties, possibly including the dealer* $D$*. Moreover, it is a statistically-correct and perfectly-secret protocol. Assuming the error probability of protocols* $\mathsf{scg}$ *and* $\mathsf{vsh}$*, as* $\epsilon_{\mathsf{scg}}$ *and respectively* $\epsilon_{\mathsf{vsh}}$*, we have* $\epsilon \leq t(2t+1)\epsilon_{\mathsf{scg}} + (t+3)\epsilon_{\mathsf{vsh}}$.

**Lemma 3 (Efficiency).** *In* $\mathcal{F}_{\mathsf{vsh}}$*-SIFR model, protocol* $\mathsf{tsh}$ *has a complexity of* $O\Big(\mathsf{poly}(n \log |\mathbb{F}|) \log(1/\epsilon)\Big)$.

*Simplifications in Cryptographic Setting.* Protocol $\mathsf{tsh}$ can be simplified in the cryptographic setting significantly, using the specifics of the VSS realization. In particular, the SCGs can be avoided altogether and further two rounds are enough to complete theTSS protocol. In fact, we prove a stronger statement– VSS and any single-input functionality has the same round complexity in cryptographic setting. However, the latter uses the VSS in a non-black-box way (hence does not lead to a one-round protocol in $\mathcal{F}_{\mathsf{vsh}}$-SIFR model). We postpone these details to the full version [7].

**Degree-2 Computation** Here we show how to compute a degree-2 computation of the following form: $y = x^\alpha x^\beta + \sum_{k=1}^{n} r^k$, where $x^\alpha$ and $x^\beta$ are the inputs of $P_\alpha$ and $P_\beta$ respectively and $r^k$ is an input of $P_k$ for $k \in \{1,\ldots,n\}$. This extended computation was proven to be complete for any polynomial-time computation. The goal is abstracted as a functionality $\mathcal{F}_{\mathsf{deg2c}}$ below and the protocol appears subsequently for the computation of a single $y$. We assume the output is given to everyone for simplicity. The functionality can be modified to take a random input from the rightful recipient $P_\gamma$ and $y$ can be sent out in blinded form using the randomness as the blinder. The realisation of this slightly modified functionality can obtained relying on the realisation of the below functionality and additionally asking $P_\gamma$ to run a VSS on a random polynomial (with a uniform random element $m^\gamma$ in the constant term). The value $y$ is then reconstructed in blinded form to everyone with $m^\gamma$ as the blinder, which only $P_\gamma$ can unblind. Thus, we assume $y$ be dispatched to all in $\mathcal{F}_{\mathsf{deg2c}}$.

**Functionality $\mathcal{F}_{\mathsf{deg2c}}$**

$\mathcal{F}_{\mathsf{deg2c}}$ receives $x^\alpha$ from $P_\alpha$, $x^\beta$ from $P_\beta$ and $r^k$ from $P_k \in \mathsf{P}$. It computes $y = x^\alpha x^\beta + \sum_{k=1}^{n} r^k$ and returns $y$ to every party.

**Figure 5**: Functionality $\mathcal{F}_{\mathsf{deg2c}}$

Recall that the high-level idea our protocol is to generate $\langle x^\alpha x^\beta + \sum_{\ell=1}^{n} r^\ell \rangle$ from $[[x^\alpha]], [[x^\beta]], \{[[r^\ell]]\}_{\ell \in \{1,\ldots,n\}}$ and reconstruct the secret $x^\alpha x^\beta + \sum_{\ell=1}^{n} r^\ell$ via its second-level $t$-sharings. A bunch of VSS instances are invoked to generate $[[x^\alpha]], [[x^\beta]], \{[[r^\ell]]\}_{\ell \in \{1,\ldots,n\}}$ in the first round. Ignoring the additive terms, the major task boils down to generating $\langle x^\alpha x^\beta \rangle$ from $[[x^\alpha]]$ and $[[x^\beta]]$. Local product of the shares and share-shares of these two sharings results in a non-randomized $2t$-sharing in both the first and second level. To compute a $\langle \cdot \rangle$-sharing, we (a) use the Beaver's trick to compute $t$-sharing of the product-share $x_i^\alpha x_i^\beta$ from the $t$-sharing of shares $x_i^\alpha$ and $x_i^\beta$ and then (b) randomize the first-level $2t$-degree product polynomial. For the latter, we use the existing techniques via VSS (for example see [8]). We only recall the functionality $\mathcal{F}_{\langle 0 \rangle}$ responsible for generating a $\langle 0 \rangle$, and mention that it can be implemented in one round in the $\mathcal{F}_{\mathsf{vsh}}$-SIFR model. Let $\mathsf{zsh}$ denote a one-round $\mathcal{F}_{\mathsf{vsh}}$-SIFR protocol for zero-sharing (these protocols are termed as ZSS protocols).

**Functionality $\mathcal{F}_{\langle 0 \rangle}$**

Given a set of parties $\mathsf{C} \subset \mathsf{P}$ that are controlled by ideal adversary $\mathcal{A}$, $\mathcal{F}_{\langle 0 \rangle}$ receives $\{s_i\}_{i \in \mathsf{C}}$ and $\{s_{ij}\}_{i \in \{1,\ldots,n\}; j \in \mathsf{C}}$. It picks a random polynomial of degree at most $2t$, $f(x)$, such that– (i) $f(0) = 0$ and (ii) $f(i) = s_i$ for $i \in \mathsf{C}$. It further picks a set of random polynomials $\{f_i(x)\}_{i \in \{1,\ldots,n\}}$ of degree at most $t$ such that for each $f_i(x)$– (a) $f_i(0) = f(i)$ and (ii) $f_i(j) = s_{ij}$ for all $j \in \mathsf{C}$. It sends $(f(i), f_i(x))$ to every $P_i$.

**Figure 6**: Functionality $\mathcal{F}_{\langle 0 \rangle}$

To achieve the former task, every $P_i$ generates $([a^i], [b^i], [c^i])$ such that $(a^i, b^i, c^i)$ are random and independent of the actual inputs and satisfy $c^i = a^i b^i$ using an instance of protocol $\mathsf{tsh}$. Recall that while generating the sharings takes is done in the first round, the verification of the product relation takes place in the second round. Beaver's trick requires reconstruction of $u_i = (x_i^\alpha - a^i)$, $v_i = (x_i^\beta - b^i)$ first to compute $[y_i] = [x_i^\alpha x_i^\beta]$ as $u_i v_i + u_i[b^i] + v_i[a^i] + [c^i]$ and subsequently, degree-2 computation requires reconstruction of $y_i$ (the randomized version of it) which, if correct, is a share of the first-level $2t$-sharing of the product $x^\alpha x^\beta$. Therefore, the above approach leads a 3 round protocol. To conclude within 2 rounds, we need a reconstruction mechanism that achieves– (a) for an honest $P_i$, the reconstruction is robust and $y_i$ is the correct share (b) for a corrupt $P_i$, either the reconstruction fails or $y_i$ is the correct share. This reconstruction is enabled via SCGs. For the reconstruction of $y_i$, $P_i$ in the role of Alice, $P_j$ in the role of Bob run an SCG to allow every $P_k$ learn $y_{ij}$, the $j$th share-share of $y_i$. The input of $P_i$ is $(u_i, v_i)$ and the $j$th share-shares, $a_j^i, b_j^i, c_j^i$ of $a, b, c$. The input of $P_j$ is $j$th share-shares of $a, b, c$. The function $f$ computes $y_{ij} = u_i v_i + u_i b_j^i + v_i a_j^i + c_j^i$ and outputs $(u_i, v_i, y_{ij})$. With all the inputs ready at the end of the first round, we compute the offline phase in the first round as well, while the online phase can be executed in the second round. During the online phase, we also make sure $P_k$, as Carol, holds $u_i, v_i$, by reconstructing these values from their $t$-sharing. This allows $P_k$ to make sure that $P_i$ used the values $u_i$ and $v_i$ as an input to the SCG, thus making sure that the value extracted from the SCG is either $y_{ij}$ or $\perp$, even when $P_i$ is corrupt. The protocol appears in Fig 7 and the proof that it realizes functionality $\mathcal{F}_{\mathsf{deg2c}}$ (Theorem 6) in the full version of the paper [7].

