Electronic Colloquium on Computational Complexity, Report No. 23 (2025)

How to Share an NP Statement or Combiners for Zero-Knowledge Proofs

Benny Applebaum* Eliran Kachlon*

February 24, 2025

Abstract

In Crypto'19, Goyal, Jain, and Sahai (GJS) introduced the elegant notion of *secret-sharing of an* **NP** *statement* (NPSS). Roughly speaking, a t-out-of-n secret sharing of an **NP** statement is a reduction that maps an instance-witness pair to n instance-witness pairs such that any subset of $(t-1)$ reveals no information about the original witness, while any subset of t allows full recovery of the original witness. Although the notion was formulated for general $t \leq n$, the only existing construction (due to GJS) applies solely to the case where $t = n$ and provides only computational privacy. In this paper, we further explore NPSS and present the following contributions.

Definition. We introduce a refined definition of information-theoretically secure NPSS. This notion can be seen as a cryptographic variant of standard **NP**-reductions and can be compiled into the GJS definition using any one-way function.

Construction. We construct information-theoretic t-out-of-n NPSS for any values of $t \leq n$ with complexity polynomial in n. Along the way, we present a new notion of secure multiparty computation that may be of independent interest.

Applications. Our NPSS framework enables the *non-interactive combination* of n instances of zero-knowledge proofs, where only t_s of them are sound and only t_z are zero-knowledge, provided that $t_s + t_z > n$. Our combiner preserves various desirable properties, such as the succinctness of the proof. Building on this, we establish the following results under the minimal assumption of one-way functions: 1. **Standard NIZK implies NIZK in the Multi-String Model** (Groth and Ostrovsky, J. Cryptology, 2014), where security holds as long as a majority of the n common reference strings were honestly generated. Previously, such a transformation was only known in the *common random string* model, where the reference string is uniformly distributed. 2. A **Designated-Prover NIZK in the Multi-String Model**, achieving a strong form of two-round Multi-Verifier Zero-Knowledge in the honest-majority setting. 3. A **three-round secure multiparty computation protocol** for general functions in the honest-majority setting. The round complexity of this protocol is optimal, resolving a line of research that previously relied on stronger assumptions (Aharonov et al., Eurocrypt'12; Gordon et al., Crypto'15; Ananth et al., Crypto'18; Badrinarayanan et al., Asiacrypt'20; Applebaum et al., TCC'22).

*Tel-Aviv University, Israel `bennyap@post.tau.ac.il`, `elirn.chalon@gmail.com`. Supported by ISF grant no. 2805/21 and by the European Union (ERC-2022-ADG) under grant agreement no.101097959 NFITSC.

ISSN 1433-8092

Contents

1 Introduction

Threshold secret sharing [Sha79, Bla79] allows to distribute a data item into n shares, requiring a minimum number of t participants to combine their shares to reconstruct the secret, while smaller groups cannot access it. In Crypto'19, Goyal, Jain, and Sahai [GJS19] introduced an elegant extension of this central concept to the case where one wants to share an **NP** *statement*. Roughly speaking, a *t-out-of-n secret sharing of an* **NP** *statement* (NPSS) is a reduction that maps an instance/witness pair into n instance/witness pairs such that any set of $(t-1)$ of them reveals no information on the original witness and any t out of them allow to fully recover the original witness.

From a complexity theory perspective, NPSS can be viewed as a natural cryptographic analog of standard **NP**-reduction. From a cryptographic point of view, NPSS allows us to non-interactively *combine* multiple zero-knowledge protocols in such a way that, even if some instances are "flawed," the final outcome remains secure. As we will later see, this kind of "combiner" proves to be highly effective, offering a clean and straightforward solution to several existing problems in the literature.

Goyal et al. [GJS19] considered *computationally-private* NPSS in which privacy holds only against polynomial-time adversaries and correctness/recovery may be violated with negligible probability. They presented an n-out-of-n NPSS and a "gap-construction" with a privacy threshold of $t_p < n/3$ and recovery threshold of $t_c > 2n/3$ based on non-interactive commitments. However, until now, it was unknown whether this notion could be realized beyond these concrete values, and whether any meaningful notion of information-theoretic NPSS could be achieved at all.

In this paper, we introduce new refined definitions of information-theoretic secure NPSS, present the first efficient constructions of NPSS for any values of $t \leq n$, and describe new applications in the domain of zero-knowledge proofs and secure multiparty computation. We continue with a detailed account of our results.[1]

1.1 How to Define NPSS?

While the notion of NPSS is intuitively clear, the formal definition requires some care. Unlike the original definition of [GJS19] which is inherently limited to computational security, we define an information-theoretic notion of NPSS and show that our variant can be compiled into the [GJS19] variant based on any one-way function. (See Section 1.4 for more details.) Jumping ahead, working with the "right" definition turns to be crucial both for deriving efficient NPSS constructions and for employing it in different applications.

Syntactically, our NPSS definition is composed of four efficient algorithms: a deterministic instance mapper R, a randomized assignment mapper W, a randomized simulator Sim and a deterministic decoder Dec. Given a circuit-SAT instance f, the instance mapper deterministically

[1] NPSS were originally introduced as a tool for *amplifying* the security of non-interactive zero-knowledge proofs (NIZK) where the goal is to simultaneously reduce the soundness and zero-knowledge errors. Unfortunately, this use of NPSS for NIZK amplification [GJS19] turns to be flawed (as noted by the authors, see [Jai24, BG24]). Our paper shows that NPSS is sufficiently strong for the task of *combining* many NIZK instances out of which some are faulty. Roughly speaking, the main difference between the two scenarios is that the "bad event" in which the security of a weak-NIZK fails may depend on the input (i.e., the statement and its witness). In a companion paper [AK25], we introduce and construct a stronger variant of *Leakage-Resilient NPSS*, and show how to use it as an amplifier for NIZK. These results employ a different set of techniques that are not addressed here.

maps f into n circuit-SAT instances $F = (f_1, ..., f_n)$ over a *common set of variables* such that the followings hold:

- *Correctness and* $(t-1)$*-privacy:* Given a satisfying assignment x for f, the randomized assignment mapper W samples n partial assignments $(y_1, \dots, y_n)$ that *consistently satisfy* $(f_1, ..., f_n)$ in the sense that y_i satisfies f_i and there exists some global assignment y that agrees with all the partial assignments. Furthermore, the marginal distribution of every collection of $t-1$ partial assignments $(y_{i_1}, \dots, y_{i_{t-1}})$ leaks no information about the witness x, i.e., the joint distribution can be efficiently sampled based on f via the simulator $\mathsf{Sim}(f, \{i_1, \dots, i_{t-1}\})$.

- *t-recovery:* Given any collection of t partial assignments $y_{i_1}, \dots, y_{i_t}$ that consistently satisfy $f_{i_1}, \dots, f_{i_t}$, the decoder Dec efficiently recovers an assignment x that satisfies f. This implies a strong *soundness* property: if f is unsatisfiable, it is impossible to consistently satisfy more than t of the instances in F.

To get a better understanding of this notion, it is useful to consider two unsuccessful naive implementations and see why they fail (assuming that $\mathbf{P} \neq \mathbf{NP}$). Let us first take f_i and y_i to be a copy of f and x, respectively. This construction satisfies the correctness and recovery requirements, but fails to satisfy the privacy condition for every $t > 1$. (Indeed, if privacy holds then whenever f is satisfiable one can efficiently recover a satisfying assignment by using the simulator.) Alternatively, one can try to secret-share the witness x into $(y_1, \dots, y_n)$ via t-out-of-n secret sharing (e.g., Shamir's scheme [Sha79]). In this case the privacy requirement holds, but it is infeasible to define a sequence of instance $(f_i)_{i\in[n]}$ that satisfy both the correctness and the recovery property. Intuitively, each predicate f_i has only local view of y_i and so it is hard to correlate their global behavior. Formally, if f is satisifiable then every predicate f_i must accept the "zero share" $\mathbf{0}$ (otherwise, correctness is violated with some probability), but if f is unsatisfiable then, by recovery/soundness, some f_i must locally reject $\mathbf{0}$. Therefore, given an efficient mapping from f to F we can efficiently decide the satisfiability of f by checking if the all-zero string satisfies all the predicates f_i, putting $\mathbf{NP}$ in $\mathbf{P}$.

Generalizing the above discussion, privacy and recovery induce two conflicting requirements. On one hand, the partial assignments should not overlap by too much (some entropy on the global assignment y should be left even after seeing $t-1$ of them), and, on the other hand, every pair of partial assignments must share some common variables (otherwise local satisfiability implies global satisfiability as in the second example). Overall, the $\mathbf{NP}$ instances and their witnesses should be correlated in a non-trivial way that globally reflects the satisfiability of f but locally hides it.

Remark 1.1 (The use of partial assignments)**.** *Each circuit f_i depends only on a subset S_i of the set of common variables y, and, accordingly, the partial assignment y_i will keep the variables outside S_i unassigned. Interestingly, in our constructions, the assignment mapper also keeps some of the S_i variables unassigned in y_i, and choose these variables "dynamically" based on the randomness of the witness mapping procedure W. (We conjecture that this is inherent to every construction of threshold NPSS.) This means that the circuit f_i should be well defined over partial assignments $y_i \in \{0,1,*\}^k$. For this, we extend Boolean circuit logic in the following natural way: if any gate input is $*$, the output defaults to $*$, except in simple cases like $\wedge(0,*) = \wedge(*,0) = 0$ and $\vee(1,*) = \vee(*,1) = 1$. As usual, y_i satsifies f_i if $f_i(y_i) = 1$.*

Remark 1.2 (An alternative perspective)**.** *It is instructive to think about the NPSS formulation as a* multi-verifier zero-knowledge oracle-based proof system *where a prover wishes to prove to n verifiers that f is satisfiable. The prover publishes a proof y on a public board and hides each bit of y under a card. To convince the verifiers that f is satisfiable, the prover reveals the bits of y_i to the ith verifier, who announces whether y_i satisfies f_i. In the latter case, the verifier certifies his claim by revealing the bits of the false witness to the other verifiers. The verifiers reject the proof if some verifier revealed an un-satisfied instance, and accept it otherwise. The privacy property of the NPSS guarantees* perfect zero-knowledge *against every set of $t-1$ verifiers, whereas the recovery property guarantees* perfect soundness *against any coalition that contains a cheating prover together with $n-t-1$ cheating verifiers. In the special case where $n = 2t + 1$, we get both perfect soundness and perfect zero-knowledge at the presence of honest majority among the verifiers.*

This somewhat resembles the notion of zero-knowledge probabilistic-checkable-proofs (ZK-PCP) [KPT97, DFK+92]. Indeed, these two primitives can be described under the same framework with the difference that NPSS optimizes one set of parameters and standard constructions of ZK-PCP's optimize a different set of parameters. We further mention that, despite the similarity in names, NPSS is completely unrelated to the notion of *Secret-Sharing for* **NP** *access structures* [KNY14]. (See Section 1.4 for a detailed comparison between NPSS and these notions.)

1.2 Main Results

Our main result is an unconditional construction of NPSS for arbitrary threshold.

Informal Theorem 1.1 (NPSS exist)**.** *There is a t-out-of-n NPSS for arbitrary choices of t and n. In particular, there are universal NPSS algorithms that take t and n as auxiliary inputs, and in time $\mathrm{poly}(n, |f|)$ generate instances f_i of size $\mathrm{poly}(n) \cdot |f|$, where $|f|$ is the size of the given circuit-SAT instance f.*

The theorem yields NPSS with perfect correctness and perfect recovery (zero error) as well as perfect privacy (the simulated distribution is identical to the real distribution). See Section 3.1 for formal details. We will elaborate on the proof of this theorem in Section 1.3, but for now, let us turn to a discussion of its applications. (Some of the readers may choose to switch the order and read first the technical overview of the NPSS construction in Section 1.3.)

1.2.1 Multi-String Non-Interactive Zero-knowledge Proofs

A Zero-Knowledge proof system [GMR89] is considered *non-interactive* if the prover can generate a proof without any interaction with the verifier [BFM88]. Specifically, given a public **NP** statement and a witness, the prover generates a proof π and sends it to the verifier, who then decides whether to accept it. The absence of interaction makes this approach extremely powerful, with numerous important applications, ranging from CCA security [NY90, DDN91] and signatures [BMW03, BKM06], to cryptocurrencies [SCG+14].

