

Your Reputation’s Safe with Me: Framing-Free Distributed Zero-Knowledge Proofs

Carmit Hazay
Bar-Ilan University
carmit.hazay@biu.ac.il

Muthuramakrishnan Venkatasubramaniam
Georgetown University
vmuthu@gmail.com

Mor Weiss
Bar-Ilan University
mor.weiss@biu.ac.il

Abstract

Distributed Zero-Knowledge (dZK) proofs, recently introduced by Boneh et al. (CRYPTO’19), allow a prover \mathcal{P} to prove NP statements on an input x which is *distributed* between k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$, where each \mathcal{V}_i holds only a piece of x . As in standard ZK proofs, dZK proofs guarantee *Completeness* when all parties are honest; *Soundness* against a malicious prover colluding with t verifiers; and *Zero Knowledge* against a subset of t malicious verifiers, in the sense that they learn nothing about the NP witness and the input pieces of the honest verifiers.

Unfortunately, dZK proofs provide no correctness guarantee for an honest prover against a subset of maliciously corrupted verifiers. In particular, such verifiers might be able to “frame” the prover, causing honest verifiers to reject a true claim. This is a significant limitation, since such scenarios arise naturally in dZK applications, e.g., for proving honest behavior, and such attacks are indeed possible in existing dZKs (Boneh et al., CRYPTO’19).

We put forth and study the notion of *strong completeness* for dZKs, guaranteeing that true claims are accepted even when t verifiers are maliciously corrupted. We then design strongly-complete dZK proofs using the “MPC-in-the-head” paradigm of Ishai et al. (STOC’07), providing a novel analysis that exploits the unique properties of the distributed setting.

To demonstrate the usefulness of strong completeness, we present several applications in which it is instrumental in obtaining security. First, we construct a certifiable version of Verifiable Secret Sharing (VSS), which is a VSS in which the dealer additionally proves that the shared secret satisfies a given NP relation. Our construction withstands a constant fraction of corruptions, whereas a previous construction of Ishai et al. (TCC’14) required $k = \text{poly}(t)$. We also design a *reusable* version of certifiable VSS that we introduce, in which the dealer can prove an *unlimited* number of predicates on the *same* shared secret.

Finally, we extend a compiler of Boneh et al. (CRYPTO’19), who used dZKs to transform a class of “natural” semi-honest protocols in the honest-majority setting into maliciously secure ones with abort. Our compiler uses *strongly-complete* dZKs to obtain *identifiable* abort.

Keywords: Distributed Zero Knowledge, Secure Multiparty Computation, MPC-in-the-head, Verifiable Secret Sharing, Identifiable Abort.

Contents

1	Introduction	2
1.1	Our Contribution	4
1.1.1	Strongly-Complete dZK Proofs	4
1.1.2	Applications	6
1.2	Highlights of Our Techniques	7
1.2.1	dZK Proofs From MPC-in-the-Head	8
1.2.2	Certifiable VSS (cVSS) and Reusable cVSS	11
1.3	Open Problems and Future Directions	12
1.4	Paper Organization	12
1.5	Related Works	12
2	Preliminaries	16
2.1	Distributed Zero-Knowledge (dZK) Proofs	17
2.2	Secure Multi-Party Computation (MPC) Protocols	19
3	Checking Membership in a Robust Code	20
3.1	A Batched Verifiable Secret Sharing (VSS) Scheme	24
4	dZK Proofs from Secure MPC Protocols	25
4.1	Instantiations and Extensions	33
4.1.1	Revisiting [IKOS07] in the Distributed Setting	33
4.2	The Case of dZK Without Strong Completeness	36
5	Applications	38
5.1	Certifiable Verifiable Secret Sharing	38
5.2	Reusable cVSS	44
5.3	From Semi-Honest Security to Identifiable Abort Via dZK Proofs	50
5.3.1	Security With Identifiable Abort	50
5.3.2	MPC Terminology and Notation	51
5.3.3	The Compiler	52
5.3.4	Comparison With Previous Works	56
6	Ideal dZK and Connections with Verifiable Relation Sharing	57
6.1	The Ideal dZK Functionality	57
6.2	Connection between dZK and VRS	60

1 Introduction

Zero-Knowledge (ZK) Proofs, namely proofs that yield nothing but their validity, are an essential component of many cryptographic systems. Recently, [BBC⁺19b] introduced a *distributed* model for ZK proofs which captures the types of proof systems that appear in many existing application scenarios such as anonymous messaging [CBM15], verifiable function secret sharing [BGI16], and systems for privately computing aggregate statistics [CB17]. This distributed model proved particularly suited for proving honest behaviour in Multi-Party Computation (MPC) protocols [BBC⁺19b, BGIN19, BGIN20, BGIN21].

ZK proofs. A standard ZK proof [GMR85] is a 2-party protocol between a prover \mathcal{P} and a verifier \mathcal{V} . Both parties are Probabilistic Polynomial Time (PPT), and have a joint input x . The prover’s goal is to convince \mathcal{V} that $x \in \mathcal{L}$ for some language \mathcal{L} (i.e., a subset of strings). When \mathcal{L} is an NP language, \mathcal{P} is additionally given a witness for the membership of x in \mathcal{L} . A ZK proof guarantees the following properties. (1) *Correctness*, meaning if $x \in \mathcal{L}$ and both parties honestly follow the protocol, then \mathcal{V} outputs accept (with high probability). (2) *Soundness*, namely if $x \notin \mathcal{L}$ then any (possibly malicious and computationally unbounded) \mathcal{P}^* can only cause \mathcal{V} to accept x with small probability. (3) *Zero Knowledge (ZK)*, guaranteeing that even a (possibly malicious and computationally unbounded) \mathcal{V}^* learns nothing from the execution, except that $x \in \mathcal{L}$. In particular, \mathcal{V} learns nothing about the corresponding witness. This is formalized by requiring the existence of a PPT simulator $\text{Sim}_{\mathcal{V}^*}$ who on input x can simulate the entire view of \mathcal{V}^* – consisting of x , the random coin tosses of \mathcal{V}^* , and the messages it received from the honest \mathcal{P} .

Distributed ZK (dZK) proofs [BBC⁺19b] generalize standard ZK proofs to a distributed setting with multiple verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$, where the input statement x is distributed among them but no verifier knows x in full. (For example, x could be a secret sharing of some secret, and each \mathcal{V}_i holds a share).¹ Parties are connected via secure point-to-point channels as well as a broadcast channel. In fact, our protocols have the additional feature that following the first round in which the prover sends a single message to each verifier, all communication is over the broadcast channel and private point-to-point channels are not needed (except in the first round).

The standard properties of completeness and soundness naturally extend to the distributed setting, where completeness should hold when all parties are honest, and soundness should hold against a cheating prover \mathcal{P}^* that colludes with a subset of at most t verifiers. As for zero knowledge, in the distributed setting one would generally wish to hide not only the witness w , but also the parts of the input statement held by the honest parties. More specifically, assume the input statement x is distributed between the verifiers, where each verifier \mathcal{V}_i holds an *input piece* $x^{(i)}$. Then we require that a subset $T \subseteq [k], |T| \leq t$ of possibly malicious verifiers learn nothing except $(x^{(i)})_{i \in T}$ and the fact that $x \in \mathcal{L}$, in the sense that there exists a PPT simulator Sim that can simulate their entire view in the protocol given only $(x^{(i)})_{i \in T}$.

Boneh et al. [BBC⁺19b] constructed such dZK proofs for NP whose communication is linear in the size of the verification circuit, and additionally presented sublinear dZK proofs for structured languages. (See Section 1.5 for further details.) A main feature of the dZKs of [BBC⁺19b] — which is crucial to their applications for designing efficient MPC protocols — is having sublinear communication between the verifiers. Notice that while generic MPC protocols could be used to achieve dZK, they would not generally have this feature (see Section 1.1 for further details).

¹Various works have considered other models, e.g., when security is only computational, or when the input statement is known in full to all verifiers. These models are discussed in Section 1.5, but similar to [BBC⁺19b] our focus is on information-theoretic security when the input statement is distributed between the verifiers.

Several recent works [BBC⁺19b, BGIN19, BGIN20, BGIN21] have demonstrated the usefulness of dZK proofs towards constructing *maliciously-secure* MPC protocols, i.e., ones which guarantee correctness of the outputs and privacy of the inputs of the honest parties, even when some parties deviate from the protocol specification. This is done by transforming an MPC protocol Π with semi-honest security – namely whose security only holds when all parties follow the protocol, though corrupted parties might still try to learn information about the inputs of honest parties – into a *maliciously secure* protocol $\tilde{\Pi}$. The high-level idea is to execute Π except for its final round (in which the outputs are revealed), and use dZK proofs to prove honest behavior in this execution.² In this context, “honest behaviour” of a party \mathcal{P}_i roughly means that there exist an input x_i and random coins r_i for which the messages that \mathcal{P}_i sent to the other parties are consistent with x_i, r_i , the messages \mathcal{P}_i received from the other parties, and Π . This is an NP statement (with witness x_i, r_i) which is distributed between the parties $\mathcal{P}_j, j \neq i$ because \mathcal{P}_j knows (only) the messages exchanged between $\mathcal{P}_i, \mathcal{P}_j$, and is known in full to \mathcal{P}_i .

“Framing-Free” dZK Proofs. In the 2-party setting, the properties of standard ZK proofs capture all possible corruption models, namely no corruptions (completeness), corrupt prover (soundness), or corrupt verifier (ZK). However, the corresponding properties of dZK proofs do *not* capture all possible corruption models in the *distributed* setting. Indeed, there is no guarantee for an honest \mathcal{P} when the dZK is executed in the presence of maliciously corrupted verifiers, and in particular, such corrupted verifiers could potentially *frame* the prover, i.e., cause the proof of a correct claim to fail. This corruption model was not explored in previous works on dZK proofs [BBC⁺19b], and in fact maliciously corrupted verifiers *can* frame the prover in their constructions. We note that the dZKs of [BBC⁺19b] do implicitly provide a partial guarantee in this case, since \mathcal{P} is able to identify cheating verifiers.³ However, the best one can do in this case is to identify a pair of parties – namely \mathcal{P} and one of the verifiers \mathcal{V}_i – such that at least one of them is corrupted, but it is impossible to determine which one. In particular, one cannot deduce that $x \in \mathcal{L}$, otherwise this would lead to a successful soundness-violating cheating strategy for \mathcal{P} colluding with a verifier \mathcal{V}_i : “sacrifice” \mathcal{V}_i by causing an inconsistency between \mathcal{P} and \mathcal{V}_i , which will lead the other verifiers to (falsely) conclude that $x \in \mathcal{L}$.

A “framing-free” guarantee is desirable since such situations naturally arise in applications of dZK, e.g., when using dZK proofs to prove honest behavior in MPC protocols as discussed above, and more generally in distributed systems over shared data. Indeed, when an MPC protocol Π is executed with a subset T of corrupted parties, the dZK proof of an honest $\mathcal{P}_i, i \notin T$ will be executed in the presence of corrupted verifiers (namely, the parties in T). In previous works applying dZK to prove honest behavior [BGIN20, BGIN21], the “solution” is to have the prover identify a cheating verifier whenever the proof fails, and then disqualify both parties from the next protocol execution. This “player elimination” techniques is standard in protocol design (and can even be used to obtain guaranteed output delivery), but it gives no indication as to which of the eliminated parties is corrupted. For an honest party who was eliminated from the execution in this manner, the fact that the computation can be successfully completed without it might provide little consolation.

In particular, a protocol in which framing is possible encourages attacks where the adversary targets honest parties and disqualifies them, thus excluding their inputs from the computation and consequently biasing the outcome. This is of particular importance in settings – such as voting,

²More specifically, this paradigm applies to a class of *natural* protocols which guarantee, among other things, that privacy is preserved up to the final round even in the presence of *malicious corruptions*; see Section 5.3.2.

³Roughly, this holds in their protocols because the verifiers do not have any private coins, and \mathcal{P} knows the entire input statement x .

auctions, and secure aggregation – in which elimination harms the reputation of the eliminated party, or when a biased outcome has severe consequences. Moreover, player elimination is only useful when there are repeated executions (or multiple phases) of the protocol, which is not the case in some of the application scenarios in which framing arises (see further discussion in Section 5).

Thus, “framing-free” dZK proofs are motivated not only by the goal of guaranteeing security against all possible corruption patterns, but also because such attacks naturally arise in many application scenarios of dZK.

1.1 Our Contribution

We put forth a strong completeness notion for dZK proofs which guarantees that honest provers cannot be framed. More specifically, we define *t-strong completeness* which guarantees that when the prover is honest and the verifiers hold pieces of an $x \in \mathcal{L}$ then all honest verifiers accept the proof, even in the presence of a subset of t maliciously-corrupted and computationally-unbounded verifiers. In terms of communication, we distinguish between the *proof generation phase* in which the prover distributes the proof among the verifiers (with no communication between the verifiers), and the *verification phase* in which the proof is verified. Our goal is for the total communication complexity during verification to be independent of the computation size (ideally, polynomial in the number of verifiers, the security parameter, and $\log |x|$). We call such protocols *verification efficient*. This feature is especially important when verifiers are lightweight devices and stable communication between a large number of verifiers is not available. A related attractive feature that our protocols provide (in certain settings) is that verifiers require small space to process and verify the proofs. Our protocols will have the added feature that all communication during verification is only through broadcasts.

1.1.1 Strongly-Complete dZK Proofs

We construct strongly-complete dZK proofs by employing the so-called “MPC-in-the-head” paradigm [IKOS07]. Specifically, we construct strongly-complete dZK proofs from an MPC protocol Π with t -privacy – namely, in which secrecy of the honest parties’ inputs holds in the presence of t semi-honest corruptions – and t -robustness (which, roughly, guarantees correctness of the honest parties’ outputs in the presence of t malicious corruptions, see Definition 2.8).

Our construction is informally summarized in the following theorem, where an unconditionally secure t -dZK is a dZK proof system with completeness, strong completeness, soundness, and zero-knowledge (as informally defined above) in the information-theoretic setting in the presence of t corruptions; $\hat{\mathcal{L}}$ for an NP-language \mathcal{L} consists of all robust encodings of $x \in \mathcal{L}$ (e.g., encoding x using an error correcting code with good distance);⁴ and a dZK proof system is verification efficient if the total communication complexity during verification is $\text{poly}(k, \kappa, \log |x|)$ where k denotes the number of verifiers and κ is a security parameter.

Theorem 1.1 (dZK from MPC-in-the-head – informal statement of Theorem 4.1). *Let $t, k \in \mathbb{N}$ such that $k > 6t + 2$. Let \mathcal{L} be an NP-language, and let Π be a perfectly correct, t -private and perfectly t -robust k -party protocol verifying membership in $\hat{\mathcal{L}}$. Then assuming ideal coin-tossing, there exists a 2-round unconditionally-secure verification-efficient t -dZK for $\hat{\mathcal{L}}$.*

⁴Notice that the dZK proof is for input statements that are distributed between the verifiers *using a robust encoding*. [BBC⁺19b] make the same assumption. The reason to focus on such languages is because they show [BBC⁺19a, Sec. 6.3.2] limitations on the existence of dZK proofs for languages that are not robustly encoded.

Moreover, the total proof length in our dZK proof system is quasilinear in the size of the circuit verifying membership in \mathcal{L} , and can be reduced to linear by increasing the round complexity to 3 (see Corollaries 4.3 and 4.5). Furthermore, we define the ideal dZK functionality and show that our constructions realize it (see Section 6).

We note that while strong completeness can be obtained fairly easily in the *computational* setting using standard tools such as commitments and signatures, obtaining it in the *information theoretic* setting seems significantly harder. Specifically, in the computational model the prover can commit to its messages to the verifiers, and parties can then prove consistency with respect to these commitments. (This is exactly the method used to obtain strong completeness in the GMW compiler [GMW87].) Achieving strong completeness information theoretically is much harder since the prover is not committed to its messages to the verifiers.

Theorem 1.1 gives an alternative approach towards designing dZK proofs (even *without* strong completeness) compared to previous works [BBC⁺19b], who relied on fully-linear probabilistically-checkable proofs and fully-linear interactive oracle proofs. One advantage of our approach is that the general construction of Theorem 1.1 can be instantiated with various MPC protocols to obtain dZK proofs with different tradeoffs between the parameters. This is particularly appealing since one could potentially leverage the major research effort devoted towards optimizing MPC protocols, and employ it to obtain dZK proofs whose parameters are optimized for specific applications. We demonstrate this versatility of our approach by instantiating our general transformation with two different MPC protocols, obtaining dZK proofs with different parameters. See Section 4.1 for further details.

The proof of Theorem 1.1 uses a novel analysis for MPC-in-the-head, exploiting the distributed setting, as well as a novel protocol for *batched* verifiable secret sharing. Both of these might be of independent interest. See Sections 1.2, 3, and 4 for further details.

dZK Proof Systems Without Strong Completeness. As noted above, our construction gives an alternative approach towards designing dZK proof systems. To demonstrate this, we describe a scaled-down variant of our construction *without* strong completeness (i.e., in the same security model as that considered in [BBC⁺19b]), and relying on the same assumption of ideal coin tossing) with an improved corruption threshold. This gives an alternative approach towards designing dZK proofs *without* strong completeness.

Theorem 1.2 (dZK Without Strong Completeness – informal statement of Corollary 4.6). *Let $t, k \in \mathbb{N}$ such that $k > 2t + 2$. Let \mathcal{L} be an NP-language, and let Π be a perfectly correct, t -private and perfectly t -robust k -party protocol verifying membership in $\hat{\mathcal{L}}$. Then assuming ideal coin-tossing, there exists a 2-round unconditionally-secure verification-efficient t -dZK for $\hat{\mathcal{L}}$ without strong completeness.*

dZK from Generic MPC Protocols. An alternative route towards designing dZK protocols is to view dZK as an ideal functionality (see Section 6 for further discussion of this functionality), and then use *generic fully-secure* MPC protocols to instantiate it. In slightly more details, such a functionality would take as input each verifier’s input piece, as well as all input pieces and the witness from the prover. Then, it will check that: (1) for at least $k - t$ of the verifiers, the input pieces they provided are consistent with the input pieces the prover provided; and (2) the input pieces define an instance in the relation. While this gives a generic mechanism for constructing strongly-complete dZK proofs, unfortunately, it does not yield *verification efficient* dZKs. Indeed, even when the most communication-efficient protocols (e.g., [DI06]) are used to instantiate the ideal dZK functionality, the total communication between the verifiers will be proportional to the size of the circuit verifying the relation. In contrast, in our protocol, following the initial proof

generation phase, the communication between the verifiers is *independent* of the circuit size. (See Section 1.5 for further details and comparison with generic MPC protocols.)

1.1.2 Applications

We demonstrate the usefulness of strong completeness by showing several applications of dZK proofs in which strong completeness is crucial.

Verifiable Secret Sharing (VSS) and Extensions. A (robust) t -private secret sharing scheme allows an honest dealer \mathcal{D} to distribute a secret x between a k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$, such that any t parties learn nothing about x , but when all parties come together they can reconstruct x even in the presence of t maliciously corrupted parties. A *Verifiable Secret Sharing (VSS)* scheme additionally guarantees soundness against a corrupted dealer colluding with t parties. Specifically, it guarantees that the secret shares define *some* secret x^* which the honest parties will reconstruct regardless of the shares that the corrupted parties provide during reconstruction. Ishai and Weiss [IW14] put forth the notion of *Certifiable VSS (cVSS)* which additionally guarantees that x is in some NP language \mathcal{L} (and, when the dealer is corrupted, that $x^* \in \mathcal{L}$). They construct such schemes based on zero-knowledge probabilistically checkable proofs of proximity, in which $t = k^\epsilon$ for a small $\epsilon < 1$.

We use strongly-complete dZK proofs to construct cVSS schemes, in which the corruption threshold is “inherited” from the underlying dZK. Specifically, using the dZKs of Theorem 1.1, we obtain $t < (k - 2)/6$. Very roughly, the high-level idea is for the dealer to share x using a standard secret sharing scheme, and then have all parties engage in a dZK proving that the input pieces held by the parties share an $x \in \mathcal{L}$. We note that strong completeness is essential for obtaining correctness, which in VSS and cVSS is required to hold for an honest dealer even if t parties are maliciously corrupted. Indeed, if the underlying dZK does not have strong completeness then corrupted parties who actively cheat during the dZK proof can cause it to fail, thus violating correctness. See Section 5.1 for further details.

We also introduce a *reusable* variant of cVSS, in which the dealer can prove that $x \in \mathcal{L}_i$ for a sequence of NP-languages \mathcal{L}_i . In particular, there is no bound on the *number* of languages \mathcal{L}_i , which are determined (i.e., provided to the dealer and all other parties) in an online fashion, and all membership claims are proven with relation to *the same* secret x . We construct reusable cVSS schemes from strongly-complete dZK proofs in Section 5.2.

MPC with Identifiable Abort (IA-MPC). Aborts pose a major obstacle in the malicious corruption setting, or even when parties are honest but have poor network connections. Indeed, a deviating/crashed party could potentially cause the entire computation to fail. The natural mitigation against such “denial-of-completion” attacks is to support *Identifiable Abort (IA)*, i.e., when the executions fails to complete, the parties can (publicly) identify at least one malicious/crashed party.

We use strongly-complete dZK proofs to transform a class of “natural” protocols that are secure in the *semi-honest* setting (in which even corrupted parties follow the protocol) to protocols that guarantee security with identifiable abort in the presence of *malicious* corruptions. This class of “natural” protocols was introduced by [BBC⁺19b], who used dZK proofs *without strong completeness* to transform such protocols into maliciously-secure protocols with (*non-identifiable*) abort. Thus, our compiler shows that *strong completeness* can be used to obtain *identifiable* abort. This result is summarized in the following theorem (see Theorem 5.4 for the formal statement and Definition 5.4 for the definition of a natural protocol):

Theorem 1.3 (IA-MPC from Natural Protocols – Informal). *Let $t, k \in \mathbb{N}$ such that $k > 6t + 3$, and let Π_{nat} be a natural k -party protocol computing a function f in the presence of t semi-honest corruptions.*

Then assuming ideal coin tossing, there exists a protocol Π which securely computes f with identifiable abort in the presence of t malicious corruptions.

Our compiler is very similar to the compiler of [BBC⁺19b]. Their main observation is that dZK proofs can be used to replace the standard ZK proofs used in GMW-style compilers [GMW87], and in fact seem to be a more natural tool in this context. Indeed, the ZK proofs are used to prove honest behavior in an execution of a semi-honest protocol, and this task exactly requires running a zero-knowledge proof on a distributed input.

More specifically, the high-level idea of our compiler is to execute all rounds of Π_{nat} except the final round, then run dZK proofs attesting to the honest behavior of all parties during this execution, before executing the final round of Π_{nat} to reveal the output. One notable property of our compiler is that the compiler itself *does not use any broadcasts*. In particular, all broadcasted messages in Π are either broadcasts of Π_{nat} or of the underlying dZK proofs. When instantiated with our dZK proofs of Theorem 1.1, the number of broadcast bits introduced by the dZK proofs could be as low as $k^2 \text{polylog}(\text{CC}(\Pi_{\text{nat}}))$, where $\text{CC}(\Pi_{\text{nat}})$ denotes the communication complexity of Π_{nat} . We note that the use of broadcasts is inherent to obtaining identifiable abort [CL14]. Previous works obtaining identifiable abort either built on specific maliciously-secure protocols that use a broadcast channel for every multiplication gate, or increased the number of broadcasts to equal the number of multiplication gates. This includes the generic compiler from [IOZ14] discussed next.

Our compiler gives an alternative, conceptually simple, approach towards transforming protocols with semi-honest security into maliciously-secure protocols with identifiable abort in the information-theoretic setting, compared to an existing compiler of Ishai, Ostrovsky and Zikas [IOZ14]. These approaches result in incomparable compilers. Specifically, the compiler of [IOZ14] works in the correlated randomness setting, transforming *any* semi-honest secure protocol into a maliciously-secure protocol with identifiable abort, by broadcasting *every* message of the semi-honest protocol (and proving honest behavior with relation to the broadcasted messages). Our compiler works only for “natural” protocols, but uses much fewer broadcasts. See Sections 1.5 and 5.3.4 for further discussion and comparison of these compilers.

“Framing-Free” Proofs on Committed or Secret-Shared Data. [BBC⁺19b] use dZK proofs to construct proofs on secret shared data. Special cases of such proofs have been considered in several recent works, e.g., [BGI16, CB17]. Roughly, they allow a client to secret share a (potentially large) input x among multiple servers, and then prove to the servers that x satisfies various NP statements. The construction from dZK is conceptually simple: the client plays the role of the dZK prover, and the servers play the role of the verifiers. The client first shares x using a robust secret sharing scheme, and distributes the shares between the servers. The client and servers can then engage in multiple dZK executions to prove various NP statements on x . Instantiating this construction with our *strongly complete* dZK proofs yields “framing-free” proofs on secret shared data, namely in which a subset of corrupted servers cannot cause the proof of a true statement to fail. This strengthens the proofs obtained in [BBC⁺19b] (based on dZKs *without* strong completeness), which do not provide this guarantee.

1.2 Highlights of Our Techniques

In this section we highlight the main techniques used to obtain our results.

1.2.1 dZK Proofs From MPC-in-the-Head

Our dZK proofs are based on the MPC-in-the-head paradigm, introduced by [IKOS07] in the context of constructing (standard) ZK proofs. The high-level idea of the paradigm is that an MPC protocol Π computing a predicate “ $x \in \mathcal{L}$ ” for some NP language \mathcal{L} and some public x (where the corresponding NP witness w is secret-shared between the parties executing Π), can be used to design a ZK proof for the membership of $x \in \mathcal{L}$. Indeed, an *honest* execution of Π on x will result in output 1 if and only if $x \in \mathcal{L}$. Moreover, a main observation made in [IKOS07] is that verifying that Π was honestly emulated – i.e., that the views of all parties participating in Π are globally consistent – can be done by checking *pairwise consistency* of the views. (The view of a party consists of its input, random coins tosses, and the messages it received from other parties in the execution.) That is, if the set of all parties’ views does not correspond to an honest execution of Π on x , then there is a pair of parties whose views are inconsistent with each other.

This observation immediately gives rise to the following proof system: the prover \mathcal{P} emulates “in its head” the entire execution of Π on x (and the shares of w), and commits to the views of all parties in this execution. The verifier \mathcal{V} then picks a pair of parties whose views \mathcal{P} opens, and \mathcal{V} accepts if these views are pairwise consistent, and these parties output 1 in the execution. Thus, soundness follows from the (perfect) correctness of Π , whereas if Π is private against semi-honest corruptions then the proof is also ZK, because the verifier learns only a pair of views in Π , and these reveal only two secret shares of the witness w which, in turn, reveal no information about w .

The resultant proof system has a large soundness error, i.e., the probability that \mathcal{V} accepts false claims is large. [IKOS07] then show how to reduce the soundness error by relying on a stronger correctness guarantee – known as *robustness* – which holds even in the presence of *malicious* corruptions. Roughly, robustness means that if $x \notin \mathcal{L}$ then even maliciously-corrupted parties cannot cause honest parties to output 1 in Π . In particular, while \mathcal{V} might not open a pair of inconsistent views during verification (since \mathcal{V} opens only a small subset of views), still robustness guarantees that cheating that occurred in the un-opened views cannot “propagate” and cause honest parties to accept a false claim in the execution. Consequently, if $x \notin \mathcal{L}$ then the output reported in the honest parties’ views which \mathcal{V} opened will be 0, and so \mathcal{V} will reject. This should be contrasted with the basic construction described above – using Π that is only secure against semi-honest corruptions – in which \mathcal{V} rejects only if it opened a pair of inconsistent views, namely the view of a corrupted party.

Novel Verification for MPC-in-the-Head in Distributed Settings. Our dZK proofs employ the MPC-in-the-Head paradigm, using a novel verification procedure that exploits the properties of the distributed setting.⁵ Specifically, the proof is executed between a prover \mathcal{P} that knows x and a corresponding NP witness w , and k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$, where each \mathcal{V}_i holds a *piece* $x^{(i)}$ of the input x (and \mathcal{P} knows all input pieces). To simplify the presentation, we describe here a simplified dZK proof in the *correlated randomness* model, in which an honest party samples ahead of time a random string $R = (R_1, \dots, R_k)$ from a pre-defined distribution \mathcal{D} , and gives R_i to \mathcal{V}_i . We explain below how to remove this assumption. We stress that the final dZK proof (Figure 3 in Section 4) is *in the plain model* and does *not* use correlated randomness.

The dZK proof proceeds as follows. The correlated randomness consists of (long) random masks r_{ij} for every pair of verifiers $\mathcal{V}_i, \mathcal{V}_j$, where $R_i = (r_{ij})_{j \neq i, j \in [k]}$. The prover emulates “in its head” a k -party protocol Π computing the predicate $(x^{(1)}, \dots, x^{(k)}) \in \mathcal{L}$ as in the 2-party ZK proof described above. However, instead of committing to the views, \mathcal{P} sends the i ’th view to \mathcal{V}_i . The

⁵See Section 1.5 for a comparison between our construction and other constructions using this technique in the two-party and in other distributed settings.

parties then jointly execute the following verification procedure:

1. Each \mathcal{V}_i checks *local* consistency of its view, namely that the input reported in the view is $x^{(i)}$, and that the output of the i th party given this view is 1. If the view is not locally consistent then \mathcal{V}_i broadcasts a complaint against the prover. Let C_1 denote the set of verifiers that complained against the prover.
2. Each pair $\mathcal{V}_i, \mathcal{V}_j$ check *pairwise* consistency of their views by comparing the messages exchanged between parties i, j in Π .⁶ This pairwise consistency check is done *publicly*, by having $\mathcal{V}_i, \mathcal{V}_j$ broadcast the values to be compared, masked using r_{ij} .
3. The prover broadcasts complaints against verifiers who broadcasted incorrect messages in Step 2. Let C_2 denote the set of verifiers against whom the prover complained.
4. Finally, each verifier \mathcal{V}_i determines its output as follows. If $|C_1 \cup C_2| > t$, or there exist $i, j \notin C_1 \cup C_2$ who broadcasted inconsistent messages in Step 2, then reject. Otherwise, accept.

We note that while this describes the main steps in the dZK proof, the actual construction is more involved, in several respects. First, to reduce the communication complexity, instead of sending in Step 2 all the messages exchanged in Π , the verifiers send information-theoretic MACs of these values. More specifically, the messages exchanged between i, j are interpreted as the coefficients of a univariate polynomial, and the MAC is the evaluation of this polynomial at a random point (see Step 3 in Figure 3). This requires the verifiers to jointly sample a random element of a sufficiently large field. The resultant protocol therefore uses an ideal coin-tossing functionality as in [BBC⁺19b] (and makes *minimal* use of it). Second, by using MACs we can eliminate the correlated randomness and have *the prover* provide random masks as part of the proof, and moreover each mask will consist of a *single* field element. Since the random masks are chosen *before* the MAC key is sampled, inconsistent views will, with overwhelming probability, result in inconsistent MACs, even when the prover chooses the masks.

Finally, the input pieces held by the verifiers should constitute an encoding of the underlying input x in some robust code, and the parties need to verify that their input pieces are indeed “close” to a valid encoding. This is done using a standard technique for batch-testing of code membership. Specifically, the verifiers broadcast a random linear combination of the codeword symbols they hold, which they also mask with a random codeword (masking is needed to guarantee ZK). Soundness of this test has been studied in several previous works [AHIV17, BBHR18, BCI⁺20] (and plays an important role in improving the concrete efficiency of succinct ZK arguments). However, relying on the analysis directly in our distributed setting will not guarantee strong completeness (only identifiable abort). We refine this soundness analysis to apply in the distributed verification setting while guaranteeing strong completeness. See Section 4 for further details and the full construction.

We note that the dZK proofs of [BBC⁺19b] also require the input pieces to form a robust encoding, and they show [BBC⁺19a, Sec. 6.3.2] some limitations on the existence of dZK proofs when the input statement is *not* robustly encoded, at least when security should hold against collusions of the prover and verifiers, as we consider in this work.