**Protocol** deg2c

**Inputs:** $P_\alpha$ and $P_\beta$ input $x^\alpha$ and respectively $x^\beta$. In addition, $P_\ell$ inputs $r^\ell$ for $\ell \in \{1, \ldots, n\}$.
**Output:** Every party outputs $y = x^\alpha x^\beta + \sum_{\ell=1}^{n} r^\ell$.

**R1** The parties do the following in parallel

- *(VSS calls)* $P_\alpha$ picks a symmetric bivariate polynomial of degree at most $t$ in each variable $X^\alpha(x,y)$ with $X^\alpha(0,0) = x^\alpha$ and initiates an instance of $\mathcal{F}_{\mathsf{vsh}}$. $P_\beta$ picks $X^\beta(x,y)$ with $X^\beta(0,0) = x^\beta$ and initiates an instance of $\mathcal{F}_{\mathsf{vsh}}$. Each $P_\ell$ picks $R^\ell(x,y)$ with $R^\ell(0,0) = r^\ell$ and initiates an instance of $\mathcal{F}_{\mathsf{vsh}}$ with input $R^\ell(x,y)$.
- *(TSS calls)* Every $P_i$ initiates an instance of $\mathsf{tsh}$, denoted as $\mathsf{tsh}^i$, with inputs $(a^i, b^i, c^i)$, randomly chosen, yet satisfying product relation $c^i = a^i b^i$.
- *(ZSS call)* The parties initiate an instance of $\mathsf{zsh}$, which is concluded by the end of the round.
- *(SCG offline calls)* For every triple $(i,j,k)$, $P_i$ in the role of Alice, $P_j$ in the role of Bob, and $P_k$ in the role of Carol execute $\mathsf{scg.off}^{ijk}$ for function $f : \mathbb{F}^2 \times \mathbb{F}^{n+4} \to \mathbb{F}^3$ defined by

$$f\Big((\mathsf{u},\mathsf{v}),(\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{o},\{\mathsf{w}^\ell\}_{\ell\in\{1,\ldots,n\}})\Big) = (\mathsf{u},\mathsf{v},\mathsf{uv}+\mathsf{ub}+\mathsf{va}+\mathsf{c}+\mathsf{o}+\sum_{\ell=1}^{n}\mathsf{w}^\ell). \tag{3}$$

- *(Local computation)* At the end of this round, we have $[[x^\alpha]]$, $[[x^\beta]]$, $\{[[r^\ell]]\}_{\ell\in\{1,\ldots,n\}}$, $\{[a^i],[b^i],[c^i]\}_{i\in\{1,\ldots,n\}}$ from $\{\mathsf{tsh}^i\}_{i\in\{1,\ldots,n\}}$, and $\langle 0\rangle$ (we denote the $i$th share as $o_i$ and $j$th share-share of $o_i$ as $o_{ij}$).

**R2** The parties do the following:

- *(TSS completion)* The parties run **R2** of $\{\mathsf{tsh}^i\}_{i\in\{1,\ldots,n\}}$.
- *(SCG online calls)* For every pair $(i,j)$, we assign the arguments of $f$ of Equation 3 as: $\mathsf{u} = (x_i^\alpha - a^i)$, $\mathsf{v} = (x_i^\beta - b^i)$, $\mathsf{a} = a^i_j$ ($j$th share of $a^i$), $\mathsf{b} = b^i_j$ ($j$th share of $b^i$), $\mathsf{c} = c^i_j$ ($j$th share of $c^i$), $\mathsf{o} = o_{ij}$ ($j$th share-share of $i$th share of 0 corresponding to the generated $\langle 0\rangle$), $\mathsf{w}^\ell = r^\ell_j$ ($j$th share of $r^\ell$). For every $(i,j,k)$ $P_i$, as Alice, inputs $(\mathsf{u},\mathsf{v})$ and $(\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{o},\{\mathsf{w}^\ell\})$, while $P_j$, as Bob, inputs $(\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{o},\{\mathsf{w}^\ell\})$ to the execution of $\mathsf{scg.on}^{ijk}$.
- *(Recovering $\mathsf{u},\mathsf{v}$ for all)* For every $P_i$, the parties run two instances of $\mathsf{rec}$ for $[x_i^\alpha - a^i]$ and $[x_i^\beta - b^i]$ to recover the values for all.
- *(Local computation)* Every $P_k$ initiates a set $\mathsf{L}$ to $\mathsf{P}$ and acts as follows. $P_k$ runs the local computation steps of $\{\mathsf{tsh}^i\}_{i\in\{1,\ldots,n\}}$. For every $(i,j,k)$, run the local computation for $\mathsf{scg.on}_{ijk}$ using the SCG output $(u_{ij}, v_{ij}, y_{ij})$ (which can be $\perp$). An $\mathsf{scg}^{ijk}$ is called *silent* if the output of $P_k$ is $\perp$. For every $P_i \in \mathsf{L}$, $P_k$ does the following.
  - Exclude $P_i$ from $\mathsf{L}$, if $P_i$ is discarded, as a dealer, in $\mathsf{tsh}^i$.
  - Exclude $P_i$ from $\mathsf{L}$ if there exists some $j$ such that $u_{ij} \neq (x_i^\alpha - a^i)$ or $v_{ij} \neq (x_i^\beta - b^i)$.
  - Exclude $P_i$ from $\mathsf{L}$, if at least $t+1$ executions of $\{\mathsf{scg}^{ijk}\}_j$ are silent. Otherwise let $\mathsf{L}_i$ denote the set of all $j$s (which is at least $n-t$) for non-silent circuits.
  - Exclude $P_i$ from $\mathsf{L}$, if the values $\{y_{ij}\}_{j\in\mathsf{L}_i}$ do not lie on a polynomial of degree $t$.
  - For every $P_i \in \mathsf{L}$, let the the constant term of the unique polynomial defined by $\{y_{ij}\}_{j\in\mathsf{L}_i}$ be $y_i$.