To make this concept feasible, both the prover and the verifier have access to a public key, typically referred to as a *common reference string* (CRS), which is assumed to be generated by a *trusted party*. This trust assumption is crucial: if the CRS is generated maliciously, the soundness of the system is compromised, allowing for fake proofs of false statements to be generated using the simulator. In some NIZKs, this may also lead to attacks on the zero-knowledge property. These vulnerabilities may even arise if the CRS is sampled honestly but the private coins used during the sampling procedure are kept as a trapdoor.

The standard solution to this problem is to distribute the trust among multiple "authorities" by sampling the CRS through the use of a secure multiparty computation (MPC) protocol. If a majority of the authorities are honest, we can ensure a "good CRS," for which the NIZK remains secure. However, this solution is highly interactive and can be quite costly in terms of communication and computation, particularly when the CRS-sampling procedure is complex.

In [GO14], Groth and Ostrovsky proposed an elegant solution to this problem that allows the trust to be "split" among several authorities in a non-interactive way. Formally, they introduce the Multi-String (MS) model, where instead of relying on a single CRS, we have a sequence of CRSs $(\mathsf{crs}_1, \dots, \mathsf{crs}_n)$, each sampled locally by a different authority and published once and for all. The prover and verifier access these strings during proof generation and verification, and security holds as long as a majority of the CRSs are generated honestly. This means that even if a minority of the CRSs were generated maliciously, in collaboration with the prover or the verifier, the soundness and zero-knowledge guarantees still hold.[2] Conveniently, the authorities do not need to communicate with one another or even be aware of each other's existence.

Groth and Ostrovsky [GO14] constructed MS-NIZK either based on specific algebraic intractability assumptions related to groups with a bilinear map or from any NIZK in the *common random string* model where the CRS is sampled *uniformly*. However, constructing MS-NIZK from general NIZK systems where the CRS is structured remains an open problem. This question is motivated by both theoretical and practical concerns since several existing constructions of NIZKs [PS19, CCH$^+$19, BKM20, LPWW20], including some of the most practical ones [PHGR16, BCG$^+$13, DFGK14], rely on highly-structured CRSs for which the MPC-based solution tends to be expensive.[3] We resolve this problem via the aid of NPSS.

Informal Theorem 1.2 (MS-NIZK from NIZK)**.** *Assuming the existence of one-way functions, any NIZK for* $\mathbf{NP}$ *can be efficiently converted into an MS-NIZK for* $\mathbf{NP}$*.*

It is not hard to show that MS-NIZK implies NIZK. Therefore, assuming one-way functions, the theorem shows that the two notions are essentially equivalent.[4]

Proof sketch. We sketch the proof and leave the full details to Section 4. Given the MS-CRS $\mathsf{crs} = (\mathsf{crs}_1, \dots, \mathsf{crs}_n)$, a circuit-SAT instance f and a witness x, the prover uses t-out-of-n NPSS with $t = \lceil (n+1)/2 \rceil$ to generate the statements $f_1, \dots, f_n$ and to sample the partial assignments $y_1, \dots, y_n$ that are all consistent with some assignment $y \in \{0,1\}^m$. The prover uses a statistically-binding bit-commitment scheme to generate a commitment $(Y[i])_{i\in[m]}$ to each bit of the assignment $(y[i])_{i\in[m]}$, and generates a sequence of proofs $\pi = (\pi_1, \dots, \pi_n)$ where π_i is generated by running the NIZK's prover with crs_i for the statement

[2]More generally, [GO14] consider a refined version of the definition in which the threshold (number of honestly generated CRS) that is needed for zero-knowledge may be different from the soundness threshold as long the sum of the threshold is strictly larger than n. Our results extend to this setting as well (See Section 4).

[3]Assuming the existence of one-way functions, NIZK in the uniform CRS model are equivalent to so-called *Zaps* [DN07] whereas structured-CRS NIZKs can be currently based on a wider family of assumptions while achieving additional features [PS19, CCH$^+$19, BKM20, LPWW20].

[4]In fact, the one-way assumption can be replaced with the assumption that $\mathbf{NP}$ is non-trivial in the sense that it cannot be emulated by polynomial-size circuits for infinitely often input lengths. Since the latter assumption, together with NIZK for $\mathbf{NP}$, implies the existence of one-way functions [HN24].

> (f_i, Y): "there exists a partial assignment that satisfies f_i which is consistent with the committed string Y"

whose witness is y_i and the openings of the corresponding commitments. The proof consists of π and Y. (The prover verifies that each of the proofs pass verification and if this is not the case, she replaces the flawed proof π_i with the witness y_i). The verification is performed in the natural way. Given $(f, \mathsf{crs}, \pi, Y)$, use the NPSS to compute $f_1, \ldots, f_n$ and, for each i, verify that the statement (f_i, Y) is accepted by the NIZK's verifier with respect to the string crs_i.

Completeness is immediate. For soundness, note that if the proof is accepted then it is also accepted with respect to $n - (t-1) \geq t$ honestly-generated CRS's and so, by the soundness of the NIZK and the binding property of the commitment, there exist at least t partial assignments that consistently satisfy t of the f_i's, which by the recovery property of the NPSS implies that f is satisfiable. Intuitively, zero-knowledge relies on the privacy of the NPSS: even if $t-1$ of the CRS's are corrupted and the corresponding witnesses are leaked, the original witness x remains hidden. This intuition can be translated to an efficient simulator whose security relies on the hiding property of the commitments. Finally, we realize the commitments based on one-way functions by adopting Naor's commitments [Nao89] to MS-CRS setting. $\square$

Apart of the use of commitments, the above transformation is information-theoretic, and so even advanced properties of NIZK are essentially preserved assuming that an appropriate type of commitments is being used. For example, we can lift a standard non-interactive statistical zero-knowledge argument-of-knowledge to the multi-string setting by instantiating the transformation with statistically-hiding commitments that can be based on collision-resistance hash functions [DPP98a, HM96a]. Under the same assumption, we also get a *communication-efficient* reduction: the new proof consists of n proofs of circuit-SAT instances whose size is larger than f by a $\mathrm{poly}(n, \kappa)$ factor where κ is the security parameter. In particular, if the original proof system is *somewhat succinct* (resp., *succinct*), i.e., the length of a proof for S-size circuit is S^ϵ for some $\epsilon < 1$ (resp., $\mathrm{poly}(\log(S), \kappa)$), then so is the new system (assuming that $n = \mathrm{poly}(\kappa)$ and that f is sufficiently large compared to κ). (See Section 4 for more detains and for additional extensions.)

In contrast, the transformation proposed in [GO14] (for a uniformly chosen CRS) inherently results in computational zero-knowledge, even if the original NIZK possesses statistical zero-knowledge properties. It also fails to maintain any level of succinctness, even when the number of CRS instances is constant. Indeed, this construction applies the NIZK-to-Zaps transform of [DN07] that, in order to maintain the same soundness error, has to blow-up the proof size by a factor that is at least linear in the instance size.

1.2.2 NIZK combiners

A "downgraded" version of Theorem 1.2 allows us to prove the following basic result about NIZK combiners.

Informal Theorem 1.3 (non-interactive combiners)**.** *Assuming the existence of one-way functions, it is possible to take n NIZK candidates and in time $\mathrm{poly}(n, \kappa)$, generate a new NIZK whose security holds as long as a majority of the candidates are secure (sound and zero-knowledge) regardless of the (in)security properties of the other candidates. Moreover, assuming collision resistance hash functions, the transformation preserves succinctness.*

The theorem generalizes to the case where t_s of the candidates are sound and t_z of the candidates are zero-knowledge as long as $t_s + t_z > n$, i.e., at least one candidate is both sound and zero-knowledge. (See Theorem 4.9.) To the best of our knowledge, this is the first NIZK combiner that preserves succinctness, and has complexity that is $\mathrm{poly}(n, \kappa)$. Alternative constructions of NIZK combiners are implicit in works on NIZK amplifications [GJS19, BG24], however, the complexity of these combiners is super-polynomial (in fact, sub-exponential) in n.[5], Moreover, the latter construction does not preserve succinctness even for constant n.

1.2.3 Two-Round Distributed Zero-Knowledge and Three-Round MPC

We move on to the case of zero-knowledge proofs in *distributed settings* where a single prover interacts with multiple verifiers – a model that was extensively studied in the past decade with several useful applications (see [AKP22b] for references). In the simplest variant, known as *multi-verifier zero-knowledge* proofs (MVZK) [BD91], the prover tries to prove a public $\mathbf{NP}$ statement to n verifiers such that zero-knowledge (resp., soundness) should hold even if the adversary corrupts up to t of the verifiers (resp., the prover and up to $t-1$ of the verifiers).

Since no trusted setup is assumed, non-trivial languages (outside $\mathbf{BPP}$) do not admit single-round MVZK protocols even for $t = 1$, nor do they admit two-round MVZK protocols with a dishonest majority [GO94] (see the discussion in [AKP22b]). Still, in the presence of honest majority one can hope for two-round protocols, or even for an offline/online MVZK in which the first *offline* round is independent of the statement that is chosen adaptively at the beginning of the second *online* round. Indeed, as noted by [ACGJ18], MS-NIZK give rise to such online/offline honest-majority MVZK for $\mathbf{NP}$. By Theorem 1.2, we can base MS-NIZK on NIZK. However, the mere existence of NIZK currently relies on public-key assumptions (e.g., [BFM88, FLS90, GOS12, SW14, PS19, CCH$^+$19]) or on private-key primitives with non-standard properties [KRR17, CCRR18, HL18, BKM20] such as correlation intractability [CGH04].

The question of basing two-round honest-majority MVZK on one-way functions, the minimal cryptographic assumption, was studied by [AKP22b]. They partially resolved the question with two main caveats: They had to rely on *injective* one-way function with *sub-exponential hardness* and the complexity of their protocol grows exponentially with the number of parties. By using NPSS, we overcome these caveats. Along the way, we derive stronger efficiency properties, and significantly simplify the construction.

Informal Theorem 1.4. *Assuming one-way functions, there exists online/offline honest-majority MVZK.*

Furthermore, the protocols follow a simple Verifiers/Prover communication pattern: In the first round, the verifiers broadcast public messages and send private information to the prover, while the prover remains silent. In the second round, the verifiers remain silent, and the prover publishes a proof. Verification is based solely on the prover's message and the public messages from the verifiers.

Unlike the constructions in [AKP22b] the verifiers do not have to communicate with each other, and in fact, they can become "inactive" after the first round. Viewing the offline broadcast message of each verifier as a CRS, we note that the communication pattern of our NVZK is very similar to the one obtained by MS-NIZK. The main difference is that the public CRS is accompanied by

[5]Both constructions use an iterative approach that alternates between zero-knowledge amplification and soundness amplification for a constant number of levels. It can be shown that this approach implicitly realizes majority by a constant-depth circuit with AND/OR gates with unbounded-fan-in, which is known to require sub-exponential size (see, e.g., [FSS84, OW07]).

a secret message ("private key") that is sent from each verifier to the prover. In the setting of a single authority, such a protocol is also known as *designated-prover NIZK* (dp-NIZK) for adaptively chosen single theorem [SMP87, BGT20]. Accordingly, Theorem 1.4 provides a dp-NIZK in the MS setting. To prove the theorem, we first construct a (single-theorem) dp-NIZK in the CRS model based on one-way functions, and then use NPSS to lift it to the MS model. We further show that one-way functions suffice for dp-NIZKs with either perfect-soundness and computational-ZK or for perfect-ZK and computational soundness. Previous constructions of dp-NIZK relied on stronger assumptions such as statistically-hiding commitments or sub-exponential hard one-way functions [BGT20, AKP22b], and failed to achieve perfect zero-knowledge. (See Remark 5.5.)

The construction of MVZKs enables solutions to several additional problems where prior approaches relied on stronger assumptions or encountered the caveats mentioned earlier. Most notably, building on [AKP22b, AKP22a], we construct an optimal, 3-round secure multiparty computation (MPC) protocol for general functions. This protocol achieves full security — including guaranteed output delivery — even in the presence of an actively corrupted minority. Our construction relies on one-way functions, concluding a line of research that previously depended on stronger assumptions for such protocols [AJL$^+$12, GLS15, BJMS20, ACGJ18, AKP22a].