The Security Analysis. Proving security of our dZK proof is more complex than in standard (in particular, 2-party) settings of MPC-in-the-Head, and requires a novel analysis which combines

⁶The messages sent from party i to party j appear explicitly in the view of party j , and the messages it sent to party i can be computed from its view.

techniques from the VSS literature. Intuitively, this is because while in the analysis of ZK proofs such as those of [IKOS07] the verifier can safely reject if an inconsistency is detected, we cannot immediately reject because inconsistencies might be due to corrupted verifiers trying to “frame” an honest prover (and so rejecting in this case would violate strong completeness). Thus, the strong completeness guarantee leads to a much more intricate soundness analysis.

Proving strong completeness is fairly simple, and it follows from the fact that all complaints arise from the corrupted parties. That is, either a corrupted party falsely complaining that its view is not locally consistent, or a corrupted verifier broadcasting an incorrect MAC in Step 2, causing the prover to complain against it. Thus, we will have $|C_1 \cup C_2| \leq t$ in Step 4, and all other parties’ views will be pairwise consistent, so all honest verifiers will accept.

The soundness analysis, however, is much more involved. At a high level, it proceeds by showing that if $(x^{(1)}, \dots, x^{(k)}) \notin \mathcal{L}$ then there exists a subset H of parties which constitutes an honest majority in the execution of Π with input pieces $(x^{(1)}, \dots, x^{(k)})$, and therefore their outputs in Π would be 0. Thus, the checks performed in Step 4 would fail and all honest verifiers would reject. More specifically, in the analysis we gradually eliminate verifiers (alternatively, parties in Π , since there is a correspondence between the dZK verifiers and the parties in Π) until we are left with the set H . We stress that unlike MPC applications employing the “player elimination” technique, we *do not actually eliminate* any verifier from the computation, but rather this “elimination” is only done *in the analysis*. Moreover, an “eliminated” verifier is not necessarily corrupted – for example, it might be an honest verifier who received an incorrect view from the prover – but rather these are verifiers whose views in Π might not correspond to honest strategies, and therefore cannot be relied on for verification.

More specifically, the set H is obtained as follows. First, since the verifiers check that their input pieces are close to a valid codeword, if the test passes then we are guaranteed that the input pieces of the honest verifiers are at most t -far from the code, in the following sense. There exists a subset T , $|T| \leq t$ of honest verifiers, and a valid codeword, such that the input pieces of all honest verifiers $i \notin T$ are identical to the corresponding pieces of the codeword. (Intuitively, the parties in T hold “incorrect” input pieces.) We then eliminate the verifiers in T .

Next, in the remaining set of $\geq k - t$ verifiers, there are at most t corrupted verifiers (i.e., corrupted in the dZK), and we eliminate them as well. These verifiers need to be eliminated because they cannot be relied on to honestly check their views. In particular, they might not complain against the prover, even if their output in Π is 0, or they might cheat in Step 2, sending messages which are not actually consistent with their views. We note that the existence of such corrupted verifiers *in the dZK execution* is also the reason that we need to rely on *robust* MPC protocols even though we seemingly check *all* views in Π (indeed, the views held by the corrupted verifiers are never checked). Finally, we eliminate the (at most t) verifiers in $C_1 \cup C_2$.

We thus remain with $\geq k - 3t$ *honest* verifiers, whose views are both locally and pairwise consistent. Their views therefore correspond to an execution of Π on $(x^{(1)}, \dots, x^{(k)})$ with an honest majority (when $k > 6t + 2$), and so the robustness of Π guarantees that the outputs of these parties must be correct. (This description is a gross over simplification of the actual analysis, see Section 4 for further details, and for a clarification why we need $k > 6t + 2$ instead of $k > 6t$.)

Finally, our verification procedure provides a strong ZK guarantee – verifiers learn nothing beyond the view of the corresponding party in Π (whereas in [IKOS07] the verifier learns multiple views).

1.2.2 Certifiable VSS (cVSS) and Reusable cVSS

Certifiable VSS (cVSS) schemes follow naturally from strongly-complete dZK proofs, using a standard robust secret sharing scheme. Specifically, in a t -robust secret sharing scheme, reconstruction succeeds even if t parties provide incorrect shares. We note that many standard VSS schemes employ robust secret sharing as a building block. Moreover, as discussed in Section 1.2.1, robustly encoding the input seems necessary in dZKs with security against coalitions of the prover and a subset of verifiers.

Our cVSS scheme for an NP-language \mathcal{L} consists of a dealer \mathcal{D} and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$. The dealer shares its secret x using the robust secret sharing scheme, and distributes the shares among the parties. The parties then run a “code membership” test to check that their shares are “close” to a valid secret sharing of some secret. Then, the parties execute a dZK, in which \mathcal{P}_i 's input piece is its share, attesting to the fact that the shared secret is in \mathcal{L} . If the dZK fails then the parties revert to some fixed sharing of an arbitrary $x^* \in \mathcal{L}$. Reconstruction is performed by simply running the reconstruction procedure of the underlying secret sharing scheme (and correcting errors if necessary). The strong completeness of the dZK guarantees that when the dealer is honest, corrupted parties cannot “frame” the dealer during the dZK test. Since the code membership test has a similar strong completeness guarantee, the cVSS is correct (in the presence of t active corruptions).

Our cVSS scheme is incomparable to the cVSS of [IW14]. Specifically, the communication during the verification part of their sharing phase (i.e., the part corresponding to executing the code test and the dZK in our cVSS) is polylogarithmic in the total number of parties, and the total communication during sharing is linear. In contrast, when instantiated with the dZK proofs of Theorem 1.1, the communication of our cVSS scheme during sharing would be at least quadratic. However, to contend with t corruptions, the number of parties in the cVSS of [IW14] must be a (large) polynomial in t , whereas our cVSS has $k = O(t)$. Therefore, in many settings, our cVSS might have lower overall communication complexity due to the smaller number of parties it employs. Moreover, our cVSS can be generalized to the *reusable* setting, as we now discuss.

Reusable cVSS. We generalize the notion of cVSS to allow the dealer to prove *multiple* NP statements – which are determined in an online fashion – on the *same* shared secret, using the same secret shares. In particular, the scheme now includes a *Prove* phase that can be executed following the Sharing phase an unlimited number of times. In each Prove phase the parties are given an NP language \mathcal{L} , and \mathcal{D} is additionally given a corresponding witness, and the dealer proves to the parties that the secret their shares encode is in \mathcal{L} .

We note that several subtleties arise when defining reusability, and in particular, reusable cVSS is not a strict strengthening of cVSS. The main reason for this is that since during the Sharing phase the parties still do not know all the NP languages which will be used during the Prove phases, we cannot generally guarantee that the secret x which the shares will reconstruct to will be in all NP languages (and in fact, it might be the case that there exists no such x). Instead, binding only guarantees that when the Sharing phase terminates, even a malicious dealer \mathcal{D}^* is committed to *some* secret x , but there is no further guarantee on x . Binding additionally guarantees that \mathcal{D}^* cannot prove false claims about x , namely if $x \notin \mathcal{L}$ then an execution of the Prove phase with language \mathcal{L} will fail. This should be contrasted with standard cVSS which isn't reusable, i.e., it can be executed only for a single NP language \mathcal{L} , but whose binding property guarantees that the secret x which will be reconstructed, satisfies $x \in \mathcal{L}$. Further subtleties are discussed in Section 5.2.

Our reusable cVSS scheme operates similarly to the cVSS scheme described above. Specifically, to share x the dealer secret shares it using a robust secret sharing scheme, and the parties then run a code membership test on the shares. Each Prove phase with NP language \mathcal{L} consists of running

a dZK for the claim that the shares reconstruct to a secret in \mathcal{L} , and reconstruction is by running the reconstructor of the underlying secret sharing scheme. See Section 5.2 for further details.

1.3 Open Problems and Future Directions

Our work gives rise to many interesting questions in the context of dZK and MPC-in-the-Head. First, we did not explore the possibility of obtaining more efficient constructions for simple NP languages, e.g., with low degree. In particular, using an appropriate MPC instantiation, it might be possible to design special-purpose dZKs for simpler languages, with sublinear communication complexity and improved computational complexity. Round complexity is another important complexity measure. Our construction achieves a 2-round dZK assuming ideal coin-tossing, and leaves open the question of proving this is optimal, or further improving the round complexity as in the computational setting for related proof systems [AKP22]. Finally, it would be interesting to find further applications of dZK proofs, and in particular of strongly-complete ones.

1.4 Paper Organization

In Section 2 we introduce basic preliminaries. In Section 3 we analyze our batch code membership test. In Section 4 we present our dZK proof construction. In Section 5 we present applications of strongly-complete dZK, and in Section 6 we discuss the connection between dZK and Verifiable Relation Sharing (VRS).

1.5 Related Works

Zero-Knowledge Proofs in Distributed Settings. The notion of ZK in distributed settings has been extensively explored in a recent sequence of works [CBM15, CB17, BBC⁺19b, BGIN20, BGIN21]. The motivation for such models is that they present a useful abstraction that captures many scenarios naturally arising in distributed computation. The first two works discussed how to embed distributed ZK into real-world applications such as anonymous broadcast messaging practical at a large scale [CBM15], and a federated learning system, denoted by Prio, with input certification to securely compute aggregate statistics [CB17]. The latter system has been deployed in various real world scenarios. For instance, Mozilla uses a modified version of Prio to privately collect web usage statistics, and Apple and Google use Prio for their exposure notifications express (ENX) system. Nevertheless, the model considered for both [CBM15, CB17] is limited because they assume that the verifiers are semi-honest, and moreover they only consider a specific functionality. On the other hand, their dZKs achieve information-theoretic security.

Different settings have been considered in this context, depending on whether the input statement is known in full to all verifiers (starting with the work of [BD91], and more recently in, e.g., [GO07, YW22, BJO⁺22, AKP22]), or distributed between them (as in the distributed ZK proofs discussed below); and whether the resultant scheme is information-theoretically, or only computationally, secure while optimizing different parameters of the proof system.

Another related model is that of Verifiable Relation Sharing (VRS) [GIKR02], which is similar to the model of dZK proofs considered in this work, in the sense that the input statement is distributed between the verifiers, but differs from it because the prover chooses the statement and the verifiers' shares (whereas in dZK proofs the prover has no control over the input statement). Works on VRS [GIKR02, AKP20a, AKP22] consider both the information theoretic [GIKR02] and the computational setting [AKP20a, AKP22], with progressively improving corruption thresholds.

Specifically, the VRS of [GIKR02] is for $k \geq 6t$, which the latter pair of works improved by moving to the computational setting. More specifically, [GIKR02] obtain a 2-round perfectly-secure VRS protocol whose communication complexity (and in particular, the communication between the verifiers) scales with the circuit size. This protocol can be made to be verification efficient (i.e., where the communication between the verifiers is independent of the circuit size) using the MAC-based verification techniques used in our dZK proofs. This requires coin tossing, and also relaxing security to statistical. [AKP22] obtain a 2-round VRS with computational security against $t < (k - 1)(1/2 - \varepsilon)$ for an arbitrarily small $\varepsilon > 0$, assuming non-interactive commitments (their protocol is not verification-efficient).⁷ Their result extends to any *single input functionality*, resolving the round complexity of such functionalities. (As noted in Section 1.3, the round complexity of dZK – which is not a single input functionality – is not yet resolved for optimal thresholds.)

It is instructive to note that this difference in who chooses the input statement makes VRS and dZK incomparable. Indeed, dZK can be used to prove correctness “after the fact” (namely, after the parties already have their inputs fixed), while in VRS the prover chooses the inputs. Therefore, when used in settings when parties already hold their inputs, VRS necessitates some external mechanism for verifying consistency of the parties’ inputs, and the inputs provided by the prover.⁸ On the other hand, both primitives can be useful in constructing similar applications, such as the certified VSS primitive discussed above. We note that while VRS can also be used to obtain IA-MPC, the construction from dZK is conceptually simpler and more efficient (only additively increases the round and communication complexities). Using VRS complicates the protocol (as it requires sending the protocol messages as part of the VRS), and also increases the round and communication complexities by a multiplicative factor that grows with the respective complexities of the underlying VRS. See Section 6 for further discussion of the connection between the two primitives.

Distributed Zero-Knowledge Proofs. Out of the multitude of distributed models for ZK, the focus of this work is on the “distributed zero-knowledge” (dZK) proofs presented in [BBC⁺19b]. In dZK, the input statement is distributed between the verifiers (where no verifier knows it in full), and security is unconditional. [BBC⁺19b] consider two possible corruption models in the context of soundness: a malicious prover interacting with honest verifiers (“setting I”), and a malicious prover colluding with a subset of verifiers (“setting II”). (In this work we consider only the latter corruption model.) They design dZK systems based on fully-linear probabilistically checkable proofs or fully-linear interactive oracle proofs. Specifically, assuming ideal coin tossing, they construct dZKs for “*low-degree languages*” in which the communication and round complexities are logarithmic in the size of the statement.

Assuming ideal coin-tossing, they also provide a 2-round construction for arbitrary circuits in which the communication complexity is proportional to the circuit size, and ZK (soundness, respectively) holds against t corrupted verifiers ($t - 1$ corrupted verifiers colluding with the prover, respectively), where $k > 2t$. We note that their constructions do not achieve strong completeness. In Section 4.2 we adapt our techniques to obtain a 2-round dZK scheme (assuming ideal coin tossing) without strong completeness with ZK (soundness, respectively) against t corrupted verifiers (t corrupted verifiers colluding with the prover, respectively) for $k > 2(t + 1)$, where the

⁷[AKP22] also obtain a fully information-theoretically secure VRS assuming ideal non-interactive commitments, as well as a computationally sound and statistically ZK (statistically sound and computationally ZK, respectively) VRS based on computationally binding and statistically hiding (statistically binding and computationally hiding, respectively) non-interactive commitments [App22].

⁸In the computational setting one can use standard tools such as commitments to help resolve disputes between parties, but in the information theoretic setting this seems to require a more sophisticated dispute-resolution sub-protocol.

communication complexity is quasi-linear in the circuit size, and can be reduced to linear with one additional round. See Section 4.2 for further details and comparison with the results of [BBC⁺19b].

Following the formalization of [BBC⁺19b], several follow-up works [BGIN19, BGIN20, BGIN21] explored the applicability of dZKs in the context of MPC, starting with [BGIN19] that focused on the three-party setting with an honest majority. This simpler case excludes the corruption model of a prover colluding with a verifier, and therefore only requires a simpler dZK. Building on [BGIN19], the protocol introduced in [BGIN20] works in the honest majority setting for a constant number of parties by applying the sublinear dZK from [BBC⁺19b]. [BGIN21] extends the techniques from [BGIN20] to the dishonest majority setting with preprocessing.

dZK as an ideal functionality As mentioned before, our notion of dZK can be specified as an ideal functionality (See Section 6.1), and such a functionality can be realized generically using a fully secure MPC protocol against active adversaries. Starting from the works of [BGW88, CCD88], we have known that against a threshold of up to $t < n/3$ active corruptions, any functionality can be realized where the round complexity is proportional to the “depth” of the computation.⁹ If we are willing to assume exponential communication (in the depth and/or number of parties), several works have shown how to achieve this in constant rounds [IK02, ABT19, ACGJ18, ACGJ19]. Since the dZK functionality can be expressed as a constant-depth function (specifically, a depth-2 function), the most relevant works are [AKP20a, AKP20b], who show how to achieve tight threshold and round complexity with perfect and statistical security (respectively). Specifically, [AKP20a] show how to achieve 3-round actively-secure MPC with statistical security against $t < n/3$ active corruptions where the communication is exponential in the number of parties (for constant depth circuits), and [AKP20b] show how to achieve 4-round perfect security against $t < n/3$ active corruptions that is fully polynomial time (for constant-depth circuits). Both these results are tight up to corruption thresholds and security level (perfect or statistical). However, all these works will result in communication between the verifiers that is proportional to the circuit size, so they are not verification efficient.

MPC-in-the-head is a powerful technique, originally introduced in [IKOS07] as a novel approach towards designing zero-knowledge proofs, based on MPC protocols. The high level idea is to mimick the environment for which the MPC was designed to run, and essentially emulate its execution. This approach gives a generic construction of zero-knowledge proofs from general MPC protocols with certain properties, and compliments the prevalent use of zero-knowledge proofs in designing MPC protocols. Informally, the prover emulates “in its head”, on shared inputs, the MPC protocol that validates the relation to be proven, and commits to the internal view of each party (which includes their secret inputs and randomness, and incoming messages). The verifier then asks the prover to open a subset of these views, which the verifier checks for pairwise consistency. ZK of the proof system follows from the privacy property of the underlying MPC, whereas soundness follows from the MPC robustness (or correctness). Following this seminal work, this approach has been improved and optimized [GMO16, AHIV17, CDG⁺17, KKW18, BFH⁺20, GSV21], leading to different MPC protocols and optimizations. More specifically, [GMO16, CDG⁺17] followed the line of work from [IKOS07] based on semi-honest MPC protocols. The Ligero proof system [AHIV17] showed a sublinear zero-knowledge protocol based on an honest majority MPC protocol, where the proof complexity grows with $\sqrt{|C|}$ (where C is the circuit verifying the relation), which was later optimized in [BFH⁺20, GSV21]. Finally, [KKW18] considered an instantiation based on a semi-honest pro-

⁹Depth of a circuit is the maximal number of sequential multiplications (in the corresponding field/ring) from any input to any output in the circuit.

tol in the preprocessing model.

COMPARISON WITH OUR MPC-IN-THE-HEAD CONSTRUCTION. Our novel verification technique for the distributed setting has several advantages over MPC-in-the-head techniques used in the 2-party setting. First, it does not require commitments, and in particular gives information-theoretic security. Commitments are not needed because sending the views in Π to the verifiers effectively commits the prover to these views (at least, to the ones given to honest verifiers). Moreover, usually (e.g., this is the case in [IKOS07]) the soundness error depends on the number of parties. Indeed, the soundness error depends on the size of the challenge space, namely the number of possible subsets of views which the verifier opens, where obtaining $\text{negl}(s)$ soundness error requires opening $\Omega(s)$ views. Since a single verifier receives all the opened views, the privacy parameter of the system, and consequently the total number of parties, increases proportionally to s , and the communication complexity of Π (i.e., the view size) increases accordingly. In contrast, in our verification procedure *all* views are simultaneously checked, but each verifier sees a single view. Thus, the privacy parameter, and the total number of parties in Π , is *independent* of the security parameter. Instead, the soundness error is roughly proportional to the ratio between the communication complexity of Π and the size of the field used to generate the MACs. Obtaining $\text{negl}(s)$ soundness error thus requires the field size to be super-polynomial in s and the communication complexity $CC(\Pi)$ of Π . Therefore, the overall communication in Step 2 (Section 1.2.1) would be only *polylogarithmic* in s .

COMPARISON WITH OTHER MPC-IN-THE-HEAD CONSTRUCTIONS IN A DISTRIBUTED SETTING. [AKP22] employ the MPC-in-the-head paradigm in the *distributed* computational setting. Similar to [IKOS07], the prover in their construction commits to the views in the MPC. However, instead of opening a subset of views (and checking their consistency), [AKP22] exploit the distributed setting to simultaneously check consistency of *all* views. Nonetheless, their method of doing so differs significantly from ours. Specifically, they check *local* consistency of each view (roughly, verifying that the party honestly generated its messages given its input and randomness), by running an appropriate MPC with a “committee” (i.e., a subset) of the parties. In contrast, we check pairwise consistency between views by revealing a view in full to a single party and have the parties compare their messages in the clear (i.e., without any MPC computation). We are thus able to avoid using commitments and get information-theoretic security.

Identifiable Abort MPC (IA-MPC). Secure computation in the dishonest majority setting has a significant limitation: it *inherently* cannot prevent even a single deviating party from causing the protocol to fail [Cle86]. While guaranteed output delivery is possible when there is an *honest* majority, aborts still create substantial obstacles. In particular, obtaining guaranteed output delivery often incurs a large overhead in rounds and communication complexity due to the player elimination technique. A natural solution to the problem of parties repeatedly failing the protocol is to support *identifiable abort*. That is, if the protocol fails to complete, it must provide a method to (publicly) identify at least one malicious / crashed party. Identifying cheaters is highly non-trivial for concretely efficient protocols [IOZ14, SF16, BOS16, CFY17, BOSS20, BMMM20, Bra21, SSY22] since the parties must reach consensus on the cheater’s identity. This property is very useful for deterring malicious behavior; in particular, when *penalties* are used against malicious parties, as is the case with smart contracts that run on distributed ledgers and realize a bulletin board.

Amongst these works, only [IOZ14] introduces a *generic* compiler from any semi-honestly secure MPC protocol which uses correlated randomness into a similar protocol which is secure with identifiable abort against malicious adversaries. This compiler works by broadcasting each semi-honest message, together with a zero-knowledge proof of consistency with that party’s committed input and correlated randomness obtained from the setup phase. It therefore increases the broad-

cast complexity of the semi-honest protocol proportionally to the size of the computation. More recent works (e.g., [BOS16, SF16, BOSS20]) on identifiable abort have refined this approach but the overall communication complexity of these approaches is $\Omega(\kappa \cdot |C|)$ to generate correlated randomness in the offline phase, and $\Omega(|C|)$ in the online phase, where κ the (computational) security parameter and $|C|$ is the circuit size. While the more recent works achieve identifiable abort property for specific protocols (e.g., SPDZ-type protocols), the work of [IOZ14] presents a generic compiler for any semi-honest protocol in the correlated-randomness model. In this work, we use dZKs to provide a compiler in similar vein to achieve identifiable abort, but for a class of protocols in the honest-majority setting.

Verifiable Secret Sharing (VSS) is an extension of standard secret sharing, and an important building block in designing secure MPC protocols. A large body of works have studied VSS in different settings, e.g., synchronous vs. asynchronous networks, and information-theoretic vs. computational security. The complexity of VSS has likewise been extensively analyzed (see [CCP21] for a survey).

In this work we consider VSS in two settings: *batch VSS* in the synchronous setting, where a single dealer shares a batch of secrets; and *certifiable VSS* where in addition to sharing a secret the dealer proves membership of the secret in some language. The round complexity of VSS in the synchronous setting was studied in [GIKR02], who constructed two-round VSS for $k > 4t$ and 3-round VSS for $k > 3t$, where k is the total number of parties, and t is the number of corruptions. These schemes are both optimal, up to the usage of broadcasts. [KKK09] improved the 3-round protocol to additionally get optimal usage of broadcast (specifically, it is used in a single round). In the batched setting, [BGR98] design a protocol to batch-verify degrees of polynomials over large fields, assuming all verifiers are honest. Although they do not obtain all the properties of a VSS scheme, their construction is essentially the blueprint for our batched VSS. Implicit in [DI06] is a batch VSS scheme for $t = \Omega(k)$ in which the dealer can share multiple (packed) secrets over constant-sized fields.

2 Preliminaries

Notation. \mathbb{F} denotes a finite field. A *language* \mathcal{L} over \mathbb{F} is a subset $\mathcal{L} \subseteq \mathbb{F}^*$. For a pair of vectors $v, u \in \mathbb{F}^k$, we denote their Hamming distance by $d(u, v) = |\{i : u_i \neq v_i\}|$. We associate with a code $\mathcal{C} \subseteq \mathbb{F}^k$ an *encoding procedure* Enc and a *decoding procedure* Dec such that for every x , $\text{Dec}(\text{Enc}(x)) = x$. We will also allow for encoding to be randomized (this would be useful for our applications of dZK, see Section 5.1). We use PPT to denote probabilistic polynomial time computation. For a distribution \mathcal{D} , sampling according to \mathcal{D} is denote by $X \leftarrow \mathcal{D}$, or $X \in_R \mathcal{D}$. For a pair of random variables X, Y , we use $X \equiv Y$ to denote that X, Y are identically distributed. For random variables X and Y over a finite domain Ω , the *statistical distance* between them is defined as

$$\text{SD}(X, Y) = \frac{1}{2} \sum_{w \in \Omega} |\Pr[X = w] - \Pr[Y = w]|.$$

X and Y are ε -*statistically close* if their statistical distance is at most ε . Ensembles $\{X_s\}_s, \{Y_s\}_s$ are *statistically close*, denoted $X_s \approx Y_s$, if there exists an $\epsilon(s) = \text{negl}(s)$ such that X_s, Y_s are $\epsilon(s)$ -close for every s .

The Ideal Coin-Tossing Functionality. For modularity, our construction will employ an ideal implementation of coin-tossing, a standard primitive that generates unpredictable, public ran-

domness. Specifically, for a field \mathbb{F} we define $\mathcal{F}_{\text{coin}}^{\mathbb{F}}$ to be a randomized functionality that takes no input, and outputs to all parties a uniformly random $r \in_R \mathbb{F}$. We omit \mathbb{F} from the notation when it is clear from the context, and abuse notation by allowing the functionality to toss multiple coins in parallel. That is, we will write “the parties call $\mathcal{F}_{\text{coin}}$ to generate $r \in \mathbb{F}^m$ ” to denote that the parties make m parallel calls to $\mathcal{F}_{\text{coin}}^{\mathbb{F}}$.

Coding notation. For a code $\mathcal{C} \subseteq \mathbb{F}^k$ and vector $v \in \mathbb{F}^k$, denote by $d(v, \mathcal{C})$ the minimal distance of v from \mathcal{C} , namely $d(v, \mathcal{C}) = \min_{u \in \mathcal{C}} d(v, u)$, and denote by $\Delta(v, \mathcal{C})$ the set of positions in which v differs from such a closest codeword (in case of ties, take the lexicographically first closest codeword). We further denote, for a vector set $V \subseteq \mathbb{F}^k$ and a code \mathcal{C} , $\Delta(V, \mathcal{C}) = \bigcup_{v \in V} \{\Delta(v, \mathcal{C})\}$, and denote by $d(V, \mathcal{C})$ the minimal distance between a V and the code \mathcal{C} , namely $d(V, \mathcal{C}) = \min_{v \in V} \{d(v, \mathcal{C})\}$. Our constructions will employ robust codes which, intuitively, are error correcting.

Definition 2.1 (Robust Code). A code $\mathcal{C} \subseteq \mathbb{F}^k$ is (ε, t) -robust if for every $u \in \mathcal{C}$ and for every $v \in \mathbb{F}^k$ such that $d(u, v) \leq t$, $\Pr[\text{Dec}(u) = \text{Dec}(v)] \geq 1 - \varepsilon$. \mathcal{C} is perfectly t -robust if it is $(0, t)$ -robust.

Distributed Inputs, Distributed Relations, and Distributed Languages. Let $n \in \mathbb{N}$ be a length parameter, \mathbb{F} be a finite field, and $\mathcal{C} \subseteq \mathbb{F}^k$ be a robust code with encoding procedure Enc and decoding procedure Dec . The following notions are defined for a fixed n , but naturally extends to a family of length parameters by using families of codes. For an input $x \in \mathbb{F}^n$, a corresponding k -distributed input $X \in \mathbb{F}^{k \times n}$ is a matrix such that for every $i \in [n]$, the i 'th column $X[i]$ of X satisfies $x_i = \text{Dec}(X[i])$ (intuitively, the i 'th column of X encodes the i 'th symbol x_i of x , possibly with some errors).¹⁰ We will write $X = (x^{(1)}, \dots, x^{(k)})$, where for every $i \in [k]$, the *input piece* $x^{(i)}$ is the i 'th row of X (i.e., it is the list of i 'th symbols in the codewords encoding x_1, \dots, x_n).

Definition 2.2 (Distributed Languages and Relations). For a language $\mathcal{L} \subseteq \mathbb{F}^n$ over \mathbb{F} , the corresponding k -distributed language $\widehat{\mathcal{L}}_{\mathcal{C}}$ over \mathbb{F} with relation to \mathcal{C} is defined as

$$\widehat{\mathcal{L}}_{\mathcal{C}} = \left\{ X = \left(x^{(1)}, \dots, x^{(k)} \right) : (x_1, \dots, x_n) \in \mathcal{L}, \right. \\ \left. \text{where } x_i = \text{Dec}(X[i]) \text{ for all } 1 \leq i \leq n \right\}.$$

For an NP-relation $\mathcal{R} = \mathcal{R}(x, w) \in \mathbb{F}^n \times \mathbb{F}^*$ over \mathbb{F} , the corresponding k -distributed relation $\widehat{\mathcal{R}}_{\mathcal{C}}$ with relation to \mathcal{C} is defined as

$$\widehat{\mathcal{R}}_{\mathcal{C}} = \left\{ \left(\left(x^{(1)}, \dots, x^{(k)} \right), w \right) : ((x_1, \dots, x_n), w) \in \mathcal{R}, \right. \\ \left. \text{where } x_i = \text{Dec}(X[i]) \text{ for all } 1 \leq i \leq n \right\}.$$

We call $\widehat{\mathcal{L}}_{\mathcal{C}}, \widehat{\mathcal{R}}_{\mathcal{C}}$ the k -distributed language and the k -distributed relation that correspond to \mathcal{L}, \mathcal{R} , respectively. When \mathcal{C} is clear from the context, we omit it and simply write $\widehat{\mathcal{L}}, \widehat{\mathcal{R}}$. For a (distributed) NP-relation \mathcal{R} , we denote $\mathcal{L}(\mathcal{R}) = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$.

2.1 Distributed Zero-Knowledge (dZK) Proofs

Following Boneh et al. [BBC⁺19b], we consider a distributed setting in which a single prover \mathcal{P} interacts with k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$. Each verifier \mathcal{V}_i holds a *piece* $x^{(i)} \in \mathbb{F}^*$ of a distributed input

¹⁰Notice that if Enc is randomized then x might have several corresponding k -distributed inputs X .

$(x^{(1)}, \dots, x^{(k)})$ encoding some input $x \in \mathbb{F}^*$, and the prover's goal is to convince the verifiers that $(x^{(1)}, \dots, x^{(k)}) \in \widehat{\mathcal{L}}$ for some language \mathcal{L} . We assume that $(x^{(1)}, \dots, x^{(k)})$ is known to the prover. When \mathcal{L} is an NP language, the prover additionally knows a witness w for the fact that $x \in \mathcal{L}$. The parties can communicate over point-to-point channels, as well as a broadcast channel.

Similar to standard (i.e., 2-party) ZK proofs, the system should satisfy completeness (when all parties are honest), zero knowledge (against a subset of corrupted verifiers), and soundness. The two latter properties have several possible interpretations in the distributed setting, as we now explain.

In terms of ZK, following [BBC⁺19b] we require that a subset of corrupted verifiers learn nothing on the NP witness, *as well as on the input pieces of the honest verifiers*. This is formalized by requiring, as in the standard setting, the existence of an efficient simulator that can simulate the corrupted verifiers' views given only their input pieces. This provides a strong ZK property which is meaningful also for languages and relations in P . There are also two possible interpretations of the soundness property. We choose to consider the stronger requirement of soundness against a corrupted prover colluding with a subset of verifiers, namely for $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, the honest verifiers should reject with high probability. ([BBC⁺19b] consider also a weaker notion in which soundness is only required to hold when all verifiers are honest.)

Another concern naturally arises in this distributed setting: that of corrupted verifiers trying to "frame" an honest prover, namely trying to cause the honest verifiers to reject a true claim $(x^{(1)}, \dots, x^{(k)}) \in \widehat{\mathcal{L}}$. We require that they succeed only with small probability. This property, which we call *strong completeness*, was not required in the distributed model of [BBC⁺19b] (and their dZK proofs do not obtain it), but will be needed for the applications discussed in Section 5. This discussion is summarized in the following definition:

Definition 2.3 (Distributed Zero-Knowledge Proofs). *Let $\widehat{\mathcal{R}} = \widehat{\mathcal{R}}((x^{(1)}, \dots, x^{(k)}), w)$ be a k -distributed relation over a finite field \mathbb{F} . A k -verifier $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ distributed Zero-Knowledge proof $((\varepsilon_p, \varepsilon_r, t_p, t_r)$ -dZK) Π_{dist} for \mathcal{R}_k consists of a prover \mathcal{P} and verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$ satisfying the following:*

- **Syntax.** *The input of each \mathcal{V}_i is an input piece $x^{(i)}$, and the input of \mathcal{P} is $(x^{(1)}, \dots, x^{(k)})$ and a witness w such that $((x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}$.¹¹ The parties interact in rounds over point-to-point channels and a broadcast channel, where the messages sent by a party in round i are determined given a next-message function, and depend on its input, randomness, and messages it received in previous rounds. The protocol terminates after a fixed number of rounds, and each verifier outputs either accept or reject, based on its view (which consists of its input and the messages it received throughout the execution).*
- **Completeness.** *For every $(x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}$, when all parties are honest then all verifiers accept in the execution of Π on $((x^{(1)}, \dots, x^{(k)}), w)$ with probability 1.*
- **(ε_r, t_r) -Strong Completeness.** *For every $(x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}$, in an execution of Π on $((x^{(1)}, \dots, x^{(k)}), w)$ with an honest prover, except with probability at most ε_r all honest verifiers accept, even if t_r verifiers are corrupted, computationally unbounded, and may arbitrarily deviate from the protocol.*
- **(ε_r, t_r) -Soundness.** *For every (possibly malicious and unbounded) prover \mathcal{P}^* , and any $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}(\widehat{\mathcal{R}})$, even if a subset C of at most t_r verifiers are maliciously corrupted, computationally unbounded and colluding with \mathcal{P} , then except with probability ε_r all honest verifiers reject in the execution of Π_{dist} on input $(x^{(1)}, \dots, x^{(k)})$ with prover \mathcal{P}^* colluding with the verifiers in C .*

¹¹We note that w may also be the empty string, e.g., if $\widehat{\mathcal{R}}$ corresponds to a language in P .