  Finally, $P_k$ uses $\{y_i\}_{P_i\in\mathsf{L}}$ to interpolate the $2t$-degree polynomial holding output $y$ in the constant term and outputs $y$.

**Figure 7**: Protocol $\mathsf{deg2c}$

**Theorem 6 (Security).** *Protocol* $\mathsf{deg2c}$ *realises functionality* $\mathcal{F}_{\mathsf{deg2c}}$*, except with probability* $\epsilon$*, tolerating a static adversary* $\mathcal{A}$ *corrupting* $t$ *parties. Moreover, it is a statistically-correct and perfectly-secret protocol. Assuming the error probability of protocols* $\mathsf{scg}$ *and* $\mathsf{vsh}$*, as* $\epsilon_{\mathsf{scg}}$ *and respectively* $\epsilon_{\mathsf{vsh}}$*, we have* $\epsilon \leq (nt + 5n + 2)\epsilon_{\mathsf{vsh}} + (n+1)t(2t+1)\epsilon_{\mathsf{scg}}$.

**Theorem 7 (Efficiency).** *In* $\mathcal{F}_{\mathsf{vsh}}$*-SIFR model, protocol* $\mathsf{deg2c}$ *has a complexity of* $O\Big(\mathsf{poly}(n\log|\mathbb{F}|)\log 1/\epsilon)\Big)$.

## 4 Verifiable Secret Sharing

Here, we introduce a new statistical VSS and recall the existing cryptographic VSS of [10]. In the latter section, we also suggest a simplified computational TSS protocol that is devoid of the SCGs.

In this section, the underlying field for sharing, $\mathbb{F}$, can be taken to be an arbitrary finite field of size $q > n$. We let $\kappa$ denote a statistical security parameter that guarantees a correctness error of $2^{-\Omega(\kappa)}$ (and perfect secrecy), and always take $\kappa$ to be super-logarithmic in the number of parties, i.e., $\kappa = \omega(\log n)$. We

assume without loss of generality that $\log|\mathbb{F}| > \Omega(\kappa)$, and if this is not the case, we lift $\mathbb{F}$ up to a sufficiently-large extension field. Finally, we assume that basic arithmetic operations over $\mathbb{F}$ can be implemented with polynomial complexity in the $\log|\mathbb{F}|$. As usual, we fix the resiliency $t$ to $\lfloor (n-1)/3 \rfloor$.

### 4.1 Statistical VSS

In this section, we construct the first 2-round statistical VSS that produces $[[s]]$ of $D$'s secret from $\mathbb{F}$. The existing 2-round VSS of [46, 1] does not generate $[[\cdot]]$-sharing and further the set of secrets that are allowed to be committed is $\mathbb{F} \cup \{\perp\}$. The latter implies that a corrupt $D$ has the liberty of not committing to any secret or put differently, the committed secret can be $\perp$. A natural consequence of being able to produce $[[\cdot]]$-sharing is that the reconstruction turns to a mere one-round communication of shares followed by error correction, unlike the complicated approach taken in [46, 1].

As a stepping stone towards a statistical VSS, we first build two weaker primitives– interactive signature and weak commitment.

**Interactive Signature** An interactive signature protocol is a three-phase protocol (distribute, verify and open), involving four entities– a dealer $D \in \mathsf{P}$, an intermediary $I \in \mathsf{P}$, a receiver $R \in \mathsf{P}$ and a set of verifiers $\mathsf{P}$. In the distribute phase, the dealer $D$, on holding a secret, *distributes* the secret and a signature on the secret to intermediary $I$ and private verification information to each party $P_i$ in $\mathsf{P}$. In the verify phase, $I$ and the verifiers $\mathsf{P}$ together verify if the secret and signature verify with the verification information. In the open phase, $I$ opens the message and signature to $R$ and the verifiers open verification information to $R$ who verifies and accepts the message if it verifies correctly. Intuitively, we require four properties from the primitive– (a) *privacy* of the secret till the end of execution of the three phases when $D, I, R$ are honest and at most $t$ of the verifiers are corrupt; (b) *unforgeability* of honest $D$'s secret in the open phase against the collusion of a corrupt $I$ and at most $t$ corrupt verifiers; (c) *nonrepudiation* of the secret after the verify phase succeeds against the collusion of a corrupt $D$ and at most $t$ corrupt verifiers, i.e. an honest $R$ accepts an honest $I$'s secret and signature after a successful verify phase and a corrupt $D$, colluding with $t$ verifiers cannot repudiate to not have sent the message to $I$ during distribute phase; and lastly (a) *correctness* i.e. $R$ outputs $D$'s secret when $D$ and $I$ are honest. We give the formal definition below.

**Definition 6 (Interactive Signature Scheme (ISS)).** *In an interactive signature scheme (ISS) amongst a set of $n$ parties $\mathsf{P}$, there is a distinguished party $D \in \mathsf{P}$ that holds an input $s$ picked over a field $\mathbb{F}$, referred to as a secret. The scheme involves three more entities apart from $D$, an intermediary $I \in \mathsf{P}$, a receiver $R \in \mathsf{P}$ and a set of verifiers $\mathsf{P}$ and consists of three phases, a* distribute*, a* verify *and an* open *phase. In the beginning, $D$ holds $s$ and each party including the dealer holds an independent random input.*

- *Distribute: In this phase, $D$ sends private information (computed based on its secret and randomness) to a designated intermediary $I \in \mathsf{P}$ and to each of the verifiers in $\mathsf{P}$.*
- *Verify: In this phase, $I$ and the verifiers interact to ensure that the information received from $D$ are consistent. This phase ends with a public accept or reject, indicating whether verification is successful or not.*
- *Open: Here, $I$ and each verifier in $\mathsf{P}$ send the the information received from $D$ in distribute phase to a designated receiver $R \in \mathsf{P}$ that applies a verification function to conclude if the message sent by $I$ can be accepted or not. The output of this phase is considered only upon a successful execution of verify phase.*

*A three-phase, $n$-party protocol as above is called a $(1-\epsilon)$-secure ISS scheme, if for any adversary $\mathcal{A}$ corrupting at most $t$ parties amongst $\mathsf{P}$, the following holds:*

- *Correctness: If $D$ and $I$ are honest, the verify phase will complete with a success and an honest $R$ accepts and outputs $s$ in the open phase.*
- *$\epsilon$-nonrepudiation: If $I$ and $R$ are honest and the verify phase has completed with a success, then $R$ accepts and outputs $s'$ sent by $I$ in the open phase, except with probability $\epsilon$.*

- *$\epsilon$-unforgeability: If $D$ and $R$ are honest, then $R$ accepts and outputs $s'$ sent by $I$ in the open phase only if $s' = s$, except with probability $\epsilon$.*
- *Privacy: If $D, I, R$ are honest, then at the end of the protocol the adversary's view is identical for any two secrets $s$ and $s'$. Denoting $\mathcal{D}_s$ as $\mathcal{A}$'s view during the ISS scheme when $D$'s secret is $s$, the privacy property demands $\mathcal{D}_s \equiv \mathcal{D}_{s'}$ for any $s \neq s'$.*

We would like to note that the existence of a similar primitive, known as information-checking protocol (ICP) [51, 20, 25]. ICP is played amongst three entities a dealer $D$, an intermmediary $INT$ and a receiver $R$, where the verification information is held by $R$ alone. In a variant of ICP [48, 47], $R$ is replaced with the set of parties $\mathsf{P}$, similar to our definition, but the secret and the signature are disclosed in the public. We introduce the definition above that suits best for our protocols using ISS as the building block.