Additionally, if the parties have access to a collision-resistant hash function, we can achieve *statistical everlasting security* for any $\mathbf{NC}^1$ functionality. This means that the protocol remains secure against adversaries who are computationally bounded during execution but become computationally unbounded afterward. (See discussion in [AKP22b].) This extension builds on a statistical zero-knowledge version of Theorem 1.4. (See Section 5 for details.)

1.3 Technical Overview of the Main Theorem

AND-NPSS. We start with the simplest case of 2-out-of-2 NPSS, hereafter referred to as AND-NPSS. Roughly, our goal is to map f into two statements f_1 and f_2 over a joint set of variables such that each single statement f_i is independently satisfiable, but the two are simultaneously satisfiable if and only if f is satisfiable. More accurately, f should be mapped to a pair of distributions (Y_1, Y_2) over partial assignments such that any assignment x that satisfies f can be randomly mapped to a pair of consistent assignments (y_1, y_2) such that y_i satisfies f_i and its marginal distribution is identical to Y_i.

We rely on the MPC-in-the-head paradigm of Ishai, Kushilevitz, Ostrovsky and Sahai [IKOS09] with a minor, yet important, tweak. Let $\pi = \pi_f$ be a 2-party protocol that given an input x_1 for P_1 and x_2 for P_2 returns the value $f(x_1 \oplus x_2)$ to both parties. Given an assignment x that satisfies f, secret-share x into x_1 and x_2 via additive secret sharing, and sample a *full transcript* y for $\pi(x_1, x_2)$. That is, sample random tapes r_1 and r_2 for P_1 and P_2, generate all the messages m_i that P_i sends to the other party, and define

$$y := (x_1, r_1, x_2, r_2, m_1, m_2).$$

The function f_i checks that the partial view of P_i that is defined by y leads to an output of 1. That is, $f_1(x_1, r_1, m_1, m_2)$ runs the protocol π on (x_1, r_2) sets the incoming messages according to m_2 and computes the outgoing messages and the final output, if the outgoing messages are consistent with m_1 and the final output is 1, the function accepts its input. The predicate f_2 is defined similarly. Crucially, the function depends only on inputs that are revealed to P_i during π, and so if the protocol is *1-private* (i.e., the view of every single party that plays honestly can be perfectly simulated) then the NPSS is also private. The 2-recovery and perfect correctness properties of the NPSS hold assuming that the protocol π is *perfectly correct* in the sense that for every input (x_1, x_2) and every pair of random tapes (r_1, r_2) the outcome $\pi((x_1, r_1), (x_2, r_2))$ equals to $f(x_1 \oplus x_2)$.

Coping with OT-messages. It is known that non-trivial functions cannot be computed with perfect correctness and 1-privacy in the "plain-model" [CK89]. However, the celebrated GMW theorem [GMW87] provides such a protocol for every function in the *oblivious-transfer* (OT) hybrid model. In this model, P_1 and P_2 share an OT channel that operates as follows: the sender sends a pair of bits (z_0, z_1) the receiver chooses a bit s and receives the message z_s. The sender learns no information on the selection bit, and the receiver learns nothing on the message z_{1-s}. For simplicity, assume that the protocol π involves a single OT-channel call, with P_1 as the sender and P_2 as the receiver. The NPSS witness y is extended with the OT-triple (z_0, z_1, s). The verification f_1 of the sender's view consists of checking that (z_0, z_1) are indeed the OT-messages that are being sent given input (x_1, r_1) and the incoming messages m_2. Accordingly, f_1 depends on P_1's view that includes (z_0, z_1) in addition to the input, randomness and standard messages.

The situation for the receiver P_2 is more subtle. In order to check that the partial view of P_2 is consistent, the function f_2 needs the value of z_s as an input, where s is the selection bit that is computed by f_2 based on (x_2, r_2, m_1). Since s is determined by the inputs to f_2, it implies that f_2 syntactically depend on both z_0 and z_1. However, we cannot simulate a satisfying assignment for f_2 that assigns values to both variables. (Indeed, if the message z_{1-s} is leaked in π privacy completely fails, since otherwise we get a protocol in the plain model!) The key insight is to use a *partial assignment*. This allows us to assign only the necessary inputs for f_2 while keeping other inputs unassigned in order to preserve privacy. Specifically, we generate a view for P_2 that includes (x_2, r_2, m_1, s, z_s), leaving z_{1-s} as unassigned (represented by $*$). The function $f_2(x_2, r_2, m_1, s, z_1, z_2)$ takes this partial assignment and checks consistency while ignoring the value of z_{1-s}. Using the extended logic defined in Remark 1.1, we can define a circuit $f_2(x_2, r_2, m_1, s, z_1, z_2)$ that computes the desired functionality (tests if a given view is valid and leads to 1) even when z_{1-s} is set to $*$. In particular, one can retrieve the value z_s via the subformula $(s \wedge z_1) \vee (\neg s \wedge z_0)$. This relaxation of Boolean logic allows us to perfectly realize the AND-NPSS gadget.

Comparison to [GJS19]. It is instructive to compare the above construction to the NPSS solution of [GJS19]. Unlike our approach, [GJS19] uses cryptographic commitments to encapsulate the extended transcript y based on the original MPC-in-the-head template. In their construction, the commitment string Y is hard-wired to the statements f_1 and f_2, making it part of their specification. The inputs of f_1, f_2 are the "openings" of the commitments to the relevant parts of the view (e.g., f_2 gets opening for $x_2, r_2, m_1, m_2, s, z_s$). Since the commited assignment is fixed as part of the function's description, this approach circumvents partial assignments: The function f_i depends on k inputs, with the simulator generating a complete assignment for these inputs.[6] On the downside, the construction is only computationally-secure. Moreover, the number of inputs in f_i is polynomially larger than the number of variables in f. In contrast, since the GMW compiler has constant computational overhead, our approach increases the number of variables only by a

[6]We note that in all of our applications, the partial assignments in our NPSS will be eventually wrapped by commitments as well. However, as we will see, it is beneficial to "postpone" the use of commitments as much as possible.

constant factor — a property that will later prove crucial. (Less importantly, the use of commitments requires from [GJS19] to define the NPSS syntax in a slightly more liberal way as discussed in Section 1.4.)

Generalized NPSS. Our next goal is to construct t-out-of-n NPSS. In fact, it will be convenient to generalize the notion of NPSS and define its properties with respect to a monotone predicate $\chi : 2^{[n]} \to \{0,1\}$ for which $\chi(A) \leq \chi(B)$ whenever $A \subseteq B$. A χ-NPSS is defined similarly to t-out-of-n NPSS, except that for every subset $S \subseteq [n]$ of statements, privacy holds if $\chi(S) = 0$, meaning that S is *unauthorized*, and recovery holds if $\chi(S) = 1$ meaning that S is *authorized*. The notion of t-out-of-n NPSS corresponds to the case where χ is taken to be the t-out-of-n threshold function.

Following the traditional secret-sharing literature (see [BL88, Bei11]), it is natural to construct χ-NPSS recursively by scanning a monotone formula for χ in a gate-by-gate manner starting from the output gate. For instance, in order to realize n-out-of-n NPSS use an AND-tree, share f in the root into f_1, f_2 and continue recursively with each statement separately. Unfortunately, this approach fails to satisfy the recovery/soundness condition due to the use of partial assignments. For example, suppose that f is AND-shared into (g, h) and then g is AND-shared into (g_1, g_2) and h is AND-shared into (h_1, h_2). One would like to argue that (g_1, g_2, h_1, h_2) form a 4AND-sahring of f. However, the first two instances (g_1, g_2) share no common variables with the (h_1, h_2) and so no consistency is enforced between the g_i's and the h_i's. As a result, even if f is not satisfiable, it is possible to generate partial assignments (y_1, y_2, y_3, y_4) that consistently satisfy the tuple (g_1, g_2, h_1, h_2). (Use the simulator to sample a pair of inconsistent assignments (a, b) that separately satisfy (g, h), and apply the assignment mapper on a and b seperately to generate (y_1, y_2) and (y_3, y_4).) This problem can be avoided if we use the commitment-based construction of [GJS19]. However, the commitment-based approach suffers from a polynomial blow-up in the number of variables, and therefore it can only be used for constant-depth formulas that are too weak to compute non-trivial threshold functions where t is a constant fraction of n. (See, e.g., [FSS84, OW07]).

Back to MPC. The key idea is to apply recursion at the "MPC-level" and then move to NPSS via the MPC-in-the-head transform. Since MPC is an interactive object, the meaning of recursion is somewhat subtle. Moreover, to make this work, we will have to carefully define a non-standard notion of NPSS-compatible MPC. Details follow.

Generalizing the previous AND-NPSS construction, we introduce the following notion of χ-*secure* protocols for computing a public-output function f. In such a protocol, the following properties hold for every subset $S \subset [n]$:

1. **(Perfect passive privacy for zero-sets)**: If $\chi(S) = 0$, the joint view of coalition S can be perfectly simulated.

2. **(Perfect correctness with abort against an active adversary)**: If $\chi(S) = 1$, the output of parties in S is "correct" even if an unbounded adversary actively corrupts all other parties. Specifically, the adversary may cause the honest parties to abort, but if the honest parties agree on an output z, there exists some input $x_{\bar{S}}$ for the corrupted parties that, together with the honest parties' input x_S, produces z under f. This correctness must hold with probability 1, meaning that even with full knowledge of the honest parties' information, the adversary cannot force an incorrect output $z \notin \text{Image}(f(x_S, \cdot)) \cup \{\perp\}$.

Unlike standard notions of MPC where privacy and correctness should hold against the same collection of adversarial coalitions, here each of these properties is defined with respect to a different collection of coalitions. Just like in the case of secret-sharing, this notion has an all-or-nothing flavor: Every coalition either (passively) learn noting about the output, or has enough information to avoid an an incorrect output even in the presence of an active adversary that corrupts all other parties. The latter correctness property is tailored to the NPSS setting; it ensures that if the views of an authorized set S are self-consistent and yield an output of 1, then f must be satisfiable. Notably, the GMW protocol satisfies these properties for the AND-predicate. In this case, the correctness property simplifies to standard perfect correctness, as the only authorized set is the set containing all parties. Another simple example is the OR-predicate, for which the privacy condition is trivial, since any non-empty subset $S \subset [2]$ satisfies the OR predicate. OR-MPC can be realized via a "replication protocol," where each party is given the entire input x and outputs $f(x)$.

The combination of these two protocols turns out to be sufficient for realizing an arbitrary monotone predicate χ. To see this, express χ as a monotone circuit using AND and OR gates, and construct the protocol recursively, layer by layer, through *party virtualization* (see, e.g., [HM00, CDI+13]). For instance, let χ_d be an AND-tree with depth d, and assume we have a χ_d-secure protocol π_d for f. We can construct a χ_{d+1}-secure protocol π_{d+1} by replacing each party P in π_d with a pair of parties (P_1, P_2) that securely emulate P in π_d using a 2-party AND-secure protocol. Since π_d employs an OT channel, P_1 and P_2 must also emulate P's OT communication, but this is manageable. Since our basic AND/OR protocols incur only constant computational overhead, we achieve efficient χ-secure protocols as long as χ is computable by a monotone circuit of logarithmic depth. Since threshold functions admit such explicit circuits (e.g., based on optimal sorting networks [AKS83]), we get efficient t-out-of-n NPSS for arbitrary parameters t, n.

1.4 NPSS and Other Related Models

Our NPSS vs. the NPSS definition of [GJS19]. Recall that in our NPSS definition, the instance f is mapped into $(f_i)_{i\in[n]}$ via a (deterministic) instance mapper R, and the assignment x is mapped to to the partial assignments $(y_i)_{i\in[n]}$ by the randomized assignment mapper W. In [GJS19] these two algorithm are combined into a single randomized sharing algorithm M that given an instance/witness pair (f, x) and some public parameters (generated by a setup algorithm), outputs n instance/witness pairs $(f_i, y_i)_{i\in[n]}$. This algorithm is accompanied by a verification algorithm V and by a recovery algorithm, where the former verifies that the instances $(f_i)_{i\in[n]}$ are "valid" with respect to f (and the public parameters), and the latter provides t-recovery, conditioned on successful verification. The simulator should be able to sample all the instance $(f_i)_{i\in[n]}$ jointly with any subset of $t-1$ assignments. Note that the above syntax can be derived from our syntax via simple manipulations (by letting $M = (R, W)$ and by letting V accept $(f_i)_{i\in[n]}$ if they are consistent with $R(f)$). The important difference is that [GJS19] define the instances $(f_i, y_i)_{i\in[n]}$ on *disjoint* sets of variables and so, unlike our setting, every y_i is a fully defined assignment and no consistency property is required between the assignment. Instead, consistency is enforced by the verification algorithm based on the *instances* $(f_i)_{i\in[n]}$.