- (ε_p, t_p) -**Distributed Zero Knowledge (dZK)**. For every adversary \mathcal{A} corrupting a subset T of at most t_p verifiers, there exists a PPT simulator Sim such that for every $((x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}$, it holds that

$$SD\left(\text{Sim}\left(\left(x^{(j)}\right)_{j \in T}\right), \text{View}_{\Pi, \mathcal{A}}\left(\left(x^{(1)}, \dots, x^{(k)}\right), w\right)\right) \leq \varepsilon_p$$

where $\text{View}_{\Pi, \mathcal{A}}(x^{(1)}, \dots, x^{(k)}, w)$ denotes the view of the adversary \mathcal{A} in an execution of Π with an honest prover on inputs $(x^{(1)}, \dots, x^{(k)}, w)$.

The following notation will be useful.

Notation 1 (t -dZK). We say that a protocol between a prover \mathcal{P} and k verifiers is a t -dZK proof, if it is a $(\text{negl}(s), \text{negl}(s), t, t)$ -dZK proof, where s is a statistical security parameter.

Next, we describe a special structure of dZK proofs, which the systems constructed in this work satisfy. Specifically, the execution is divided into a proof generation phase in which the prover sends a proof share to each verifier, and a verification phase in which the proof shares are verified. We are particularly interested in dZK proofs in which the communication during the verification phase is independent of the size of the verification circuit.

Definition 2.4. We say that a dZK proof is verification efficient if it is a dZK between the prover and k verifiers, whose execution can be divided into a proof generation phase in which the prover sends a message to each verifier, and a verification phase in which all parties interact, and moreover, the communication complexity during the verification phase is $\text{poly}(k, s, \log n)$, where s is a statistical security parameter, and n is the input length. In particular, the communication complexity during verification is independent of the size of the computation.

Remark 2.1 (Round Complexity of dZK Proofs). Our dZK proofs are designed in the coin-tossing hybrid model, in which parties can obtain truly random coins by calling an ideal coin-tossing oracle. Calls to this oracle are done separately from communication rounds, namely parties do not exchange any messages in rounds during which the oracle is called. When counting the round complexity of a dZK proof, the rounds in which the oracle is called are not counted towards the round complexity of the system. Thus, if the coin-tossing oracle is replaced with a secure MPC implementation of coin-tossing, the round complexity of the resultant system will be the sum of the round complexity of the dZK and of the secure coin-tossing.

2.2 Secure Multi-Party Computation (MPC) Protocols

Secure Multi-Party Computation (MPC) protocols allow a set of parties to jointly compute a function of their inputs while guaranteeing correctness of the outputs, and preserving privacy of the inputs, even in the presence of (maliciously) corrupted parties. In this section we set some notation and terminology relating to MPC protocols, which will be used in subsequent sections.

Let Π be an MPC protocol between parties P_1, \dots, P_k . The *view* View_i of party P_i consists of its input, random coin tosses, and all the messages it received throughout the protocol execution.

We will need a notion of *pairwise consistency* between a pair of views, requiring that the messages which the corresponding pair of parties exchanged during the protocol (as reported in the views) are consistent. This is formalized in the following definition:

Definition 2.5 (Pairwise Consistent Views). A pair of views $\text{View}_i, \text{View}_j$ of parties P_i, P_j in an MPC protocol Π is pairwise consistent if the outgoing messages from P_i to P_j implicit in View_i are identical to the incoming messages from P_i to P_j reported in View_j , and vice versa.

Definition 2.6 (ε -Correctness). We say that a k -party protocol Π realizes a deterministic k -party functionality $f(x^{(1)}, \dots, x^{(k)})$ with ε -correctness if for every $x^{(1)}, \dots, x^{(k)}$,

$$\Pr \left[\exists i \in [k] : y_i \neq f(x^{(1)}, \dots, x^{(k)}) \right] \leq \varepsilon$$

where y_i denotes the output of P_i in Π .

We say that Π is perfectly correct if it is ε -correct for $\varepsilon = 0$.

Definition 2.7 ((ε, t) -Privacy). Let $1 \leq t < k$. We say that a protocol Π realizing a k -party functionality f is (ε, t) -private if for every subset $C \subset [k]$ of size $|C| \leq t$ there exists a PPT simulator Sim_C such that for every $x^{(1)}, \dots, x^{(k)}$,

$$SD \left(\text{View}_C(x^{(1)}, \dots, x^{(k)}), \text{Sim}_C \left(\left\{ x^{(i)} \right\}_{i \in C}, f(x^{(1)}, \dots, x^{(k)}) \right) \right) \leq \varepsilon$$

where $\text{View}_C(x^{(1)}, \dots, x^{(k)})$ denotes the joint view of the parties in C (including their inputs, random coin tosses, and the messages they received) in a semi-honest execution of Π in which the parties have inputs $x^{(1)}, \dots, x^{(k)}$.

We say that Π is perfectly t private if it is $(0, t)$ -private.

Definition 2.8 ((ε, t) -Robustness). Let f be a k -party functionality whose outputs are in $\{0, 1\}$, and let $1 \leq t < k$. We say that a protocol Π realizing f is (ε, t) -robust if for every subset $C \subset [k]$ of size $|C| \leq t$ and for every $x^{(1)}, \dots, x^{(k)}$, the following holds. If there exist no $x^{(1')}, \dots, x^{(k')}$ such that: (1) $x^{(i)} = x^{(i')}$ for every $i \notin C$, and (2) $f(x^{(1')}, \dots, x^{(k')}) = 1$, then except with probability ε , all parties $i \notin C$ output 0 in an execution of Π in which the honest parties have inputs $\{x^{(i)}\}_{i \notin C}$, even if the parties in C are maliciously corrupted, colluding, and computationally unbounded.

We say that Π is perfectly t robust if it is $(0, t)$ -robust.

3 Checking Membership in a Robust Code

In this section, we describe and analyze a batch code membership test. As we show in Section 3.1 below, this test yields a *batched* Verifiable Secret Sharing (VSS) scheme – which allows the dealer to share multiple secrets simultaneously – in which the complexity of verifying the shares is *independent* of the batch size. First, we establish some notations.

Our test pertains to Reed-Solomon (RS) codes, defined next. We recall that a $[k, \delta, d]$ code refers to a linear code over some underlying field \mathbb{F} where k is block length, δ is the message length (dimension) and d is the minimal distance.

Definition 3.1 (Reed-Solomon Code). For positive integers k, δ , finite field \mathbb{F} , and a vector $\eta = (\eta_1, \dots, \eta_k) \in \mathbb{F}^k$ of distinct field elements, the code $\text{RS}_{\mathbb{F}, k, \delta, \eta}$ is the $[k, \delta, k - \delta + 1]$ linear code over \mathbb{F} that consists of all k -tuples $(p(\eta_1), \dots, p(\eta_k))$ where p is a polynomial of degree $< \delta$ over \mathbb{F} .

Since we are interested in batch code-membership testing, it will be convenient to view m -tuples of codewords in a linear code L as codewords in an interleaved code L^m . We formally define this notion below.

Definition 3.2 (Interleaved code). Let $L \subset \mathbb{F}^k$ be a $[k, \delta, d]$ linear code over \mathbb{F} . We let L^m denote the $[mk, m\delta, d]$ (interleaved) code over \mathbb{F} whose codewords are all $k \times m$ matrices U such that every column $U[i]$ of U satisfies $U[i] \in L$. For $U \in L^m$ and $j \in [k]$, we denote by U_j the j th symbol (row) of U .

Definition 3.3 (Encoded message). Let $L = \text{RS}_{\mathbb{F},k,\delta,\eta}$ and $\zeta = (\zeta_1, \dots, \zeta_\ell)$ be a sequence of distinct elements of \mathbb{F} for $\ell \leq \delta$. For $u \in L$ we define the message $\text{Dec}_\zeta(u)$ to be $(p_u(\zeta_1), \dots, p_u(\zeta_\ell))$, where p_u is the polynomial (of degree $< \delta$) corresponding to u (i.e., $p_u(x) = \sum_{i=0}^{\delta-1} u_i x^i$). For $U \in L^m$ with columns $U[1], \dots, U[m] \in L$, we let $\text{Dec}_\zeta(U)$ be the length- $m\ell$ vector $x = (x_{11}, \dots, x_{1\ell}, \dots, x_{m1}, \dots, x_{m\ell})$ such that $(x_{i1}, \dots, x_{i\ell}) = \text{Dec}_\zeta(U[i])$ for $i \in [m]$. Finally, when ζ is clear from the context, we say that U encodes x if $x = \text{Dec}_\zeta(U)$.

Private (Interleaved) RS Codes. We will in fact use a *private* variant of the (interleaved) RS code. Intuitively, in a t -private code encoding is randomized, and any subset of t codeword symbols reveals no information about the encoded message (when the codeword was randomly generated).¹²In particular, privacy requires a randomized encoding procedure Enc . We will sometime explicitly state the randomness r used for encoding, denoted by $\text{Enc}(\cdot; r)$. Formally,

Definition 3.4 (Private Code). Let $t, k \in \mathbb{N}$, and $\varepsilon \in [0, 1]$. A code $\mathcal{C} \subseteq \mathbb{F}^k$ with a randomized encoding procedure Enc is (ε, t) -private if for every x, x' , and every subset $\mathcal{I} \subseteq [k]$ of size $|\mathcal{I}| \leq t$, it holds that $\text{SD}(\text{Enc}(x)|_{\mathcal{I}}, \text{Enc}(x')|_{\mathcal{I}}) \leq \varepsilon$, where $\text{Enc}(x)|_{\mathcal{I}}$ denotes the restriction of $\text{Enc}(x)$ to the coordinates in \mathcal{I} , and the distance is over the randomness used to encode x, x' . We say that \mathcal{C} is perfectly t -private if it is $(0, t)$ -private.

Intuitively, to guarantee that the RS codeword reveals no information about the underlying secret, we rely on a *randomized* version of the code which concatenates the message with randomness before encoding it. Specifically, we use the following private version of RS codes (for our purposes, $\ell = 1$ suffices; but the notion easily extends to larger ℓ).

Definition 3.5 (Randomized Reed-Solomon (RRS) Code). For positive integers k, δ , finite field \mathbb{F} , and a vector $\eta = (\eta_1, \dots, \eta_k) \in \mathbb{F}^k$ of distinct field elements, the code $\text{RRS}_{\mathbb{F},k,\delta,\eta}$ is defined by the following encoding and decoding procedures.

- Enc is a PPT procedure that on input $x \in \mathbb{F}$ samples $r \leftarrow \mathbb{F}^{\delta-1}$ and applies the encoding procedure of the RS code $\text{RS}_{\mathbb{F},k,\delta,\eta}$ (Definition 3.1) to (x, r) . That is, it computes $(p(\eta_1), \dots, p(\eta_k))$ where $p(y) = x + \sum_{i=1}^{\delta-1} r_i y^i$.
- Dec is a deterministic procedure that on input a purported codeword $c \in \mathbb{F}^k$ applies the decoding procedure of the RS code $\text{RS}_{\mathbb{F},k,\delta,\eta}$ and (if decoding succeeds) outputs the first symbol of the decoded message.

We will need the following simple fact regarding the RRS code, which follows from the properties of Shamir's secret sharing.

Fact 3.1 (RRS is Robust and Private). For every positive integers k, δ such that $k > 3(\delta - 1)$, any finite field \mathbb{F} , and any vector $\eta = (\eta_1, \dots, \eta_k) \in \mathbb{F}^k$ of distinct field elements, the code $\text{RRS}_{\mathbb{F},k,\delta,\eta}$ is $(0, \delta - 1)$ -private and $(0, \delta - 1)$ -robust.

Moreover, the code is $(0, \delta - 1)$ -private for any $k \geq \delta$.

Batch Verification of RS Codewords. We describe a simple procedure for batch verification of membership in the (randomized) RS code, namely membership in the *Interleaved RS (IRS)* code $\text{RS}_{\mathbb{F},k,\delta,\eta}^m$. First, we recall a Lemma from [BCI⁺20] regarding IRS codes.

¹²We note that a private (robust) code is equivalent to a (robust) secret-sharing scheme – to share, simply randomly encode the message, to robustly reconstruct, decode it while correcting errors if needed.

Lemma 3.2. [BCI⁺20, Theorem 1.2] Let $L = \text{RS}_{\mathbb{F},k',\delta,\eta}$ be a Reed-Solomon code with minimal distance $d = k' - \delta + 1$ and e a positive integer such that $e < d/2$. Suppose $d(U', L^m) > e$. Then, for a random w^* in the column-span of U' , we have

$$\Pr[d(w^*, L') \leq e] \leq k'/|\mathbb{F}|.$$

The IRS Test

Let $L = \text{RS}_{\mathbb{F},k,\delta,\eta}$ and $\delta = t + \ell - 1$. The IRS test is executed between a prover \mathcal{P} and k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$. Let $x \in \mathbb{F}^{\ell \times m}$ (we think of x as a batch of m length- ℓ secrets), and let $U \in L^m$ such that $\text{Dec}_\zeta(U) = x$. In the protocol, \mathcal{P} has input U , and each verifier \mathcal{V}_i has as input the i th row U_i of U . The protocol proceeds as follows.

1. \mathcal{P} samples a random vector $r_b \in \mathbb{F}^\ell$ and a random codeword $U^* \in L$ such that $\text{Dec}_\zeta(U^*) = r_b$. It sends U_i^* to \mathcal{V}_i .
2. The verifiers call $\mathcal{F}_{\text{coin}}$ to obtain a random $r \in_R \mathbb{F}^m$.
3. Each verifier \mathcal{V}_i computes $w_i = \sum_{j=1}^m r_j \cdot U_{i,j} + U_i^*$ and broadcasts w_i .
4. Denote $w = (w_1, \dots, w_k)$. The verifiers accept iff $d(w, L) \leq t$.

Figure 1: Verifying Membership in the IRS Code with t Corruptions

Our batched code-membership test, which we call the *IRS test*, is described in Figure 1. Its properties are summarized in the following theorem.

Theorem 3.1 (IRS Test, Figure 1). *Let $k > 4\delta$ where $\delta = t + \ell$. Then, the protocol described in Figure 1 satisfies the following properties, even if t verifiers are maliciously corrupted.*

- **Correctness.** *If $U \in L^m$ (i.e. the shares held by the parties form a valid codeword), and the prover is honest, then all honest verifiers accept with probability 1.*
- **Soundness/Commitment.** *If $U \notin L^m$, then except with probability $(k - t + 2)/|\mathbb{F}|$ one of the following hold even if the prover and t verifiers are maliciously corrupted and colluding.*
 - *All honest verifiers reject.*
 - *Let H denote the set of honest parties, and let L' and L'^m denote the restrictions of the codes L and L^m (respectively) to the coordinates corresponding to the parties in H . Let U' denote the restriction of U to the coordinates held by the parties in H . Then $d(U', L'^m) \leq t$, and there exists a unique codeword $\tilde{U} \in L'^m$ that agrees with U' on $H - \Delta(U', L'^m)$.*
- **Secrecy.** *For every x, x' , and any subset $T \subseteq [k]$, $|T| \leq t$, we have $\text{View}_T(x) \equiv \text{View}_T(x')$, where $\text{View}_T(x)$ denotes the view of the parties in T in an execution of the protocol on a random encoding U of x (i.e., U is random subject to $\text{Dec}_\zeta(U) = x$), in which the prover and the verifiers $\mathcal{V}_i, i \notin T$ are honest.*

Proof: We prove that our scheme satisfies each of the properties:

Correctness ($U \in L^m$): In this case, the verifiers hold a valid codeword. Since we have at most t malicious verifiers, $d(w, L) \leq t$ where w is the result of the test computed in Step 3. Therefore, the honest verifiers will output accept w.p. 1.

Soundness ($U \notin L^m$): Let H denote the set of honest parties. We consider the restrictions L' and L'^m of the codes L and L^m (respectively) to the coordinates corresponding to the parties in H . Let U' denote the restriction of U to the coordinates held by the parties in H , and w' be the restriction of w to the shares held by H . We consider two cases:

- Case 1: $d(U', L'^m) > t$. Let V denote the matrix containing U and the masking codeword U^* . Since $d(U, L^m) > t$, we have $d(V, L^{m+1}) > t$. Using Lemma 3.2 it holds that, except with probability $(k - t)/|\mathbb{F}|$, the codeword obtained via a random linear combination of V 's rows has distance $> t$ from a valid codeword. If w were a random linear combination of the rows of V , then we would be done. However, recall that in the actual IRS test, U^* has multiplier 1 in the random linear combination (instead of a random multiplier). Observe that if – except with probability $(k - t)/|\mathbb{F}|$ – a random linear combination of V 's rows has more than t errors, then by an averaging argument, there exists a fixed $r^* \neq 0$ such that conditioned on U^* 's multiplier being r^* , the resulting linear combination has more than t errors except with probability $(k - t + 2)/|\mathbb{F}|$. Scaling-down these linear combinations by r^* , the number of errors in the resultant codeword does not change, and U^* 's multiplier becomes 1. Since this distribution of random combinations is identical to the IRS test we can conclude that the soundness of the IRS test will be $(k - t + 2)/|\mathbb{F}|$.
- Case 2: $d(U', L'^m) \leq t$. If the test fails, then soundness holds. Otherwise, there is a unique codeword $U^* \in L^m$ that agrees with U' on $H - \Delta(U', L'^m)$. This is because the distance of the code L^m is bigger than $2t$.

Secrecy: Recall that the malicious verifiers in T holds at most t rows of U , and in the test they obtain a linear combination of the form $w = (r||1)^T U''$, where $U'' \in \mathbb{F}^{k \times (m+1)}$ is a matrix whose columns are codewords in L , and the last column of U'' encodes a random value. In particular, w encodes a random value. At a high-level, secrecy follows from a combination of the t -privacy of the underlying Reed-Solomon code (which guarantees that the shares given to the parties in T reveal no information about x) and the use of a blinding vector r_b (which guarantees that the codeword symbols revealed during verification reveal nothing about x , because the encoded messages are blinded by r_b).

More formally, when $U^x, U^{x'}$ are *random* encodings of x, x' respectively, then any t rows of $U^x, U^{x'}$ are identically distributed (see Fact 3.1 above). Therefore, we can condition on the codeword symbols $(U_i^*)_{i \in T}$ which the prover sent to the verifiers in T in the first round of the IRS test. Under this conditioning, the following is a perfect simulation for $\text{View}_T(x)$: sample a codeword $w^b \in L$ (encoding a random value r_b) which is random subject to agreeing with $(U_i^*)_{i \in T}$, and provide $(w_i^b)_{i \in T}$ to the verifiers in T as the messages from the prover in Step 1. The proof of perfect simulation proceeds by a sequence of hybrids, where \mathcal{H}_0 is $\text{View}_T(x)$, \mathcal{H}_1 replaces the codeword w generated during verification with an encoding w'' of a random value, where the encoding is random subject to being consistent with the shares of the parties in T . (That is, for every $i \in T$, $w''_i = \sum_{j=1}^m r_j \cdot U_{i,j}^x + w_i^b$.) The hybrids are identically distributed because distinguishability implies that for a fixed x one can distinguish between random encodings of $x||r_b$ and $x||r'_b$, given only the shares of the parties in T . This contradicts the privacy of the RRS code. Next, the hybrid \mathcal{H}_2 is obtained by replacing $(U^x[i])_{i \in T}$ with $(U^{x'}[i])_{i \in T}$. Then $\mathcal{H}_1, \mathcal{H}_2$ are identically distributed because $(U^x[i])_{i \in T}, (U^{x'}[i])_{i \in T}$ are identically distributed, and the other values in the distributions are independent of these shares. Finally, \mathcal{H}_3 is $\text{View}_T(x')$, and is identically distributed to \mathcal{H}_2 using the same argu-

ment as the one used to show that $\mathcal{H}_0, \mathcal{H}_1$ are identically distributed.

Extension for Constant-Sized Fields. The protocol and analysis above relies on the field size $|\mathbb{F}|$ being sufficiently large. In order to accommodate secret sharing over small fields, such as Algebraic-Geometric codes [CC06], we extend our protocol and analysis to holds for small fields as follows:

First, we modify the protocol where we will have the verifiers obtain $\sigma = k + \kappa$ random vectors r_1, \dots, r_σ in $\{0, 1\}^m$ for a statistical security parameter σ , and execute Steps 3 and 4 for each of the σ random vectors. Let w_1, \dots, w_σ be the vectors collected in Step 4. We modify Step 4 to have the verifiers accept only if $|\cup_{i=1}^\sigma \Delta(w_i, L)| \leq t$. Completeness and zero-knowledge essentially follow using the same argument as above (when independent blinding vectors are used in each of the σ iterations.) Soundness on the other hand requires a different analysis. To argue soundness, we essentially follow the argument presented in [DI06]. More formally, it suffices to show that the probability that $|\cup_{i=1}^\sigma \Delta(w_i, L)| \leq t$ and all the honest parties not in $\cup_{i=1}^\sigma \Delta(w_i, L)$ do not have a valid codeword in L^m is small.

In more detail, consider an arbitrary subset H of the honest parties and suppose that for some column i in U , the parties in H do not have shares consistent with L . Denote by $v_1(H)$ the restriction of the i th column to the shares held by H . Consider any one of the σ random vectors r_j and fix all of its values except its value in column i . Denote the codeword obtained by combining all the columns except column i by v' . Now we have that either $v'(H)$ or $v'(H) + v_1(H)$ must be inconsistent with L . Therefore, w.p $1/2$ over the r_j , $H \cap \Delta(w_j, L)$ is not empty. This implies that except with probability $1/2^\sigma$, $H \cap \cup_{i=1}^\sigma \Delta(w_i, L)$ is not empty. Taking a union bound over all possible subsets H , we get that, except with probability $1/2^\sigma \cdot 2^k = 1/2^\kappa$, if a subset of the honest parties H has a column inconsistent with L , then $H \cap \cup_{i=1}^\sigma \Delta(w_i, L)$ is not empty. This implies that except with probability $1/2^\kappa$, we have that the codeword held by all honest parties not in $\cup_{i=1}^\sigma \Delta(w_i, L)$ is a valid codeword in L' .

Therefore, we have the following corollary:

Corollary 3.2 (IRS Test, Small Fields). *Let $k > 4\delta$ where $\delta = t + \ell - 1$ and let κ be a statistical security parameter. Then, the modified protocol described above satisfies correctness, secrecy as in Theorem 3.1 and the following soundness guarantee.*

Soundness: *If $U \notin L^m$, then except with probability $1/2^\kappa$ one of the following hold even if the prover and t verifiers are maliciously corrupted and colluding.*

- *All honest verifiers reject.*
- *Let H denote the set of honest parties, and let L' and L'^m denote the restrictions of the codes L and L^m (respectively) to the coordinates corresponding to the parties in H . Let U' denote the restriction of U to the coordinates held by the parties in H . Then $\Delta(U', L'^m) \leq t$, and there exists a unique codeword $\tilde{U} \in L'^m$ that agrees with U' on $H - \Delta(U', L'^m)$.*

3.1 A Batched Verifiable Secret Sharing (VSS) Scheme

In this section, we show that the IRS test of Figure 1 easily gives rise to a batched VSS scheme. This construction is implicit in [BGR98, DI06], and our main contribution is in refining the analysis for large fields to obtain better thresholds t (using recent advances on testing Reed-Solomon codes [BCI⁺20]), as well as showing that the test satisfies all the properties of VSS.

Roughly, our batch VSS allows the dealer to VSS-share multiple secrets simultaneously by encoding the secrets using the RS code, and then all parties run the IRS test to verify that the sharing was honestly generated.

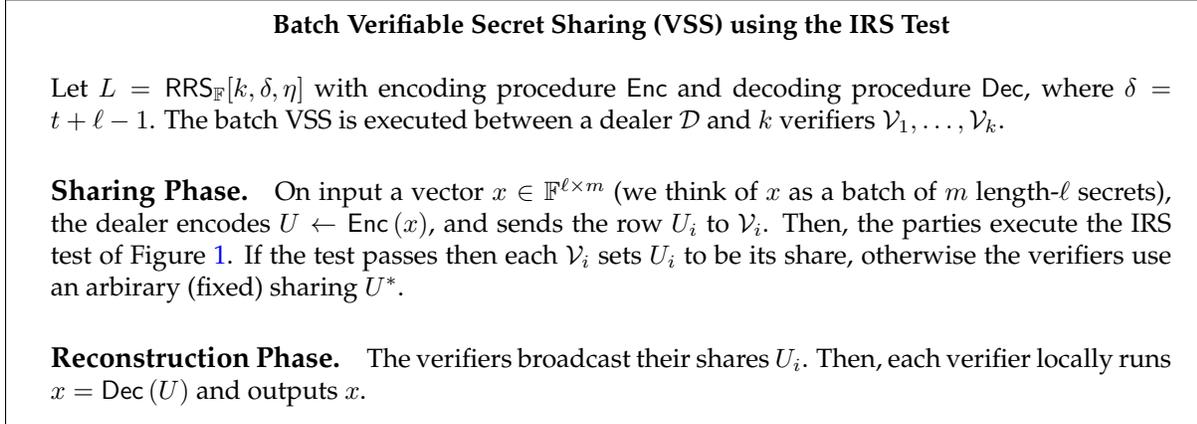


Figure 2: A batch VSS protocol for t corruptions

4 dZK Proofs from Secure MPC Protocols

In this section we describe our dZK proofs and prove Theorem 1.1. We first describe the generic construction from MPC protocols. Then, in Section 4.1 we instantiate the generic construction with specific MPC protocols which yield a dZK with short proofs. Finally, in Section 4.2, we describe a variant of our dZK protocol which obtains only the standard notion of completeness (i.e., does not guarantee strong completeness), with an improved corruption threshold compared to our strongly-complete dZK.

Overview of the dZK Proof System. Let $\mathcal{R} = \mathcal{R}(x, w)$ be a relation over \mathbb{F} . Our dZK proves membership in the corresponding k -distributed relation $\widehat{\mathcal{R}}_{\text{RRS}}$ (see Definition 2.2 in Section 2, and Definition 3.5 in Section 3). Roughly, we employ the MPC-in-the-head paradigm in the following way. The prover generates the proof by emulating “in its head” an MPC protocol Π which checks membership in $\widehat{\mathcal{R}}_{\text{RRS}}$. More specifically, Π is a $(k + 1)$ -party protocol between $\mathcal{P}_0, \dots, \mathcal{P}_k$, in which every $\mathcal{P}_i, i > 0$ has input $x^{(i)}$ and \mathcal{P}_0 holds a corresponding witness w , and the protocols checks whether $((x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}_{\text{RRS}}$. The emulation of Π results in views $\text{View}_0, \dots, \text{View}_k$ of the parties, and the prover sends $\text{View}_i, i \in [k]$ to \mathcal{V}_i (notice that View_0 is not given to any verifier). The verifiers then verify the proof by performing the following. First, they run the IRS test (Figure 1) to verify that their input pieces are close to an RRS codeword. If so, the verifiers call $\mathcal{F}_{\text{coin}}$ to sample a public random value r which will be used when checking pairwise consistency of the views. More specifically, every pair of verifiers exchange short authentication tags which are computed from their views using r . The proof is accepted if these checks pass, allowing for a small (at most t) number of “errors”. This “error tolerance” is essential to guaranteeing strong completeness, namely that corrupted verifiers cannot “frame” an honest prover. We note that this “error tolerance” significantly complicates the soundness analysis. Indeed, even if an inconsistency was revealed, the verifiers cannot immediately reject because that might violate strong completeness. The soundness analysis thus needs to show that a malicious prover cannot exploit the error tolerance to convince verifiers of false claims.

dZK from Secure MPC Protocols

For an NP relation \mathcal{R} over \mathbb{F} , let $\widehat{\mathcal{R}}_{\text{RRS}}$ be the corresponding k -distributed relation (see Definition 2.2, and Definition 3.5), and let $\widehat{\mathcal{L}} := \widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$.^a The dZK proof system is executed between a prover \mathcal{P} and k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$. It employs a $(k+1)$ -party MPC protocol Π for $\widehat{\mathcal{R}}_{\text{RRS}}$, and is parameterized by a bound $t < (k-2)/6$ on the number of corrupt verifiers.

Proof Generation. The prover \mathcal{P} on input $((x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}_{\text{RRS}}$ operates as follows:

1. Runs Π “in its head” with parties P_0, P_1, \dots, P_k holding inputs $w, x^{(1)}, \dots, x^{(k)}$ (respectively).^b That is, it honestly emulates the operations of all parties in Π . Let $\text{View}_1, \dots, \text{View}_k$ denotes the views of P_1, \dots, P_k in this execution, *excluding their inputs*. That is, P_i 's view consists of its coin tosses, and all the messages it received throughout the execution.
2. For every pair $i < j$ of verifiers, picks $r_{ij} \leftarrow \mathbb{F}$.
3. Emulates the prover in Step 1 of the IRS test of Figure 1 (with $\ell = 1$) to generate the messages m_1, \dots, m_k which the prover sends to $\mathcal{V}_1, \dots, \mathcal{V}_k$.
4. For every $i \in [k]$, sends $\text{View}_i, \{r_{ij}\}_{i < j}, \{r_{ji}\}_{j < i}, m_i$ to \mathcal{V}_i .

Verification.

1. The verifiers execute the IRS test of Figure 1 (with $\ell = 1$) on their input pieces $x^{(1)}, \dots, x^{(k)}$, using m_1, \dots, m_k as the messages from \mathcal{P} . For each \mathcal{V}_i , if the i 'th verifier rejects in the IRS test then \mathcal{V}_i outputs reject.
2. The verifiers call $\mathcal{F}_{\text{coin}}$ to obtain a random $r \in_R \mathbb{F}$.
3. Every \mathcal{V}_i performs the following, for every $j \neq i$. Let z_1^j, \dots, z_l^j denote the field elements exchanged between P_i, P_j in the execution of Π , as they appear in View_i . (The messages from P_j to P_i appear in View_i . The messages from P_i to P_j can be computed from View_i .) Let $r'_{ij} := r_{ij}$ if $i < j$, otherwise $r'_{ij} := r_{ji}$. Then \mathcal{V}_i broadcasts $m_{ij} := p_{ij}(r)$ where $p_{ij}(x) := \sum_{f=1}^l z_f^j \cdot x^f + r'_{ij}$.
4. Every \mathcal{V}_i checks local consistency of View_i , by checking that the output of P_i given input $x^{(i)}$ and the messages reported in View_i is 1. If View_i is not locally consistent then \mathcal{V}_i broadcasts a complaint against \mathcal{P} and rejects. Let C_1 denote the set of verifiers who broadcasted a complaint against \mathcal{P} .^c
5. \mathcal{P} broadcasts a set C_2 of parties which it claims are corrupted (i.e., broadcasted false m_{ij} values). Let $C := C_1 \cup C_2$.
6. Every verifier \mathcal{V}_i checks that:
 - (a) $|C| \leq t$.
 - (b) for every $j, l \notin C_2, m_{jl} = m_{lj}$.

If one of the tests failed, then \mathcal{V}_i outputs reject. Otherwise, it outputs accept.

^aWe note that $\widehat{\mathcal{L}}$ is a subset of the code obtained by instantiating Definition 3.2 with the RRS code; this is because in $\widehat{\mathcal{L}}$, not only is every column a RRS codeword, but the underlying encoded message is also in \mathcal{L} .