We now present an ISS scheme where the three phases will require one round each and importantly the verify and open phase can be run in parallel, making the whole scheme consume only two rounds. At a very high level, $D$ hides its secret in a high-degree polynomial and gives out the polynomial as its signature to $I$. A bunch of *secret* evaluation points and evaluation of the signature polynomial on those points are given out as verification information to the verifiers. The idea of using secret evaluation points dates back to Tompa and Woll [53]. The verification is now enabled via cut-and-choose proof, though public disclosure of a padded form of the signature polynomial by $I$ and evaluations of it by the verifiers on a set of randomly selected points. The high-degree of the polynomial and the padding ensure that the privacy of the secret and signature is maintained during the verification. Lastly, the opening simply involves revealing the signature polynomial and the remaining secret evaluation points and the evaluations to the designated receiver $R$ that simply checks if the polynomial and the evaluations are consistent or not. It should be noted that a cheating $I$, exercising its rushing capability, may try to foil the cut-and-choose proof during the verify phase. Nevertheless, we show that such an adversary will be caught, with overwhelming probability, during the opening phase. We present our protocol iSig and state its properties below. For more details, see the full version of this paper [7].

**Protocol** iSig

**Inputs:** $D$ has input $s$ in the beginning of distribute phase. All parties share a statistical security parameter $1^\kappa$.
**Output:** Every party outputs `Success` or `Failure` in the end of verify phase. $R$ outputs $s'$ or $\perp$ in the end of open phase and all other parties output nothing. If $D$ is honest, then $s' = s$.

**R1 (distribute phase):** $D$ does the following.
- $D$ chooses a random polynomial $f(x)$ over $\mathbb{F}$ of degree at most $n\kappa + 1$, where $\kappa$ is the statistical security parameter, with $f(0) = s$. It further picks a random polynomial $r(x)$ over $\mathbb{F}$ of degree at most $n\kappa + 1$.
- $D$ picks $n\kappa$ random, non-zero, distinct elements from $\mathbb{F}$, denoted by $\alpha_{i1}, \ldots, \alpha_{i\kappa}$ for $i \in \|n\|$.
- $D$ sends $f(x)$ and $r(x)$ to $I$ and $\{(\alpha_{ij}, f_{i,j} = f(\alpha_{ij}), r_{ij} = r(\alpha_{ij}))\}_{j \in \|\kappa\|}$ to $P_i$.

**R2 (verify phase):** The parties do the following.
- $I$ picks a random *non-zero* value $c \in \mathbb{F}$ and broadcasts polynomial $g(x) = f(x) + cr(x)$ and $c$. Each verifier $P_i$ chooses a random subset of $\kappa/2$ indices $L_i \subset \{\kappa\}$ and broadcasts $\{(\alpha_{ij}, f_{ij}, r_{ij})\}_{j \in L_i}$.
- We say $P_i$ *accepts* $I$ if $g(\alpha_{ij}) = f_{ij} + cr_{ij}$ for all $j \in L_i$. Every $P_j$ (including $D$, $I$ and $R$) outputs `Success` if at least $2t + 1$ $P_i$ accepts and `Failure` otherwise.

**R2 (open phase):** The parties do the following.
- $I$ sends $f(x)$ to $R$. Let $\bar{L}_i := \|\kappa\| \setminus L_i$ denote the complement of $L_i$. Each verifier $P_i$ sends to $R$ the set $\{(\alpha_{ij}, f_{ij})\}_{j \in \bar{L}_i}$.
- We say $P_i$ *reaccepts* $I$ if (a) it accepted $I$ in verify phase and (ii) $f(\alpha_{ij}) = f_{ij}$ for at least $\kappa/8$ of the indices $j \in \bar{L}_i$.

  $R$ outputs $s = f(0)$ if (a) at least $t + 1$ $P_i$ reaccepts AND (b) it outputted `Success` in verify phase, and $\perp$ otherwise.

**R2 (Public open phase for non-rushing adversaries):** The parties act exactly as in the private open phase, execept that the informations are broadcasted. Every party $P_k$ reaches at the same output as $R$ would in the private open phase.

**Figure 8**: Protocol iSig

**Lemma 4.** *The Protocol* iSig *is* $(1 - 2^{-\Omega(\kappa)})$*-secure ISS tolerating a static adversary* $\mathcal{A}$ *corrupting* $t$ *parties, possibly including the dealer* $D, I$ *and* $R$*. Moreover, the protocol achieves perfect privacy, and perfect correctness, and can be implemented in time* $\text{poly}(n, \kappa, \log |\mathbb{F}|)$*.*

**Weak Commitment** As a stepping stone towards VSS, we first build a weaker primitive called weak commitment (WC) [8]. WC and opening are distributed information-theoretic variant of cryptographic commitment schemes. It also can be viewed as a (weaker) variant of the typical building block of VSS, known as Weak Secret Sharing (WSS). WC has a clean goal of ensuring that– for a unique secret $s$, at least $t+1$ honest parties must hold the shares of the secret. WSS, on the other hand, ensures that a unique secret must be committed in the sharing phase so that either the secret or $\perp$ will be reconstructed latter during the distributed reconstruction phase. It is noted that a committed secret in WC needs the help of the dealer for its opening, unlike the secret committed in WSS. With a simpler instantiation, weak commitment and opening are sufficient to build a VSS scheme.

The dealer $D$ starts with a polynomial of degree at most $t$ and generates $\lfloor\cdot\rceil$-sharing of its constant term through the input polynomial. For an honest $D$, WC in fact produces $[\cdot]$-sharing of the constant term. We abstract out the need in terms of a functionality $\mathcal{F}_{\mathsf{wcom}}$ given in Fig 9 and present the protocol realizing the functionality below. The dealer sends a polynomial $g(x)$ and a set $\mathsf{P}'$, indicating who should receive a share, to the functionality. An honest $D$ will send $g(x)$ of degree at most $t$ and $\mathsf{P}' = \mathsf{P}$. When a corrupt $D$ sends either a polynomial which is of degree more than $t$ or a set of size less than $n-t$ (denying shares to at least $t+1$ honest parties), all the parties receive $\perp$ from the functionality.

**Functionality $\mathcal{F}_{\mathsf{wcom}}$**

$\mathcal{F}_{\mathsf{wcom}}$ receives $g(x)$ and a set $\mathsf{P}'$ from $D \in \mathsf{P}$.

- If $g(x)$ has degree more than $t$ or $|\mathsf{P}'| < n - t$, it sends $\perp$ to every $P_i$.
- Else it sends $g(i)$ to every $P_i \in \mathsf{P}'$ and $\perp$ to everyone else.