It is not hard to show that this syntax cannot be realized with information-theoretic security, unless the polynomial-hierarchy collapses. Indeed, such an information-theoretic scheme allows to reduce a circuit-SAT statement f to the **SZK**-complete *statistical-distance* problem [SV97] as follows. Given an **NP** statement f, consider the circuit $C_{f,S}(r)$ that, for a $(t-1)$-subset $S \subset [n]$

invokes the NPSS-simulator $\mathsf{Sim}(f, S; r)$ with r as its random tape, and given the answer $(F = (f_i)_{i\in[n]}, (y_i)_{i\in S})$, it outputs the F-part if y_i satisfies f_i for every $i \in S$, and outputs S otherwise. Fix some two distinct $(t-1)$-subsets $S_0 \neq S_1$, and let D_b denote the distribution sampled by the circuit $C_{f,S_b}(r)$ on a randomly chosen input r. If f is satisfiable then the distributions are statistically close (assuming that the NPSS is statistically-private), and if f is unsatisfiable then the distributions must be disjoint. Indeed, if F is in the support of both distributions then $(f_i)_{i\in S_0\cup S_1}$ are all satisfiable, and (by the t-recovery property) the function f must be satisfiable as well.

Moving on with the comparison between the two notions, we note that any NPSS that is aligned with our definition can be compiled into an NPSS that satisfies the above notion via the use of non-interactive commitments. In fact, this can be based on CRS-based commitments whose existence follows from any one-way function [Nao91]. The new sharing algorithm uses $R(f)$ and $W(x)$ to generate the instances $(f_i)_{i\in[n]}$ and partial assignments $(y_1, \ldots, y_n)$, that are all consistent with some global assignment y. The algorithm then commits to every bit y (denote the commitments by C), and outputs $(g_1, ..., g_n)$ and $(z_1, ..., z_n)$ defined in the following way: g_i takes a subset of openings to C, opens the corresponding commitments (all the commitments are hard-wired to g_i), and verifies that the partial assignment defined by the opened commitments satisfies f_i. The assignment z_i is the subset of openings corresponding to the partial assignment y_i. The verification algorithm simply verifies that $g_1, ..., g_n$ are well defined with respect to the same set of commitments C. It is not hard to see that if the verification succeeds, then the underlying partial assignments are consistent, unless the adversary violated the binding property of the commitment scheme.

Overall, our definition encapsulates the information-theoretic core of the computational NPSS of [GJS19]. This can be viewed as part of a general endeavor in separating information-theoretic components from cryptographic compilers (see [Ish20]). Indeed, as we saw, the use of information-theoretic abstraction is beneficial both for the construction and for the applications (e.g., by allowing modular usage of different types of commitments).

NPSS vs. ZK-PCP. Roughly speaking, in ZK-PCP a prover who claims that x satisfies f, randomly maps x to a longer proof $y \in \{0,1\}^m$ such that a verifier can be convinced that the witness x is valid by reading a subset $I \subset [m]$ of the bits of y and applying predicate to $y[I]$. The set I is sampled according to some predefined distribution. The honest-ZK property asserts that if I is sampled properly the bits $y[I]$ can be efficiently simulated, possibly with some statistical error, even without holding the witness x.

Every t-out-of-n NPSS implies such an honest-verifier ZK-PCP where the prover samples y as the proof, and the verifier samples $t-1$ tests $(f_1, \ldots, f_{t-1})$ out of F and asks to see the corresponding partial assignments.[7] This transformation achieves perfect zero-knowledge against an honest verifier, perfect completeness, and a soundness error of $1/\binom{n}{t}$ which is optimal given the zero-knowledge property. In fact, any ZK-PCP with the above properties is an NPSS. On the downside, the resulting ZK-PCP is non-local (each test may query polynomially many bits of the proof) and zero-knowledge may completely break down if the verifier is malicious. In contrast, in the context of ZK-PCP [KPT97, DFK+92] one typically tries to minimize the query complexity of the honest

[7]Strictly speaking, ue to the use of partial assignments, we have to let the prover decide which locations I_i of y to reveal in order to convince that f_i is satisfied. Alternatively, we could let the verifier read the locations adaptively since one can define an order $(j_1, \ldots, j_k)$ over the entries of I_i such that each index j_ℓ can be computed based on the content of $y[j_1], \ldots, y[j_{\ell-1}]$.

verifier (e.g., to a constant) and try to obtain zero-knowledge against a malicious verifier that is allowed to make polynomially-many queries. The exact errors in soundness and zero-knowledge are typically not important. Hence, NPSS and ZK-PCP attempt to optimize different parameters and we currently do not know how to move from one setting to the other.

NPSS vs. secret-sharing for $\mathbf{NP}$ access structure. The notion of NPSS may superficially resemble the problem of *Secret-Sharing for* $\mathbf{NP}$ *access structures* [KNY14], though these two notions are actually very far from each other. In short, given a monotone function f in $\mathbf{NP}$, the latter notion allows to share a secret s among n parties such that a coalition can recover the secret if and only if it is accepted by f. That is, the secret can be arbitrary and the role of the monotone $\mathbf{NP}$ function f is to determine which parties are authorized to recover s. In contrast, in our case the "secret" must be a satisfying assignment for the (possibly non-monotone) $\mathbf{NP}$ function and the set of "authorized" shares is determined by a simple threshold function. Moreover, we present information-theoretic constructions for NPSS that are unlikely to exist in the case of secret-sharing for $\mathbf{NP}$-access structure unless the polynomial-hierarchy collapse [Nao06] .

2 Secure Multiparty Computation

In this section we construct MPC protocols that will be used as building blocks in the construction of NPSS in Section 3. In Section 2.1 we formally define the notion of a protocol, and also provide security definitions. In Section 2.2 we present our server-substitution generators, that are the main building blocks of our construction, and in Section 2.3 we present our MPC protocols. Throughout this section we measure computational complexity in the unit-cost RAM model.

2.1 Basic Definitions

In the following section, we define MPC in the client-server model and present our non-standard security definitions. Since we will later manipulate protocols (via different forms of compositions), we will have to carefully define the syntax of a "protocol". This can be done in multiple ways, and our concrete formalization follows the framework of Hirt and Maurer [HM00].

2.1.1 Clients, Servers and Protocols

Client and servers. We consider two types of parties: clients and servers. Every party, either a client or a server, can communicate with all other parties and perform computations. The only difference between the clients and the servers is that only clients are allowed to receive inputs and provide outputs, while the servers have no inputs and outputs.

Functionality. An n-client (deterministic) functionality $\mathcal{F}$ is a function that takes n inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and provides n outputs $\mathbf{y}_1, \ldots, \mathbf{y}_n$. We think of $\mathbf{x}_i$ as the input of the i-th client, and of $\mathbf{y}_i$ as the output of the i-th client.

Variable space, view and abort flag. We assume that the parties communicate over private point-to-point channels and use a broadcast channel to announce an "abort". For concreteness, we formalize the syntax of a protocol via the framework of [HM00, CDI$^+$13]. We assume a global *variable space* $\mathcal{X}$. A variable $x \in X$ can take values from the binary field $\mathbb{F}_2$. Every value generated during a protocol execution, including inputs, local data and outputs, is assigned to a variable. We assume that every variable is assigned a value exactly once, and can't be re-written. In addition, every variable belongs only to a single party, and we let the *view* of a party p, denoted $\mathsf{view}(p)$, be the values of all the variables that belong to p. For a variable $x \in \mathcal{X}$, we denote the value assigned to x by $\mathsf{val}(x)$.

We also assume the existence of a public *abort flag*, denoted flag, that all the parties can access. The value of the flag is set to zero at the beginning of the execution, and at any time each party can raise the flag (i.e., set its value to 1). Once the value of the abort flag is set to 1, it cannot be changed. Jumping ahead, at the end of the protocol the clients output "abort" if and only if the flag is turned on.[8]

Protocols. A protocol π among a set $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ of clients $\mathcal{C}$ and servers $\mathcal{S}$ over the variable space $\mathcal{X}$ is a sequence $d_1, \ldots, d_L$ of *statements*. There are several types of statements:

- An *input statement* $\mathsf{input}(c, i, x)$ instructs the client $c \in \mathcal{C}$ to read the i-th bit from its input tape and to assign the value to the variable $x \in \mathcal{X}$ that belongs to c.
- An *output statement* $\mathsf{output}(c, i, x)$ instructs the client $c \in \mathcal{C}$ to set its i-th output bit to be $\mathsf{val}(x)$ where x belongs to c. However, if the public abort flag is raised, i.e., $\mathsf{flag} = 1$, then the output is set to be $\perp$.
- A *transmit statement* $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ instructs a party $p_1 \in \mathcal{P}$ to send the value of the variable x_1 that belongs to p_1 to a party $p_2 \in \mathcal{P}$, and it instructs p_2 to assign the received value to the variable x_2.
- A *computation statement* $\mathsf{comp}(p, \mathsf{op}, X, x)$ instructs a party $p \in \mathcal{P}$ to perform the atomic operation op on the tuple of variables $X \subseteq \mathcal{X}$ that are in its view and to assign the result to the variable $x \in \mathcal{X}$. For concreteness, we assume that $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$ where $\{+, \times\}$ are addition and multiplication over $\mathbb{F}_2$ (in which case X contains exactly two variables), and rand is an instruction to assign a random value to x (in which case $X = \emptyset$).
- An *abort statement* $\mathsf{abort}(p, x)$ instructs a party $p \in \mathcal{P}$ to raise the abort flag flag if the value of the variable x in the view of p is 1.

Let $\mathcal{G}$ be a 2-party functionality, that takes an input $\alpha \in \{0,1\}^{\ell_1}$ from the first party, an input $\beta \in \{0,1\}^{\ell_2}$ from the second party, computes $\gamma = \mathcal{G}(\alpha, \beta)$, where $\gamma \in \{0,1\}^{\ell_3}$, and returns γ only to the second party (the first party has no output). A protocol π in the $\mathcal{G}$-hybrid model gives every pair of parties access to the functionality $\mathcal{G}$, and allows the following statement:

- A *$\mathcal{G}$-statement* $\mathsf{func}(\mathcal{G}, p_1, p_2, X^1, X^2, X^3)$ for parties $p_1, p_2 \in \mathcal{P}$, a tuple $X^1 = (x^1_1, \ldots, x^1_{\ell_1})$ of variables that belong to p_1, a tuple $X^2 = (x^2_1, \ldots, x^2_{\ell_2})$ of variables that belong to p_2, and a tuple $X^3 = (x^3_1, \ldots, x^3_{\ell_3})$ of variables that belong to p_2, instructs the parties to use the functionality $\mathcal{G}$ in the following way. Party p_1 inputs $\alpha \in \{0,1\}^{\ell_1}$ defined by the values of the variables in X^1, and p_2 inputs $\beta \in \{0,1\}^{\ell_2}$ defined by the values of the variables in X^2. The output from the functionality $\gamma = \mathcal{G}(\alpha, \beta) \in \{0,1\}^{\ell_3}$ is then given by the functionality to p_2 that assigns γ to the variables in X^3.

[8]One could emulate such a flag using a broadcast channel. However, for technical reasons it will be convenient to use the flag mechanism and avoid the use of a broadcast channel.

Throughout, we only consider protocols π that are well defined, i.e., variables are only accessed after they were defined, and the protocol does not attempt to re-write a variable. In addition we assume that all the input statement appear first as a prefix of the protocol, and that every input bit is read at most once; we assume that all the output statements appear at the end as a suffix of π, and that every output bit is written at most once; and we assume that those are the only input/output statements in the protocol. Finally, we assume that for every choice of inputs to the parties, in an honest execution of π the abort flag is never raised.

2.1.2 Security Definition

From now on, let $\mathcal{F}$ be an n-client functionality, let π be a protocol over the set of parties $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ that includes n clients $\mathcal{C}$ and m servers $\mathcal{S}$.