^bWe note that if \mathcal{R} is a relation in \mathbb{P} then Π can be a protocol for k parties P_1, \dots, P_k ; see Remark 4.3.

^cWe note that Steps 3 and 4 can be implemented in a single round.

Figure 3: A t -dZK Protocol for $k > 6t + 2$

Our dZK proof is described in Figure 3. The following theorem (which is a formal statement of Theorem 1.1) summarizes its properties.

Theorem 4.1 (dZK from MPC-in-the-head). *Let $t_p, t_r, k \in \mathbb{N}$ such that $k > 6t_r + 2$. Let $\widehat{\mathcal{R}}_{\text{RRS}}$ be a k -distributed relation over a field \mathbb{F} , and let Π be a perfectly correct, (ε_p, t_p) -private and perfectly $(3t_r + 1)$ -robust k -party protocol for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then the proof system Π_{dist} of Figure 3 is an $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -dZK for $\widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$, for*

$$\varepsilon_r = \max \left\{ \varepsilon', \binom{k}{2} \frac{N}{|\mathbb{F}|} \right\}$$

where ε' denotes the error of the IRS test (specified in Theorem 3.1), and N bounds the total number of field elements exchanged between a pair of parties in Π .

Proof: Denote $\widehat{\mathcal{L}} := \widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$. We show that the proof system is complete, strongly complete, sound and has ZK.

Completeness. Follows directly from the completeness of Π and the perfect correctness of the IRS test. In particular, if Π has perfect correctness then Π_{dist} also has perfect completeness, whereas if the statistical correctness error of Π is at most ε_{Π} then so is the completeness error of Π_{dist} .

Strong Completeness. Let $(x^{(1)}, \dots, x^{(k)}) \in \widehat{\mathcal{L}}$, and assume that a set C' of at most t verifiers are corrupted. We show that all honest verifiers output accept (with probability 1). Notice first that by the correctness of the IRS test (Theorem 3.1), the check in Step 1 of the verification phase in Figure 3 passes (with probability 1). Secondly, since \mathcal{P} is honest then the views $\text{View}_1, \dots, \text{View}_k$ are the views in an honest execution of Π on inputs $(x^{(1)}, \dots, x^{(k)})$. Therefore, all parties output 1 in Π , so no honest party broadcasts a complaint against \mathcal{P} . Moreover, $m_{i,j} = m_{j,i}$ for every honest i, j , and since \mathcal{P} can compute all the values $m_{i,j}$, then if a (corrupted) verifier \mathcal{V}_i broadcasts an incorrect value, the prover will broadcast a complaint against \mathcal{V}_i . Consequently, $C_1 \cup C_2 \subseteq C'$, so $|C_1 \cup C_2| \leq t$ and $m_{i,j} = m_{j,i}$ for every $i, j \notin C'$. Consequently all honest verifiers accept the proof.

Soundness. Let $(x^{(1)}, \dots, x^{(k)}) \in \widehat{\mathcal{L}}$ denote the input pieces of the verifiers in an execution of Π_{dist} , and assume that a corrupted prover \mathcal{P} colludes with a set C' of at most t_r corrupted verifiers. Since $(x^{(1)}, \dots, x^{(k)}) \in \widehat{\mathcal{L}}$, then either the decoding of $(x^{(1)}, \dots, x^{(k)})$ fails, or it decodes to $x \notin \mathcal{L}$. We show that except with probability $\varepsilon_r := \max \left\{ \varepsilon', \binom{k}{2} \frac{N}{|\mathbb{F}|} \right\}$, all honest verifiers output reject, where ε' denotes the soundness error of the IRS test (Figure 1), and N bounds the total number of field elements exchanged between a pair of parties in Π .

We say that the input pieces $\{x^{(i)}\}_{i \notin C'}$ of the honest parties are *compliant* with a valid encoding of some x' if there exists an $(x'^{(1)}, \dots, x'^{(k)})$ which is a valid encoding of x' , and $x'^{(i)} := x^{(i)}$ except for $i \in C'$ and at most t_r parties $i \notin C'$. (That is, using the notation of Theorem 3.1, if \widetilde{X}' denotes the restriction of $X' = (x'^{(1)}, \dots, x'^{(k)})$ to the parties not in C' , then the distance between \widetilde{X}' , and the restriction of the interleaved RS code to the parties not in C' , is at most t_r .) We consider two possible cases.

Case (1): there exists no x' (even an $x' \notin \mathcal{L}$) such that the input pieces $\{x^{(i)}\}_{i \notin C'}$ are compliant with a valid encoding of x' . Then by the soundness of the IRS test (Theorem 3.1) executed in Step 1 of the verification phase in Figure 3, except with probability ε' all honest verifiers output reject.

Case (2): there exists an x' such that the input pieces $\{x^{(i)}\}_{i \notin C'}$ are compliant with some valid encoding of x' . Let $(x'^{(1)}, \dots, x'^{(k)})$ denote this encoding of x' . Notice that in this case the distance $d((x^{(1)}, \dots, x^{(k)}), (x'^{(1)}, \dots, x'^{(k)}))$ between the input pieces $(x^{(1)}, \dots, x^{(k)})$ of the parties

and $(x^{(1)}, \dots, x^{(k)})$ is at most $2t_r$. Since the RS decoder can correct $(k-1)/2 \geq 2t_r$ errors, then $\text{Dec}(x^{(1)}, \dots, x^{(k)}) = x'$. Since $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, this implies that $x' \notin \mathcal{L}$.

Notice that the views $\{\text{View}_i\}_{i \notin C'}$ which \mathcal{P} sent to the honest verifiers effectively information-theoretically commit \mathcal{P} to these values. We will show that there exists a subset H of at least $k - 3t_r$ honest verifiers such that their views are: (1) pairwise consistent (i.e., for every pair $\mathcal{V}_i, \mathcal{V}_j \in H$, the messages sent from \mathcal{V}_i to \mathcal{V}_j according to View_j are the messages which an honest \mathcal{V}_i in Π would send given its view View_i and input x_i ; and (2) $H \subseteq [k] \setminus (C_2 \cup C')$. In particular, H constitutes an honest majority in Π with inputs $\{x^{(i)}\}_{i \in H}$ and so by the robustness of Π , all parties in H output 0 in Π . Indeed, $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, and moreover there exists no choice of input pieces $x^{(i)}$ for the parties not in H for which the resultant distributed input is in $\widehat{\mathcal{L}}$. This means that $H \subseteq C_1$, and so all honest verifiers output reject in Step 6 of the verification phase in Figure 3 (because $t_r < k - 3t_r \leq |H| \leq |C_1|$, so the check in Step 6a fails). We proceed with the formal argument.

As noted above, the messages which \mathcal{P} sent to the verifiers commit \mathcal{P} to the views of the honest verifiers $i \in [k] \setminus C'$, and this is done *before* the random value r is sampled in Step 2 of the verification phase of Figure 3. By Lemma 4.1, for every $i, j \notin C'$ such that $\text{View}_i, \text{View}_j$ are inconsistent, except with probability $n_{i,j}/|\mathbb{F}|$ it will hold that $m_{i,j} \neq m_{j,i}$, where $n_{i,j}$ is the number of field elements exchanged between P_i, P_j in Π (see Lemma 4.1). Let bad denote the event that for some $i, j \notin C'$, $\text{View}_i, \text{View}_j$ are inconsistent but $m_{i,j} = m_{j,i}$. Then, using the union bound (over all pairs of honest verifiers),

$$\Pr_{r \leftarrow \mathbb{F}}[\text{bad}] = \Pr_{r \leftarrow \mathbb{F}}[\exists i, j \notin C' \text{ s.t. } \text{View}_i, \text{View}_j \text{ are inconsistent but } m_{i,j} = m_{j,i}] \leq \binom{k - |C'|}{2} \cdot \frac{\max_{i,j} \{n_{i,j}\}}{|\mathbb{F}|}.$$

Let $\varepsilon'' := \binom{k - |C'|}{2} \cdot \frac{\max_{i,j} \{n_{i,j}\}}{|\mathbb{F}|}$. In the following, we condition on $\overline{\text{bad}}$.

We note first that we can assume that for every $i, j \notin C'$, if $\text{View}_i, \text{View}_j$ are inconsistent then at least one of i, j is in C_2 . Indeed, since we have conditioned on $\overline{\text{bad}}$, if $\text{View}_i, \text{View}_j$ are inconsistent then necessarily $m_{i,j} \neq m_{j,i}$. Therefore, if $i, j \notin C_2$ then the check in Step 6b of the verification phase of Figure 3 fails, and so all honest verifiers output reject, and soundness holds. We note that this assumption implies that for every $i, j \notin C' \cup C_2$, their views $\text{View}_i, \text{View}_j$ are consistent.

Second, we can further assume that $|C_2| \leq t_r$. Indeed, if $|C_2| > t_r$ then all honest verifiers output reject because the check in Step 6a of the verification phase in Figure 3 fails. Therefore, the set $H' := [k] \setminus C_2$ has size at least $k - t_r$, and it contains at most t_r corrupted verifiers. Let H'' denote the set of honest verifiers in H' (i.e., $H'' = H' \setminus C'$), then $|H''| \geq k - 2t_r$. Moreover, by the case assumption, $\{x^{(i)}\}_{i \notin C'}$ are complaint with a valid encoding $(x^{(1)}, \dots, x^{(k)})$ of x' . Therefore, there exists a subset $H \subseteq H''$ of at least $|H''| - t_r \geq k - 3t_r$ verifiers such that for every $i \in H$: (1) $x^{(i)} = x'^{(i)}$, and (2) \mathcal{V}_i is honest.

Recall that we have shown above that $\text{View}_i, \text{View}_j$ are consistent for every $i, j \in H$ (this holds for all $i, j \notin C' \cup C_2$, and $H \subseteq [k] \setminus (C' \cup C_2)$). Therefore, the views of the verifiers in H correspond to an execution of Π in which at most $3t_r + 1$ parties are corrupted (these include the corrupted verifiers, as well as the parties in C_2 , the parties in $H'' \setminus H$ – whose input pieces are inconsistent with the encoding of x' – and P_0),¹³ and where the inputs of the honest parties are consistent with $(x^{(1)}, \dots, x^{(k)})$. (Here, we use the fact that \mathcal{V}_i determines its view using also its input piece $x^{(i)}$.) Since $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, but $\{x^{(i)}\}_{i \in H}$ are complaint with $(x^{(1)}, \dots, x^{(k)})$ – which is a valid encoding of $x' \notin \mathcal{L}$ – then because $k - 1 \geq 6t_r$ then the error-correction of the RS code guarantees

¹³We note that if the dZK is run for a relation in \mathcal{P} , then P_0 is not needed, and the prover can use an MPC protocol with k parties. In this case, the corruption threshold improves to $k > 6t_r$, and Π is only required to be $(\varepsilon_r, 3t_r)$ -robust see Remark 4.3 below.

that any $3t_r$ errors in $(x^{(1)}, \dots, x^{(k)})$ can be corrected. In particular, this implies that regardless of the choice $(x''^{(i)})_{i \in [k] \setminus H}$ of inputs for the parties in $[k] \setminus H$, the resultant “codeword” will decode to x' , meaning $(x''^{(1)}, \dots, x''^{(k)}) \notin \widehat{\mathcal{L}}$ (where $x''^{(i)} := x^{(i)}$ for every $i \in H$). Therefore, the perfect $(3t_r + 1)$ -robustness of Π guarantees that the parties in H output 0 in Π with probability 1.

Zero-Knowledge. Let \mathcal{I} denote a set of at most t_p corrupted verifiers in an execution of Π_{dist} with an honest prover. Since the r_{ij} 's for honest $\mathcal{V}_i, \mathcal{V}_j$ are random and unknown to the corrupted verifiers, they serve as one-time pads, so m_{ij}, m_{ji} reveal no information about $\text{View}_i, \text{View}_j$. Moreover, for every $i \in \mathcal{I}$, and every $j \in [k]$, $m_{j,i}$ is computable from r_{ij} and View_i , which are both known to \mathcal{V}_i . Moreover, when \mathcal{P} is honest, all complaints (either of verifiers against the prover, or of the prover against verifiers) originate from the corrupted verifiers. Finally, the ZK property of the IRS test (Theorem 3.1) guarantees that the messages which the verifiers in \mathcal{I} receive during the execution of the IRS test (Step 1 of the Verification phase) are perfectly simulatable given only $\{x^{(i)}\}_{i \in \mathcal{I}}$. Therefore, the entire view of the corrupted verifiers can be efficiently simulated from their input pieces $\{x^{(i)}\}_{i \in \mathcal{I}}$ and views $\{\text{View}_i\}_{i \in \mathcal{I}}$. We proceed to formally describe the simulator.

The simulator Sim uses as the simulator Sim_Π for the parties in \mathcal{I} in an execution of Π , and the simulator Sim_{RS} of the IRS test. Sim on input $(x^{(i)})_{i \in \mathcal{I}}$ operates as follows:

- Runs $\text{Sim}_\Pi \left((x^{(i)})_{i \in \mathcal{I}} \right)$ to obtain simulated views $(\text{View}_i^S)_{i \in \mathcal{I}}$.
- For every $i \in \mathcal{I}$ and every $j \neq i$, picks a random mask r'_{ij} .
- Provides $(\text{View}_i^S)_{i \in \mathcal{I}}$ and $(r'_{ij})_{i \in \mathcal{I}, j \neq i}$ to the corrupted verifiers as the proof shares sent from the prover.
- Executes the IRS test with the corrupted verifiers (Step 1 of the Verification phase), using $\text{Sim}_{\text{RS}} \left((x^{(i)})_{i \in \mathcal{I}} \right)$ to simulate the messages which they obtain from the honest parties.
- Picks a random $r' \leftarrow \mathbb{F}$ and sends it to the corrupted verifiers as the output of $\mathcal{F}_{\text{coin}}$.
- Executes Step 3 of the Verification phase with the corrupted verifiers, providing random values m_{ij} as the ones broadcasted by honest i, j , and honestly computing the values m_{ji} for honest j and $i \in \mathcal{I}$.¹⁴ In particular, Sim obtains from the corrupted verifiers the values m_{ij} they would have broadcasted.
- Uses the messages $m_{ij}, i \in \mathcal{I}, j \neq i$ to determine the set C_2 of parties against which the prover complained.
- Outputs the simulated views of the parties in \mathcal{I} , consisting of: $(x^{(i)}, \text{View}'_i)_{i \in \mathcal{I}}, (r'_{ij})_{i \in \mathcal{I}, j \neq i}, r'$, all the messages m_{ij}, m_{ji} , the set C_2 , and the simulated messages which the honest parties sent to the verifiers in \mathcal{I} during the IRS test.

We now show that the simulated and actual views are ε_p -statistically close, though a sequence of hybrids. Notice that $r', (m_{ij}, m_{ji})_{i, j \notin \mathcal{I}}$ and $(r'_{ij})_{i \in \mathcal{I}, j \neq i}$ are identically distributed in both distributions, so we can condition on these values.

\mathcal{H}_0 : This is the real world views of the parties in \mathcal{I} .

¹⁴This can be done because Sim knows the messages sent from \mathcal{P}_j to \mathcal{P}_i , since they appear in View_i^S , and it can use the next message function of \mathcal{P}_i to compute which message an honest \mathcal{P}_i would have sent to \mathcal{P}_j .

\mathcal{H}_1 : \mathcal{H}_1 is obtained from \mathcal{H}_0 by replacing the real views of the corrupted parties during the IRS test with the simulated views generated by Sim_{RS} .

Then $\text{SD}(\mathcal{H}_0, \mathcal{H}_1) = 0$ by the perfect ZK of the IRS test. Indeed, all other values included in $\mathcal{H}_0, \mathcal{H}_1$ can be computed from these views, the real views of \mathcal{I} in Π , and the values we have conditioned on, and applying a function (that has the views in Π , and the values we have conditioned on, hard-wired into it) to the random variables does not increase the statistical distance.

\mathcal{H}_2 : \mathcal{H}_2 is obtained from \mathcal{H}_1 by replacing the real views of the corrupted parties in the execution of Π with the simulated views generated by Sim_{Π} .

Then $\text{SD}(\mathcal{H}_1, \mathcal{H}_2) = \varepsilon_p$ by the (ε_p, t_p) -privacy of Π . Indeed, the (ε_p, t_p) -privacy of Π guarantees that the views of the corrupted parties in a (semi-honest) execution of Π can be efficiently simulated – up to ε_p statistical distance – from $\{x^{(i)}\}_{i \in \mathcal{I}}$. Moreover, all other values included in $\mathcal{H}_1, \mathcal{H}_2$ can be computed from these views and the values we have conditioned on, and applying a function to the random variables does not increase the statistical distance.

We conclude that Π_{dist} has (ε_p, t_p) -dZK. ■

The proof of Theorem 4.1 used the following lemma, which bounds the probability that for a pair of inconsistent $\text{View}_i, \text{View}_j$ of honest verifiers $\mathcal{V}_i, \mathcal{V}_j$ it holds that $m_{i,j} = m_{j,i}$.

Lemma 4.1. *Let $\mathcal{V}_i, \mathcal{V}_j$ for $1 \leq i < j \leq k$ be honest verifiers in the dZK protocol Π_{dist} of Figure 3, and assume that in Step 4 of the proof generation phase of Π_{dist} a corrupted prover \mathcal{P} sent inconsistent views $\text{View}_i, \text{View}_j$ to $\mathcal{V}_i, \mathcal{V}_j$ (respectively). Let $n_{i,j}$ denote the number of field elements exchanged between P_i, P_j in Π .¹⁵ Then the following holds for the values $m_{i,j}, m_{j,i}$ broadcasted in Step 3:*

$$\Pr_{r \leftarrow \mathbb{F}} [m_{i,j} = m_{j,i}] \leq \frac{n_{i,j}}{|\mathbb{F}|}.$$

Proof: The lemma follows immediately from the Schwartz–Zippel Lemma. Indeed, let $z_1^i, \dots, z_{n_{i,j}}^i$ (respectively, $z_1^j, \dots, z_{n_{i,j}}^j$) denote the field elements exchanged between P_i, P_j in Π according to View_i (respectively, View_j). Then by the assumption of the lemma, $(z_1^i, \dots, z_{n_{i,j}}^i) \neq (z_1^j, \dots, z_{n_{i,j}}^j)$.

Let r_{ij}^i, r_{ij}^j denote the random masks which \mathcal{P} sent to $\mathcal{V}_i, \mathcal{V}_j$ (respectively) in Step 4 of the proof generation phase in Figure 3 (notice that if the prover is corrupted then it might be the case that $r_{ij}^i \neq r_{ij}^j$). Then the polynomial $g_{i,j}(y) := (r_{ij}^i - r_{ij}^j) + \sum_{f=1}^{n_{i,j}} (z_f^i - z_f^j) \cdot y^f$ is not identically zero over \mathbb{F} , so by the Schwartz-Zippel lemma,

$$\Pr_{r \leftarrow \mathbb{F}} [g_{i,j}(r) = 0] \leq \frac{n_{i,j}}{|\mathbb{F}|}.$$

The lemma now follows because $m_{i,j} = m_{j,i}$ if and only if $g_{i,j}(r) = 0$ for the random $r \leftarrow \mathbb{F}$ generated in Step 2 of the verification phase of Π_{dist} . (In particular, r was chosen *after* $g_{i,j}$ was fixed.) ■

Remark 4.2. *We note that in Theorem 4.1, we could make due with an MPC protocol Π which is robust against a maliciously corrupted P_0 colluding with a subset of at most $2t_r$ maliciously corrupted parties*

¹⁵The field elements exchanged between P_i, P_j can be computed from the views. For example, the messages which P_i obtained from P_j appear explicitly in View_i , and the messages which P_i sent to P_j can be efficiently computed from View_i . We note that since $\text{View}_i, \text{View}_j$ are inconsistent, it might be the case that the number of field elements exchanged according to View_i is different from the number of elements exchanged according to View_j . In this case we set $n_{i,j}$ to be the larger of the two.

(instead of robustness against arbitrary subsets of $2t_r + 1$ parties). Indeed, in the proof of Theorem 4.1, at most $2t_r$ of the parties P_1, \dots, P_k might be considered “corrupted”, and the the additional $2t_r + 1$ corrupted party is always P_0 (see the proof of the soundness property).

Remark 4.3 (Improved Corruption Threshold for Relations in \mathbf{P}). *When the dZK proof of Figure 3 is executed on a k -distributed relation $\widehat{\mathcal{R}}$ corresponding to a relation \mathcal{R} in \mathbf{P} , then the party P_0 is not needed. Indeed, in this case the protocol Π could be between k parties P_1, \dots, P_k , each holding an input piece. Consequently, in the soundness proof the number of possibly corrupted parties in the execution of Π reported in the views (which \mathcal{P} sent to the verifiers in Step 4 of the proof generation phase in Figure 3) is at most $3t_r$, so in this case it suffices for Π to be $3t_r$ -robust, and that $k > 6t$.*

Remark 4.4 (Reducing the Round Complexity). *For simplicity of the presentation, we chose to separate different parts of the dZK proof of Figure 3 into separate communication rounds. However, we note that the round complexity of the dZK can be easily reduced, by executing the batched VSS verification test of Step 1 in parallel to the tests performed in Step 3 onward. Batched VSS verification (Figure 1) consists of a call to $\mathcal{F}_{\text{coin}}$, and a single communication round of broadcasts. Moreover, the braodcasts of Step 3 and Step 4 can be executed in a single communication round. Therefore, except for the messages from the prover, the dZK will consist of a single communication round of broadcasts from the verifiers. Reducing the number of rounds in this manner preserves the security of the scheme. Indeed, soundness is preserved because the verifiers have no private inputs or coins, so their entire view in the dZK is known to the prover. (This is proved formally in Corollary 4.2 below.) ZK is preserved because both the secrecy of the IRS test, and the ZK of the dZK of Figure 3 (without reducing the rounds) hold with straight-line black-box simulators, and so ZK is preserved under general concurrent composition [KLR06]; see also Remark 4.5 below. For this reason, strong completeness is also preserved. (Indeed, ZK implies full simulation, since the outputs of the honest parties in this case is 1; therefore security holds under general concurrent composition by [KLR06].)*

The complexity of our scheme. The communication complexity of our scheme grows with the communication and randomness complexities of the underlying MPC protocol Π . In particular, the prover’s first message corresponds to the collections of all views introduced by the emulated MPC, whose accumulated size is exactly the sum of the communication and randomness complexities of Π , while the verification phase is proportional to $O(k^2 \cdot \log |\mathbb{F}|)$ and is independent of the NP circuit evaluated in Π . As we demonstrate below, the communication complexity depends on the actual MPC instantiation. The round complexity of our dZK is 3 for protocols with perfect robustness where the first message is sent by the prover, followed by the broadcasted message from the verifiers (these include the broadcasts of Steps 3 and 4, and the broadcasts of the batch VSS verification), where the second message from the prover concludes the protocol. In terms of broadcast complexity, the number of broadcast bits is $O(k^2 \log |\mathbb{F}|)$, and depends on the desired error for the resultant scheme. Specifically, it could be as low as $k^2 \text{polylog}(\text{CC}(\Pi))$, where $\text{CC}(\Pi)$ denotes the communication complexity of Π . Indeed, obtaining negligible error requires that $N/|\mathbb{F}|$ is negligible, where N is the number of field elements exchanged between a pair of parties in Π . Setting, e.g., $|\mathbb{F}| = 2^{\log^2 \text{CC}(\Pi)}$, the number of broadcasted bits would be $O(k^2 \log |\mathbb{F}|) = k^2 \cdot \text{poly log CC}(\Pi)$. In particular, our dZK scheme is verification efficient (see Definition 2.4).

Strongly-Complete dZK Proofs in 2 Rounds. The round complexity of our dZK proofs can be further reduced to 2 rounds by only slightly increasing the communication complexity during verification. However, the (asymptotic) communication complexity remains unchanged, and in particular the system is still verification efficient. This is summarized in the following corollary.

Corollary 4.2 (2-Round dZK Proofs (using Ideal Coin-Tossing)). *Let $t_p, t_r, k \in \mathbb{N}$ such that $k > 6t_r + 2$. Let $\widehat{\mathcal{R}}_{\text{RRS}}$ be a k -distributed relation over a field \mathbb{F} , and let Π be a perfectly correct, (ε_p, t_p) -private and perfectly $(3t_r + 1)$ -robust k -party protocol for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then there exists a 2-round, verification-efficient $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -dZK for $\widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$, for*

$$\varepsilon_r = \max \left\{ \varepsilon', \binom{k}{2} \frac{N}{|\mathbb{F}|} \right\}$$

where ε' denotes the error of the IRS test (as specified in Theorem 3.1), and N is a bound on the total number of field elements exchanged between a pair of parties in Π . Moreover, the verification phase consists only of broadcasting of $O(k^2)$ field elements.

The high-level idea is to compress the rounds during the verification phase of Figure 3 by collapsing the verifier complaint round (Steps 3 and 4 of the verification phase) and the prover complaint round (Step 5 of the verification phase). This can be done by having the prover broadcast all the MAC values that should have been sent in Step 3.

Proof of Corollary 4.2: We explain how to compress the rounds of the protocol of Figure 3 into two rounds. The proof generation phase is identical to that of Figure 3, and the first interaction round consists of the prover sending the proof shares, and the prover messages in the IRS test, as in Step 4 of the proof generation of Figure 3. Following this step, the parties execute the verification phase, during which the verifiers call the coin tossing oracle to generate the coins needed for the IRS test, as well as the random r used to generate the messages of Step 3. In the second (and final) interaction round, the parties perform the following:

1. Each verifier \mathcal{V}_i sends his broadcasted messages of Step 3 of the IRS test (Figure 1).
2. Each verifier \mathcal{V}_i sends the MACs it sends in Step 3 of the verification phase in Figure 3, as well as the message it broadcasts in Step 4.
3. The prover \mathcal{P} broadcasts, for every pair $\mathcal{V}_i, \mathcal{V}_j, i < j$ of verifiers, the MAC value $r'_{i,j}$ which these verifiers should have broadcasted.
4. Each verifier \mathcal{V}_i computes the set C_1 as in Figure 3. Additionally, it computes the set C_2 as the set of all verifiers \mathcal{V}_i such that there exists a verifier $\mathcal{V}_j \neq \mathcal{V}_i$ for which $r_{ij} \neq r'_{i,j}$ if $i < j$ ($r_{ij} \neq r'_{j,i}$ if $j < i$). \mathcal{V}_i then makes its decision based on these sets (instead of the ones computed in Figure 3).

Clearly, the protocol has 2 rounds, and the communication complexity during verification is as specified in the corollary's statement (see paragraph above about the complexity of our dZK scheme). We now prove the protocol is a strongly-complete dZK.

Completeness follows identically to the proof of Theorem 4.1.

Strong completeness. When the prover is honest and at most t servers are corrupted, then the IRS test is executed on a valid codeword, and the corrupted verifiers can only affect t of its symbols. This holds regardless of the random challenge used in the test, and any other information available to the corrupted servers. Therefore, the IRS test passes even when it is executed in parallel to the rest of the protocol. Moreover, C contains only corrupted verifiers (by definition, and this is regardless of the fact that rushing corrupted verifiers can first see the MACs sent by the prover) and therefore $|C| \leq t$ so all honest verifiers accept.

Zero-Knowledge. There are two differences in this context between the 2-round protocol and the protocol of Figure 3: (1) the messages of the IRS test are broadcasted in parallel to the broadcasts checking the validity of the MPC in the head execution; and (2) the prover broadcasts are sent in parallel to the broadcasts of the verifiers. Regarding (2), when the prover is honest then the MACs he broadcasts are identical to the ones sent by the honest verifiers, so this does not affect the simulation. As for (1), the messages broadcasted by the honest parties during the IRS test and when verifying the MPC in the head execution are independent of each other, and independent of the operations of corrupted parties (since corrupted verifiers only communicate with the honest parties in the final communication round). Since the IRS test and the MPC in the head verification can be simulated using straight-line simulators, zero-knowledge holds even when these two tests are performed in parallel.

Soundness. We show the protocol is sound by reduction to the security of the dZK of Figure 3. The high-level reason is that the prover has full knowledge in the execution, namely it knows all the input pieces, the witness, and the outcomes of the coin tosses. Therefore, if there were a winning strategy for the prover in the compressed protocol, causing the honest verifiers to accept with probability $> \epsilon_r$, then this strategy could have been used by the prover in the protocol of Figure 3, contradicting Theorem 4.1. If the prover had the same knowledge throughout the execution of the protocol in both protocols, then the reduction would have been immediate. The only difference here is that the random coins used to check the MPC in the head execution (Step 2 of the verification phase of Figure 3) are known to the adversary in the compressed protocol *before the messages of the IRS test are sent*. However, the main point here is that the coins are revealed *after* the prover has “committed” (by sending messages to the honest parties in the first round) to his messages in the IRS test, as well as to the codeword symbols to be used in the test. Once these values have been committed, the outcome of the IRS test is determined solely by the coin toss and is independent of how the corrupted parties operate. Therefore, soundness is preserved. ■

4.1 Instantiations and Extensions

We first note that our dZK proofs remain secure under parallel composition. This observation will be used in Section 5.3 when our dZK proofs are used to build maliciously-secure protocols with identifiable abort.

Remark 4.5 (Parallel Composition for dZK). *Notice that we prove zero-knowledge of our dZK protocol (Theorem 4.1) via a straight-line black-box simulator. [KLR06] prove that if security is proved via a straight-line simulator, where all honest parties receive their inputs before the protocol begins, then the protocol is secure under general concurrent composition. Since in the context of parallel composition, the inputs are indeed fixed for the honest parties before the execution begins, our dZK protocol is secure under parallel composition.*

4.1.1 Revisiting [IKOS07] in the Distributed Setting

The seminal work of [IKOS07, IKOS09] has shown how to use secure Multi-Party Computation (MPC) protocols to construct ZK argument systems, where different instances of MPC protocols imply different proof lengths and soundness analyses. This so-called “MPC in the head” paradigm has attracted much attention lately due to its broad applicability and practicality [GMO16, AHIV17, CDG⁺17, KKW18, BFH⁺20, GSV21]. In this section we instantiate our protocol with two information theoretic protocols from [DI06] and [DIK10]. The former instantiation requires coping with statistical robustness (and further requires revisiting a claim from [IKOS07]),

while the latter one is perfectly robust (and so we can directly use the soundness analysis of Theorem 4.1 and Corollary 4.2). In more details, let \mathcal{L} be a language in NP, and let $\mathcal{R}(x, w)$ be the corresponding NP-relation. Let f be the following k -argument function (for $k \geq 3$), corresponding to \mathcal{R} :

$$f(x, w_1, \dots, w_k) = \mathcal{R}(x, w_1 \oplus \dots \oplus w_k).$$

Namely, f is a k -party functionality, where the first argument x is a public input known to all k parties, w_i is a private input of player P_i , and all parties receive the output.

An Instantiation Based on Perfectly Robust MPC. Our first instantiation is based on the perfectly secure MPC protocol of [DIK10, Thm. 1]. The communication complexity of the protocol is $O(\log k \cdot \log |C| \cdot |C|) + d^2 \cdot \text{poly}(k, \log |C|)$ field elements, where C is the computed circuit, $|C|$ is the number of multiplication gates in C and d is the depth of C . The corruption threshold of this protocol is $t < k/3$. Then [DIK10] implies the following corollary (see Notation 1 for the definition of t -dZK).

Corollary 4.3. *Let $k \in \mathbb{N}$, and let \mathbb{F}_k be a field of size k^l for a sufficiently large $l \in \mathbb{N}$. Then the following holds for any k -distributed relation $\widehat{\mathcal{R}}_{\text{RRS}}$ that can be verified by a circuit C over \mathbb{F}_k . Assuming ideal coin-tossing, there exists a 2-round $\Omega(k)$ -dZK proof with (total) proof length $O(\log k \cdot \log |C| \cdot |C|) + d^2 \cdot \text{poly}(k, \log |C|)$ field elements, where $|C|$ denotes the number of gates in C , and d denotes its depth. Furthermore, the communication complexity during verification of the proof shares is k^2 field elements.*

Coping with Statistical Robustness. Recall that Theorem 4.1 requires perfect robustness which implies that if there exist no choice of inputs for the corrupted parties, which – together with the inputs of the honest parties – satisfies the relation, then for every set of random strings the outcome would indeed be 0. To contend with statistical robustness error, [IKOS07] needed to refine their soundness analysis. This is due to the fact that the malicious prover knows all the randomness selected for the emulated parties, including the random coins of the honest parties, which are *unknown* to an adversary attacking the protocol (who only knows the inputs and randomness of corrupted parties). In such a case, even a single corrupted party may potentially cause an incorrect output. Moreover, using a coin tossing oracle to choose the randomness used by the prover would not help here since upon seeing the entire randomness, a corrupted prover may still coordinate an attack on the protocol, leveraging the fact that it knows all inputs and randomness.