**Figure 9**: Functionality $\mathcal{F}_{\mathsf{wcom}}$

At a high level, $D$, on holding a polynomial $g(x)$ of degree at most $t$, initiates the protocol by picking a *symmetric* bivariate polynomial $G(x, y)$ of degree $t$ in both variables uniformly at random over $\mathbb{F}$ such that $G(x, 0)$ and $G(0, y)$ are the same as the input polynomial $g(x)$ (with change of variable for $G(0, y)$). Following some of the existing WSS/VSS protocols based on bivariate polynomials [32, 42], $D$ sends $g_i(x) = G(x, i)$ to party $P_i$ and in parallel the parties exchange random pads to be used for pairwise consistency checking of their common shares. When a bivariate polynomial is distributed as above, a pair of parties $(P_i, P_j)$ will hold the common share $G(i, j)$ via their respective polynomials $g_i(x)$ and $g_j(x)$. Namely, $g_i(j) = g_j(i) = G(i, j)$. A pair $(P_i, P_j)$ is marked to be in conflict when the padded consistency check fails. In addition, $D$ runs an ISS protocol for every ordered pair $(i, j)$ with $P_i$ as the intermediary and $P_j$ as the receiver for secret $G(i, j)$. This allows $D$ to pass a signature on $G(i, j)$ to $P_i$ who can later use the signature to convince $P_j$ of the receipt of $G(i, j)$. $(D, P_i)$ are marked to be in conflict when one of the $n$ instances with $P_i$ as the intermediary results in failure. Now a set of non-conflicting parties, $\mathsf{W}$, of size $n-t$, including $D$, is computed (using a deterministic clique finding algorithm). Due to pair-wise consistency of the honest parties in $\mathsf{W}$, their polynomials together define a unique symmetric bivariate polynomial, say $G'(x, y)$ and an underlying degree $t$ univariate polynomial $g'(x) = G'(x, 0)$, the latter of which is taken as $D$'s committed input. For an honest $D$,

such a set exists and can be computed (in exponential time in $n$), albeit, it may exclude some honest parties. The possibility of exclusion of some of the honest parties makes this protocol different from existing 3-round constructions where $D$ gets to resolve inconsistencies in round 3 and therefore an honest party is never left out of such a set. The honest parties in $\mathsf{W}$ output the constant term of their $g_i(x)$ polynomials received from $D$ as the share of $g'(x)$. An honest outsider recomputes its $g'_i(x)$ interpolating over the non-$\perp$ outcomes from interactive signatures (as a receiver) corresponding to intermediaries residing in set $\mathsf{W}$. When $D$ is honest, the correct $g_i(x)$ can be recovered this way, thanks to the unforgeability of the signature and as a result, every honest party will hold a share of $g(x)$. For a corrupt $D$, while non-repudiation allows honest parties in $\mathsf{W}$ to convey and convince an honest outsider about their common share, the corrupt parties in $\mathsf{W}$ can inject any value as their common share. As a result, the interpolated polynomial may be an incorrect polynomial of degree more than $t$. In this case, an honest outsider may not be able to recover its polynomial $g'_i(x)$ and share of $g'(x)$. Protocol $\mathsf{swcom}$, which realizes functionality $\mathcal{F}_{\mathsf{wcom}}$ (Lemma 5) is described in Fig. 10. For more details, see the full version of this paper [7].

We point out that the error in the outputs of the honest parties in WC are totally inherited from the underlying ISS instances.

**Protocol** $\mathsf{swcom}$

**Inputs:** $D$ has input $g(x)$. All parties share a statistical security parameter $1^\kappa$.

**Output:** The parties output $[g(0)]$ if $D$ is honest and $\lfloor g'(0) \rceil$ otherwise for some $g'(x)$ of degree at most $t$. The parties output $\perp$, if $D$ is discarded.

**R1:** $D$ and every party $P_i$ do the following in parallel.
- $D$ chooses a random symmetric bivariate polynomial $G(x, y)$ of degree at most $t$ in each variable such that $G(x, 0) = g(x)$. $D$ sends to each $P_i$ the polynomial $g_i(x) = G(x, i)$.
- For every ordered pair $(P_i, P_j)$, $D$ initiates the distribute phase of one instance of $\mathsf{iSig}$, denoted as $\mathsf{iSig}_{ij}$, with $P_i$ as the intermediary, $P_j$ as the receiver and $G(i, j)$ as the secret (and with security parameter $1^\kappa$).
- Each $P_i$ picks a random polynomial $r_i(x)$ of degree at most $t$ and sends $r_{ij} = r_i(j)$ to every $P_j$.

**R2:** Each $P_i$ sets its share $s_i = g_i(0)$. For each ordered pair $(i, j)$, the parties $P_i$ and $P_j$ broadcast $m_i(x) = g_i(x) + r_i(x)$ and $m_{ij} = r_{ij} + g_j(i)$ respectively. For each ordered pair $(i, j)$, the parties execute the verify and open phases of $\mathsf{iSig}_{ij}$ and let $P_j$ outputs $g'_{ij}$ or $\perp$ in $\mathsf{iSig}_{ij}$.

**Local Computation:** A pair $(P_i, P_j)$ is called *conflicting* pair if $m_i(j) \neq m_{ij}$ or $m_j(i) \neq m_{ji}$. A pair $(D, P_i)$ is called *conflicting* pair if any of the $\mathsf{iSig}_{ij}$ instances for $j \in \|n\|$ results in `Failure`. Compute a set, $\mathsf{W}$, of $n - t$ pairwise non-conflicting parties including $D$ deterministically (a clique finding algorithm can be used). If no such set exists, then $D$ is discarded and $\mathsf{W}$ is reset to $\varnothing$. Otherwise, every $P_i \notin \mathsf{W}$ computes a polynomial $g'_i(x)$ interpolating over $\{g'_{ji}\}_{P_j \in \mathsf{W}}$. If degree of $g'_i(x)$ is more than $t$, then $P_i$ resets $s_i$ to $\perp$. Otherwise, $P_i$ resets $g_i(x) = g'_i(x)$ and $s_i = g'_i(0)$.

**Figure 10**: Protocol $\mathsf{swcom}$

**Lemma 5.** *Protocol* $\mathsf{swcom}$ *realises functionality* $\mathcal{F}_{\mathsf{wcom}}$*, except with probability* $\epsilon$*, tolerating a static adversary* $\mathcal{A}$ *corrupting* $t$ *parties, possibly including the dealer* $D$*. Moreover, it is a statistically-correct and perfectly-secret protocol. Assuming the error probability of protocol* $\mathsf{iSig}$ *as* $\epsilon_{\mathsf{iSig}}$*, we have* $\epsilon \leq (2t+1)^2\epsilon_{\mathsf{iSig}} = O(n^2\epsilon_{\mathsf{iSig}})$*. The communication complexity is* $\mathrm{poly}(n, \kappa, \log|\mathbb{F}|)$*, and the computational complexity is exponential in* $n$ *and polynomial in* $\kappa$ *and* $\log|\mathbb{F}|$*.*

While we never need to reconstruct a $\lfloor\cdot\rceil$-shared secret, non-robust reconstruction can be enabled by allowing $D$ to broadcast the committed polynomial and the parties their shares. The $D$'s polynomial is taken as the committed one if $n - t$ parties' share match with it. Clearly an honest $D$'s opened polynomial will be accepted and a non-committed polynomial will always get rejected.