Our security definition is parameterized by a monotone predicate $\chi : 2^{\mathcal{P}} \to \{0, 1\}$ referred to as an *access structure*. Here monotonicity means that $\chi(A) \leq \chi(B)$ for every $A, B \subseteq \mathcal{P}$ that satisfy $A \subseteq B$. Our security definition consists of two parts. The first part requires standard perfect privacy and perfect correctness against a passive adversary that corrupts subsets Z for which $\chi(Z) = 0$. (Intuitively, as in the case of secret sharing these coalitions correspond to "unauthorized coalitions".)

Definition 2.1 (χ-passive security). *We say that π computes $\mathcal{F}$ with* χ-passive security *if there exists a simulator* Sim *such that for all inputs $(x_1, \dots, x_n)$ to the clients the following holds:*

- (Perfect Correctness) *With probability 1, the output of the i-th clients in an execution of π is y_i, where $(y_1, \dots, y_n) = \mathcal{F}(x_1, \dots, x_n)$.*
- (Perfect Privacy) *For any set $Z \subseteq \mathcal{P}$ of corrupt parties that satisfies $\chi(Z) = 0$ it holds that*

$$(\mathsf{view}(p))_{p \in Z} \equiv \mathsf{Sim}(Z, (x_i)_{c_i \in Z \cap \mathcal{C}}, (y_i)_{c_i \in Z \cap \mathcal{C}})$$

  where $(y_1, \dots, y_n) = \mathcal{F}(x_1, \dots, x_n)$. In addition, the running time of Sim *is polynomial in n and the number of statements in π.*

The second part of the definition guarantees *perfect correctness* for authorized coalitions S for which $\chi(S) = 1$, even in the presence of an active adversary corrupts all other parties $Z := \mathcal{P} \setminus S$. That is, the honest parties in S will *never* generate an erroneous output but the adversary may force them to output an "abort" symbol. This means that in every execution of π in the presence of the active adversary, where the clients hold inputs $x = (x_1, \dots, x_n)$, either the abort flag is raised, or the outputs of the honest clients are consistent with $\mathcal{F}(x'_1, \dots, x'_n)$ for some assignment $x' = (x'_1, \dots, x'_n)$ that is consistent with x with respect to the honest clients (i.e., $x'_i = x_i$ if the i-th client is honest). In fact, we even require the existence of an efficient extractor so that whenever the honest clients do not abort, the assignment $(x'_1, \dots, x'_n)$ can be extracted from the views of the honest parties. Before formalizing this non-standard notion, let us formally define the notion of active adversary in our concrete syntactic framework.

Active adversarial behavior. As usual, an active adversary that corrupts a set Z of parties is an adversary that can make the corrupted parties in Z deviate from the protocol. Since our non-standard notion of correctness should hold with probability 1, it is enough to require that it holds with respect to *degenerate active adversaries*, that specify *before* the execution of π what messages the corrupt parties send to the honest parties. Indeed, an active adversary that violates correctness with positive probability can be transformed into a degenerate active adversary that violates correctness with positive probability. Since correctness is satisfied when the abort flag is raised, we may also assume that the adversary never raises the abort flag. Finally, we mention that we allow the active adversary more power: for every $\mathcal{G}$-statement where the first party is corrupt and the second party that receives the output is honest, we let the adversary *choose* the output of the functionality $\mathcal{G}$.

Formally, let $\pi = (d_1, \ldots, d_L)$ be a protocol over $\mathcal{P}$. An *active adversarial behavior* with respect to a set Z of corrupt parties, is a protocol $\tilde{\pi} = (\tilde{d}_1, \ldots, \tilde{d}_L)$ such that for every $i \in [L]$, (1) if the i-th statement is a transmit statement $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ where $p_1 \in Z$ and p_2 is honest, then the statement is marked by a message m, so that in an execution of $\tilde{\pi}$ party p_1 ignores the original statement and instead sends m to p_2, (2) if the i-th statement is a $\mathcal{G}$-statement $\mathsf{func}(\mathcal{G}, p_1, p_2, X^1, X^2, X^3)$ where p_1 is corrupt and p_2 is honest, then the statement is marked by an output γ, so that in an execution of $\tilde{\pi}$ the output of $\mathcal{G}$ is always γ, (3) if the i-th statement is $\mathsf{abort}(p, x)$ for $p \in Z$, then the statement is ignored, and (4) otherwise, $\tilde{d}_i = d_i$ is executed honestly.

Definition 2.2 (χ-perfect active correctness with abort)**.** *We say that π computes $\mathcal{F}$ with* χ-perfect active correctness with abort*, if there exists a deterministic extractor* Ext *such that for every set of $S \subseteq \mathcal{P}$ of honest parties that satisfies $\chi(S) = 1$, all active adversarial behaviors $\tilde{\pi}$ with respect to the corrupt parties $Z := \mathcal{P} \setminus S$, all inputs $(x_1, \ldots, x_n)$ to the clients, and every execution of $\tilde{\pi}$, one of the following holds:*

- *The output of all honest clients is $\perp$.*
- *There exists a vector of inputs $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ such that (1) $x_i^* = x_i$ for all honest clients $c_i \in \mathcal{C} \cap S$, (2) the outputs of the honest clients are consistent with $\mathcal{F}(\mathbf{x}^*)$, and (3)* Ext*, given the set Z and the view of all honest parties, outputs $\mathbf{x}^*$.*

In addition, the running time of Ext *is polynomial in n and the number of statements in π.*

Our default notion of security, denoted by χ-dual security, requires privacy for authorized sets and correctness for unauthorized sets.

Definition 2.3 (χ-dual security)**.** *We say that π computes $\mathcal{F}$ with* χ-dual security*, if π provides both χ-passive security and χ-perfect active correctness with abort.*

2.1.3 Protocol Assignment Mapping

Let $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ be a set of parties. A *protocol assignment mapping* σ of $\mathcal{P}$ is function $\sigma : \mathcal{S} \to \mathcal{C}$. For a protocol π among $\mathcal{P}$, we define the mapped protocol $\sigma(\pi)$ to be the same protocol, where in each statement all involved servers are replaced with the mapped clients. We note that $\sigma(\pi)$ is a protocol among the set of clients $\mathcal{C}$, and without any server.

For a subset of clients $Z \subseteq \mathcal{C}$ we define the inverse mapping

$$\sigma^{-1}(Z) = \{s \in \mathcal{S} : \sigma(s) \in Z\}.$$

For a monotone predicate $\chi : 2^{\mathcal{P}} \to \{0,1\}$, we define the mapped predicate $\chi_\sigma : 2^{\mathcal{C}} \to \{0,1\}$ to be

$$\chi_\sigma(Z) := \chi(Z \cup \sigma^{-1}(Z)),$$

for every $Z \subseteq \mathcal{C}$. The following claim, proved in Appendix A.1, will be useful later on.

Claim 2.4 (Replacing servers with clients)**.** *Let σ be a protocol assignment mapping, let $\mathcal{F}$ be an n-client functionality, and let π be a protocol among $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ that computes $\mathcal{F}$ with χ-dual security. Then $\pi' = \sigma(\pi)$ is a protocol among $\mathcal{P}' = \mathcal{C}$ that computes $\mathcal{F}$ with χ_σ-dual security. In addition, the simulator Sim' and extractor Ext' of π' can be computed efficiently given σ (say, represented as an array), the simulator Sim and the extractor Ext for π. The number of statements in π' is $|\pi|$, and the running time of Sim' and Ext' is $O(n \cdot t_{\mathsf{Sim}})$ and $O(n \cdot t_{\mathsf{Ext}})$, respectively, where t_{Sim} is the running time of Sim, and t_{Ext} is the running time of Ext.*

2.2 Server-Substitution Generators

A *$\mathcal{P}'$-server substitution generator G* takes as an input a protocol π among a set $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ of clients and servers, together with the identity of some server $\tau \in \mathcal{S}$, and returns a new protocol π' among a set $(\mathcal{P} \setminus \{\tau\}) \cup \mathcal{P}'$ of clients and servers. We will always assume that $\mathcal{P}'$ is a set of servers that is *disjoint* of $\mathcal{P}$ and if this is not the case, we simply rename the servers in $\mathcal{P}'$.

In the following sections we present two server-substitution generators in the (bit) OLE-hybrid model, that will be used as a basic building block in the construction of our MPC protocols. Recall that the OLE functionality takes from the first party two bits (a, b) and from the second party a bit x and returns the bit $(a \cdot x) \oplus b$ to the second party.[9]

2.2.1 The $\vee$-Generator

Let π be a protocol among $\mathcal{P}$ that computes a functionality $\mathcal{F}$ with χ-dual security. The $\vee$-generator $G_\vee$ takes the protocol π and a server $\tau \in \mathcal{P}$, and generates a new protocol π' where τ is replaced with two new servers: A and B. At a high level, both A and B take the role of τ and hold a copy of its view: Every message sent to τ is now sent both to A and B, and every message sent from τ is sent by both A and B. Throughout, we make consistency checks for incoming and outgoing messages of A and B, and abort if some inconsistency is found.

For passive security, we note that if the view of either A or B is leaked, then it is equivalent to the leakage of the view of τ in π. This means that the new protocol is passively-secure against a set $Z \subseteq \mathcal{P}'$ if either (1) $A, B \notin Z$, and the original protocol is secure against Z, or (2) if ($A \in Z$ or $B \in Z$), and π is passively-secure against $(Z \setminus \{A, B\}) \cup \{\tau\}$.

As for perfect active correctness with abort, we note that as long as one of A or B is honest, then this party guarantees an honest emulation of τ. This means that we obtain perfect correctness with abort against $Z \subseteq \mathcal{P}'$ if either (1) $A, B \in Z$ and π has perfect active security with abort against $(Z \setminus \{A, B\}) \cup \{\tau\}$, or (2) either A or B is not in Z, and the original protocol is secure against $Z \setminus \{A, B\}$.

We present $G_\vee$ in Figure 1.

Generator $G_\vee$

- **Inputs:** A protocol π over the set of parties $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$, and a server $\tau \in \mathcal{S}$.

[9] The use of the OLE-hybrid model instead of the OT-hybrid model is just for convenience, since the two models are equivalent (an OT call can be emulated by an OLE call and vice versa.)

- **Procedure:** The generator generates a protocol π' over the set of parties $(\mathcal{P} \setminus \{\tau\}) \cup \{A, B\}$, where A and B are servers that do not belong to $\mathcal{P}$. The protocol π' follows the statements of the protocol π, where the servers A and B each hold a copy of the local view of τ. That is, we keep the invariant that for every variable x that belongs to τ, A and B hold corresponding variables x^A and x^B, respectively, with the same value.

  Formally, for every $i = 1, \ldots, |\pi| + 1$, the i-th statement d_i in π is replaced with the following set of statements in π':

  – If the statement d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$, then in π' the party p sends $v := \mathsf{val}(x_1)$, both to A and to B. Denote by v^A the value that A received and by v^B the value that B received. We let A assign v^A to x_2^A, and B assigns v^B to x_2^B.

    In addition, A sends v^A to B and B verifies that $v^A = v^B$. If the verification fails, B raises the abort flag flag.

  – If the statement d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ then let x_1^A and x_1^B be the corresponding variables that A and B hold, with values v^A and v^B, respectively. A and B send v^A and v^B, respectively, to p, and p assigns v^A to x_2.

    In addition, p verifies that $v^A = v^B$. If the verification fails then p raises the abort flag flag.

  – If the statement is $\mathsf{comp}(\tau, *, (x_1, x_2), x)$ and $* \in \{+, \times\}$ then A assigns to the variable x^A the value $(v_1^A * v_2^A)$ and B assigns to the variable x^B the value $(v_1^B * v_2^B)$, where $v_1^A, v_2^A, v_1^B, v_2^B$ are the values of the variables $x_1^A, x_2^A, x_1^B, x_2^B$, respectively.

  – If the statement is $\mathsf{comp}(\tau, \mathsf{rand}, \emptyset, x)$ then A samples a random bit $r \leftarrow \{0, 1\}$, and assigns r to x^A. In addition, A sends r to B, and B assigns r to x^B.

  – If the statement d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$ then we execute two instances of OLE. In the first instance A inputs v_1^A, v_2^A and p inputs v_3, where v_1^A, v_2^A and v_3 are the values of x_1^A, x_2^A and x_3, respectively. In the second instance B inputs v_1^B, v_2^B and p inputs v_3, where v_1^B, v_2^B and v_3 are the values of x_1^B, x_2^B and x_3, respectively. Let y^A (resp., y^B) be the value of the output that p received from the first (resp., second) instance of OLE, and let p assign y^A to x_4.