They therefore revised their soundness analysis, requiring that the underlying protocol has a two-phase structure and the prover can therefore likewise “corrupt” the MPC parties in two phases, where corruptions in the second phase take place adaptively. The verifier’s challenge in this proof is divided into two parts, where in the first step the verifier engages in a coin-tossing protocol with the prover to fix the randomness of the MPC parties. (Since we work in the $\mathcal{F}_{\text{coin}}$ -hybrid model, we can instead call $\mathcal{F}_{\text{coin}}$ to generate these coins.) The challenge of the second step is required to sample the set of parties whose views will be opened for the consistency check. For the MPC-in-the-head instantiation, [IKOS07] defined a topology with a single input client I and k output parties P_1, \dots, P_k . We adopt their definition of two-phase protocols.

Definition 4.1 (Adaptively robust two-phase protocol). *Let $t_r \in \mathbb{N}$ and let Π be a $k + 1$ -party two-phase MPC protocol (involving an input client I and k parties P_i), and let f be a function computed by an input client I and k parties P_1, \dots, P_k . We say that Π realizes f with adaptive (ε_r, t_r) -robustness if any computationally unbounded adversary can only win the following game with probability ε_r . First, the adversary picks a false statement x (such that $(x, w) \notin \mathcal{R}$ for all w), a set T_1 of at most t_r corrupted parties,*

and random inputs r_i for all uncorrected parties. Now the adversary runs Phase 1 of Π , arbitrarily controlling I and the parties in T_1 . Once Phase 1 terminates, $\mathcal{F}_{\text{coin}}$ is invoked, generating a random challenge r . Based on r , the adversary can corrupt at most $t_r - |T_1|$ additional parties, and continues to interact with the honest parties during Phase 2 of the protocol. The adversary wins if some party that was never corrupted outputs 1.

If ε_r is negligible, then we say that Π is t_r -adaptive statistical robust. Adapting this instantiation to the distributed setting can be done as follows. The proof generation phase stays the same except that it only captures Phase 1 from [IKOS07]. In Step 2, the parties call $\mathcal{F}_{\text{coin}}$ to generate the coins r used by the emulated MPC parties in Phase 2. The prover then runs another phase of proof generation in which it generates the parties' views in Phase 2 of the MPC protocol when using randomness r . The check of Step 4 additionally verifies consistency with r for the computations performed during Phase 2 of the protocol. Note that the round complexity has increased by one round due to the additional MPC phase (whereas in [IKOS07] this modification requires two additional rounds). This gives the following theorem, where $\widehat{\mathcal{R}}_{\text{RS}}$ is defined in Definition 2.2 (Section 2).

Theorem 4.4. *Let $s \in \mathbb{N}$ be a statistical security parameter, and let $t_p, t_r, k \in \mathbb{N}$ such that $k > 6t_r + 2$. Let $\widehat{\mathcal{R}}_{\text{RRS}}$ be a k -distributed relation over a field \mathbb{F} , and let Π be a perfectly correct, (ε_p, t_p) -private and t_r -adaptive statistical robust k -party protocol for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then the proof system Π_{dist} of Figure 3 (subjected to the revisions described above), is a 3-round $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -dZK for $\widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$, for*

$$\varepsilon_r = \max \left\{ \varepsilon', \binom{k}{2} \frac{N}{|\mathbb{F}|} + \text{negl}(s) \right\}$$

where ε' denotes the error of the IRS test (as specified in Theorem 3.1), N is a bound on the total number of field elements exchanged between a pair of parties in Π , and $\text{negl}(s)$ is the robustness error of Π .

Constant rate dZK. Based on the statistically-robust MPC instantiation from [DI06], [IKOS07] constructed a constant rate zero-knowledge protocol where the proof length is a constant multiple of the circuit size $|C|$ used to verify the underlying NP relation. We shall discuss next how to adapt this result to the distributed setting. Note first that the network topology in [DI06] is defined by a constant number of input clients who secret share their inputs to a large set of servers. The servers securely evaluate the circuit layer by layer, while ensuring robustness using correctness tests that are carried out with the clients (which will imply statistical robustness). If all the tests are verified successfully, then the servers reveal to the clients the output sharing. Finally, to get constant rate, [IKOS07] embedded the computation in [DI06] over small fields, instantiated with Algebraic-Geometric codes [CC06]. For that we need to consider a modified IRS test whose analysis holds for such small fields (see Corollary 3.2). Finally, the overall size of the views generated in both phases is $O(|C|) + \text{poly}(k, \log |C|)$. This yields the following corollary.

Corollary 4.5. *Let \mathbb{F} be a constant-sized field. Then assuming ideal coin-tossing, any k -distributed relation $\widehat{\mathcal{R}}_{\text{RRS}}$ that can be verified by a circuit C over \mathbb{F} has a 4-round $\Omega(k)$ -dZK proof. Moreover, the (total) proof length is $O(|C|) + \text{poly}(k, \log |C|)$ field elements, where $|C|$ denotes the number of gates in C , and the communication complexity during verification is $O(k^2 + s)$ field elements, where s is a statistical security parameter.*

4.2 The Case of dZK Without Strong Completeness

In this section we describe a dZK protocol *without strong completeness* – but with a better corruption threshold – based on the techniques employed to construct our strongly-complete scheme (Figure 3). This protocol achieves all properties of Definition 2.3 except for strong completeness, and in particular it is a secure dZK in “setting II” of [BBC⁺19a] (i.e., where soundness holds against corrupted prover *and* verifiers). Thus, this gives an alternative construction to the ones presented in [BBC⁺19a]. Specifically, we prove the following, where we refer to a dZK protocol without strong completeness as a *weak* dZK.

Corollary 4.6 (2-Round Weak dZK (using Ideal Coin-Tossing)). *Let $t_p, t_r, k \in \mathbb{N}$ such that $k > 2t_r + 2$. Let $\widehat{\mathcal{R}}_{\text{RRS}}$ be a k -distributed relation over a field \mathbb{F} , and let Π be a perfectly correct, (ε_p, t_p) -private and perfectly $(t_r + 1)$ -robust k -party protocol for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then there exists a 2-round, verification-efficient weak $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -dZK for $\widehat{\mathcal{L}}(\widehat{\mathcal{R}}_{\text{RRS}})$, for $\varepsilon_r = O\left(\binom{k}{2} \frac{N}{|\mathbb{F}|}\right)$, where N is a bound on the total number of field elements exchanged between a pair of parties in Π . Moreover, the verification phase consists only of broadcasting of $O(k^2)$ field elements.*

The high-level idea is that if strong completeness is not required then verifiers can immediately reject if an inconsistency is identified, namely there is no need for dispute resolution, as we do in Figure 3. This allows us to remove some of the checks performed in Figure 3, and to use a simplified IRS test to check the inputs. As a result, we can handle a higher corruption threshold.

Proof of Corollary 4.6: We first describe the weak dZK system, which is a variant of the dZK system of Figure 3. The protocol consists of 2 rounds. In the first round, the prover executes the proof generation phase as in Figure 3. In the second round, the parties execute the following verification phase (all messages are broadcasted in a single communication round):

1. Each verifier \mathcal{V}_i sends his broadcasted messages of Step 3 of the IRS test (Figure 1, here $\ell = 1$).
2. Each verifier \mathcal{V}_i sends the MACs it sends in Step 3 of the verification phase in Figure 3, as well as the message it broadcasts in Step 4.
3. Each verifier accepts if and only if the following tests pass:
 - (a) All MAC pairs are consistent, i.e., that for every $i \neq j$ it holds that $r_{ij} = r_{ji}$.
 - (b) No verifier broadcasted a complaint against the prover.
 - (c) The messages broadcasted by the verifiers as part of the IRS test form a *valid codeword*. In particular, here we *modify* Step 4 of the IRS test of Figure 1, where the verifier accept if and only if $w \in L$ (using the notations of Figure 1), otherwise it rejects.

Clearly, the protocol has only 2 rounds and the communication complexity during verification is as stated (the proof is similar to that of Corollary 4.2). We now prove that the protocol satisfies the properties of a dZK proof (except for strong completeness).

Completeness follows identically to the proof of Theorem 4.1.

Zero-Knowledge follows identically to the proof of Corollary 4.2 (because the same messages are sent in the dZK described above, and the dZK proof of Corollary 4.2).

Soundness. Let $C, |C| \leq t$ denote the set of corrupted verifiers, and $H = [k] \setminus C$ denote the honest parties. We say that the input pieces $\{x^{(i)}\}_{i \notin C}$ of the honest parties are *compliant* with a valid encoding of some x' if there exists an $(x'^{(1)}, \dots, x'^{(k)})$ which is a valid encoding of x' , and $x'^{(i)} := x^{(i)}$ for every $i \in H$.

We first show that the modified IRS test described above (in which the verifiers reject if the broadcasted word is not a codeword) satisfies the following soundness guarantee. Assume that $k > 2t_r$. (Using the notation of Section 3, we have $\ell = 1$ and $\delta = t_r$) If there exists no x' such that the input pieces $\{x^{(i)}\}_{i \in H}$ of the honest parties are compliant with a valid encoding of x' , then except with probability $1/|\mathbb{F}|$ all honest verifiers reject the test, even when the prover colludes with the parties in C. This is because when $k > 2t_r$ then the RS code has distance at least $k - t_r + 1 > t_r$. Therefore, except with probability $1/|\mathbb{F}|$, the linear combination computed by the honest parties is not compliant with any valid codeword. Consequently, the word broadcasted in Step 3 of Figure 1 would not be a codeword, regardless of the messages broadcasted by the corrupted verifiers during the modified IRS test (see, e.g., [DI06, Lemma 2] for a full proof).

We consider two possible cases.

Case (1): there exists no x' (even an $x' \notin \mathcal{L}$) such that the input pieces $\{x^{(i)}\}_{i \in H}$ are compliant with a valid encoding of x' . Then by the soundness of the modified IRS test (discussed above), all honest verifiers output reject except with probability $1/|\mathbb{F}|$.

Case (2): there exists an x' such that the input pieces $\{x^{(i)}\}_{i \in H}$ are compliant with some valid encoding of x' . Let $(x'^{(1)}, \dots, x'^{(k)})$ denote this encoding of x' . Notice that in this case the distance $d((x^{(1)}, \dots, x^{(k)}), (x'^{(1)}, \dots, x'^{(k)}))$ between the input pieces $(x^{(1)}, \dots, x^{(k)})$ of the parties and $(x'^{(1)}, \dots, x'^{(k)})$ is at most t_r . Since the RS decoder can correct $(k - 1)/2 \geq t_r$ errors, then $\text{Dec}(x^{(1)}, \dots, x^{(k)}) = x'$. Since $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, this implies that $x' \notin \mathcal{L}$.

Notice that the views $\{\text{View}_i\}_{i \in H}$ which \mathcal{P} sent to the honest verifiers effectively information-theoretically commit \mathcal{P} to these values. Since this is done *before* the random coin toss (used to compute the MACs) Lemma 4.1 guarantees that for every $i, j \in H$ such that $\text{View}_i, \text{View}_j$ are inconsistent, except with probability $n_{i,j}/|\mathbb{F}|$ their MACs will be different, where $n_{i,j}$ is the number of field elements exchanged between P_i, P_j in Π (see Lemma 4.1). Let bad denote the event that for some $i, j \in H$, $\text{View}_i, \text{View}_j$ are inconsistent but their MACs are equal. Then, using the union bound (over all pairs of honest verifiers),

$$\Pr_{r \leftarrow \mathbb{F}}[\text{bad}] = \Pr_{r \leftarrow \mathbb{F}}[\exists i, j \in H \text{ s.t. } \text{View}_i, \text{View}_j \text{ are inconsistent but their MACs are equal}]$$

is at most $\binom{k-|C|}{2} \cdot \frac{\max_{i,j} \{n_{i,j}\}}{|\mathbb{F}|}$. Let $\varepsilon'' := \binom{k-|C|}{2} \cdot \frac{\max_{i,j} \{n_{i,j}\}}{|\mathbb{F}|}$.

Conditioned on bad, if the honest verifiers accept, then the views $\{\text{View}_i\}_{i \in H}$ are all pairwise consistent, and therefore they correspond to an execution of Π with at least $k - t_r - 1$ honest parties (i.e., an honest majority), in which the inputs of the honest parties are consistent with $(x^{(1)}, \dots, x^{(k)})$ (because the view of every honest verifier is also locally consistent). Since $(x^{(1)}, \dots, x^{(k)}) \notin \widehat{\mathcal{L}}$, but $\{x^{(i)}\}_{i \in H}$ are compliant with $(x'^{(1)}, \dots, x'^{(k)})$ – which is a valid encoding of $x' \notin \mathcal{L}$ – then because $k - 1 \geq 2t_r$ then the error-correction of the RS code guarantees that any t_r errors in $(x^{(1)}, \dots, x^{(k)})$ can be corrected. In particular, this implies that regardless of the choice $(x''^{(i)})_{i \in C}$ of inputs for the corrupted parties, the resultant “codeword” will decode to x' , meaning $(x''^{(1)}, \dots, x''^{(k)}) \notin \widehat{\mathcal{L}}$ (where $x''^{(i)} := x^{(i)}$ for every $i \in H$). Therefore, the perfect $(t_r + 1)$ -robustness of Π guarantees that the parties in H output 0 in Π with probability 1. ■

By instantiating Theorem 4.6 with the protocols of [DIK10] and [DI06] as in Section 4.1, we obtain the following corollaries:

Corollary 4.7 (2-Round weak dZK, Quasilinear Communication). *Let $k \in \mathbb{N}$, and let \mathbb{F}_k be a field of size k^l for a sufficiently large $l \in \mathbb{N}$. Then the following holds for any k -distributed relation $\widehat{\mathcal{R}}_{\text{RRS}}$ that can be verified by a circuit C over \mathbb{F}_k . Assuming ideal coin-tossing, there exists a 2-round weak t -dZK proof for $k > 2(t + 1)$, with (total) proof length $O(\log k \cdot \log |C| \cdot |C|) + d^2 \cdot \text{poly}(k, \log |C|)$ field elements, where*

$|C|$ denotes the number of gates in C , and d denotes its depth. Furthermore, the communication complexity during verification of the proof shares is k^2 field elements.

Corollary 4.8 (2-Round weak dZK, Linear Communication). *Let \mathbb{F} be a constant-sized field. Then assuming ideal coin-tossing, any k -distributed relation $\widehat{\mathcal{R}}_{\text{RRS}}$ that can be verified by a circuit C over \mathbb{F} has a 4-round weak t -dZK proof for $k > 2(t+1)$. Moreover, the (total) proof length is $O(|C|) + \text{poly}(k, \log |C|)$ field elements, where $|C|$ denotes the number of gates in C , and the communication complexity during verification is $O(k^2 + s)$ field elements, where s is a statistical security parameter.*

Comparison with [BBC⁺19a]. Theorem 4.6 gives a dZK proof *without strong completeness* for any language encoded using the RRS code. However, it extends to languages encoded using any robust linear code, as long as the code is t -private in the sense that any t codeword symbols reveal no information about the underlying message. Boneh et al. [BBC⁺19a, Thm. 6.16] design a dZK proof for encoded languages, based on fully-linear IOPs. To obtain a 2-round dZK (in the coin-tossing hybrid model), their construction needs to be instantiated with a fully linear PCP. They construct such a PCP for general circuits (see [BBC⁺19a, Thm. 4.3], where the G -gates are simply multiplication gates). The proof size is then linear in the size of the circuit verifying the relation, the communication complexity during verification is $O(k)$ field elements, and the proof is ZK against $k - 1$ verifiers, and sound against a prover colluding with $t < k/2$ verifiers. In comparison, we get ZK and soundness against $t < (k - 2)/2$ verifiers with proof length which is quasi-linear in the circuit size and $O(k^2)$ field elements communicated during verification. Thus, our dZK obtains comparable corruption thresholds (for soundness) and slightly worse communication complexity. This might indicate that the reason for the lower corruption threshold in our strongly-complete verification-efficient dZK (roughly $k/6$ compared to $k/2$ in dZKs without strong completeness) could be due to the strong completeness property, and not the use of MPC in the head.

5 Applications

5.1 Certifiable Verifiable Secret Sharing

In this section we describe our construction of a certifiable VSS (cVSS) scheme. We first recall the definition of cVSS [IW14]. Intuitively, cVSS is a certifiable variant of VSS, in which the dealer \mathcal{D} not only commits to some secret x , but also proves that it satisfies some predicate, namely that $x \in R$ for some (NP) language L . similar to standard VSS, cVSS is a $(k + 1)$ -party two-phase protocol consisting of a Sharing phase in which \mathcal{D} commits to x by distributing shares between the parties $\mathcal{P}_1, \dots, \mathcal{P}_k$, and a Reconstruction phase in which the parties reconstruct x from their shares. cVSS guarantees correctness and secrecy as in standard VSS, i.e., if the dealer is honest then the shares will reconstruct to x even if t parties are corrupted, and any subset of t corrupted parties learn nothing about x during the Sharing phase. cVSS guarantees a stronger notion of binding compared to VSS. Indeed, VSS guarantees that when the Sharing phase terminates, there exists some x^* which will necessarily be reconstructed during the Reconstruction phase, even if \mathcal{D} and a subset of t of the parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ maliciously cheat throughout the protocol execution. cVSS additionally guarantees that $x^* \in L$. Finally, cVSS has a zero-knowledge guarantee which extends the secrecy property of standard VSS. Specifically, any subset of t parties learn nothing about the NP witness (in cases where it exists, namely if L is an NP language) throughout the execution, namely not even when x is revealed in the Reconstruction phase. This is formalized in the following definition.

Definition 5.1 (cVSS). Let $\varepsilon \in [0, 1]$, let $t_p, t_r, k \in \mathbb{N}$, and let $\mathcal{R} = \mathcal{R}(x, w)$ be an NP relation with a corresponding NP-language $\mathcal{L}_{\mathcal{R}}$. We say that a 2-phase protocol Π between a dealer \mathcal{D} and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ is an $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -Certifiable Verifiable Secret Sharing $((\varepsilon_p, \varepsilon_r, t_p, t_r)$ -cVSS) scheme for \mathcal{R} if it satisfies the following.

- **Syntax.** The protocol has two phases: a Sharing phase and a Reconstruction phase. In the sharing phase, the dealer \mathcal{D} has input x and a corresponding witness w , and the parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ have no input. The output of \mathcal{P}_i in this phase is a share Share_i , and \mathcal{D} has no output. In the reconstruction phase, the input of each \mathcal{P}_i is Share_i , and \mathcal{D} has no input. The output of the Reconstruction phase is an $x' \in \mathcal{L}(\mathcal{R})$.¹⁶
- **Strong Correctness.** For every $(x, w) \in \mathcal{R}$, in an execution of Π with an honest dealer \mathcal{D} that has input (x, w) , except with probability ε_r the output of the reconstruction phase is x , even if t parties are corrupted, computationally unbounded, and may arbitrarily deviate from the protocol.
- **Binding.** For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_r$, the following holds except with ε_r failure probability over the randomness of the sharing phase. At the end of the sharing phase of Π in which the parties in $\{\mathcal{P}_1, \dots, \mathcal{P}_k\} \setminus T$ are honest, there exists a unique $x^* \in \mathcal{L}_{\mathcal{R}}$ such that the output of the parties in $\{\mathcal{P}_1, \dots, \mathcal{P}_k\} \setminus T$ in the reconstruction phase will be x^* , regardless of the messages sent by \mathcal{D} and the parties in T during the reconstruction phase. This holds even if \mathcal{D} and the parties in T are corrupted, computationally unbounded, and arbitrarily deviate from the protocol throughout the computation (including during the sharing phase).
- **Secrecy.** For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_p$, there exists a simulator Sim_T such that for every $(x, w) \in \mathcal{R}$, $\text{SD}(\text{Sim}_T(|x|), \text{View}_T(x, w)) \leq \varepsilon_p$, where $\text{View}_T(x, w)$ denotes the joint view of the parties in T in an execution of the sharing phase of Π with an honest dealer \mathcal{D} that has input (x, w) , and in which the parties in T may be corrupted, computationally unbounded, and may arbitrarily deviate from the protocol.
- **Zero Knowledge.** For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_p$, there exists a simulator Sim_T^f such that for every $(x, w) \in \mathcal{R}$, $\text{SD}(\text{Sim}_T^f(x), \text{View}_T^f(x, w)) \leq \varepsilon_p$, where $\text{View}_T^f(x, w)$ denotes the joint view of the parties in T in a full execution of Π (including the sharing and reconstruction phases) with an honest dealer \mathcal{D} that has input (x, w) , and in which the parties in T may be corrupted, computationally unbounded, and may arbitrarily deviate from the protocol.

When $\varepsilon_p = \varepsilon_r$ and $t_p = t_r$ we say that the scheme is an (ε, t) -cVSS, or simply that it is a t -cVSS.

Overview of Our cVSS Scheme. At a high level, our cVSS scheme – described in Figure 4 – works by having the dealer encode its secret x using the Randomized RS (RRS) code of Definition 3.5, and distribute the shares between the parties. Then, the parties verify their shares by running the IRS test of Figure 1 to check the validity of the encoding, then running a dZK proof attesting to the fact that the encoded secret satisfies the predicate.

The following theorem summarizes the properties of the cVSS scheme of Figure 4.

Theorem 5.1 (cVSS from dZK). Let $t \in \mathbb{N}$, and $\text{RRS} = \text{RRS}_{\mathbb{F}, k, t+1, \eta}$ for $k > \max\{3(t+1), 4t\}$ and some $\eta \in \mathbb{F}^k$. Let \mathcal{R} be an NP relation, and let Π_{dist} be k -verifier $(\varepsilon_p, \varepsilon_r, t, t)$ -dZK for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then the cVSS scheme of Figure 4, when instantiated with \mathcal{C} and Π_{dist} as the underlying building blocks, is an $(\max\{\varepsilon_r, \varepsilon'\}, \varepsilon_p, t, t)$ -cVSS for \mathcal{R} , where ε' is the soundness error of the IRS test (Theorem 3.1).

¹⁶We assume that $\mathcal{L}(\mathcal{R})$ is not empty, in which case a description of such an x' can be included in the description of the protocol.

A Certifiable Verifiable Secret Sharing Scheme

Building Blocks. The scheme is used to prove membership of the secret shared value in an NP relation \mathcal{R} . It employs a dZK proof system Π_{dist} for the relation $\widehat{\mathcal{R}}_{\text{RRS}}$ (see Definition 2.2 in Section 2, and Definition 3.5 of the RRS code), between a prover \mathcal{P} and verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$.

Sharing Phase. Sharing is executed between the dealer \mathcal{D} with input a secret x and a witness w such that $(x, w) \in \mathcal{R}$, and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ that have no input. The output of each \mathcal{P}_i is a share Share_i . This phase is executed as follows.

1. \mathcal{D} picks a random string r for the encoding procedure Enc of the RRS code, encodes $(x^{(1)}, \dots, x^{(k)}) \leftarrow \text{Enc}(x; r)$, and sends $x^{(i)}$ to \mathcal{P}_i for every $1 \leq i \leq k$.
2. The parties run the IRS test of Figure 1 using the inputs $x^{(1)}, \dots, x^{(k)}$.^a
3. The parties emulate the dZK Π_{dist} , where \mathcal{D} plays the role of the dZK prover \mathcal{P} with input $(x^{(1)}, \dots, x^{(k)})$ and witness w , and each \mathcal{P}_i emulates \mathcal{V}_i with input $x^{(i)}$.
4. When the emulation of Π_{dist} terminates, \mathcal{P}_i determines its share Share_i as follows. If at most t parties reject in Step 2, and \mathcal{V}_i accepted in Π_{dist} , then $\text{Share}_i := x^{(i)}$. Otherwise, Share_i is set to the i 'th share in a pre-determined (and fixed) encoding of an arbitrary $x' \in \mathcal{L}(\mathcal{R})$.

Reconstruction Phase. Each \mathcal{P}_i operates as follows.

1. Broadcasts its share Share_i . Let $\text{Share}'_1, \dots, \text{Share}'_k$ denote the broadcasted shares.
2. Locally computes $x' = \text{Dec}(\text{Share}'_1, \dots, \text{Share}'_k)$ (where Dec is the decoding procedure of the RRS code) and outputs x' .

^aWe note that if Π_{dist} is the dZK proof of Figure 3 then this step can be skipped, because the dZK itself internally runs the IRS test.

Figure 4: A t -cVSS Scheme

Proof: We prove that the cVSS scheme of Figure 4 satisfies the properties of Definition 5.1.

Strong correctness. If $(x, w) \in \mathcal{R}$ and \mathcal{D} is honest, then $((x^{(1)}, \dots, x^{(k)}), w) \in \widehat{\mathcal{R}}_{\text{RRS}}$ so the correctness of the IRS test (Theorem 3.1) guarantees that all honest parties accept in Step 2. Therefore, only the corrupted parties may reject in this phase, so there are at most t rejections in that step. Moreover, the strong completeness of Π_{dist} guarantees that except with probability ε_r , Step 3 of the sharing phase passes, so every honest \mathcal{P}_i outputs $\text{Share}_i = x^{(i)}$ in Step 4 of the sharing phase. Since at most t parties are corrupted during the reconstruction phase (and the shares generated by \mathcal{D} form a valid encoding of x), then the shares broadcasted in Step 1 of the Reconstruction phase are t -close to an encoding of x , and so the robustness of the RRS code guarantees that all honest parties output x during the Reconstruction phase.

Binding. The analysis here is somewhat similar to (though much simpler than) the analysis of our dZK system (Theorem 4.1). Let \mathcal{D} be a corrupted dealer colluding with a subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t$, and let $(x^{(i)})_{i \notin T}$ denote the shares which \mathcal{D} sent to the honest parties in Step 1 of the sharing phase. We say that the input pieces $\{x^{(i)}\}_{i \notin T}$ of the honest parties are *compliant* with a valid encoding of some x' if there exists an $(x'^{(1)}, \dots, x'^{(k)})$ which is a valid encoding of x' , and $x'^{(i)} := x^{(i)}$ except for $i \in T$ and at most t parties $i \notin T$. We consider two possible cases.

Case (1): there exists no x' such that the input pieces $\{x^{(i)}\}_{i \notin T}$ are compliant with a valid encoding of x' . Then by the soundness of the IRS test (Theorem 3.1), except with probability ε' Step 2 of the Sharing phase fails, and so all honest parties output a fixed encoding of some arbitrary $x^* \in \mathcal{L}(\mathcal{R})$. Therefore, the shares broadcasted in the Reconstruction phase are t -close to a valid encoding of x^* (all errors are concentrated in the shares of the parties in T , who may use incorrect shares), so the robustness of the RRS code guarantees that x^* will be reconstructed.

Case (2): there exists an x' such that the input pieces $\{x^{(i)}\}_{i \notin T}$ are compliant with some valid encoding $(x^{(1)}, \dots, x^{(k)})$ of x' . In this case, if the IRS test fails then the binding holds as in case (1). Otherwise, the IRS test passes. Moreover, in this case for every choice $x''^{(i)}, i \in T$ of input pieces for the corrupted parties, we have $d((x''^{(1)}, \dots, x''^{(k)}), (x^{(1)}, \dots, x^{(k)})) \leq 2t$, where $x''^{(i)} = x^{(i)}$ for every $i \notin T$. Since the RRS decoder can correct $(k-1)/2 \geq 2t$ error, then $\text{Dec}(x^{(1)}, \dots, x^{(k)}) = x'$ regardless of the input pieces which the parties in T provide to the decoder.

We consider three sub cases. First, if $x' \notin \mathcal{L}(\mathcal{R})$ then there exists no choice of input pieces for the parties in T , for which the resultant distributed input will be in $\widehat{\mathcal{R}}_{\text{RRS}}$. Therefore, the soundness of Π_{dist} guarantees that except with probability ε_r , the execution in Step 3 of the Sharing phase fails, and so as in case (1) all honest parties output shares of $x^* \in \mathcal{L}(\mathcal{R})$, and x^* will be reconstructed. Second, if $x' \in \mathcal{L}(\mathcal{R})$ but the execution of Π_{dist} failed, then similarly to the previous sub case, x^* will be reconstructed during the Reconstruction phase. Thirdly, if $x' \in \mathcal{L}(\mathcal{R})$ and the execution of Π_{dist} succeeded, then since $x^{(i)} := x^{(i)}$ except for $i \in T$ and at most t parties $i \notin T$, the input pieces used during reconstruction are at most $2t \leq (k-1)/2$ far from a valid encoding of x' (here, we use the fact that $k > 4t$). Since the RRS decoder can correct $(k-1)/2$ errors, x' will be reconstructed. In summary, since cases (1) and (2) are disjoint, the binding error is $\max\{\varepsilon_r, \varepsilon'\}$.

Secrecy. Let T be a subset of corrupted parties of size $|T| \leq t_p$. We describe a simulator Sim_T that simulates the view of the parties in T during the sharing phase. The simulator uses the simulator $\text{Sim}_T^{\text{dist}}$ (Sim_T^{RS} , respectively) for an execution of Π_{dist} (the IRS test, respectively) in which the parties in T are corrupted. Intuitively, Sim will simulate the shares of parties in T using a random encoding of the all-0 string, and then emulate Sim_T^{RS} and $\text{Sim}_T^{\text{dist}}$ on these shares. Formally, $\text{Sim}_T(|x|)$ operates as follows:

- Encodes $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}) \leftarrow \text{Enc}(0^{|x|})$, and provides $(\tilde{x}^{(i)})_{i \in T}$ to the corrupted parties as the messages sent by the dealer in Step 1 of the Sharing phase.
- Executes the IRS test (Step 2 of the Sharing phase) with the parties in T , using $\text{Sim}_T^{\text{RS}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the test. Let $(\widetilde{\text{View}}_i^{\text{RS}})_{i \in T}$ denote the views of the parties in T during this step.
- Executes Π_{dist} with the parties in T , using $\text{Sim}_T^{\text{dist}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the dZK. Let $(\widetilde{\text{View}}_i^{\text{dist}})_{i \in T}$ denote the views of the parties in T during this step.
- Outputs $(\tilde{x}^{(i)}, \widetilde{\text{View}}_i^{\text{RS}}, \widetilde{\text{View}}_i^{\text{dist}})_{i \in T}$ as the views of the corrupted parties during the Sharing phase.

We now show that for every $(x, w) \in \mathcal{R}$ it holds that $\text{SD}(\text{Sim}_T(|x|), \text{View}_T(x, w)) \leq \varepsilon_p + \varepsilon'_p$ through a sequence of hybrids.

\mathcal{H}_0 : This is the real-world view $\text{View}_T(x, w)$.

\mathcal{H}_1 : In \mathcal{H}_1 , we replace the views of the parties in T during the execution of Π_{dist} with the simulated views generated by $\text{Sim}_T^{\text{dist}}$ (in particular, the input pieces given to $\text{Sim}_T^{\text{dist}}$ are the actual shares generated by \mathcal{D} in the real execution).

Then $SD(\mathcal{H}_0, \mathcal{H}_1) \leq \varepsilon_p$ by the (ε_p, t_p) -ZK of Π_{dist} . Indeed, the ZK of Π_{dist} guarantees that the real and simulated views of the parties in T during an execution of Π_{dist} are ε_p -statistically close, and the entire hybrid distributions can be generated from the real/simulated views in Π_{dist} by the function that has the real views during the IRS test hard-wired into it.¹⁷

\mathcal{H}_2 : In \mathcal{H}_2 , we replace the views of the parties in T during the execution of the IRS test with the simulated views generated by Sim_T^{RS} (in particular, the input pieces given to Sim_T^{RS} are the actual shares generated by \mathcal{D} in the real execution).