**The Statistical VSS** VSS allows a dealer to distributedly commit to a secret in a way that the committed secret can be recovered robustly in a reconstruction phase. Our VSS protocol $\mathsf{vsh}$ allows a dealer $D$ to generate

double $t$-sharing of the constant term of $D$'s input bivariate polynomial $F(x, y)$ of degree at most $t$ and therefore allows robust reconstruction via Read-Solomon (RS) error correction, unlike the weak commitment scheme swcom.

At a high level, protocol svsh proceeds in the same way as the weak commitment scheme wcom, except that each blinder polynomial is now committed via an instance of swcom. A happy set, V, is formed in the same way. Two conflicting honest parties cannot belong to V, implying all the honest parties in V are pairwise consistent and together define a unique symmetric bivariate polynomial, say $F'(x, y)$ and an underlying degree $t$ univariate polynomial $f'(x) = F'(x, 0)$, the latter of which is taken as $D$'s committed input. A crucial feature that vsh offers by enforcing the W set of every party in V to have an intersection of size at least $n - t$ with V, is that the blinded polynomial of a corrupt party from V is consistent with $F'(x, y)$. This follows from the fact that the shares (pads) that the parties in W receive as a part of wcom remain unchanged, implying $n - 2t \geq t + 1$ of the honest parties in V ensure the consistency of the blinded polynomial of the corrupt party. This feature crucially enables an honest party $P_i$ that lies outside V (in case of a corrupt dealer) to extract out her polynomial $f'_i(x) = F'(x, i)$ and thereby completing the double $t$-sharing of $f'(0)$. To reconstruct $f'_i(x)$, $P_i$ looks at the blinded polynomial of all the parties in V who kept her happy in their respective weak commitment instances (implying her share did not change). For each such party, the blinded polynomial evaluated at $i$ and subtracted from $P_i$'s share/pad from the underlying wcom instance, allows $P_i$ to recover one value on $f'_i(x)$. All the honest parties in V (which is at least $t + 1$) contribute to one value each, making sure $P_i$ has enough values to reconstruct $f'_i(x)$. A corrupt party in V, being committed to the correct polynomial as per $F'(x, y)$, with respect to the parties in its W set, cannot inject a wrong value. Protocol vsh, which realizes functionality $\mathcal{F}_{\mathsf{vsh}}$ (Theorem 8), is now described in Fig. 11. See the full version of this paper [7] for more details.

We point out that the error in the outputs of the honest parties in VSS are totally inherited from the underlying WC and in turn the ISS instances.

**Protocol** svsh

**Inputs:** $D$ has input $F(x, y)$, a symmetric bivariate polynomial of degree at most $t$.

**Output:** The parties output $[[F(0, 0)]]$ when $D$ is honest and $[[F'(0, 0)]]$ otherwise where $F'(x, y)$ is a bivariate polynomial of degree at most $t$.

**R1** $D$ and every party $P_i$ do the following in parallel.
- $D$ sends to each $P_i$ the polynomial $f_i(x) = F(x, i)$.
- Each party $P_i$ picks a random polynomial $h_i(x)$ of degree at most $t$ and initiates an instance of swcom, denoted as $\mathsf{swcom}_i$ as a dealer with polynomial $h_i(x)$.

**R2** For each ordered pair $(i, j)$, $P_i$ and $P_j$ broadcast $p_i(x) = f_i(x) + h_i(x)$ and $p_{ij} = h_{ij} + f_j(i)$ respectively, where $h_{ij}$ is the share of $P_j$ in $\mathsf{swcom}_i$. In parallel, parties execute **R2** of $\mathsf{swcom}_i$ for all $i \in \{1, \ldots, n\}$.

**Local Computation** The parties execute local computation step for every $\mathsf{swcom}_i$ for $i \in \{1, \ldots, n\}$. A pair $(P_i, P_j)$ is called *conflicting* pair if $p_i(j) \neq p_{ij}$ or $p_j(i) \neq p_{ji}$. Compute a set, V, of $n - t$ pairwise non-conflicting parties including $D$ deterministically such that $|\mathsf{V} \cap \mathsf{W}_i| \geq n - t$ for every $P_i \in \mathsf{V}$, where $\mathsf{W}_i$ denote the set of non-conflicted parties in $\mathsf{swcom}_i$ (a clique finding algorithm can be used). If no such set exists, then $D$ is discarded and a default sharing is assumed and reset V to P. Otherwise, every $P_i \notin \mathsf{V}$ resets polynomial $f_i(x)$ to the degree $t$ polynomial interpolated over the values $\{p_j(i) - h_{ji}\}_{P_j \in \mathsf{V}; P_i \in \mathsf{W}_j}$ (where $p_j(x)$ was broadcasted by $P_j$ in **R2** and $P_i$ has its share $h_{ji}$ from $\mathsf{swcom}_j$). Finally, every $P_i$ outputs $f_i(0)$ and $f_i(x)$.

**Figure 11**: Protocol svsh

**Theorem 8.** *Protocol* svsh *realises functionality* $\mathcal{F}_{\mathsf{vsh}}$*, except with probability* $\epsilon$*, tolerating a static adversary* $\mathcal{A}$ *corrupting* $t$ *parties, possibly including the dealer* $D$*. Moreover, it is a statistically-correct and perfectly-secret protocol. Assuming the error probability of protocol* iSig *as* $\epsilon_{\mathsf{iSig}}$*, we have* $\epsilon \leq n(2t + 1)^2\epsilon_{\mathsf{iSig}}$*. The communication complexity is* $\mathrm{poly}(n, \kappa, \log |\mathbb{F}|)$*, and the computational complexity is exponential in* $n$ *and polynomial in* $\kappa$ *and* $\log |\mathbb{F}|$*.*

It is easy to note that svsh generates $[[F(0,0)]]$ via the set of polynomials $\left\{F(x,0), \{f_i(x)\}_{i\in\{1,\ldots,n\}}\right\}$. Plugging in the above VSS in the deg2c protocol, we get a 3-round MPC for degree-2 computation.