    In addition, p verifies that $y^A = y^B$. If the verification fails then p raises the abort flag flag.

  – If the statement d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ then we execute two instances of OLE. In the first instance p inputs v_1, v_2 and A inputs v_3^A, where v_1, v_2 and v_3^A are the values of x_1, x_2 and x_3^A, respectively. In the second instance p inputs v_1, v_2 and B inputs v_3^B, where v_1, v_2 and v_3^B are the values of x_1, x_2 and x_3^B, respectively. Let y^A (resp., y^B) be the value of the output that A (resp., B) received from the first (resp., second) instance of OLE, and let A assign y^A to x_4^A, and B assign y^B to x_4^B.

    In addition, A sends y^A to B and B verifies that $y^A = y^B$. If the verification fails, B raises the abort flag flag.

  – If the statement d_i is $\mathsf{abort}(\tau, x)$ then A executes $\mathsf{abort}(A, x^A)$ and B executes $\mathsf{abort}(B, x^B)$.

  – For any other statement d_i, the statement remains the same in π.

- **Outputs:** The generator outputs the protocol π' over the set of parties $(P \setminus \{\tau\}) \cup \{A, B\}$.

Figure 1: Generator $G_\vee$

Complexity and parallel applications. We note that a single application of $G_\vee$ incurs a constant overhead in the number of instructions, as each instruction in π is compiled into a constant number c of instructions in π'. In fact, even if we repeatedly apply $G_\vee$ on any subset of original servers $\{s_1, \ldots, s_k\} \subseteq \mathcal{S}$, i.e., we compute $\pi_i := G_\vee(\pi_{i-1}, s_i)$ for all $i = 1, \ldots, k$ where $\pi_0 = \pi$, the overhead per instruction is still constant. Indeed, every original instruction refers to at most two servers, say s_i and s_j for $i < j$, so it will be compiled into a constant number of c instructions in π_i, and then each of these new instruction will be compiled into a constant number of c instructions in π_j. In addition, these instructions will not be affected by any compilation π_ℓ for $\ell \notin \{i, j\}$, and therefore the total overhead per instruction is at most $c^2 = O(1)$. We think of such an application of $G_\vee$ as *parallel application* as it does not involve the substitution of servers that are generated throughout these applications.

Lemma 2.5. *Let π be a protocol among $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ that computes a functionality $\mathcal{F}$ with χ-dual security, and let $\tau \in \mathcal{S}$ be a server. Then $\pi' := G_\vee(\pi, \tau)$ is a protocol among the set of parties $\mathcal{P}' = (\mathcal{P} \setminus \{\tau\}) \cup \{A, B\}$ that computes π with $\chi_\vee$-dual security, where*

$$\chi_\vee(S) = \begin{cases} \chi(S), & \textit{if } A, B \notin S \\ \chi((S \setminus \{A, B\}) \cup \{\tau\}), & \textit{otherwise,} \end{cases}$$

for every set $S \subseteq \mathcal{P}'$.

Moreover, there is a constant C and an efficient algorithm $\mathsf{comp}_\vee$ *that takes as an input (1) a protocol π over $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ together with its simulator* Sim *and extractor* Ext*, and (2) a list of servers $(s_1, \ldots, s_k)$ from $\mathcal{S}$; The algorithm outputs a protocol π' together with its simulator* Sim' *and extractor* Ext' *such that the following holds: (1) $\pi' = \pi_k$, where $\pi_0 = \pi$ and $\pi_i = G_\vee(\pi_{i-1}, s_i)$ for all $i = 1, \ldots, k$, (2) if the number of instructions in π is L then the number of instructions in π' is at most $C \cdot L$, and (3) if the running time of* Sim *and* Ext *is bounded by T, then the running time of* Sim' *and* Ext' *is bounded by $C \cdot T$.*

We prove Lemma 2.5 in Appendix B.

2.2.2 The $\wedge$-Generator

Let π be a protocol among $\mathcal{P}$ that computes a functionality $\mathcal{F}$ with χ-dual security. The $\wedge$-generator $G_\wedge$ takes the protocol π and a server $\tau \in \mathcal{P}$, and generates a new protocol π' where τ is replaced with two new servers: A and B. At a high level, the view of τ is secret shared among A and B, and every local computation of τ is performed by a secure computation of A and B in the OLE-hybrid model, in a similar way to the GMW protocol [GMW87]. Every message sent to τ is secret shared among A and B, and every message m sent from τ is emulated by letting A and B send their shares of m.

For passive security, we note that if the view of at most one of $\{A, B\}$ is leaked then the view of τ remains secure. However, if both views are leaked, then the view of τ can be recovered from the secret sharing. This means that the new protocol is passively-secure against a set $Z \subseteq \mathcal{P}'$ if either (1) $A, B \in Z$, and π is passively-secure against $(Z \setminus \{A, B\}) \cup \{\tau\}$, or (2) if $A \notin Z$ or $B \notin Z$, and the original protocol is secure against $Z \setminus \{A, B\}$.

As for perfect active correctness with abort, we note that as long both A and B are honest, then π is emulated honestly. This means that we obtain perfect correctness with abort against $Z \subseteq \mathcal{P}'$ if either (1) $A \in Z$ or $B \in Z$ and π is perfect active security with abort against $(Z \setminus \{A, B\}) \cup \{\tau\}$, or (2) both A and B are not in Z, and the original protocol is secure against Z.

We present $G_\wedge$ in Figure 2.

Generator $G_\wedge$

- **Inputs:** A protocol π over the set of parties $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$, and a server $\tau \in \mathcal{S}$.
- **Procedure:** The generator generates a protocol π' over the set of parties $(\mathcal{P} \setminus \{\tau\}) \cup \{A, B\}$, where A and B are servers that do not belong to $\mathcal{P}$. The protocol π' follows the statements of the protocol π, where the view of τ is secret shared among the servers A and B. That is, we keep the invariant that for every variable x that belongs to τ, the new servers A and B hold corresponding variables x^A and x^B whose values form a 2-out-of-2 secret sharing of the value of x.

  Formally, for every $i = 1, \dots, |\pi| + 1$, the i-th statement d_i in π is replaced with the following set of statements in π':

  - If the statement d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$, then in π' the party p takes the value $v \in \{0, 1\}$ of x_1, and samples a 2-out-of-2 secret sharing of v, denoted (v^A, v^B). Then p sends v^A to A and v^B to B. We let A assign v^A to x_2^A, and B assigns v^B to x_2^B.
  - If the statement d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ then let x_1^A and x_1^B be the corresponding shares that A and B hold, with values v^A and v^B, respectively. A samples a random bit $r \leftarrow \{0, 1\}$ and sends r to B. A and B send $y^A := v^A \oplus r$ and $y^B := v^B \oplus r$, respectively, to p, and p assigns the value $y^A \oplus y^B$ to the variable x_2.[a]
  - If the statement is $\mathsf{comp}(\tau, +, (x_1, x_2), x)$ then A assigns to the variable x^A the value $v_1^A \oplus v_2^A$ and B assigns to the variable x^B the value $v_1^B \oplus v_2^B$, where $v_1^A, v_2^A, v_1^B, v_2^B$ are the values of the variables $x_1^A, x_2^A, x_1^B, x_2^B$.
  - If the statement is $\mathsf{comp}(\tau, \times, (x_1, x_2), x)$ then the parties execute two instances of OLE in the following way. Let $v_1^A, v_2^A, v_1^B, v_2^B$ be the values of the variables $x_1^A, x_2^A, x_1^B, x_2^B$. We let A sample two random bits $v_3^A \leftarrow \{0, 1\}$ and $r \leftarrow \{0, 1\}$. In the first OLE execution A inputs $(v_1^A, v_1^A \cdot v_2^A \oplus v_3^A \oplus r)$, B inputs v_2^B and receives the output y_1. In the second OLE execution A inputs (v_2^A, r), B inputs v_1^B and receives the output y_2. Finally, A assigns v_3^A to x_3^A, and B assigns $y_1 \oplus y_2 \oplus v_1^B \cdot v_2^B$ to x_3^B.
  - If the statement is $\mathsf{comp}(\tau, \mathsf{rand}, \emptyset, x)$ then A samples a random bit $v^A \leftarrow \{0, 1\}$, and assigns v^A to x^A. Similarly, B samples a random bit $v^B \leftarrow \{0, 1\}$, and assigns v^B to x^B.
  - If the statement d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$ then we let A sample a random bit $r \leftarrow \{0, 1\}$ and send r to B. We execute two instances of OLE. In the first instance A inputs $(v_1^A, v_2^A \oplus r)$ and p inputs v_3, where v_1^A, v_2^A and v_3 are the values of x_1^A, x_2^A and x_3, respectively. In the second instance B inputs $(v_1^B, v_2^B \oplus r)$ and p inputs v_3, where v_1^B, v_2^B and v_3 are the values of x_1^B, x_2^B and x_3, respectively. Let y^A (resp., y^B) be the value of the output that p received from the first (resp., second) instance of OLE, and let p assign $y^A \oplus y^B$ to x_4.
  - If the statement d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ then we let p sample a random bit $r \leftarrow \{0, 1\}$, and we execute two instances of OLE. In the first instance p inputs $(v_1, v_2 \oplus r)$ and A inputs v_3^A, where v_1, v_2 and v_3^A are the values of x_1, x_2 and x_3^A, respectively. In the second instance p inputs (v_1, r) and B inputs v_3^B, where v_1, v_2 and v_3^B are the values of x_1, x_2 and x_3^B, respectively. Let y^A (resp., y^B) be the value of the output that A (resp., B) received from the first (resp., second) instance of OLE, and let A assign y^A to x_4^A, and B assign y^B to x_4^B.
  - If the statement d_i is $\mathsf{abort}(\tau, x)$ then let v^A (resp., v^B) be the value of the variable x^A (resp., x^B). We let A send v^A to B, and then B recovers $v = v^A \oplus v^B$ and raises the abort flag if $v = 1$.
  - For any other statement d_i, the statement remains the same in π.
- **Outputs:** The generator outputs the protocol π' over the set of parties $(P \setminus \{\tau\}) \cup \{A, B\}$.

[a] In the compilation of this statement we let A and B re-randomize the shares v^A and v^B. The goal of this re-randomization is just to simplify the description of the simulator in the proof of Lemma 2.6.

Figure 2: Generator $G_\wedge$

The following lemma captures the security of $G_\wedge$. The "Moreover" part shows that when the transformation is applied on parallel to a set of servers, the overhead per instruction is constant, just like in the case of $G_\vee$.

Lemma 2.6. *Let π be a protocol among $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ that computes a functionality $\mathcal{F}$ with χ-dual security, and let $\tau \in \mathcal{S}$ be a server. Then $\pi' := G_\wedge(\pi, \tau)$ is a protocol among the set of parties $\mathcal{P}' = (\mathcal{P} \backslash \{\tau\}) \cup \{A, B\}$ that computes π with with $\chi_\wedge$-dual security, where*

$$\chi_\wedge(S) = \begin{cases} \chi((S \setminus \{A,B\}) \cup \{\tau\}), & \text{if } A, B \in S \\ \chi(S \setminus \{A,B\}), & \text{otherwise,} \end{cases}$$

for every set $S \subseteq \mathcal{P}'$.

Moreover, there is a constant C and an efficient algorithm $\mathsf{comp}_\wedge$ *that takes as an input (1) a protocol π over $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ together with its simulator* Sim *and extractor* Ext, *and (2) a list of servers $(s_1, \ldots, s_k)$ from $\mathcal{S}$; The algorithm outputs a protocol π' together with its simulator* Sim′ *and extractor* Ext′ *such that the following holds: (1) $\pi' = \pi_k$, where $\pi_0 = \pi$ and $\pi_i = G_\wedge(\pi_{i-1}, s_i)$ for all $i = 1, \ldots, k$, (2) if the number of instructions in π is L then the number of instructions in π' is at most $C \cdot L$, and (3) if the running time of* Sim *and* Ext *is bounded by T, then the running time of* Sim′ *and* Ext′ *is bounded by $C \cdot T$.*

We prove Lemma 2.6 in Appendix C.