Then $SD(\mathcal{H}_1, \mathcal{H}_2) = 0$ by the perfect ZK of the IRS test. This holds because the simulated views in Π_{dist} can be computed from $(x^{(i)})_{i \in [k]}$, and applying a function to the random variables does not increase the statistical distance.

\mathcal{H}_3 : In \mathcal{H}_3 , we replace the shares sent to the parties in T in Step 1 of the Sharing phase with the corresponding input pieces in a random encoding $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ of $|x|$ (and, in particular, the inputs given to $\text{Sim}_T^{\text{dist}}, \text{Sim}_T^{\text{RS}}$ are $(\{\tilde{x}^{(i)}\}_{i \in T})$). Notice that \mathcal{H}_3 is identical to $\text{Sim}_T(|x|)$.

Then $SD(\mathcal{H}_2, \mathcal{H}_3) = 0$ by the perfect t_p -privacy of the code. (Indeed, all values in the hybrids can be efficiently generated from the input pieces of the parties in T .)

Zero Knowledge. The proof of the ZK property is similar to the secrecy proof, except that now the simulator is given the input x and can therefore generate an encoding of x to be used in the simulation. We proceed with the formal argument. Let T be a subset of corrupted parties of size $|T| \leq t_p$. The simulator Sim_T^f uses the simulator $\text{Sim}_T^{\text{dist}}$ (Sim_T^{RS} , respectively) for an execution of Π_{dist} (the IRS test, respectively) in which the parties in T are corrupted. $\text{Sim}_T^f(x)$ operates as follows:

- Encodes $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}) \leftarrow \text{Enc}(x)$.
- Executes the IRS test (Step 2 of the Sharing phase) with the parties in T , using $\text{Sim}_T^{\text{RS}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the test. Let $(\widetilde{\text{View}}_i^{\text{RS}})_{i \in T}$ denote the views of the parties in T during this step.
- Executes Π_{dist} with the parties in T , using $\text{Sim}_T^{\text{dist}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the dZK. Let $(\widetilde{\text{View}}_i^{\text{dist}})_{i \in T}$ denote the views of the parties in T during this step.
- Outputs $(\tilde{x}^{(i)})_{i \in T}$ as the shares sent by \mathcal{D} to the parties in T in Step 1 of the Sharing phase, $(\widetilde{\text{View}}_i^{\text{RS}})_{i \in T}$ as the views of the parties in T in the IRS test (Step 2 of the Sharing phase),

¹⁷In particular, we note that replacing the views in Π_{dist} (from real to simulated) cannot affect the behaviour of the parties in T during the IRS test, which occurs before the execution of Π_{dist} .

$\left(\widetilde{\text{View}}_i^{\text{dist}}\right)_{i \in T}$ as the views during the execution of Π_{dist} in Step 3 of the Sharing phase, and $\left(\tilde{x}^{(i)}\right)_{i \notin T}$ as the shares broadcasted in Step 1 of the Reconstruction phase.

We now show that for every $(x, w) \in \mathcal{R}$ it holds that $\text{SD}\left(\text{Sim}_T^f\left(\left(x^{(i)}\right)_{i \in T}\right), \text{View}_T^f(x, w)\right) \leq \varepsilon_p$, using a sequence of hybrids. Since the input pieces are identically distributed in the real execution and the simulation (in both cases these are random encodings of x), then we can condition on these values. Therefore, we can repeat the hybrid argument from the secrecy case, showing that $\text{SD}(\mathcal{H}_0, \mathcal{H}_2) \leq \varepsilon_p$. This proves ZK since in this case \mathcal{H}_2 is identical to $\text{Sim}_T^f(x)$. \blacksquare

Remark 5.1 (Equivocal cVSS). *We note that if the code \mathcal{C} is equivocal, then our cVSS is also equivocal. Specifically, we say that \mathcal{C} is equivocal if there exists an efficient resampling algorithm S such that for every $(x^{(1)}, \dots, x^{(k)}) \in \mathcal{C}$, and every subset $\mathcal{I} \subseteq [k]$, $|\mathcal{I}| \leq t_p$ it holds that $S\left(\mathcal{I}, x', \left(x^{(i)}\right)_{i \in \mathcal{I}}\right)$ outputs an encoding $(x'^{(1)}, \dots, x'^{(k)})$ which is random subject to the constraint that $x'^{(i)} = x^{(i)}$ for every $i \in \mathcal{I}$.*

Given such a code, our cVSS scheme of Figure 4 has the following equivocation property which is a strengthening of the ZK property of Definition 5.1: for every subset T of at most t parties there exists a simulator Sim_T^e such that for any x , and any input shares $\left(x'^{(i)}\right)_{i \in T}$ generated by the secrecy simulator $\text{Sim}_T(|x|)$, the simulated views generated by $\text{Sim}_T^e\left(x, \left(x'^{(i)}\right)_{i \in T}\right)$ are ε'_p -statistically close to $\text{View}_T^f(x, w)$, and moreover the shares which the parties in T obtained from \mathcal{D} in Step 1 of the Sharing phase in $\text{Sim}_T^e\left(x, \left(x'^{(i)}\right)_{i \in T}\right)$ are $\left(x'^{(i)}\right)_{i \in T}$.

Complexity of Our cVSS Scheme. The dealer can send to the parties in Step 1 of the Sharing phase, also the prover messages in the IRS test (Step 2) and the dZK (Step 3). Then, the parties can execute in parallel the remaining steps of the IRS test and the dZK. (This does not affect security, see Remark 4.4) Using our dZKs of Corollary 4.3, the cVSS sharing phase will have 3 rounds, and except for the dealer messages, all communication is through the broadcast of $O(\log k \cdot \log |C| \cdot |C|) + d^2 \cdot \text{poly}(k, \log |C|)$ field elements, where C is an arithmetic circuit of depth d that verifies membership in $\widehat{\mathcal{R}}_{\text{RRS}}$. This gives the following corollary.

Corollary 5.2. *Let $k \in \mathbb{N}$, and let \mathbb{F}_k be a field of size k^l for a sufficiently large $l \in \mathbb{N}$. Let \mathcal{R} be a relation, and let C be a circuit over \mathbb{F}_k verifying membership in $\widehat{\mathcal{R}}_{\text{RRS}}$. Assuming ideal coin-tossing, $\widehat{\mathcal{R}}_{\text{RRS}}$ has a 3-round $\Omega(k)$ -cVSS scheme, in which $O(\log k \cdot \log |C| \cdot |C|) + d^2 \cdot \text{poly}(k, \log |C|)$ field elements are broadcasted during the Sharing phase, where $|C|$ denotes the number of gates in C , and d is its depth.*

Comparison with the cVSS of [IW14]. Ishai and Weiss [IW14] design a cVSS scheme based on Zero-Knowledge Probabilistically Checkable Proof of Proximity (ZK-PCPP).¹⁸ They separate the Sharing phase into two phases, a sharing phase in which the dealer distributes shares between the parties (this corresponds to Step 1 of the Sharing phase in Figure 4) and a verification phase in which the parties interact to determine the validity of the sharing. This is done to capture the fact that their verification phase requires only polylogarithmic communication in the total number of parties (whereas the communication during this part of our cVSS is at least linear in k , and in fact

¹⁸Their cVSS scheme is defined in the plain model in a system which includes a designated receiver who determines the outcome of the cVSS, but the receiver is only used to generate random coin tosses, so in the $\mathcal{F}_{\text{coin}}$ -hybrid model it can be replaced with a call to $\mathcal{F}_{\text{coin}}$.

quadratic when using the dZK of Section 4). However, the total number of parties in their scheme is a (large) polynomial in the corruption bound t and the input length n , whereas in our cVSS the number of parties is only a constant times larger than the corruption bound. In particular, due to this relationship between k, t , in many natural scenarios the communication required to verify the sharing might actually be smaller in our cVSS.

5.2 Reusable cVSS

In this section we describe a generalization of cVSS with the attractive feature of *reusability*. Roughly, a reusable cVSS scheme consists of an initial sharing phase in which the dealer distributes secret shares between the parties, following which the dealer can prove – in an online fashion – an unlimited number of claims on these secret shares. These proofs are guaranteed to all hold with respect to *the same* secret shares distributed in the initial sharing phase. Finally, a reconstruction phase allows the parties to reconstruct the shared secret. This is formalized in the following definition.

Definition 5.2 (Reusable cVSS). *Let $\varepsilon \in [0, 1]$, and $t_p, t_r, k \in \mathbb{N}$. We say that protocol Π between a dealer \mathcal{D} and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ is an $(\varepsilon_p, \varepsilon_r, t_p, t_r)$ -reusable cVSS scheme if the following holds.*

- **Syntax.** *The protocol has a Sharing phase, followed by any number of Prove phases, and finally a Reconstruction phase. In the Sharing phase, \mathcal{D} has input x , and the parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ have no input. The output of \mathcal{P}_i in this phase is a share Share_i , and \mathcal{D} has no output. Following the Sharing phase, there could be multiple invocations of a Prove phase. In each invocation all parties are given the description of an NP relation \mathcal{R} , and \mathcal{D} is given a witness w such that $(x, w) \in \mathcal{R}$. The output of \mathcal{P}_i in this phase is either accept or reject, and \mathcal{D} has no output. In the Reconstruction phase, the input of each \mathcal{P}_i is its outputs in all previous phases, and \mathcal{D} has no input. The output of the sharing phase is an x' as well as accept or reject (intuitively, this additional output indicates whether x' satisfies all the predicates of the Prove phase).*
- **Strong Correctness.** *For every x , every sequence $\mathcal{R}_1, \dots, \mathcal{R}_m$ of NP relations, and every sequence w_1, \dots, w_m of witnesses such that $(x, w_i) \in \mathcal{R}_i$ for every $1 \leq i \leq m$, the following holds except with probability $m \cdot \varepsilon_r$. In an execution of Π with an honest dealer \mathcal{D} that has input x , where the Prove phase is executed m times with relations $\mathcal{R}_1, \dots, \mathcal{R}_m$ and witnesses w_1, \dots, w_m , the output of the reconstruction phase is x , accept, even if t parties are corrupted, computationally unbounded, and may arbitrarily deviate from the protocol.*
- **Binding.** *For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_r$, the following holds except with ε_r failure probability over the randomness of the Sharing phase. At the end of the Sharing phase of Π in which the parties in $\{\mathcal{P}_1, \dots, \mathcal{P}_k\} \setminus T$ are honest, there exists a unique x^* such that the output of the parties in $\{\mathcal{P}_1, \dots, \mathcal{P}_k\} \setminus T$ in the Reconstruction phase will be x^* , regardless of the messages sent by \mathcal{D} and the parties in T during the Reconstruction phase. Moreover, if the prove phase is executed with some \mathcal{R} such that $x^* \notin \mathcal{L}(\mathcal{R})$ then the honest parties will output reject in the Reconstruction phase. This holds even if \mathcal{D} and the parties in T are corrupted, computationally unbounded, and arbitrarily deviate from the protocol throughout the computation (including during the sharing and prove phases).*
- **Secrecy.** *For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_p$, there exists a simulator Sim_T such that for every x , every sequence $\mathcal{R}_1, \dots, \mathcal{R}_m$ of NP relations, and every sequence w_1, \dots, w_m of witnesses such that $(x, w_i) \in \mathcal{R}_i$ for every $1 \leq i \leq m$,*

$$SD(\text{Sim}_T(|x|, \mathcal{R}_1, \dots, \mathcal{R}_m), \text{View}_T(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)) \leq m \cdot \varepsilon_p$$

where $\text{View}_T(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)$ denotes the joint view of the parties in T in an execution of the Sharing phase of Π with an honest dealer \mathcal{D} that has input x , followed by m execution of the Prove phase for relations $\mathcal{R}_1, \dots, \mathcal{R}_m$ in which the honest \mathcal{D} has inputs w_1, \dots, w_m (respectively). This holds even if the parties in T are corrupted, computationally unbounded, and arbitrarily deviate from the protocol.

- **Zero Knowledge.** For every subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t_p$, there exists a simulator Sim_T^f such that for every x , every sequence $\mathcal{R}_1, \dots, \mathcal{R}_m$ of NP relations, and every sequence w_1, \dots, w_m of witnesses such that $(x, w_i) \in \mathcal{R}_i$ for every $1 \leq i \leq m$,

$$\text{SD}\left(\text{Sim}_T^f(x, \mathcal{R}_1, \dots, \mathcal{R}_m), \text{View}_T^f(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)\right) \leq m \cdot \varepsilon_p$$

where $\text{View}_T^f(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)$ denotes the joint view of the parties in T in a full execution of Π (including the Sharing phase, any number of Prove phases, and the Reconstruction phase) with an honest dealer \mathcal{D} that has input x , and the Prove phase is executed m times with relations $\mathcal{R}_1, \dots, \mathcal{R}_m$ and the honest \mathcal{D} has inputs w_1, \dots, w_m (respectively). This holds even if the parties in T are corrupted, computationally unbounded, and arbitrarily deviate from the protocol.

When $\varepsilon_p = \varepsilon_r$ and $t_p = t_r$ we say that the scheme is an (ε, t) -reusable cVSS, or simply that it is a t -reusable cVSS.

Discussion and Comparison with cVSS. A few remarks are in order. First, notice that the notions of cVSS and reusable cVSS are incomparable. Indeed, cVSS offers the additional property of reusability, which allows the dealer \mathcal{D} to prove membership of its secret shared x in multiple NP languages which are determined in an *online* fashion. That is, \mathcal{D} does not need to know the identity of these NP languages ahead of time, and following the initial sharing phase it will be able to successfully perform the Prove phase for every \mathcal{L} such that $x \in \mathcal{L}$ (provided it is given an appropriate witness). On the other hand, cVSS is not reusable – the dealer can only prove membership in a *single* NP language, whose identity is known in advance – but it provides a stronger binding guarantee. Indeed, at the end of the Sharing phase, the parties hold a sharing of an $x^* \in \mathcal{L}$. This should be contrasted with the binding guarantee of reusable cVSS, which guarantees that \mathcal{D} is committed to some secret x' , but guarantees nothing more (in particular, x' is not guaranteed to be in any specific NP language). Thus, reusable cVSS does not capture cVSS as a special case, even if during the execution a single Prove phase is executed for some NP language \mathcal{L} (because the secret x' that will be reconstructed in the Reconstruction phase might not be in \mathcal{L}).

Second, we note that this weaker binding guarantee of reusable cVSS is nonetheless inherent. Indeed, since the relations used during the Prove phase are unknown during the Sharing phase, one cannot guarantee that the unique secret determined in the Sharing phase will satisfy them. One could alternatively allow for the secret to change following the execution of the Prove phases, so that the final secret determined following the Sharing and all Prove phases will satisfy all the relations, but this would be inconsistent with the general spirit of VSS which requires that a single secret is committed to by the dealer, and cannot later be changed.¹⁹ Moreover, additional complications arise in this case, since it might be the case that there exists *no* secret that simultaneously satisfies all relations (and even if there is, it might not be feasible for the parties to find one). We stress, however, that though the reconstructed secret might not satisfy *any* of the relations used

¹⁹In particular, allowing for such “adjustments” of the secret might lead to possible attacks by a corrupted dealer, that could purposely cause a Prove phase to fail as a method of changing the secret that would be reconstructed.

in the Prove phase, still the binding guarantee is meaningful since in this case the dealer cannot convince the parties that the secret satisfies the relations.

Third, notice that the error in the protocol accumulates with the number of invocations of the Prove phase, *except* for the binding property. Intuitively, this is because for binding once the dealer is identified as corrupted then all parties can reject and use a fixed sharing of an arbitrary value, regardless of the outcome of consequent Prove phases. This should be contrasted, for example, with the strong correctness property - in which as long as the dealer is still considered honest, each invocation of the Prove phase might have a small failure probability (i.e., incorrectly identifying the dealer as corrupted), and these errors accumulate.

Finally, the definition of reusable cVSS can be naturally extended and generalized in several respects. First, it can be strengthened to guarantee that the parties know, at the end of the prove phase, exactly which of the relations used during the Prove phase are satisfied by the secret that will be reconstructed. Second, reusable cVSS can be used in settings in which the entity sharing the secret, and the entity proving it satisfied certain predicates, are different entities. In fact, the Prove phase can be executed by different entities, and these entities do not need to share a state or any information, other than that all proving entities must know the entire secret sharing generated during the Sharing phase. We note that our reusable cVSS scheme of Figure 5 satisfies these stronger notions.

Comparison with the cVSS of [IW14]. We note that the cVSS scheme of [IW14] is not reusable, and does not seem to easily extend to the reusable setting. Indeed, their construction roughly works as follows. The dealer encodes its secret using a private and robust code, and generates a Zero-Knowledge Probabilistically Checkable Proof of Proximity (ZK-PCPP) for the claim that the encoding is close to an encoding of an x which satisfies the relation, i.e., that the encoding is in $\mathcal{L}(\widehat{\mathcal{R}}_{\text{RRS}})$. Each party is then given either a symbol of the encoding or of the ZK-PCPP. The parties then check that their shares indeed encode an $x \in \mathcal{L}(\mathcal{R})$ by jointly running the ZK-PCPP verifier, namely they call $\mathcal{F}_{\text{coin}}$ to obtain randomness r for the verifier, and determine the queries q_1, \dots, q_m which the verifier makes to its input (i.e., the encoding) and the ZK-PCPP. The parties holding these symbols broadcast them, and each party can locally verify that the ZK-PCPP verifier would accept. In particular, this verification phase reveals to all parties some symbols of the encoding of x , which were not part of their secret share of x . Therefore, this encoding can only be used to prove an (a-priori bounded) number of relations. Once this bound is reached, the dealer must refresh the encoding, which requires an elaborate proof to show that the refreshed encoding encode the same secret x .

Overview of Our Reusable cVSS. Our reusable cVSS scheme follows roughly the structure of our cVSS scheme of Figure 4. Specifically, during the Sharing phase the dealer randomly encodes x and distributes the codeword symbols between the parties, and the parties then execute the IRS test (Figure 1) to check that the codeword is a valid encoding. If this phase fails then the parties use an arbitrary valid encoding as their secret shares. In each Prove phase for a relation \mathcal{R} , the parties use the same input pieces (secret shares) generated during the Sharing phase, and execute a dZK for the claim that these input pieces are a valid encoding of an x such that $x \in \mathcal{L}(\mathcal{R})$. Finally, in the Reconstruction phase all parties broadcast their shares, and each party locally decodes the secret. Each party additionally outputs either accept or reject based on whether all Prove phases were successful or not. The scheme is provided in Figure 5.

The following theorem summarizes the properties of the reusable cVSS scheme of Figure 5.

A Reusable Certifiable Verifiable Secret Sharing Scheme

Building Blocks. The scheme is run between a dealer \mathcal{D} and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$.

Sharing Phase. Sharing is executed between the dealer \mathcal{D} with input a secret x , and k parties $\mathcal{P}_1, \dots, \mathcal{P}_k$ that have no input. The output of each \mathcal{P}_i is a share Share_i . This phase is executed as follows.

1. \mathcal{D} picks a random string r for the encoding procedure Enc of the RRS code, encodes $(x^{(1)}, \dots, x^{(k)}) \leftarrow \text{Enc}(x; r)$, and sends $x^{(i)}$ to \mathcal{P}_i for every $1 \leq i \leq k$.
2. The parties run the IRS test of Figure 1 using their inputs $x^{(1)}, \dots, x^{(k)}$.
3. When the IRS test terminates, \mathcal{P}_i determines its share Share_i as follows. If the IRS test passed then $\text{Share}_i := x^{(i)}$. Otherwise, Share_i is set to the i 'th share in a pre-determined (and fixed) encoding of the all-0 string.

Prover Phase. This phase can be executed repeatedly for different NP relations \mathcal{R} , which are given as input to all parties. The dealer has input x, r, w where r is the random string used during the Sharing phase, and $(x, w) \in \mathcal{R}$. Each \mathcal{P}_i has as input its share Share_i from the Sharing phase. The phase uses a dZK proof $\Pi_{\mathcal{R}}$ for $\widehat{\mathcal{R}}_{\text{RRS}}$ (see Definition 2.2 in Section 2), and proceeds by having the dealer emulate the dZK prover with input $(\text{Share}_1, \dots, \text{Share}_k)$ and witness w , and each \mathcal{P}_i emulates \mathcal{V}_i with input Share_i . When the emulation ends, each \mathcal{P}_i outputs accept or reject similar to the output of \mathcal{V}_i .

Reconstruction Phase. Each \mathcal{P}_i operates as follows.

1. Broadcasts its share Share_i . Let $\text{Share}'_1, \dots, \text{Share}'_k$ denote the broadcasted shares.
2. Locally computes $x' = \text{Dec}(\text{Share}'_1, \dots, \text{Share}'_k)$ and outputs x' . Additionally, if all outputs of \mathcal{P}_i in the Prove phase were accept then \mathcal{P}_i outputs accept, otherwise it outputs reject.

Figure 5: A t -reusable cVSS for $k = 6t + 2$ parties

Theorem 5.3 (Reusable cVSS from dZK). *Let $t \in \mathbb{N}$, and $\text{RRS} = \text{RRS}_{\mathbb{F}, k, t+1, \eta}$ for $k > \max\{3(t+1), 4t\}$ and some $\eta \in \mathbb{F}^k$. If for every relation \mathcal{R} used during the Prove phase of Figure 5 there exists a k -verifier $(\varepsilon_p, \varepsilon_r, t, t)$ -dZK proof for $\widehat{\mathcal{R}}_{\text{RRS}}$, then the scheme of Figure 5 is a $(\varepsilon_r + \varepsilon', \varepsilon_p, t, t)$ -reusable cVSS, where ε' is the soundness error of the IRS test (Theorem 3.1).*

Proof: We prove that the scheme of Figure 5 satisfies the properties of Definition 5.2.

Strong correctness follows from the strong completeness of the underlying dZK proofs, the correctness of the IRS test, and the error correction of the RRS code. Indeed, the correctness of the IRS test guarantees that Step 2 of the Sharing phase passes with probability 1, even if t parties are corrupted, and so $\text{Share}_i = x^{(i)}$ for every honest \mathcal{P}_i . Conditioned on the Sharing phase passing, each Prove phase will pass except with probability ε_r (even if t parties cheat) due to the strong completeness of the dZK proofs. If the Prove phase is executed m times then using the union bound, except with probability $m \cdot \varepsilon_r$, all invocations of the Prove phase pass. Conditioned on the even that all previous phases passed, the robustness of the RRS code guarantees that x will be reconstructed (with probability 1) because the shares broadcasted during the Reconstruction phase have at most t errors. Therefore overall the correctness error is $m \cdot \varepsilon_r$.

Binding. Let \mathcal{D} be a corrupted dealer colluding with a subset $T \subseteq \{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ of size $|T| \leq t$,

and let $(x^{(i)})_{i \notin T}$ denote the shares which \mathcal{D} sent to the honest parties in Step 1 of the sharing phase. Similar to the binding proof of our cVSS scheme (Theorem 5.1), we consider two possible cases based on whether the input pieces $\{x^{(i)}\}_{i \notin T}$ of the honest parties are *compliant* with a valid encoding of some x' , namely if there exists a valid encoding $(x'^{(1)}, \dots, x'^{(k)})$ of x' , such that $x'^{(i)} := x^{(i)}$ except for $i \in T$ and at most t parties $i \notin T$.

Case (1): there exists no such x' . Then by the soundness of the IRS test (Theorem 3.1), except with probability ε' Step 2 of the Sharing phase fails, so all honest parties output a fixed encoding of some arbitrary x^* . Moreover, since at most t parties cheat during reconstruction, the broadcasted shares would be t -close to a valid encoding of x^* , so x^* will be reconstructed (by the robustness of the RRS code). If the Prove phase is executed for some \mathcal{R} such that $x^* \notin \mathcal{L}(\mathcal{R})$ then the soundness of $\Pi_{\mathcal{R}}$ guarantees that except with probability ε_r , the output of the honest parties in that Prove phase (and consequently also in the Reconstruction phase) would be reject. Therefore, overall the soundness error in this case is $\varepsilon' + \varepsilon_r$.

Case (2): there exists an x' such that the input pieces $\{x^{(i)}\}_{i \notin T}$ are compliant with a valid encoding of x' . In this case, if the IRS test fails then the binding holds as in case (1). Otherwise, the IRS test passes. Moreover, in this case for every choice $x''^{(i)}, i \in T$ of input pieces for the corrupted parties, we have $d((x''^{(1)}, \dots, x''^{(k)}), (x'^{(1)}, \dots, x'^{(k)})) \leq 2t$, where $x''^{(i)} = x^{(i)}$ for every $i \notin T$. Since the RRS decoder can correct $(k-1)/2 \geq 2t$ error, then $\text{Dec}(x^{(1)}, \dots, x^{(k)}) = x'$ regardless of the input pieces which the parties in T provide to the decoder. Furthermore, if the Prove phase is executed for some \mathcal{R} such that $x' \notin \mathcal{L}(\mathcal{R})$ then the soundness of $\Pi_{\mathcal{R}}$ guarantees that except with probability ε_r , the output of the honest parties in that Prove phase (and consequently also in the Reconstruction phase) would be reject. Therefore, overall the soundness error in this case is $\varepsilon' + \varepsilon_r$.

Secrecy. Let T be a subset of corrupted parties of size $|T| \leq t$. We describe a simulator Sim_T that simulates the view of the parties in T during the Sharing and Prove phases. Let $\mathcal{R}_1, \dots, \mathcal{R}_m$ denote the relations used in the m Prove phases. The simulator uses the simulators $\text{Sim}_T^1, \dots, \text{Sim}_T^m$ for executions of $\Pi_{\mathcal{R}_1}, \dots, \Pi_{\mathcal{R}_m}$ in which the parties in T are corrupted, as well as the simulator Sim_T^{RS} for an execution of the IRS test in which the parties in T are corrupted. Intuitively, Sim will simulate the shares of parties in T using a random encoding of the all-0 string, and then emulate $\text{Sim}_T^{\text{RS}}, \text{Sim}_T^1, \dots, \text{Sim}_T^m$ sequentially on these shares. Formally, $\text{Sim}_T(|x|, \mathcal{R}_1, \dots, \mathcal{R}_m)$ operates as follows:

- Encodes $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}) \leftarrow \text{Enc}(0^{|x|})$.
- Executes the IRS test (Step 2 of the Sharing phase) with the parties in T , using $\text{Sim}_T^{\text{RS}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the test. Let $(\widetilde{\text{View}}_i^{\text{RS}})_{i \in T}$ denote the views of the parties in T during this step.
- For every $1 \leq j \leq m$, executes $\Pi_{\mathcal{R}_j}$ with the parties in T , using $\text{Sim}_T^j((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in $\Pi_{\mathcal{R}_j}$. Let $(\widetilde{\text{View}}_i^j)_{i \in T}$ denote the views of the parties in T during this step.
- Outputs $(\tilde{x}^{(i)}, \widetilde{\text{View}}_i^{\text{RS}}, \widetilde{\text{View}}_T^1, \dots, \widetilde{\text{View}}_T^m)_{i \in T}$ as the views of the corrupted parties during the Sharing and Prove phases.

We now show that for every x , and every w_1, \dots, w_m such that $(x, w_i) \in \mathcal{R}_i$ for every $1 \leq i \leq m$,

it holds that $\text{SD}(\text{Sim}_T(|x|, \mathcal{R}_1, \dots, \mathcal{R}_m), \text{View}_T(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)) \leq m \cdot \varepsilon_p$, through a sequence of hybrids.

\mathcal{H}_0 : This is the real-world view $\text{View}_T(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m)$.

$\mathcal{H}_1^i, 1 \leq i \leq m+1$: In \mathcal{H}_1^i , we replace the views of the parties in T during the executions of $\Pi_{\mathcal{R}_i}, \dots, \Pi_{\mathcal{R}_m}$ with the simulated views generated by $\text{Sim}_T^i, \dots, \text{Sim}_T^m$ (in particular, the input pieces given to the simulators are the actual shares generated by \mathcal{D} in the real execution). Notice that $\mathcal{H}_1^{m+1} := \mathcal{H}_0$.

Then $\text{SD}(\mathcal{H}_1^{i-1}, \mathcal{H}_1^i) \leq \varepsilon_p$ for every $1 \leq i \leq m+1$, by the (ε_p, t) -ZK of Π_{dist} . Indeed, a distinguisher D between $\mathcal{H}_1^{i-1}, \mathcal{H}_1^i$ implies a distinguisher D_i between the simulated and actual views of the parties in T in an execution of $\Pi_{\mathcal{R}_i}$. D_i has the following values hard-wired into it: the real views of T in $\Pi_{\mathcal{R}_j}$ for $j < i$; the simulated views of T in $\Pi_{\mathcal{R}_j}$ for $j > i$; and the real views of T during the IRS test. Thus, given the (real or simulated) views of the parties in T in an execution of $\Pi_{\mathcal{R}_i}$, it can generate the entire hybrid distribution,²⁰ and run D on it.

Using the union bound, it follows that $\text{SD}(\mathcal{H}_0, \mathcal{H}_1^1) \leq m \cdot \varepsilon_p$.

\mathcal{H}_2 : In \mathcal{H}_2 , we replace the views of the parties in T during the execution of the IRS test with the simulated views generated by Sim_T^{RS} (in particular, the input pieces given to Sim_T are the actual shares generated by \mathcal{D} in the real execution).

Then $\text{SD}(\mathcal{H}_1^1, \mathcal{H}_2) = 0$ by the perfect ZK of the IRS test. This holds because the views during the Prove phases can be computed given only $(x^{(i)})_{i \in [k]}$, and applying a function to the random variables does not increase the statistical distance.

\mathcal{H}_3 : In \mathcal{H}_3 , we replace the shares sent to the parties in T in Step 1 of the Sharing phase with the corresponding input pieces in a random encoding $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ of $0^{|x|}$ (and, in particular, the inputs given to the simulators $\text{Sim}_T^{\text{RS}}, \text{Sim}_T^1, \dots, \text{Sim}_T^m$ are $(\{\tilde{x}^{(i)}\}_{i \in T})$). Notice that \mathcal{H}_3 is identical to $\text{Sim}_T(|x|, \mathcal{R}_1, \dots, \mathcal{R}_m)$.

Then $\text{SD}(\mathcal{H}_2, \mathcal{H}_3) = 0$ from the perfect t_p -privacy of the RRS code. (Indeed, the hybrids can be efficiently generated from the – real or simulated – input pieces of the parties in T .)

Zero Knowledge. The proof of the ZK property is similar to the secrecy proof, except that now the simulator is given the input x and can therefore generate an encoding of x to be used in the simulation. We proceed with the formal argument. Let T be a subset of corrupted parties of size $|T| \leq t$, and let $\mathcal{R}_1, \dots, \mathcal{R}_m$ denote the relations used in the m Prove phases. The simulator uses the simulators $\text{Sim}_T^1, \dots, \text{Sim}_T^m$ for executions of $\Pi_{\mathcal{R}_1}, \dots, \Pi_{\mathcal{R}_m}$ in which the parties in T are corrupted, as well as the simulator Sim_T^{RS} for an execution of the IRS test in which the parties in T are corrupted. $\text{Sim}_T^f(x, \mathcal{R}_1, \dots, \mathcal{R}_m)$ operates as follows:

- Encodes $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}) \leftarrow \text{Enc}(x)$.
- Executes the IRS test (Step 2 of the Sharing phase) with the parties in T , using $\text{Sim}_T^{\text{RS}}((\tilde{x}^{(i)})_{i \in T})$ to determine the messages which the honest parties send to the parties in T in the test. Let $(\widetilde{\text{View}}_i^{\text{RS}})_{i \in T}$ denote the views of the parties in T during this step.

²⁰In particular, we note that replacing the views in $\Pi_{\mathcal{R}_i}$ (from real to simulated) cannot affect the behaviour of the parties in T in the executions of the IRS test and $\Pi_{\mathcal{R}_1}, \dots, \Pi_{\mathcal{R}_{i-1}}$.