### 4.2 Cryptographic VSS and Computation of any single-input functions

We briefly recall the construction of [10]. In Round 1, $D$ publicly commits to a symmetric bivariate polynomial $F(x,y)$ using a NICOM and delivers the opening corresponding to $f_i(x) = F(x,i)$ to $P_i$. The commitments are computed in a way that simple public verification suffice for the checking of pairwise consistency between the common points (such as $f_i(j)$ and $f_j(i)$). To ensure that the commitments correspond to a polynomial of degree at most $t$ in both $x$ and $y$, it suffices if the honest parties (which are $n-t$ in number) confirm that their received polynomials are consistent with their commitments and they are of degree at most $t$. If this is not true, then $P_i$'s goal is to make $D$ publicly reveal the polynomial consistent with the commitments in the second round. Towards realizing the goal, $P_i$ commits to a pad publicly and send the opening to $D$ alone during Round 1. If $D$ finds the opening inconsistent to the public commitment, then it turns unhappy towards $P_i$ and opens the commitments corresponding to $f_i(x)$ publicly. Otherwise, it blinds the opening of $f_i(x)$ using the pad and makes it public. When $P_i$'s check about $f_i(x)$ fails, she similarly turns unhappy with $D$ and opens the pad which in turn unmask the opening for $f_i(x)$. A corrupt $P_i$ cannot change the pad and dismiss an honest $D$, owing to the binding property of NICOM. A corrupt $D$ however may choose not to hand $P_i$ the correct $f_i(x)$ in Round 1 and reveal $f_i(x)$ correctly in Round 2. The above technique therefore makes the $f_i(x)$ that is consistent with the public commitment of $D$ publicly known when $D$ and $P_i$ are in conflict (and $P_i$ is honest).

In the cryptographic setting, the VSS of [10] has the special feature of making the share (the entire univariate polynomial) of a party public when they are in conflict. We can tweak the TSS protocol (Section 3.3) so that the shares for all the $t+3$ instances are made public for party $P_i$ in round 2, if $P_i$ is in conflict with $D$ (which also includes the reason that $P_i$'s share do not satisfy the relation). This allows the public verification for corrupt parties in round 2 itself and thus TSS concludes in 2 rounds, like the VSS. We, in fact, can prove a stronger version of the result– any single-input function takes 2 rounds in cryptographic setting. TSS is a special case. We present this general result below. Plugging in the above VSS,and TSS in the deg2c protocol, we get a 3-round MPC for degree-2 computation.

**Cryptographic MPC for single-input functions** In this section, we obtain a 2-round protocol for every function whose outputs are determined by the input of a single party (single-input functions). This class of functions include important tasks such as distributed ZK and VSS. While a VSS protocol will be implied from our result from this section, we have separated out VSS in the previous section, as the VSS of [10] is used in a non-blackbox way for MPC for single-input functions.

[34] reduces secure computation of a single-input function to that of degree-2 polynomials and subsequently show a 2-round construct to evaluate the latter with perfect security and threshold $t < n/6$. In this work, we complement their reduction with a 2-round protocol to evaluate a degree-2 polynomial with threshold $t < n/3$ and relying on NICOMs. Let the sole input-owner be denoted as $D \in \mathsf{P}$, the inputs be $x^1, \ldots, x^m$ and the degree-2 polynomial be $p$ (in most general form, there can be a vector of such polynomials). Broadly, the goal is to compute $2t$-sharing of $p(x^1, \ldots, x^m)$ and reconstruct the secret relying on the guidance of $D$ in 2 rounds. The protocol starts with $D$ sharing all the inputs using $m$ instances of VSS. For the guided reconstruction, $D$ locally computes the shares $p(x_i^1, \ldots, x_i^m)$ ($p$ applied on the $i$th shares of the inputs) of the degree $2t$ polynomial holding $p(x^1, \ldots, x^m)$ in the constant term and broadcasts all the $n$ points. In Round 2, apart from the checks $P_i$ conducts inside the VSS instances, it also verifies if the broadcast of $D$ is consistent with her received polynomials. If the check fails, then it becomes unhappy with $D$ in all the instances and opens the pads distributed in the VSS instances to expose all the polynomials in her share. This allows public reconstruction of the correct $p(x_i^1, \ldots, x_i^m)$. The reconstruction in Round 2 is then achieved simply by fitting a degree $2t$ polynomial over the values $p(x_i^1, \ldots, x_i^m)$– (i) if $P_i$ is not in conflict with $D$, this value is taken from $D$'s broadcast (ii) otherwise, this value is publicly recomputed as explained. If there is no such $2t$ degree polynomial, then $D$ is concluded to be corrupt and is discarded. An

honest $D$ will always broadcast the correct values $p(x_i^1, \ldots, x_i^m)$ that lie on a $2t$ degree polynomial and a corrupt unhappy $P_i$ cannot open a different value than this (due to binding property of NICOM). Lastly, since these values correspond to a non-random $2t$ degree polynomial, they are randomized using a $\langle 0 \rangle$ before broadcast. The $\langle 0 \rangle$ sharing is created by $D$ by running $t$ additional instances of VSS.

We present the functionality and the protocol below, the security proof of the latter (Lemma 6) is deferred to the full version of this paper [7]. We assume the output is given to everyone for simplicity. For a function that outputs distinct values for the parties, say $y^i$ to $P_i$, the functionality can be modified to deliver $y^i$ to $P_i$. This can be implemented by $D$ $t$-sharing ($[\cdot]$-sharing) a random pad, $pad^i$ for every party $P_i$, where the bivariate polynomial (used for sharing) and all the commitment opening are disclosed to $P_i$, who becomes unhappy when there is any inconsistency. $D$ broadcasts masked values $p(x_j^1, \ldots, x_j^m) + pad_j^i$ so that $y^i + pad^i$ gets publicly reconstructed and $y^i$ gets privately reconstructed by $P_i$ alone.

**Functionality $\mathcal{F}_{\mathsf{sif}}$**

$\mathcal{F}_{\mathsf{sif}}$ receives $x^1, \ldots, x^m$ from $D$, computes $y = p(x^1, \ldots, x^m)$ and returns $y$ to every party, where $p$ is a degree-2 polynomial in the inputs of $D$.

**Figure 12**: Functionality $\mathcal{F}_{\mathsf{sif}}$

**Protocol sif**

**Inputs:** $D$ has input $x^1, \ldots, x^m$.
**Output:** The parties output $p(x^1, \ldots, x^m)$.

**R1** $D$ picks a symmetric and random bivariate polynomial $F^j(x, y)$ with $F^j(0, 0) = x^j$ and initiates an instance of cvsh for $j \in \{1, \ldots, m\}$. It additionally picks a symmetric and random bivariate polynomial $M^j(x, y)$ and initiates an instance of cvsh for $j \in \{1, \ldots, t\}$ (used for randomization). Let $D$ sends $\{f_i^j(x), m_i^j(x)\}_{j \in \{1,\ldots,t\}}$ to $P_i$ in these cvsh instances. $D$ further broadcasts $y_i = p(f_i^1(0), \ldots, f_i^m(0)) + \sum_{j=1}^{t} i^j m_i^j(0)$. All the parties participate in these instances and perform their respective steps.

**R2** Run **R2** of all the instances. Further $P_i$ checks if the value $y_i$ broadcasted by $D$ is consistent with the received polynomials. If this check fails, it becomes *unhappy* with $D$ in all the VSS instances and opens the pads to publicly reconstruct $\{f_i^j(x), m_i^j(x)\}_{j \in \{1,\ldots,t\}}$ as per cvsh protocol. Every party recomputes $y_i$ for every $P_i$ in conflict with $D$. Let $\mathsf{V}$ be the set of parties who do not have conflict with $D$. Every party checks if $\{y_i\}_{i \in \{1,\ldots,n\}}$ lie on a $2t$ degree polynomial, where $y_i$ is broadcasted by $D$ when $P_i$ is not in conflict with $D$ and $y_i$ is the publicly recomputed value otherwise. In case of yes, then every party outputs the constant term of the polynomial. Otherwise, $D$ is discarded and $p$ evaluated on default inputs is taken as output.