2.3 From Formulas to MPC

The following theorem shows that if we iteratively apply server substitution generators according to a monotone formula χ, the resulting protocol will achieve χ-dual security. Analogous statements were proven for other notions of security (see, e.g., [CDI+13, Lemma 9.1]) and our proof follows a similar argument with adaptations to our notion of security.

Theorem 2.7. *There is a deterministic compiler C that takes as an input (1) a depth-d monotone formula χ on n inputs and a single output bit, that uses only AND and OR gates with fan-in 2, and uses no constants, and (2) an n-client functionality $\mathcal{F}$, represented as a Boolean circuit. The compiler outputs a protocol π over n clients $\mathcal{C}$ and no servers in the* OLE*-hybrid model, that computes the functionality $\mathcal{F}$ with χ-dual security. The compiler also outputs the description of the corresponding simulator* Sim *and the extractor* Ext. *The running time of C is $\mathrm{poly}(|\mathcal{F}| \cdot n \cdot 2^d)$, and the running time of* Sim *and* Ext *and the number of instructions in π is bounded by $|\mathcal{F}| \cdot \mathrm{poly}(n \cdot 2^d)$, where $|\mathcal{F}|$ is the circuit size of $\mathcal{F}$ measured by the number of wires.*[10]

We emphasize that while the running time of the compiler C is $\mathrm{poly}(|\mathcal{F}| \cdot n \cdot 2^d)$, the complexity of π, Sim and Ext depends only linearly on $\mathcal{F}$, i.e., it is $|\mathcal{F}| \cdot \mathrm{poly}(n \cdot 2^d)$. We continue with a proof of the theorem.

Proof. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of n clients, and assume that the formula χ has s gates. We begin with the description of the protocol π, and then explain how to efficiently generate π, Sim,

[10]By using standard balancing theorems for monotone formulas [Spi71, Weg83], it is possible to efficiently transform any S-size monotone formula into an equivalent formula of size $\mathrm{poly}(S)$ and depth $O(\log S)$. Therefore, the restriction to depth-d formulas can be replaced with size-S at the expense of replacing the quantity 2^d with $\mathrm{poly}(S)$. We further mention that the restriction to formulas that do not employ constants can be waived since any non-constant monotone formula over the standard AND/OR basis that employs constants can be easily transformed into an equivalent formula that does not employ constants.

and Ext. Given $\mathcal{F}$ we compute a sequence of protocols $\pi_0, \pi_1, \ldots, \pi_s, \pi_{s+1}$ and π is set to be π_{s+1}. Each protocol π_i involves the clients $\mathcal{C}$, may include additional servers $\mathcal{S}_i$, and is secure with respect to some specific monotone function χ_i. We note that the compiler outputs π_{s+1} that only involves the n clients $\mathcal{C}$ and computes $\mathcal{F}$ with χ-dual security.

Let $g_1, \ldots, g_s$ be some reverse topological ordering of the gates of χ, and let us assume that for every $i \in \{0, \ldots, d-1\}$ the gates at depth i appear before all the gates in depth $i+1$ (here, the depth is the distance from the output gate). We associate each wire of χ with a server. For a wire w, we abuse notation and use w to denote both the wire and the associated server. It will be clear from the context whether we think of w as a wire or as the associated processor.

Let χ'_i be the sub-formula defined by the gates $g_1, \ldots, g_i$, and let $\mathcal{W}_i$ be the set of input wires of χ'_i, where we define χ'_0 to be the sub-formula that includes only a single wire (the output wire of χ). We let $m_i = |\mathcal{W}_i|$ be the number of input wires and we think of χ'_i as a formula on m_i inputs in the natural way. For every $i \in \{0, \ldots, s\}$, the protocol π_i will involve the set of parties $\mathcal{P}_i := \mathcal{C} \cup \mathcal{W}_i$, i.e., the original set of clients $\mathcal{C}$ together with the servers in $\mathcal{W}_i$. We will prove that π_i is secure with respect to the predicate $\chi_i : 2^{\mathcal{P}_i} \to \{0,1\}$ that given a set of clients and servers, $S \subset \mathcal{P}_i$, outputs 1 if and only if the set of servers $S \cap \mathcal{W}_i$ (or, more precisely, its characteristic vector) satisfies the formula χ'_i. (Put differently, $\chi_i : 2^{\mathcal{P}_i} \to \{0,1\}$ extends $\chi'_i : 2^{\mathcal{W}_i} \to \{0,1\}$ in a trivial way.)

Building blocks. Our main building blocks are the server-substitution generators $G_\vee$ and $G_\wedge$ be the generators from Figure 1 and Figure 2, respectively.

The protocol π_0. The protocol π_0 over the set of n clients $\mathcal{C}$ and a single server w is the "ideal world" protocol. That is, (1) each client c_i reads its input x_i, (2) the clients send their inputs to w, (3) w computes $(y_1, \ldots, y_n) = \mathcal{F}(x_1, \ldots, x_n)$ and sends y_i to c_i, (4) c_i outputs y_i. We note that π_0 indeed computes $\mathcal{F}$ with χ_0-dual security, where $\chi_0(S) = \chi'_0(S \cap \{w\})$ is equal to 0 if and only if $w \notin S$. In addition, there is a trivial simulator Sim_0 and extractor Ext_0.

The protocol π_i. For $i = 1, \ldots, s$, let π_{i-1} be the protocol among $\mathcal{P}_{i-1} = \mathcal{C} \cup \mathcal{W}_{i-1}$ that computes $\mathcal{F}$ with χ_{i-1}-dual security, where $\chi_{i-1}(S) = \chi'_{i-1}(S \cap \mathcal{W}_{i-1})$. We construct π_i recursively, based on π_{i-1}. Let w be the output wire of g_i, and let w_A, w_B be the input wires of g_i. We define π_i in the following way: If g_i is an AND gate, then $\pi_i = G_\wedge(\pi_{i-1}, w)$; Otherwise, if g_i is an OR gate, then $\pi_i = G_\vee(\pi_{i-1}, w)$. Observe that π_i is a protocol among $\mathcal{P}_i = \mathcal{P}_{i-1} \setminus \{w\} \cup \{A, B\}$, and we rename A with w_A and B with w_B. The following claim is proved in Appendix A.2.

Claim 2.8. *It holds that π_i computes $\mathcal{F}$ with χ_i-dual security, where $\chi_i(S) = \chi'_i(S \cap \mathcal{W}_i)$.*

The protocol π_{s+1}. Observe that the protocol π_s is defined with respect to the set of clients $\mathcal{C}$ and the set of servers $\mathcal{W}_s$. We will assign servers to clients as follows. Let us associate the ith *input variable* of χ with the ith client in $\mathcal{C}$, and let $\sigma : \mathcal{W}_s \to \mathcal{C}$ be the mapping the takes the ith input wire to its corresponding input variable. (Different input wires can belong to the same formal variable.) Thinking of σ as a protocol assignment mapping, we let $\pi_{s+1} = \sigma(\pi_s)$. By Claim 2.4 we conclude that π_{s+1} computes $\mathcal{F}$ with $(\chi_s)_\sigma$-dual security. It therefore remains to prove that $(\chi_s)_\sigma = \chi$, which follows immediately from the fact that $\chi_s(S) = \chi'_s(S \cap \mathcal{W}_s)$ for all $S \subseteq \mathcal{P}_s$.

Running time. First, we observe that the complexity of π_0, Sim_0 and Ext_0 is $O(|\mathcal{F}| \cdot n)$. Recall that for every $i \in [d]$ the gates at depth i appear before all the gates in depth $i+1$, and let $i_0 = 1 < i_1 < \ldots < i_d = s$ be indices such that g_{i_j} is the last gate at depth j. That is, the gates at depth $0 < j \leq d$ are $g_{i_{j-1}+1}, \ldots, g_{i_j}$, and the gate at depth 0 is g_1. Assume without loss of generality that in every depth all AND gates appear before all OR gates.[11] Define $i_{-1} := 0$ and consider the sequence of protocols $\pi_{i_{-1}}, \pi_{i_0}, \ldots, \pi_{i_d}$, that is a subset of the original sequence $\pi_0, \ldots, \pi_s$. Let us denote by L_i the number of statements in π_i. We also denote by Sim_i and Ext_i the simulator and extractor, respectively, of π_i, and by T_i an upper bound on the running time of Sim_i and Ext_i.

We observe that for every $j \in \{1, \ldots, d\}$, π_{i_j} is obtained by parallel applications of $G_\vee$ and then $G_\wedge$ on $\pi_{i_{j-1}}$. Therefore, by Lemma 2.5 and Lemma 2.6 there is a constant C such that $L_{i_j} \leq C \cdot L_{i_{j-1}}$ and $T_{i_j} \leq C \cdot T_{i_{j-1}}$, and π_{i_j}, Sim_{i_j} and Ext_{i_j} can be efficiently computed given $\pi_{i_{j-1}}$, $\mathsf{Sim}_{i_{j-1}}$ and $\mathsf{Ext}_{i_{j-1}}$. We can therefore generate π_s, Sim_s and Ext_s in time $\mathrm{poly}(n \cdot 2^d \cdot |\mathcal{F}|)$, and the number of statements in π_s, as well as the running time of Sim_s and Ext_s, is bounded by $|\mathcal{F}| \cdot \mathrm{poly}(n \cdot 2^d)$. Finally, by Claim 2.4 we obtain that π, Sim and Ext can be efficiently computed from π_s, Sim_s and Ext_s, and their complexity is $|\mathcal{F}| \cdot \mathrm{poly}(n \cdot 2^d)$. This concludes the proof of the theorem. $\square$

3 Secret Sharing for NP Statements

In this section we present our NPSS construction. We begin with a formal definition of NPSS in Section 3.1, and present the construction in Section 3.2.

3.1 Basic Definitions

Before presenting the definition of NPSS, we need the following definitions for access structures and partial assignments

Definition 3.1 (Access structure). *An n-party* access structure *is a monotone subset $\mathscr{A} \subseteq 2^{[n]}$. A set $S \subseteq [n]$ is called* authorized *if $S \in \mathscr{A}$, and otherwise it is called* unauthorized.

In some cases we will abuse notation and identify an access structure $\mathscr{A} \subseteq 2^{[n]}$ with its characteristic (monotone) function $\chi_{\mathscr{A}} : \{0,1\}^n \to \{0,1\}$. Similarly, for a monotone function $\chi : 2^{[n]} \to \{0,1\}$, we define the access structure $\mathscr{A}_\chi$ in the natural way:

$$\mathscr{A}_\chi = \{S \subseteq [n] : \chi(1_S) = 1\},$$

where 1_S is the indicator vector of the set S.

Definition 3.2 (Partial assignments for Boolean circuits.). *Let C be a Boolean circuit on n variables. A partial assignment for C is a string $\mathbf{y} \in \{0,1,*\}^n$, where $*$ is a special symbol. The evaluation of C on $\mathbf{y}$, denoted $C(\mathbf{y})$, is a value in $\{0,1,*\}$ that is defined by evaluating the circuit from the input gates to the output gate in the natural way according to the following rule: If at least one of the input of a gate g is $*$ then the output is also $*$ except for the following cases: $0 \wedge * = * \wedge 0 = 0$ and $1 \vee * = * \vee 1 = 1$.*

Definition 3.3 (Secret sharing for an $\mathbf{NP}$ statement.). *Let $\mathscr{A}$ be an n-party access structure. An n-party $\mathscr{A}$-secret sharing for an $\mathbf{NP}$ statement is a tuple of algorithms $(R, W, \mathsf{Sim}, \mathsf{Dec})$ with the following syntax:*

[11] We mention that the final protocol π_{s+1} is independent of the order of the gates. However, this assumption will help us simplify the analysis.

- *The instance mapper R is a deterministic algorithm that given a circuit-SAT instance f outputs n circuit-SAT instances $(f_1, \dots, f_n)$ over a common set of variables $\mathbf{z} = (z_1, \dots, z_m)$.*
- *The assignment mapper W is a randomized algorithm that given a circuit-SAT instance f and an assignment $\mathbf{x}$ to f, outputs a global assignment $\mathbf{y} \in \{0,1\}^m$ together with partial assignments $\mathbf{y}_1, \dots, \mathbf{y}_n \in \{0,1,*\}^m$.*
- *The simulator* Sim *is a randomized algorithm that given a circuit-SAT instance f and an unauthorized set $Z \notin \mathscr{A}$ outputs partial assignments $(\mathbf{y}_i)_{i \in Z}$.*
- *The decoder* Dec *is a deterministic algorithm that given a circuit-SAT instance f and an authorized set S together with partial assignments $(\mathbf{y}_i)_{i \in S}$, outputs an assignment $\mathbf{x}$ for f.*

The algorithms satisfy the following properties.