- For every $1 \leq j \leq m$, executes $\Pi_{\mathcal{R}_j}$ with the parties in T , using $\text{Sim}_T^j \left((\tilde{x}^{(i)})_{i \in T} \right)$ to determine the messages which the honest parties send to the parties in T in $\Pi_{\mathcal{R}_j}$. Let $\left(\widetilde{\text{View}}_i^j \right)_{i \in T}$ denote the views of the parties in T during this step.
- Outputs $(\tilde{x}^{(i)})_{i \in T}$ as the shares sent by \mathcal{D} to the parties in T in Step 1 of the Sharing phase, $\left(\widetilde{\text{View}}_i^{\text{RS}} \right)_{i \in T}$ as the views of the parties in T during the IRS test (Step 2 of the Sharing phase), $\left(\widetilde{\text{View}}_i^j \right)_{i \in T, 1 \leq j \leq m}$ as the views during the executions of $\Pi_{\mathcal{R}_1}, \dots, \Pi_{\mathcal{R}_m}$ in the Prove phase, and $(\tilde{x}^{(i)})_{i \notin T}$ as the shares broadcasted in Step 1 of the Reconstruction phase.

We now show that for every x , and every w_1, \dots, w_m such that $(x, w_i) \in \mathcal{R}_i$ for every $1 \leq i \leq m$, it holds that $\text{SD} \left(\text{Sim}_T^f(x, \mathcal{R}_1, \dots, \mathcal{R}_m), \text{View}_T^f(x, w_1, \dots, w_m, \mathcal{R}_1, \dots, \mathcal{R}_m) \right) \leq m \cdot \varepsilon_p$, through a sequence of hybrids. Since the input pieces are identically distributed in the real execution and the simulation (in both cases these are random encodings of x), then we can condition on these values. Therefore, we can repeat the hybrid argument from the secrecy case, showing that $\text{SD}(\mathcal{H}_0, \mathcal{H}_2) \leq \varepsilon_p$. This proves ZK since in this case \mathcal{H}_2 is identical to $\text{Sim}_T^f(x, \mathcal{R}_1, \dots, \mathcal{R}_m)$. ■

5.3 From Semi-Honest Security to Identifiable Abort Via dZK Proofs

In this section we use dZK proofs to design a generic compiler from Semi-Honest (SH) security to *malicious* security with identifiable abort. This result strengthens the generic compiler of [BBC⁺19a, Th. 7.3] to achieve *identifiable abort* (whereas [BBC⁺19b] guarantee only abort). Similar to their compiler, our compiler works for a class of “natural” protocols (see Definition 5.4 below). Our compiler presents an alternative method – compared to [IOZ14] – of obtaining malicious security with identifiable abort from SH-secure protocols (see discussion below).

This section is structured as follows. We first formally define security with identifiable abort in Section 5.3.1. Section 5.3.2 contains necessary preliminaries and terminology regarding MPC protocols – including the definition of natural protocols – used by our compiler, which is described and analyzed in Section 5.3.3. Finally, we compare our compiler to the compilers of [IOZ14] and [BBC⁺19b] in Section 5.3.4.

5.3.1 Security With Identifiable Abort

We define security in the real-ideal paradigm, by comparing the real world execution with an adversary \mathcal{A} to an ideal-world execution with a *Simulator*, who is restricted to only choose inputs for the corrupted parties, and to decide whether the execution is aborted, in which case it must determine a corrupted party whose identity will be revealed to all parties. We proceed to formally define these notions.

The Real Execution describes the execution of a k -party protocol Π (computing some function f) in the presence of a (computationally unbounded) adversary \mathcal{A} . \mathcal{A} corrupts a subset $T \subseteq [k]$ of parties. Throughout the execution, the honest parties $i \notin T$ follow the protocol specification, whereas the parties in T are fully controlled by \mathcal{A} and may arbitrarily deviate from the protocol. For every choice x_1, \dots, x_k of inputs for the parties in Π , the execution of Π with \mathcal{A} defines a

random variable $\mathcal{REAL}_{\mathcal{A},\Pi}(x_1, \dots, x_k)$ consisting of the $\text{View}_{\mathcal{A}}$ of \mathcal{A} in this execution, and the outputs $(y_i)_{i \notin T}$ of the honest parties.

The Ideal Experiment is executed between a simulator Sim who corrupts a subset $T \subseteq [k]$ of parties, the honest parties $i \notin T$, and an ideal functionality \mathcal{F}_f . In this execution, Sim chooses inputs $(x'_i)_{i \in T}$ for the corrupted parties, which it sends to \mathcal{F}_f . \mathcal{F}_f then computes $y = f(x'_1, \dots, x'_k)$, where $x'_i = x_i$ for every $i \notin T$ is the input of the i th party in the execution, and sends y to Sim . Sim then sends to \mathcal{F}_f a message $m \in \{\text{deliver}, (\text{abort}, i)\}$ where $i \in T$. If $m = \text{deliver}$ then \mathcal{F}_f delivers the output y to all honest parties $i \notin T$. Otherwise, \mathcal{F}_f delivers (abort, i) to all parties.²¹ For every choice x_1, \dots, x_k of inputs for the parties, the ideal execution with Sim defines a random variable $\mathcal{IDEAL}_{\text{Sim},f}(x_1, \dots, x_k)$ consisting of the output $\text{Sim}((x_i)_{i \in T})$ of Sim in this execution, and the outputs $(y_i)_{i \notin T}$ of the honest parties.

Definition 5.3 (Security with Identifiable Abort (IA-MPC)). *Let $1 \leq t < k$. We say that a protocol Π ε -securely realizes a k -party functionality f with identifiable abort if for any adversary \mathcal{A} corrupting a subset $T \subset [k]$ of size $|T| \leq t$, there exists a PPT simulator Sim such that for every x_1, \dots, x_k ,*

$$SD(\mathcal{REAL}_{\mathcal{A},\Pi}(x_1, \dots, x_k), \mathcal{IDEAL}_{\text{Sim},f}(x_1, \dots, x_k)) \leq \varepsilon.$$

5.3.2 MPC Terminology and Notation

For an MPC protocol Π , we use Π' to denote the protocol obtained from Π by removing the last communication round. For a protocol Π , and a pair of parties $i, j \in [k]$, we use $M_{ij}^{\text{out},\Pi}$ ($M_{ij}^{\text{in},\Pi}$, respectively) to denote the messages sent from \mathcal{P}_i to \mathcal{P}_j (from \mathcal{P}_j to \mathcal{P}_i , respectively) in Π . For $i \in [k]$ we denote $M_i^{\text{out},\Pi} := (M_{ij}^{\text{out},\Pi})_{j \neq i}$, $M_i^{\text{in},\Pi} := (M_{ij}^{\text{in},\Pi})_{j \neq i}$, and $M_i^\Pi := (M_i^{\text{out},\Pi}, M_i^{\text{in},\Pi})$. For a set $T \subseteq [k]$, we denote $M_T^\Pi = (M_i^\Pi)_{i \in T}$ and $M_{\text{TH}}^\Pi := (M_{ij}^{\text{out},\Pi}, M_{ij}^{\text{in},\Pi})_{i \in T, j \notin T}$ (these are the messages sent or received by a party in T from a party outside T). We say that M_{TH}^Π is *compliant* with M_T^Π if for every $i \in T$ and every $j \notin T$, the outgoing messages from \mathcal{P}_i to \mathcal{P}_j and the incoming messages from \mathcal{P}_j to \mathcal{P}_i reported in $M_T^\Pi, M_{\text{TH}}^\Pi$ are identical. When Π is clear for the context, we omit it from the notation (e.g., writing M_i in stead of M_i^Π).

Similar to [BBC⁺19b], our compiler is designed for a class of “natural” SH protocols Π which, roughly, posses the following properties. First, Π' (i.e., Π without the final round) is private against *malicious* corruptions, in the sense that maliciously corrupted parties in Π' learn nothing about the honest parties’ inputs. Second, the messages sent (to all parties) by each \mathcal{P}_i in each round of the protocol is in a k -distributed language (see Definition 2.2 in Section 2). Formally:

Definition 5.4 (Natural Protocol). *We say that a k -party protocol Π for a function f is natural against t corruptions if the following holds.*

1. Π' – i.e., Π except the final round – can be interpreted as realizing the empty functionality (which takes no inputs and given no outputs). Moreover, this realization is with perfect t -privacy against malicious corruptions.
2. The final round of Π consists solely of broadcasts, and correctness holds against adversaries that deviate from the protocol description in this final round (as long as they follow the protocol in all previous rounds).
3. For every $i \in [k]$ there exists a k -distributed language $\widehat{\mathcal{L}}_i$ such that:

²¹If $i \notin T$ then \mathcal{F}_f sends y to all honest parties.

- (a) **Structure of $\widehat{\mathcal{L}}_i$.** Inputs to $\widehat{\mathcal{L}}_i$ are of the form $M_i = (M_i^{\text{out}}, M_i^{\text{in}})$.
- (b) $\widehat{\mathcal{L}}_i$ **corresponds to executions of Π' .** For any $T, |T| \leq t$, and any $M_T^{\Pi'} = (M_i^{\Pi'})_{i \in T}$, if $M_i^{\Pi'} \in \widehat{\mathcal{L}}_i$ for every $i \in T$ then there exists a choice of inputs for the honest parties $j \notin T$, and an honest execution of Π' in which the messages sent and received by the parties in T are given by $M_T^{\Pi'}$.
- (c) **Final round messages are simulatable.** For any adversary \mathcal{A} corrupting a subset $T, |T| \leq t$ of parties, there exists a simulator $\text{Sim} = (\text{Sim}^{\text{in}}, \text{Sim}^{\text{out}})$ consisting of a deterministic input extractor Sim^{in} and a simulator Sim^{out} of the final round broadcasts, such that for any set $x_H := (x_i)_{i \notin T}$, and any $M_{\text{TH}}^{\Pi'}$ which is compliant with some $M_T^{\Pi'} = (M_i^{\Pi'})_{i \in T}$ such that $M_i^{\Pi'} \in \widehat{\mathcal{L}}_i$ for every $i \in T$, it holds that:

$$\left\{ \begin{array}{l} \text{Sim}^{\text{out}}(y, M_{\text{TH}}^{\Pi'}) : \\ \quad (x'_i)_{i \in T} = \text{Sim}^{\text{in}}(M_{\text{TH}}^{\Pi'}) \\ \quad \forall i \notin T, x'_i := x_i \\ \quad y = f(x'_1, \dots, x'_k) \end{array} \right\} \approx \left\{ \text{FinMsg}(x_H, \mathcal{A}) \mid \text{prefix } M_{\text{TH}}^{\Pi'} \right\}$$

where \approx denotes negligible statistical distance, and $\left\{ \text{FinMsg}(x_H, \mathcal{A}) \mid \text{prefix } M_{\text{TH}}^{\Pi'} \right\}$ denotes the distribution of the messages sent to the parties in T in the final round of Π in an execution with \mathcal{A} in which the honest parties have inputs x_H , where the distribution is conditioned on the transcript of all previous rounds being consistent with $M_{\text{TH}}^{\Pi'}$.²²

5.3.3 The Compiler

In this section we describe our compiler. It is almost identical to the compiler of [BBC⁺19a, Th. 7.3], except that it adds special treatment of abort messages in the underlying SH protocol, and employs a dZK with *strong completeness* (whereas the dZK used in [BBC⁺19a] has no guarantee in the case that the prover is honest but a subset of verifiers are maliciously corrupted). The strong completeness property guarantees that when an execution of the dZK fails, we know whether this occurred due to a corrupted prover, or due to a corrupted verifier.

The properties of the compiler are summarized in the following theorem.

Theorem 5.4 (IA-MPC from Natural Protocols). *Let s be a statistical security parameter, and let Π_{nat} be a natural k -party protocol computing f (as in Definition 5.4). Let $\widehat{\mathcal{L}}_1, \dots, \widehat{\mathcal{L}}_k$ denote the languages whose existence is guaranteed by Property 3b of Definition 5.4, and let Π_1, \dots, Π_k be $(k-1)$ -verifier $(\varepsilon_p, \varepsilon_r, t, t)$ -dZK proofs for $\widehat{\mathcal{L}}_1, \dots, \widehat{\mathcal{L}}_k$ which preserve ZK under parallel composition (see Remark 4.5), respectively. Then the protocol Π of Figure 6 ε -securely realizes f with identifiable abort against t corruptions, for $\varepsilon = t \cdot \varepsilon_r + (k-1) \cdot \varepsilon_p + \text{negl}(s)$. Moreover, Π has the following complexities:*

²²We note that our definition of a natural protocol slightly differs from the definition considered in [BBC⁺19a, Def. 7.1]. Specifically, Property 2 did not appear in [BBC⁺19a], and our formulation of Property 3c is slightly different, because [BBC⁺19a] only consider functionalities in which a single party receives the output. This single-output assumption is without loss of generality in the context of security with abort (the single party that obtains the output can then broadcast it to all parties), but is not the case if we require *identifiable abort*. Indeed, if the designated party (say, \mathcal{P}_1) claims that it cannot reconstruct the output, we generally cannot determine (when using Π as a black box) whether \mathcal{P}_1 is corrupted, or whether another party cheated and thus prevented \mathcal{P}_1 from reconstructing the output. Therefore, we need to consider a slightly modified simulation guarantee (as we defined here). We note that if Π has the additional guarantee that it has identifiable abort against adversaries that only actively cheat in the *final* round, then we could – similar to [BBC⁺19a] – restrict ourselves only to functionalities in which \mathcal{P}_1 obtains the output.

IA-MPC Protocol for t Corruptions

The input to the compiler is a k -party natural protocol Π_{nat} (as in Definition 5.4). Let $\widehat{\mathcal{L}}_1, \dots, \widehat{\mathcal{L}}_k$ denote the languages whose existence is guaranteed by Property 3b of Definition 5.4, and let Π_1, \dots, Π_k be $(k-1)$ -verifier dZK proofs for $\widehat{\mathcal{L}}_1, \dots, \widehat{\mathcal{L}}_k$, respectively. The compiled protocol Π operates as follows:

1. The parties emulate Π'_{nat} (i.e., all rounds of Π_{nat} except the final round), where each party \mathcal{P}_i plays the role of the corresponding party in Π'_{nat} . Let $M_i = (M_i^{\text{out}}, M_i^{\text{in}})$ denote the outgoing and incoming messages sent to \mathcal{P}_i in this execution.
2. For every $1 \leq i \leq k$ (in parallel), the parties execute Π_i where \mathcal{P}_i plays the role of the prover and the other parties play the roles of the verifiers. In this execution, \mathcal{P}_i has input M_i and every $\mathcal{P}_j, j \neq i$ has input piece $(M_{ij}^{\text{out}}, M_{ij}^{\text{in}})$.
3. The output is computed in the following way:
 - (a) If there exists an i such that more than t verifiers output reject in Π_i , then let i_0 be the smallest such i , and all parties abort with \mathcal{P}_{i_0} as a corrupted party.
 - (b) Otherwise, all parties compute and broadcast the messages of the final round of Π_{nat} . If some party i refuses to broadcast its message then let i_0 be the smallest such i , and all parties abort with \mathcal{P}_{i_0} as a corrupted party.
 - (c) Otherwise, each party computes its output according to the output of the corresponding party in Π_{nat} .

Throughout the computation, for every pair of parties $i, j \in [k]$, if \mathcal{P}_i was supposed to send a message to \mathcal{P}_j but failed to do so, then \mathcal{P}_j sets that message to be the all-0 string.

Figure 6: Compiler from Semi-Honest Security To Malicious Security with Identifiable Abort

- $\text{Rounds}(\Pi) = \text{Rounds}(\Pi_{\text{nat}}) + \max_{i \in [k]} \text{Rounds}(\Pi_i)$.
- $\text{CC}(\Pi) = \text{CC}(\Pi_{\text{nat}}) + \sum_{i=1}^k \text{CC}(\Pi_i)$.

Proof: The claim regarding the complexities of Π follows directly from its description. We proceed to prove security, which is similar in spirit to that of [BBC⁺19a, Th. 7.3], but additionally uses the *strong* completeness of the underlying dZK proofs to show that Π has *identifiable* abort.

Let \mathcal{A} be an adversary corrupting a subset $T, |T| \leq t$ of parties in Π , and assume without loss of generality that \mathcal{A} is deterministic. We will construct a simulator $\text{Sim}^{\mathcal{A}}$ for \mathcal{A} . We can divide \mathcal{A} into 3 adversaries $\mathcal{A} = (\mathcal{A}', \mathcal{A}_{\text{dZK}}, \mathcal{A}_{\text{fin}})$ that share a state, where $\mathcal{A}', \mathcal{A}_{\text{dZK}}, \mathcal{A}_{\text{fin}}$ describe the operations of \mathcal{A} in Steps 1, 2 and 3 of Π respectively (see Figure 6). $\text{Sim}^{\mathcal{A}}$ will likewise consist of 3 simulators $\text{Sim}', \text{Sim}_{\text{dZK}}, \text{Sim}_{\text{fin}}$ that share state, corresponding to the 3 adversarial entities. Specifically, let Sim' denote the simulator for Π'_{nat} whose existence follows from Property 1 of Definition 5.4. Let $M_{\text{TH}} = (M_{ij}^{\text{out}}, M_{ij}^{\text{in}})_{i \in T, j \notin T}$ denote the simulated messages sent and received by parties $i \in T$ from honest parties $j \notin T$ in the simulation of Sim' .

Sim_{dZK} simulates Step 2 of Π and operates as follows on input $(x_i)_{i \in T}, M_{\text{TH}}$:

- For every $i \in T$, Sim_{dZK} honestly emulates an execution of Π_i with \mathcal{A}_{dZK} , in which the input piece of $\mathcal{P}_j, j \notin T$ is $x^{(j)} = (M_{ji}^{\text{out}}, M_{ji}^{\text{in}})$ (notice that Sim_{dZK} knows the input pieces of all honest parties, since they are determined by $(M_{ij}^{\text{out}}, M_{ij}^{\text{in}})$). Specifically, Sim_{dZK} honestly

emulates the honest parties to generate their messages to the parties in T , and uses \mathcal{A}_{dZK} (with state as determined by $(x_i)_{i \in T}$ and M_{TH}) to generate the messages sent by the parties in T . For every $i \in T$, let M_i^{dZK} denote the messages sent from the honest parties to the parties in T during the execution, and $(o_j^i)_{j \in [k]}$ denote the outputs of the verifiers in this execution.

- For every $i \notin T$, Sim_{dZK} uses the simulator Sim_i that can simulate the views of the parties in T in Π_i (the existence of Sim_i follows from the ZK property of Π_i). More specifically, for every $j \in T$, Sim_{dZK} uses $x^{(j)} = (M_{ji}^{\text{out}}, M_{ji}^{\text{in}})$ as the input piece of \mathcal{P}_j in the execution, and runs $\text{Sim}_i((x^{(j)})_{j \in T})$ to simulate the messages M_i^{dZK} sent from the honest parties to the parties in T during the execution of Π_i , and the outputs $(o_j^i)_{j \in [k]}$ of the verifiers.
- Denote $M^{\text{dZK}} = (M_1^{\text{dZK}}, \dots, M_k^{\text{dZK}})$, and $o^{\text{dZK}} = (o_j^i)_{i,j \in [k]}$.

Let $\mathcal{A}'' = (\mathcal{A}', \text{Sim}_{\text{dZK}}, \mathcal{A}_{\text{out}})$ denote the adversary which operates identically to \mathcal{A} in Steps 1 and 3 of Π , but in Step 2 it operates by running Sim_{dZK} on the view of \mathcal{A}' in Step 1 of Π . By Property 3c in Definition 5.4, \mathcal{A}'' has a corresponding simulator $(\text{Sim}^{\text{in}}, \text{Sim}^{\text{out}})$.

We now describe the simulator $\text{Sim}^{\mathcal{A}}$, which on input $x_T := (x_i)_{i \in T}$ operates as follows.

1. Runs $\text{Sim}'(x_T)$ to simulate the messages $M_{\text{TH}} = (M_{ij}^{\text{out}}, M_{ij}^{\text{in}})_{i \in T, j \notin T}$ exchanged between the honest and corrupt parties during the execution of Π' .
2. Runs $\text{Sim}_{\text{dZK}}((x_i)_{i \in T}, M_{\text{TH}})$ to obtain $M^{\text{dZK}}, o^{\text{dZK}}$. If there exists an $i \in [k]$ such that $\left| \left\{ o_j^i : o_j^i = \text{reject} \right\} \right| > t$ then $\text{Sim}^{\mathcal{A}}$ aborts with the smallest such index i_0 as a corrupted party (notice that in this case, the dZK proof in which \mathcal{P}_{i_0} played the role of the prover has failed, so i_0 is indeed corrupted).
3. Otherwise, it simulates the final round of Π as follows.
 - (a) Runs $\text{Sim}^{\text{in}}(M_{\text{TH}})$ to obtain effective inputs $x'_T := (x'_i)_{i \in T}$ for the corrupted parties.
 - (b) Calls the ideal functionality f with inputs x'_T as the inputs of the corrupted parties, and obtains the output y .
 - (c) Simulates the final round of Π by running $\text{Sim}^{\text{out}}(y, M_{\text{TH}})$ to simulate the messages $M^{\text{fin}} := (M_i^{\text{fin}})_{i \notin T}$ broadcasted by the honest parties in the final round of Π . Then, it uses \mathcal{A} to determine whether the parties in T broadcast messages in the final round. If there exists an $i \in T$ that should have broadcasted a message but did not do so, then $\text{Sim}^{\mathcal{A}}$ aborts with the smallest such index i_0 as a corrupted party. (Notice that by Property 2 of Definition 5.4, if all parties in T broadcast their messages then the output of the honest parties is necessarily correct, conditioned on the event that \mathcal{A}' behaved honestly in Π' .)
 - (d) Outputs $(x_T, M_{\text{TH}}, M^{\text{dZK}}, M^{\text{fin}})$.

Let y, y' denote the outputs of the honest parties in the real execution and in the simulation (respectively) with adversary \mathcal{A} . We now prove that for every choice of inputs x ,

$$\text{SD} \left(\left(\text{Sim}^{\mathcal{A}}(x_T), y' \right), (\text{View}_{\mathcal{A}}(x_T, x_H), y) \right) \leq t \cdot \varepsilon_r + (k-1) \cdot \varepsilon_p + \text{negl}(s)$$

where $\text{View}_{\mathcal{A}}(x_T, x_H) = (\text{View}', \text{View}_{\text{dZK}}, \text{View}_{\text{fin}})$ denotes the view of \mathcal{A} in Steps 1, 2 and 3 of Π (respectively) in which the honest parties have inputs x_H and the corrupted parties have inputs x_T . That is, these consist of the messages exchanged between the honest and corrupt parties in the three stages of the execution. We can assume that even the corrupted parties always send messages they are supposed to send according to Π . This is without loss of generality, since for each adversarial strategy \mathcal{A} in which a subset m of messages of corrupted parties are not sent, there exists an alternative adversarial strategy \mathcal{A}^* in which all messages are sent, and these unsent messages m are fixed to be all-0. This yields the same execution as with \mathcal{A} .

We now bound the statistical distance through a sequence of hybrids. Since x_T is identical in both distributions, we will sometimes omit it from the description of the hybrids.

\mathcal{H}_0 : This is the distribution over $(\text{Sim}^{\mathcal{A}}(x_T), y')$.

\mathcal{H}_1 : In \mathcal{H}_1 , we replace M_{TH} with View' , namely $\mathcal{H}_1 = (x_T, \text{View}', M^{\text{dZK}}, M^{\text{fin}}, y')$.

$SD(\mathcal{H}_0, \mathcal{H}_1) = 0$ because the simulation of Sim' is perfect (see Property 1 in Definition 5.4). Specifically, if $SD(\mathcal{H}_0, \mathcal{H}_1) > 0$ then there exists a choice of optimal coins r for Sim^{dZK} , Sim^{fin} which maximizes the statistical distance. Define g_{x_H} to be the function that on input the messages M exchanged between the honest and corrupt parties, operates as follows: (1) applies Sim^{dZK} (with the optimal coins r) to x_T, M , (2) applies Sim^{in} to M to obtain effective inputs $(x'_i)_{i \in T}$ for the parties in T , (3) computes $y = f(x'_1, \dots, x'_k)$ where $x'_i = x_i$ for every $i \notin T$ (g_{x_H} can compute y because it has x_H hard-wired into it), (4) applies Sim^{out} (with the optimal coins r) to y and M , and (5) outputs M , together with the outputs of Sim^{dZK} , Sim^{out} generated in Steps (2), (4). Then since applying a function to the random variables does not increase the statistical distance, we have:

$$0 < SD(\mathcal{H}_0, \mathcal{H}_1) \leq SD(\mathcal{H}_0|r, \mathcal{H}_1|r) = SD(g_{x_H}(M_{\text{TH}}, g_{x_H}(\text{View}')) \leq SD(M_{\text{TH}}, \text{View}')$$

where $\mathcal{H}_0|r, \mathcal{H}_1|r$ denote the distributions obtained by conditioning $\mathcal{H}_0, \mathcal{H}_1$ on the coins r . This is a contradiction since the simulation of Sim' is perfect.

\mathcal{H}_2 : in \mathcal{H}_2 , we replace the simulated final round messages M^{fin} with View_{fin} , namely $\mathcal{H}_2 = (x_T, \text{View}', M^{\text{dZK}}, \text{View}_{\text{fin}}, y')$.

We show that $SD(\mathcal{H}_1, \mathcal{H}_2) \leq |T| \cdot \varepsilon_r + \text{negl}(s)$ due to Property 2 of Definition 5.4. We consider two cases. First, if one of the dZK executions failed, then the hybrids abort before the final round of Π is executed, and moreover the same party is announced as corrupted in both hybrids (this is because the identity of the corrupted party is determined based on $(\text{View}', M^{\text{dZK}})$ alone). Therefore, conditioned on this case, the hybrids are identically distributed. Second, assume that all dZK executions passed. Notice that for every $i \in T$, the input pieces $x_i^{(j)}, j \notin T$ used by the honest parties in the execution of Π_i are well defined (they are determined by View'). If the execution of Π_i succeeded then the soundness of Π_i guarantees that except with probability ε_r there exists a choice $x_i^{(j)} = (M_{ji}^{\text{out}}, M_{ji}^{\text{in}})$ of input pieces for the parties $j \in T$ such that $(x_i^{(1)}, \dots, x_i^{(k)}) \in \widehat{\mathcal{L}}_i$. Using the union bound, except with probability $|T| \cdot \varepsilon_r \leq t \cdot \varepsilon_r$, there exist such choices of input pieces for all $i \in T$ which simultaneously satisfy that $(x_i^{(1)}, \dots, x_i^{(k)}) \in \widehat{\mathcal{L}}_i$. Conditioned on this event, Property 3c guarantees that $SD(M^{\text{fin}}, \text{View}_{\text{fin}}|\text{pfix}) = \text{negl}(s)$ where $\text{View}_{\text{fin}}|\text{pfix}$ denotes the distribution View_{fin} conditioned on pfix , and pfix consists of the messages exchanged between all $i \in T, j \notin T$ in the execution of Π' . Since pfix is determined by View' , which is identically distributed in both hybrids, we conclude that $SD(\mathcal{H}_1, \mathcal{H}_2) \leq |T| \cdot \varepsilon_r + \text{negl}(s)$.

\mathcal{H}_3 : in \mathcal{H}_3 we replace the simulated messages M^{dZK} sent to the corrupted parties in the dZK proofs with the actual messages View_{dZK} , namely $\mathcal{H}_3 = (x_T, \text{View}', \text{View}_{\text{dZK}}, \text{View}_{\text{fin}}, y')$.

Then $SD(\mathcal{H}_2, \mathcal{H}_3) \leq (k - |T|) \cdot \varepsilon_p \leq^{|T| \geq 1} (k - 1) \cdot \varepsilon_p$ by the ZK property of the dZK proofs (which holds under parallel composition, since exactly $k - |T|$ dZK proofs are simulated by Sim_{dZK}).

Notice that \mathcal{H}_3 is distributed identically to $(\text{View}_{\mathcal{A}}(x_T, x_H), y) = (x_T, \text{View}', \text{View}_{\text{dZK}}, \text{View}_{\text{fin}}, y)$, because y is uniquely determined by x_H and $\text{View}_{\mathcal{A}}(x_T, x_H)$.

■

5.3.4 Comparison With Previous Works

We now compare our compiler of Figure 6 (Section 5.3) with two previous works: the compiler of [BBC⁺19a, Th. 7.3] and the compiler of [IOZ14].

Comparison with the Compiler of [BBC⁺19a]. Boneh et al. [BBC⁺19a] describe a generic compiler from natural protocols with an honest majority to maliciously secure protocols with abort. They rely on dZK proofs that are *not* required to have strong completeness, namely their compiler only relies on the completeness (when all parties are honest), soundness, and ZK under parallel composition (see [BBC⁺19a, page 37]), of the underlying dZK. In contrast, our compiler obtains security with *identifiable* abort, and crucially relies on the *strong* completeness of the underlying dZK proof. Thus, our compiler obtains a stronger security guarantee by leveraging a stronger security guarantee of the underlying building block. In particular, our compiler cannot be instantiated with the dZK proofs of [BBC⁺19a] (since they do not have strong completeness). In particular, while they obtain as a corollary protocols with malicious security with abort in the honest-majority setting, we get identifiable abort when $t < (k - 2) / 6$. However, since our compiler is generic and can use any dZK proof (with strong completeness and ZK which is preserved under parallel repetition), if better dZK proofs are developed in the future, our compiler could be instantiated with them to obtain a better ratio between the number of corrupted parties t and the total number of parties k in the protocol.

Comparison with the Compiler of [IOZ14]. Ishai et al. [IOZ14] describe a generic compiler from (any) SH-secure protocols in the *Correlated Randomness (CR)* model to maliciously secure protocols with identifiable abort in the CR model. (Roughly, in the CR model, (r_1, \dots, r_k) are sampled from a specific distribution, and each party \mathcal{P}_i obtains r_i before the inputs are determined.) Their compiler is defined and analyzed in the Universally Composable (UC) security framework, and works in the *dishonest majority* setting. (In contrast, our compiler works only in the honest majority setting — and with non-optimal threshold — because strongly-complete dZK is only meaningful when there is an honest majority.) Their compiler can be applied to any SH-secure protocol Π_{SH} in the CR model, where the compiled protocol Π uses correlated randomness of its own (i.e., it requires the correlated randomness used by Π_{SH} and *some additional* correlated randomness).

When applied to a protocol with R rounds, the compiled protocol Π has $4R$ rounds, since it emulates Π_{SH} round by round, adding several rounds of checks after each round of Π_{SH} . However, we notice that when Π_{SH} is a natural protocol (see Definition 5.4) then one can use the tools developed in [BBC⁺19b] to modify their compiler such that it runs all but the final round of Π_{SH} , then checks the execution of all these rounds, and then runs the final round of Π_{SH} , in which case Π will have $R + 3$ rounds. In comparison, our compiler works in the plain model (assuming an honest majority) and does not require any correlated randomness; when instantiated with our

dZK proofs of Section 4 it only relies on an ideal coin-tossing functionality (which can be realized in the plain model), and the compiled protocol has $R + 3$ rounds. Thus, our compiler provides an alternative method of obtaining information theoretic security with identifiable abort in the *honest majority* setting. We note that our compiler is conceptually simple, basically applying parallel dZK proofs on top of Π_{SH} before the final round. This should be contrasted with the compiler of [IOZ14], whose description is much more complex, and is based on several elaborate building blocks which they develop.

6 Ideal dZK and Connections with Verifiable Relation Sharing

In this section we describe an alternative formulation of dZK proofs via an idealized functionality, and discuss the connection between dZK and Verifiable Relation Sharing (VRS) [GIKR02, AKP20a, AKP22]. We first formally define the ideal dZK functionality in Section 6.1, and prove that our protocols of Section 4 securely realize it (when instantiated with appropriate MPC protocols). Then, we formally define the notion of VRS in Section 6.2, and discuss its connections to dZK.

6.1 The Ideal dZK Functionality

We define the ideal dZK functionality with respect to robust relations (as noted in Section 1.2.1, this is inherent to settings in which completeness and/or soundness should hold in the presence of corrupted verifiers). Moreover, since in dZK (as defined in Section 2.1) the honest prover and verifiers have consistent input pieces which are valid encodings, we restrict the discussion to such inputs.

Intuitively, the ideal dZK functionality \mathcal{F}_{dZK} is parameterized by a distributed relation $\widehat{\mathcal{R}}_{\mathcal{C}}$. It takes inputs from the prover \mathcal{P} and k verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$, and returns an output to the verifiers. Each verifier \mathcal{V}_i provides its input piece $x^{(i)}$, and the prover provides all input pieces $x^{(1)}, \dots, x^{(k)}$ as well as a witness w . The functionality verifies that: (1) the input pieces provided by the prover and all verifiers are consistent;²³ (2) the input pieces are close to an encoding of some input x ; and (3) (x, w) are in the relation. This is formalized in Figure 7.