**Figure 13**: Protocol sif

**Lemma 6.** *Protocol* sif *realizes* $\mathcal{F}_{\mathsf{sif}}$ *tolerating a static adversary* $\mathcal{A}$ *corrupting* $t$ *parties, relying on NICOM.*

## References


1. Agrawal, S.: Verifiable secret sharing in a total of three rounds. Inf. Process. Lett. **112**(22), 856–859 (2012). https://doi.org/10.1016/j.ipl.2012.08.003, https://doi.org/10.1016/j.ipl.2012.08.003
2. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Round-optimal secure multiparty computation with honest majority. In: Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. pp. 395–424 (2018)
3. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Two round information-theoretic MPC with malicious security. In: Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II. pp. 532–561 (2019)

4. Applebaum, B., Brakerski, Z., Tsabary, R.: Perfect secure computation in two rounds. In: Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I. pp. 152–174 (2018)
5. Applebaum, B., Brakerski, Z., Tsabary, R.: Degree 2 is complete for the round-complexity of malicious MPC. In: Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II. pp. 504–531 (2019)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in $\mathsf{NC}^0$. SIAM Journal on Computing **36**(4), 845–888 (2006)
7. Applebaum, B., Kachlon, E., Patra, A.: The resiliency of mpc with low interaction: The benefit of making errors (2020), available at the authors' homepage
8. Applebaum, B., Kachlon, E., Patra, A.: The round complexity of perfect mpc with active security and optimal resiliency. To appear in Proc. of 61st FOCS (2020), available at https://eprint.iacr.org/2020/581
9. Asharov, G., Lindell, Y.: A full proof of the BGW protocol for perfectly secure multiparty computation. J. Cryptology **30**(1), 58–151 (2017)
10. Backes, M., Kate, A., Patra, A.: Computational verifiable secret sharing revisited. In: Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 590–609 (2011)
11. Badrinarayanan, S., Jain, A., Manohar, N., Sahai, A.: Secure MPC: laziness leads to GOD. IACR Cryptology ePrint Archive **2018**, 580 (2018), https://eprint.iacr.org/2018/580
12. Bar-Ilan, J., Beaver, D.: Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In: Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989. pp. 201–209 (1989)
13. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. **37**(2), 380–400 (2007). https://doi.org/10.1137/050641958, https://doi.org/10.1137/050641958
14. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15. Lecture Notes in Computer Science, vol. 576, pp. 420–432. Springer Verlag (1991)
15. Beaver, D.: Multiparty protocols tolerating half faulty processors. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 560–572. Springer (1989). https://doi.org/10.1007/0-387-34805-0_49, https://doi.org/10.1007/0-387-34805-0_49
16. Beaver, D., Feigenbaum, J., Kilian, J., Rogaway, P.: Security with low communication overhead. In: Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. pp. 62–76 (1990)
17. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10 (1988)
18. Blum, M.: Coin flipping by telephone. In: Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981. pp. 11–15 (1981)
19. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires omega~(log n) rounds. In: Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece. pp. 570–579 (2001)
20. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA. pp. 42–51 (1993)
21. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 11–19 (1988)
22. Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. p. 62–72. STOC '89, Association for Computing Machinery, New York, NY, USA (1989). https://doi.org/10.1145/73007.73013, https://doi.org/10.1145/73007.73013
23. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985. pp. 383–395 (1985)

24. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA. pp. 364–369 (1986)
25. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. pp. 311–326 (1999)
26. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. pp. 316–334 (2000)
27. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. J. ACM **32**(1), 191–204 (1985)
28. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada. pp. 554–563 (1994)
29. Feldman, P., Micali, S.: Byzantine agreement in constant expected time (and trusting no one). In: 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985. pp. 267–276 (1985)
30. Fitzi, M., Garay, J.A., Gollakota, S., Rangan, C.P., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings. pp. 329–342 (2006)
31. Garg, S., Ishai, Y., Srinivasan, A.: Two-round MPC: information-theoretic and black-box. In: Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I. pp. 123–151 (2018)
32. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece. pp. 580–589 (2001)
33. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing. pp. 580–589. ACM (2001)
34. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. pp. 178–193 (2002)
35. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. **25**(1), 169–192 (1996)
36. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA. pp. 25–32 (1989)
37. Ishai, Y., Kumaresan, R., Kushilevitz, E., Paskin-Cherniavsky, A.: Secure computation with minimal interaction, revisited. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II. pp. 359–378 (2015)
38. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings. pp. 174–184 (1997)
39. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA. pp. 294–304 (2000)
40. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings. pp. 244–256 (2002)
41. Ishai, Y., Kushilevitz, E., Paskin, A.: Secure multiparty computation with minimal interaction. In: Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. pp. 577–594 (2010)
42. Katz, J., Koo, C., Kumaresan, R.: Improving the round complexity of VSS in point-to-point networks. Inf. Comput. **207**(8), 889–899 (2009)
43. Kumaresan, R., Patra, A., Rangan, C.P.: The round complexity of verifiable secret sharing: The statistical case. In: Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. pp. 431–447 (2010)

44. Lamport, L., Fischer, M.: Byzantine generals and transaction commit protocols. Tech. rep., Technical Report 62, SRI International (1982)
45. Moran, T., Naor, M., Segev, G.: An optimally fair coin toss. J. Cryptology **29**(3), 491–513 (2016). https://doi.org/10.1007/s00145-015-9199-z, https://doi.org/10.1007/s00145-015-9199-z
46. Patra, A., Choudhary, A., Rabin, T., Rangan, C.P.: The round complexity of verifiable secret sharing revisited. In: Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 487–504 (2009)
47. Patra, A., Choudhary, A., Rangan, C.P.: Simple and efficient asynchronous byzantine agreement with optimal resilience. In: Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, 2009. pp. 92–101 (2009)
48. Patra, A., Rangan, C.P.: Communication and round efficient information checking protocol. CoRR **abs/1004.3504** (2010), http://arxiv.org/abs/1004.3504
49. Patra, A., Ravi, D.: On the power of hybrid networks in multi-party computation. IEEE Trans. Inf. Theory **64**(6), 4207–4227 (2018)
50. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
51. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA. pp. 73–85 (1989)
52. Sander, T., Young, A.L., Yung, M.: Non-interactive cryptocomputing for NC1. In: 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA. pp. 554–567 (1999)
53. Tompa, M., Woll, H.: How to share a secret with cheaters. In: Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. pp. 261–265 (1986)
54. Unruh, D.: Everlasting multi-party computation. J. Cryptology **31**(4), 965–1011 (2018). https://doi.org/10.1007/s00145-018-9278-z, https://doi.org/10.1007/s00145-018-9278-z
55. Yao, A.C.: Theory and applications of trapdoor functions (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. pp. 80–91 (1982)