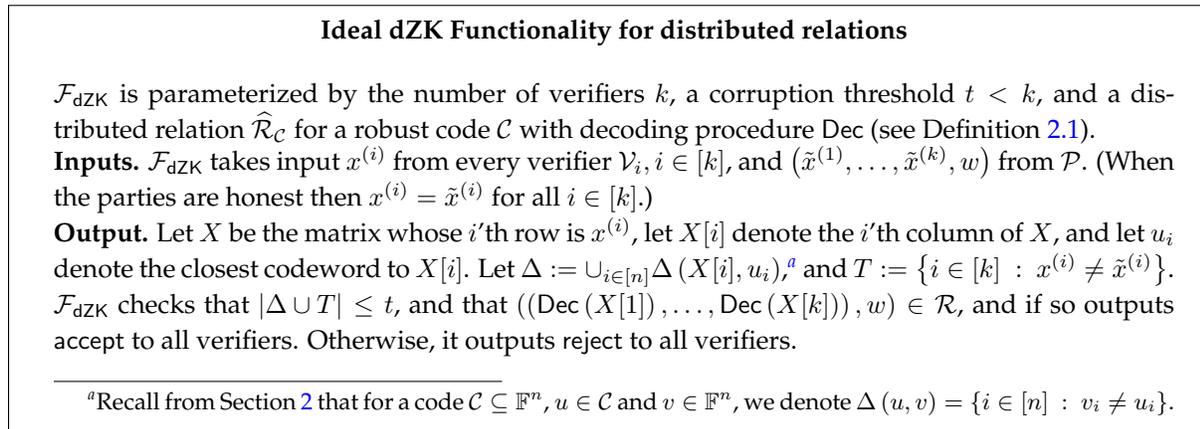


Figure 7: Ideal \mathcal{F}_{dZK} Functionality

²³A bit more specifically, since we want to capture *strongly complete* dZK proofs, the functionality \mathcal{F}_{dZK} should tolerate a small number of inconsistencies between the input pieces provided by \mathcal{P} and the verifiers.

Definition 6.1 (Securely Realizing \mathcal{F}_{dZK}). Let $k, t \in \mathbb{N}$ such that $t < k$, and $\varepsilon \in [0, 1]$. We say that a protocol Π between a prover \mathcal{P} and k verifiers t -securely realized \mathcal{F}_{dZK} for $\widehat{\mathcal{R}}_{\mathcal{C}}$ with error ε if there exists a PPT simulator Sim that interacts with the real-world adversary, such that the following holds. For any $x^{(1)}, \dots, x^{(k)}, w$, and any adversary \mathcal{A} corrupting a subset $T, |T| \leq t$ of verifiers and possibly also \mathcal{P} , we have

$$SD\left(\left(\text{View}_{\mathcal{A}}\left(\left(x^{(i)}\right)_{i \in T}, w'\right), (y_i)_{i \notin T}\right), \left(\text{Sim}\left(\left(x^{(i)}\right)_{i \in T}, (y_i)_{i \in T}, (y_i)_{i \notin T}\right)\right)\right) \leq \varepsilon$$

where: $w' = w$ if \mathcal{P} is honest, otherwise $w' = \perp$; $\text{View}_{\mathcal{A}}$ denotes \mathcal{A} 's view in the execution of Π when the honest parties have inputs $x^{(1)}, \dots, x^{(k)}, w$; and y_i is the output of verifier \mathcal{V}_i , which in the real world is determined by the execution of Π , and in the ideal world by the output of \mathcal{F}_{dZK} when the inputs of the corrupted parties are determined by Sim .

We will prove that our dZK of Figure 3 securely realizes the ideal dZK functionality of Figure 7 when the underlying MPC protocol satisfies an appropriate notion of security. Specifically, we will use the Micali-Rogaway definition of secure function evaluation [MR91]. Roughly, they define protocols to have a special *committal round* R in which the (effective) inputs of all parties are determined, in the following sense. The protocol is associated with an *effective input function* \mathcal{I} which at round R can determine each party's effective input given only the messages that party sent and received in the protocol so far. The protocol is additionally associated with an *effective output function* \mathcal{O} which can determine each party's output in the last round, given only the messages it sent and received throughout the protocol execution. The main properties we need from such protocols are the following. First, the outputs (in the real and simulated executions) are determined by the effective output function \mathcal{O} . Second, in the simulated execution the effective inputs of corrupted parties are determined according to the effective input function \mathcal{I} . Third, \mathcal{I} determines the effective input of party i given *only the messages party i sent and received*. Finally, the simulation is perfect. We note that natural perfectly-secure MPC protocols satisfy this security notion, and refer the interested reader to [DM00, Sec. 2.1] for further details.

Notation 2. We say that an MPC protocol Π is t -MR secure if it satisfies the Micali-Rogaway security definition discussed above, for adversaries corrupting at most t verifiers.

We now prove that when instantiated with a $2t$ -MR secure MPC protocol, our dZK protocol of Figure 3 securely realizes the ideal dZK functionality. We stress that we only consider distributed relations $\widehat{\mathcal{R}}_{\mathcal{C}}$ for *robust* codes \mathcal{C} , otherwise (as discussed in Section 1.2.1) membership of the input in the relation is not well defined

Theorem 6.1 (Secure Implementation of Ideal dZK). Let $t \in \mathbb{N}$ such that $k > 6t + 2$. Let $\widehat{\mathcal{R}}_{\text{RRS}}$ be a k -distributed relation over a field \mathbb{F} , and let Π be a $2t$ -MR secure protocol for $\widehat{\mathcal{R}}_{\text{RRS}}$. Then the proof system Π_{dist} of Figure 3 securely realizes the ideal dZK functionality \mathcal{F}_{dZK} for $\widehat{\mathcal{R}}_{\text{RRS}}$ with error $\binom{k}{2} \frac{N}{|\mathbb{F}|}$ where N is a bound on the total number of field elements exchanged between a pair of parties in Π .

Proof (sketch): Let Sim_{Π} denote the simulator for Π whose existence is guaranteed by the $2t$ -MR security of Π . We describe the simulator Sim , which interacts with the adversary \mathcal{A} , and consider two possible cases based on whether \mathcal{A} corrupts \mathcal{P} or not. We denote by $\mathcal{T}, |\mathcal{T}| \leq t$ the set of verifiers which \mathcal{A} corrupts and consider two cases.

Case (1): \mathcal{P} is honest. In this case, the input pieces used by \mathcal{P} are consistent with the input pieces of the honest verifiers. Since $2t$ -MR security implies *perfect correctness*, in this case the proof of the strong completeness property in Theorem 4.1 shows that all honest verifiers output accept

(with probability 1). Therefore, the outputs $\hat{y}_i, i \notin \mathcal{T}$ are trivially simulatable. As for the joint view of the corrupted verifiers, this is simulatable by Sim_Π due to the $2t$ -MR security of Π .

Case (2): \mathcal{P} is corrupted. In this case, in the first round of the dZK protocol the adversary \mathcal{A} sends to Sim the prover messages to the uncorrupted verifiers $\mathcal{V}_i, i \notin \mathcal{T}$. These, together with the inputs of the honest verifiers (which are known to Sim because they are contained in \mathcal{A} 's input), and the outcome of the coin tosses (which Sim chooses uniformly at random) fully determine the messages that the honest verifiers send in the dZK of Figure 3. Sim then emulates the execution with \mathcal{A} to determine whether the verifiers accept or reject. (We note that in the protocol of Figure 3, all honest verifiers compute the same public output predicate, and in particular they all have the same output.) If the honest verifiers reject, then simulation is trivial, since the simulator can simply provide some invalid witness w' to \mathcal{F}_{dZK} and this would cause the outputs of all honest verifiers to be reject. Moreover, the view of \mathcal{A} can be computed from the messages of the honest verifiers in this execution.

If, on the other hand, the honest verifiers output accept, then Sim must extract a valid witness to provide to \mathcal{F}_{dZK} (to cause \mathcal{F}_{dZK} to output accept to the honest verifiers). For this, Sim uses the effective input function \mathcal{I} for Π , whose existence is guaranteed by the $2t$ -MR security of Π . More specifically, since the honest verifiers accept, then there exists a set C of size $|C| \leq t$ such that for every $i, j \notin C$ the MACs $m_{i,j}, m_{j,i}$ are equal: $m_{i,j} = m_{j,i}$, and the views of i, j are locally consistent. We condition on the event that for every honest $i, j \notin C$, the views of the corresponding parties in Π are consistent (according to Lemma 4.1, this happens except with probability $\binom{k}{2} \frac{N}{|\mathbb{F}|}$). Conditioned on this event, the prover messages (which \mathcal{A} sends to Sim in the first round of the dZK protocol) define an execution of Π in which the parties in $[k] \setminus (\mathcal{T} \cup C)$ are honest. Moreover, the prover messages also determine *all the messages exchanged between the honest and corrupted parties in Π* . Crucially, this allows Sim to compute the effective inputs of all corrupted parties in Π – including P_0 and all $P_i, i \in \mathcal{T}$ – using the effective-input function \mathcal{I} . In particular, this determines an effective witness w' (which is P_0 's input in the virtual MPC).

We claim that the $2t$ -MR security of Π guarantees that w' is a valid witness (causing \mathcal{F}_{dZK} to output accept to the honest verifiers). This would complete the proof since (as noted above) Sim is able to determine the messages sent from the honest verifiers in the dZK. To see why w' is a valid witness, notice that otherwise we have found an adversarial strategy \mathcal{A}_Π (corrupting the parties in $\mathcal{T} \cup C$) violating the (perfect) $2t$ -MR security of Π . Indeed, the adversary \mathcal{A}_Π emulates \mathcal{A} to obtain the “MPC in the head” views. It uses these to generate the messages from the corrupted parties in $\mathcal{T} \cup C$ to the honest parties. With a non-zero probability, the honest parties will respond with the messages reported in these views²⁴ (otherwise, the adversary \mathcal{A}_Π gives up and aborts). If so, \mathcal{A}_Π is able to complete the execution, and the outputs of the honest parties will indeed be 1 (since we have assumed that the honest verifiers in the dZK accept, and this only happens when the honest parties in Π output 1). However, since the effective inputs determined by \mathcal{I} do not satisfy the relation $\widehat{\mathcal{R}}_{\text{RRS}}$ then the simulator Sim_Π fails to simulate the execution (because the effective inputs of the corrupted parties in the ideal execution are determined by \mathcal{I}), which contradicts the perfect $2t$ -MR security of Π . ■

²⁴Here, we also use the fact that the honest verifiers check local consistency of their MPC views. Therefore, for every $i \notin \mathcal{T} \cup C$, the view of the i th party in Π is locally consistent, meaning there exists an execution in which that party does send the messages which are implicit in its view.

6.2 Connection between dZK and VRS

In this section we discuss the connection between dZK and VRS. While these notions are quite different (for example, VRS is a single-input functionality, whereas dZK takes inputs from all parties), we show reductions between the two primitives. We first formally define VRS.

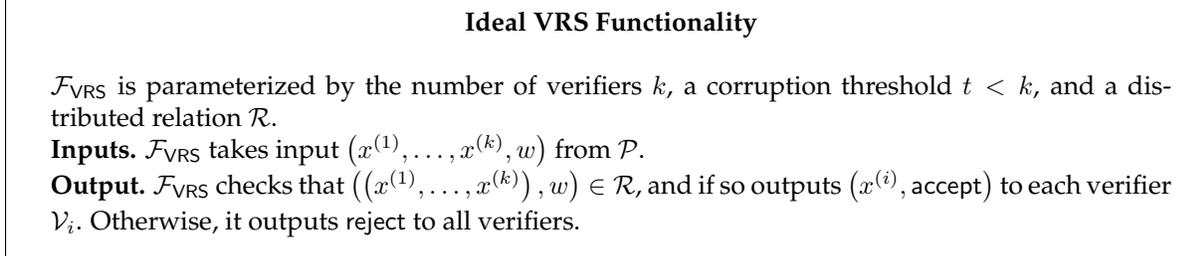


Figure 8: Ideal \mathcal{F}_{VRS} Functionality

Notation 3. Let \mathcal{R} be a distributed relation. We say that a protocol Π between a prover \mathcal{P} and verifiers $\mathcal{V}_1, \dots, \mathcal{V}_k$ is a (t, ε) -VRS for \mathcal{R} if it securely realizes the ideal VRS functionality of Figure 8 with error ε (a-la Definition 6.1).

Next, we show reductions between the notions of dZK and VRS. We begin with a general construction of VRS from dZK protocols. The reduction preserves the efficiency measures of the dZK protocol, but increases the round complexity by 1. We stress that the reduction is only for *robust relations* (as defined in Definition 2.2 in Section 2). In particular, the outputs of the honest verifiers may not correspond to *any* valid encoding (the only guarantee is that they are sufficiently close to a valid encoding such that the underlying message is well-defined and can be decoded). This might not suffice for certain applications of VRS (which might require that the output pieces form a *valid* encoding even if the prover is corrupted, a property which VRS can satisfy in general).

Claim 6.1 (dZK \Rightarrow VRS, for robust relations). Let $k, t \in \mathbb{N}$ such that $t < k$, let $\widehat{\mathcal{R}}_{\mathcal{C}}$ be a distributed relation where \mathcal{C} is a perfectly $2t$ -robust code, and let $\varepsilon \in [0, 1]$. If there exists a t -secure realization Π_{dZK} of \mathcal{F}_{dZK} for $\widehat{\mathcal{R}}_{\mathcal{C}}$ with error ε , then there exists a (t, ε) -VRS Π_{VRS} for $\widehat{\mathcal{R}}_{\mathcal{C}}$. Moreover, if Π_{dZK} has R rounds and communication complexity CC then Π_{VRS} has $R + 1$ rounds and communication $\text{CC} + n$, where n is the input length. Furthermore, if the first round of Π_{dZK} consists only of messages from \mathcal{P} to the verifiers, then Π_{VRS} will have R rounds.

We note that in our dZK of Figure 3, the first round consists solely of messages from \mathcal{P} to the verifiers, so it gives rise to 3-round VRS protocols for relations $\widehat{\mathcal{R}}_{\text{RRS}}$ with security against $t < (k - 2)/6$ corruptions.

Proof of Claim 6.1: The VRS protocol Π_{VRS} operates as follows. In the first round, the prover \mathcal{P} sends to each verifier \mathcal{V}_i the input piece $x^{(i)}$. Then, the parties execute the dZK protocol Π_{dZK} , where \mathcal{V}_i uses $x^{(i)}$ as his input piece. Clearly, the round complexity increases by 1 and the communication complexity increases by $n = \sum_{i \in [k]} |x^{(i)}|$. In case the first round of Π_{dZK} consists only of messages from the prover to the verifiers then the first round of Π_{dZK} can be executed in parallel to the prover sending the input pieces to the verifiers, and so Π_{VRS} will have only R rounds.

We now prove that Π_{VRS} securely realizes \mathcal{F}_{VRS} . Let Sim_{dZK} denote the simulator for Π_{dZK} . We construct a simulator Sim_{VRS} that interacts with an adversary \mathcal{A}_{VRS} in Π_{VRS} . \mathcal{A}_{VRS} gives rise to an adversary \mathcal{A}_{dZK} in Π_{dZK} , which operates exactly as \mathcal{A}_{VRS} does after the first round of Π_{VRS} .

Assume that \mathcal{A}_{VRS} corrupts a set $\mathcal{T}, |\mathcal{T}| \leq t$ of verifiers, then we consider two cases based on whether \mathcal{A}_{VRS} also corrupts the prover.

Case (1): \mathcal{P} is honest. Let $(x^{(1)}, \dots, x^{(k)}, w)$ denote the prover's inputs. In this case, the input pieces which \mathcal{P} sent to the verifiers in the first round are exactly the input pieces which she uses in Π_{dZK} , and honest verifiers similarly use these input pieces. Because the code is $2t$ -robust, the inputs used by the corrupted verifiers do not affect the output of \mathcal{F}_{dZK} , and so the output of \mathcal{F}_{dZK} is equal to the output of \mathcal{F}_{VRS} . Therefore, since Π_{dZK} t -securely realizes \mathcal{F}_{dZK} with error ε , we have that Π_{VRS} is a (t, ε) -VRS.

Case (2): \mathcal{P} is corrupted. In this case, an execution of Π_{VRS} with \mathcal{A}_{VRS} corresponds to an execution of Π_{dZK} with \mathcal{A}_{dZK} in which the input pieces of the honest verifiers are the values $(x^{(i)})_{i \notin \mathcal{T}}$ which \mathcal{A}_{VRS} sent to them in the first round of Π_{VRS} . Sim_{VRS} emulates the dZK simulator Sim_{dZK} , using the messages from \mathcal{A}_{VRS} to generate the messages of \mathcal{A}_{dZK} . At some point in its emulation, Sim_{dZK} generates effective inputs for the ideal functionality \mathcal{F}_{dZK} , including effective inputs for \mathcal{P} which contain input pieces $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}$. Sim_{VRS} then completes the simulation by checking the following condition. Let \tilde{X} denote the matrix whose i 'th row is $\tilde{x}^{(i)}$, let \tilde{u}_i be the closest codeword to $\tilde{X}[i]$, and let $\tilde{\Delta} := \{i \in [k] : x^{(i)} \neq \tilde{x}^{(i)}\} \cup \left(\cup_{i \in [n]} \Delta \left(\tilde{X}[i], \tilde{u}_i \right) \right)$. Then Sim_{VRS} checks that $|\tilde{\Delta}| \leq t$. If the check passes, then Sim_{VRS} uses $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}$ as its effective inputs for \mathcal{F}_{VRS} . Otherwise, Sim_{VRS} provides some invalid inputs (i.e., that don't satisfy the relation). It outputs the simulated view generated by Sim_{dZK} .

Notice that if the check passes, then by the $2t$ -robustness of \mathcal{R} , the input $x = \left(\text{Dec} \left(\tilde{X}[1] \right), \dots, \text{Dec} \left(\tilde{X}[k] \right) \right)$ encoded by $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}$ is well defined *regardless* of which input pieces the corrupted verifiers choose to use for the computation. Therefore, the security of Π_{dZK} guarantees that the joint distribution of \mathcal{A}_{dZK} 's view, together with the outputs of the honest verifiers in Π_{dZK} , is ε -statistically close to the joint distribution of the outputs of Sim_{dZK} and the honest verifiers in the ideal execution. The former is identical to the joint distribution of \mathcal{A}_{VRS} 's view, together with the outputs of the honest verifiers in Π_{VRS} . To conclude the proof, we notice that the output of \mathcal{F}_{dZK} is exactly the output of \mathcal{F}_{VRS} , except when $|\tilde{\Delta}| > t$. However, in this case the output in both the real and simulated execution is reject. ■

We now show the converse reduction, proving that VRS protocols for robust relations give rise to dZK protocols. To construct a dZK protocol for a relation $\hat{\mathcal{R}}_C$, we will use a VRS protocol for a related relation $\hat{\mathcal{R}}_C^e$ (e for *exact*) which contains only *valid encodings* of inputs in \mathcal{R} . (Recall that $\hat{\mathcal{R}}_C$ contains also *invalid* encodings, as long as they decode to valid inputs in \mathcal{R} .) This is formalized in the following notation.

Notation 4. For a relation \mathcal{R} and a code C with encoding procedure Enc , we define the relation:²⁵

$$\hat{\mathcal{R}}_C^e = \left\{ \left(\left(x^{(1)}, \dots, x^{(k)} \right), w \right) : \left((x_1, \dots, x_n), w \right) \in \mathcal{R}, \right. \\ \left. \text{where } X[i] = \text{Enc}(x_i) \text{ for all } 1 \leq i \leq n \right\}$$

Claim 6.2 (VRS \Rightarrow dZK, for robust relations).²⁶ Let $k, t \in \mathbb{N}$ such that $t < k$, let $\hat{\mathcal{R}}_C$ be a distributed relation where C is a perfectly $2t$ -robust code, and let $\varepsilon \in [0, 1]$. If there exists a (t, ε) -VRS Π_{VRS} for $\hat{\mathcal{R}}_C^e$, then there exists a t -secure realization Π_{dZK} of \mathcal{F}_{dZK} for $\hat{\mathcal{R}}_C$ with error ε . Moreover, if Π_{VRS} has R rounds and communication CC then Π_{dZK} has $R + 1$ rounds and communication $\text{CC} + k$.

²⁵In the following, recall that $X[i]$ is the i th column of X , and $x^{(i)}$ is the i th row of X .

²⁶We thank Benny Applebaum for pointing out to us this reduction (and the round-preserving one discussed below).

Proof: The dZK protocol Π_{dZK} operates as follows. The parties emulate the VRS protocol Π_{VRS} . Once the protocol terminates, if a verifier received an output $\tilde{x}^{(i)}$ which is not his input piece $x^{(i)}$ then he broadcasts a complaint against \mathcal{P} (this can be done by sending a single bit). Each verifier \mathcal{V}_i then determines his output as follows. If the output of Π_{VRS} was reject, or more than t verifiers broadcasted a complaint, then \mathcal{V}_i rejects. Otherwise, he accepts. Clearly, the round complexity increases by 1 and the communication complexity increases by at most k .

We now prove that Π_{dZK} securely realizes \mathcal{F}_{dZK} . Let Sim_{VRS} denote the simulator for Π_{VRS} . We construct a simulator Sim_{dZK} that interacts with an adversary \mathcal{A}_{dZK} in Π_{dZK} . \mathcal{A}_{dZK} gives rise to an adversary \mathcal{A}_{VRS} in Π_{VRS} , which operates exactly as \mathcal{A}_{dZK} does in the first R rounds of Π_{dZK} . Assume that \mathcal{A}_{dZK} corrupts a set $\mathcal{T}, |\mathcal{T}| \leq t$ of verifiers, then we consider two cases based on whether \mathcal{A}_{dZK} also corrupts the prover.

Case (1): \mathcal{P} is honest. In this case, the input pieces which \mathcal{P} used in Π_{VRS} are consistent with the input pieces used by the honest verifiers, and these are valid encodings of some input x . Because the code is $2t$ -robust, the inputs used by the corrupted verifiers do not affect the output of \mathcal{F}_{dZK} , and so the output of \mathcal{F}_{dZK} is equal to the output of \mathcal{F}_{VRS} . Moreover, since only corrupted verifiers will broadcast complaints in the last round, there will be at most t complaints. Therefore, since Π_{VRS} is a (t, ε) -VRS for $\widehat{\mathcal{R}}_C^\varepsilon$, then Π_{dZK} t -securely realizes \mathcal{F}_{dZK} for $\widehat{\mathcal{R}}_C$ with error ε .

Case (2): \mathcal{P} is corrupted. In this case, Sim_{dZK} emulates the VRS simulator Sim_{VRS} , using the messages from \mathcal{A}_{dZK} to generate the messages of \mathcal{A}_{VRS} . At some point in its emulation, Sim_{VRS} generates effective inputs for the ideal functionality \mathcal{F}_{VRS} , consisting of the input pieces $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}$ and witness w' provided by \mathcal{P} . If $x^{(i)} \neq \tilde{x}^{(i)}$ for more than t honest verifiers, or $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ do not form valid encodings, then Sim_{dZK} provides some invalid inputs (i.e., that don't satisfy the relation) to \mathcal{F}_{dZK} . Otherwise, it uses $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}, w'$ as its effective inputs for \mathcal{F}_{dZK} . Then, Sim_{dZK} outputs the simulated view generated by Sim_{VRS} .

Notice that if both checks pass, then by the $2t$ -robustness of \mathcal{R} , the input encoded by $\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)}$ is well defined *regardless* of the input pieces of the corrupted verifiers. Therefore, the security of Π_{VRS} guarantees that (a) the joint distribution of \mathcal{A}_{VRS} 's view, together with the outputs of the honest verifiers in Π_{VRS} , is ε -statistically close to (b) the joint distribution of the outputs of Sim_{VRS} and the honest verifiers in the ideal execution. Notice that \mathcal{A}_{dZK} 's view consists exactly of \mathcal{A}_{VRS} 's view, together with the messages broadcasted in the last round. This concludes the proof for the case that in the simulation $|\{i : \tilde{x}^{(i)} \neq x^{(i)} \wedge i \text{ is honest}\}| \leq t$ and $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ form valid encodings, since in this case the outputs of \mathcal{F}_{dZK} and \mathcal{F}_{VRS} are identical, and Sim_{dZK} can perfectly simulate the broadcasts of the last round.

We now consider the case that $|\{i : \tilde{x}^{(i)} \neq x^{(i)} \wedge i \text{ is honest}\}| > t$, or that $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ do not form valid encodings. First, if $|\{i : \tilde{x}^{(i)} \neq x^{(i)} \wedge i \text{ is honest}\}| > t$ then the outputs of the honest verifiers (in the real and ideal executions) are reject. Moreover, the view of \mathcal{A}_{VRS} is still ε -statistically close to the output of Sim_{VRS} , and Sim_{dZK} can perfectly simulate the broadcasts of the last round. Therefore, the joint distribution of the outputs of Sim_{dZK} and the honest verifiers is ε -statistically close to the joint distribution of \mathcal{A}_{dZK} 's view and the outputs of the honest verifiers in Π_{dZK} . Finally, if $(\tilde{x}^{(1)}, \dots, \tilde{x}^{(k)})$ do not form valid encodings, then by the security of Π_{VRS} , (a) is ε -statistically close to (b), and additionally the output of \mathcal{F}_{VRS} is reject. Therefore, except with probability ε , the outputs of the honest verifiers in Π_{VRS} (and consequently also in Π_{dZK}) are reject, as they are in the simulation. ■

Round-Preserving VRS to dZK Reduction. The reduction of Claim 6.2 increases the round complexity by 1. We note that one can obtain a round-preserving reduction if the VRS protocol satisfies an additional property (which is satisfied by, e.g., the VRS of [AKP22] for certain thresholds, as

well as by other natural VRS instantiations [App22]). Roughly, the property is that in the first VRS round the verifiers obtain from \mathcal{P} “tentative” input pieces which – if \mathcal{P} is honest – will be their final outputs. If the VRS satisfies this additional property then the last round of the dZK protocol Π_{dZK} (described in the proof of Claim 6.2) in which verifiers broadcast their complaints can be performed in parallel to the second VRS round. Alternatively, if at the onset of the protocol the verifiers already hold public commitments to their input pieces, then consistency with these committed values can be proven in parallel to running the VRS (without requiring any additional property from the VRS).

Acknowledgments

We thank Benny Applebaum for helpful discussions and for pointing out to us the reduction from VRS to dZK. We thank Laasya Bangalore for pointing out an error in Theorem 6.1 in an earlier version of the paper (following which we revised the acceptance criteria in Step 6 of Figure 3, and revised the statement and proof of Theorem 6.1).

The first and third authors are supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office. The first author is supported by ISF grant No. 1316/18. The first and second authors are supported by DARPA under Contract No. HR001120C0087. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. The first author is supported by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Algorand Foundation.

References

- [ABT19] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In *EUROCRYPT, Proceedings, Part II*, pages 504–531. Springer, 2019.
- [ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *CRYPTO, Proceedings, Part II*, pages 395–424. Springer, 2018.
- [ACGJ19] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Two round information-theoretic MPC with malicious security. In *EUROCRYPT, Proceedings, Part II*, pages 532–561. Springer, 2019.
- [AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkatasubramanian. Ligerio: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104, 2017.
- [AKP20a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In *TCC, Proceedings, Part II*, pages 562–594. Springer, 2020.
- [AKP20b] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *FOCS*, pages 1277–1284. IEEE, 2020.
- [AKP22] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading NIZKs with honest majority - (extended abstract). In *CRYPTO, Proceedings, Part IV*, pages 33–56. Springer, 2022.

- [App22] Benny Applebaum. Private communication. 2022.
- [BBC⁺19a] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. How to prove a secret: Zero-knowledge proofs on distributed data via fully linear PCPs. *Cryptology ePrint Archive*, Report 2019/188, 2019. <https://eprint.iacr.org/2019/188>.
- [BBC⁺19b] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO*, pages 67–97, 2019.
- [BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, pages 14:1–14:17, 2018.
- [BCI⁺20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed-Solomon codes. In *FOCS*, pages 900–909, 2020.
- [BD91] Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). In *EUROCRYPT*, pages 81–95. Springer, 1991.
- [BFH⁺20] Rishabh Bhaduria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkatasubramanian, Tiancheng Xie, and Yupeng Zhang. Liger++: A new optimized sublinear IOP. In *CCS*, pages 2025–2038, 2020.
- [BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, pages 1292–1303. ACM, 2016.
- [BGIN19] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In *CCS*, pages 869–886. ACM, 2019.
- [BGIN20] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Efficient fully secure computation via distributed zero-knowledge proofs. In *ASIACRYPT*, pages 244–276, 2020.
- [BGIN21] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Sublinear GMW-style compiler for MPC with preprocessing. In *CRYPTO*, pages 457–485, 2021.
- [BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT*, pages 236–250. Springer, 1998.
- [BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.
- [BJO⁺22] Carsten Baum, Robin Jadoul, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. Feta: Efficient threshold designated-verifier zero-knowledge proofs. In *CCS*, pages 293–306. ACM, 2022.
- [BMMM20] Nicholas-Philip Brandt, Sven Maier, Tobias Müller, and Jörn Müller-Quade. Constructing secure multi-party computation with identifiable abort. *IACR Cryptol. ePrint Arch.*, page 153, 2020.
- [BOS16] Carsten Baum, Emmanuela Orsini, and Peter Scholl. Efficient secure multiparty computation with identifiable abort. In *TCC*, pages 461–490, 2016.
- [BOSS20] Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round MPC with identifiable abort and public verifiability. In *CRYPTO*, pages 562–592, 2020.
- [Bra21] Nicholas Brandt. Tight setup bounds for identifiable abort. *IACR Cryptol. ePrint Arch.*, page 684, 2021.
- [CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *USENIX*, pages 259–282, 2017.
- [CBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *SP*, pages 321–338, 2015.

- [CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.
- [CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19. ACM, 1988.
- [CCP21] Anirudh C, Ashish Choudhury, and Arpita Patra. A survey on perfectly-secure verifiable secret-sharing. *IACR Cryptol. ePrint Arch.*, page 445, 2021.
- [CDG⁺17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *CCS*, pages 1825–1842, 2017.
- [CFY17] Robert K. Cunningham, Benjamin Fuller, and Sophia Yakoubov. Catching MPC cheaters: Identification and openability. In *ICITS*, pages 110–134, 2017.
- [CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *ASIACRYPT*, pages 466–485, 2014.
- [Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, 1986.
- [DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.
- [DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, pages 445–465, 2010.
- [DM00] Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In *CRYPTO, Proceedings*, pages 74–92. Springer, 2000.
- [GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *CRYPTO*, pages 178–193. Springer, 2002.
- [GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. In *USENIX*, pages 1069–1083, 2016.
- [GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304. ACM, 1985.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
- [GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO*, pages 323–341. Springer, 2007.
- [GSV21] Yaron Gvili, Sarah Scheffler, and Mayank Varia. BooLigero: Improved sublinear zero knowledge proofs for boolean circuits. In *FC*, pages 476–496, 2021.
- [IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256. Springer, 2002.
- [IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC, Proceedings*, pages 21–30, 2007.
- [IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
- [IOZ14] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *CRYPTO*, pages 369–386. Springer, 2014.
- [IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC*, pages 121–145, 2014.

- [KKK09] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. *Inf. Comput.*, 207(8):889–899, 2009.
- [KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *CCS*, pages 525–537, 2018.
- [KLR06] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In *STOC*, pages 109–118, 2006.
- [MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *CRYPTO, Proceedings*, pages 392–404. Springer, 1991.
- [SF16] Gabriele Spini and Serge Fehr. Cheater detection in SPDZ multiparty computation. In *ICITS*, pages 151–176, 2016.
- [SSY22] Mark Simkin, Luisa Siniscalchi, and Sophia Yakoubov. On sufficient oracles for secure computation with identifiable abort. In *SCN*, pages 494–515. Springer, 2022.
- [YW22] Kang Yang and Xiao Wang. Non-interactive zero-knowledge proofs to multiple verifiers. In *ASIACRYPT, Proceedings, Part III*, pages 517–546, 2022.