- (Correctness) *If an assignment $\mathbf{x}$ satisfies f then the following holds with probability 1: $W(f, \mathbf{x})$ outputs $(\mathbf{y}, \mathbf{y}_1, \dots, \mathbf{y}_n)$ such that each partial assignment $\mathbf{y}_i$ is consistent with $\mathbf{y}$ on all Boolean values, and $\mathbf{y}_i$ satisfies f_i.*
- (Privacy) *If $\mathbf{x}$ satisfies f then for every unauthorized set $Z \notin \mathscr{A}$, the distribution of $(\mathbf{y}_i)_{i \in Z}$ sampled by $W(f, \mathbf{x})$ is identical to the distribution of* $\mathsf{Sim}(f, Z)$.
- (Recovery) *For any authorized set $S \in \mathscr{A}$, and for any partial assignments $(\mathbf{y}_i)_{i \in S}$ that are pairwise consistent on Boolean values, it holds that if every $\mathbf{y}_i$ satisfies f_i then* $\mathsf{Dec}(f, S, (\mathbf{y}_i)_{i \in S})$ *outputs an assignment $\mathbf{x}$ that satisfies f.*

We say that an n-party NPSS is a *t-out-of-n NPSS* if the access structure $\mathscr{A} \subseteq 2^{[n]}$ contains all subsets of $[n]$ of size at least t, and no subset of $[n]$ of size less than t.

3.2 From MPC to NPSS

We continue with the NPSS construction.

Theorem 3.4. *There is a deterministic compiler that takes as an input an access structure that is represented by a depth-d monotone formula χ that uses only AND and OR gates with fan-in 2, and uses no constants. The compiler runs in time $\mathrm{poly}(n \cdot 2^d)$ and outputs the description of an n-party $\mathscr{A}_\chi$-secret sharing for an* $\mathbf{NP}$ *statement $(R, W, \mathsf{Sim}, \mathsf{Dec})$. The running time of the algorithms $(R, W, \mathsf{Sim}, \mathsf{Dec})$ is $\mathrm{poly}(|f| \cdot 2^d \cdot n)$ where f is the circuit-SAT instance that the algorithms take as an input, and $|f|$ is the circuit-size of f measured by the number of wires. In addition, on input f the algorithm R outputs circuits $(f_1, \dots, f_n)$, each of size $|f| \cdot \mathrm{poly}(2^d \cdot n)$.*

As explained in Footnote 10, the formula-depth restriction can be replaced with a formula-size restriction and the constant-free requirement can be waived.

Proof. We begin by presenting some notation.

Notation. For a circuit-SAT instance f on ℓ input bits, we define $\mathcal{F}_f$ to be the n-client functionality that (1) takes an input $\mathbf{x}_i \in \{0,1\}^\ell$ from the i-th client, (2) computes $\mathbf{x} = \mathbf{x}_1 \oplus \ldots \oplus \mathbf{x}_n$, (3) returns $f(\mathbf{x})$ to all the clients. We let C be the compiler promised by Theorem 2.7, and we denote by C_χ the compiler C when the input-formula (acsess structure) is fixed to be χ, so C_χ only takes as an input a functionality.

We assume without loss of generality that every protocol π generated by C_χ satisfies the following syntactic property. Before any OLE instruction the sender locally computes the two possible outcomes of the OLE. Formally, if the sender c_i holds the inputs $(\alpha, \beta) \in \{0,1\}^2$ and the receiver c_j holds a selection bit $\gamma \in \{0,1\}$, the sender c_i first locally computes the values $z_0 := \alpha \cdot 0 + \beta$ and $z_1 := \alpha \cdot 1 + \beta$, and only then executes the OLE instance with c_j. We call (z_0, z_1) the *possible outputs* of the OLE instance, and we call z_γ the *chosen output* of the OLE. We emphasize that z_0, z_1 are variables that are owned by the sender c_i. In addition, for every $\mathsf{transmit}(c_i, c_j, x_1, x_2)$ statement we call x_1 the *transmitted message* and x_2 the *received message*.

Given a vector of values $\mathbf{y}$ for all the variables of the protocol, the *extended view* of a party c_i consists of all the variables that belong to c_i and, in addition, all the "transmitted messages" that are directed to c_i and all the "chosen outputs" of each OLE instruction in which c_i plays the receiver. (Note that the transmitted messages and the chosen outputs are not owned by c_i.)

Construction overview. Given a circuit-SAT instance f, we use C_χ to generate a protocol π' that computes $\mathcal{F}_f$ with χ-dual security, together with the corresponding simulator Sim' and the extractor Ext'. The global assignment $\mathbf{y}$ consists of all the variables $\mathcal{X}$ in an honest execution of π'. The i-th circuit f_i takes as an input a partial assignment $\mathbf{y}_i$ that contains the extended view of the ith client c_i and verifies that it is a valid accepting transcript. That is, f_i checks that the messages transmitted to the i-th client are consistent with the received messages, that the chosen outputs of the OLE calls that are directed to c_i are consistent with the OLE outputs, and that the local computations performed by c_i are consistent with π' and that the final output is 1. To sample a global assignment $\mathbf{y}$, we invoke the protocol π' on random inputs $(\mathbf{x}_i)_{i\in[n]}$ for which $\mathbf{x}_1 + \ldots + \mathbf{x}_n = \mathbf{x}$.

Privacy is guaranteed since π' is χ-passively secure, so the views of every unauthorized set $Z \notin \mathscr{A}_\chi$ (equivalently, $\chi(Z) = 0$) can be simulated by Sim'. In addition, in an honest execution of π' all transmitted messages are equal to the received messages, and for every OLE instance, the value of the chosen output is always equal to the value of the actual output, and the receiver also knows that the location γ of the chosen output z_γ among the possible outputs (z_0, z_1) of the sender. Therefore, those values can be extracted from the simulated views.

As for recovery, we note that for every authorized set $S \in \mathscr{A}_\chi$ it holds that $\chi(S) = 1$. In addition, the use of extended views guarantees that (1) the private channels between every pair of parties in S work correctly, and (2) the OLE channels worked correctly for every pair of parties in S. Therefore, we can use the extractor Ext' to extract the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of all the clients, and compute the satisfying assignment $\mathbf{x} = \mathbf{x}_1 \oplus \ldots \oplus \mathbf{x}_n$. We continue with a formal description of the construction.

The construction. The algorithms $(R, W, \mathsf{Sim}, \mathsf{Dec})$ are defined as follows:

- **Instance mapper.** The algorithm R takes as an input an instance of circuit-SAT f, and computes $(\pi', \mathsf{Sim}', \mathsf{Ext}') = C_\chi(\mathcal{F}_f)$ over the variable space $\mathcal{X}$. It outputs n circuits $f_1, \ldots, f_n$, where f_i is defined as follows.

- *(Inputs)* It takes as an input a partial assignment $\mathbf{y}_i$ that is parsed as the values of the variables in $\mathcal{X}$, and we denote by $\mathbf{y}_i[x] \in \{0,1,*\}$ the value of the variable $x \in \mathcal{X}$ according to $\mathbf{y}_i$.
- *(Computation)* For two bits $a, b \in \{0,1\}$, let $\mathsf{eq}(a,b)$ be a sub-circuit that returns 1 if $a = b$, and 0 otherwise, i.e., $\mathsf{eq}(a,b) = (a \wedge b) \vee (\neg a \wedge \neg b)$. Observe that $\mathsf{eq}(a,*) = \mathsf{eq}(*,a) = *$ for every $a \in \{0,1,*\}$. The circuit verifies that $\mathbf{y}_i$ corresponds to a valid extended view of the client c_i, by performing the following local computations:
  1. For every variables x in the view of c_i, it verifies that $\mathbf{y}_i[x] \neq *$ by computing $\mathsf{eq}(\mathbf{y}_i[x], 0) \vee \mathsf{eq}(\mathbf{y}_i[x], 1)$.
  2. For every $\mathsf{comp}(c_i, \mathsf{op}, (x_1, x_2), x)$ statement with $\mathsf{op} \in \{+, \times\}$, it verifies that the statement was computed correctly in the view, by computing $\mathsf{eq}(\mathsf{op}(\mathbf{y}_i[x_1], \mathbf{y}_i[x_2]), \mathbf{y}_i[x])$.
  3. For every abort statement $\mathsf{abort}(c_i, x)$ it verifies that c_i did not raise the abort flag, i.e., it computes $\mathsf{eq}(\mathbf{y}_i[x], 0)$.
  4. For every $\mathsf{transmit}(c_j, c_i, x_1, x_2)$ it verifies that the transmitted message is equal to the received message by computing $\mathsf{eq}(\mathbf{y}_i[x_1], \mathbf{y}_i[x_2])$.
  5. For every $\mathsf{func}(\mathsf{OLE}, c_j, c_i, (x_1, x_2), x_3, x_4)$ it verifies that the actual output (i.e., the value of x_4) is equal to the value of the chosen output of the OLE instance. That is, we denote by (z_0, z_1) the possible outputs of the OLE, and the circuit performs the following computation
  $$[(1 \oplus \mathbf{y}_i[x_3]) \wedge \mathsf{eq}(\mathbf{y}_i[x_4], \mathbf{y}_i[z_0])] \vee [\mathbf{y}_i[x_3] \wedge \mathsf{eq}(\mathbf{y}_i[x_4], \mathbf{y}_i[z_1])].$$
  (Here we slightly abuse notation and refer to (z_0, z_1) as the variables corresponding to the possible outputs of the OLE, instead of the values of those variables.)
  6. Since $\mathcal{F}_f$ returns $f(\mathbf{x})$ to all the clients, the protocol π' contains a single output instruction of the form $\mathsf{output}(p_i, 1, x)$. The circuit verifies that $\mathbf{y}_i[x] = 1$, i.e., it computes $\mathsf{eq}(1, \mathbf{y}_i[x])$.
- *(Output)* The output is the AND over all the local computations.

- **Assignment mapper.** The algorithm W takes as an input a circuit-SAT instance f and an assignment $\mathbf{x}$ to f. It computes $(\pi', \mathsf{Sim}', \mathsf{Ext}') = C_\chi(\mathcal{F}_f)$ over the variable space $\mathcal{X}$. It then samples an n-out-of-n secret sharing of $\mathbf{x}$, denoted $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and computes an honest execution of π' where the i-th client holds $\mathbf{x}_i$ as an input. The global assignment $\mathbf{y}$ is set to be the values of all the variables in $\mathcal{X}$. The i-th partial assignment $\mathbf{y}_i$ is equal to $\mathbf{y}$ for every variable x in the extended view of c_i, and in all other variables it is equal to $*$. The algorithm outputs $(\mathbf{y}, \mathbf{y}_1, \ldots, \mathbf{y}_n)$.

- **Simulator.** The algorithm Sim takes as an input a circuit-SAT instance f, and an unauthorized set $Z \notin \mathscr{A}_\chi$. It computes $(\pi', \mathsf{Sim}', \mathsf{Ext}') = C_\chi(\mathcal{F}_f)$ over the variable space $\mathcal{X}$. It executes $(\mathsf{view}_i)_{c_i \in Z} \leftarrow \mathsf{Sim}'(Z, (\mathbf{x}_i)_{c_i \in Z}, (y_i)_{c_i \in Z})$ where for every $c_i \in Z$ the string $\mathbf{x}_i$ is a random binary string of length ℓ, and $y_i = 1$. Then, for every $c_i \in Z$ the algorithm Sim extends view_i into the extended view view_i' by (1) setting the transmitted message to be equal to the received message for every $\mathsf{transmit}(p_j, p_i, x_1, x_2)$ statement, and (2) for every OLE instance in which p_i acts as the receiver with input γ and output δ, setting the chosen output z_γ to be

δ. The extended views $(\mathsf{view}'_i)_{c_i \in Z}$ are then used to create the partial assignments $(\mathbf{y}_i)_{c_i \in Z}$, where for every x in the extended view of c_i, $\mathbf{y}_i[x]$ is equal to the value of x according to view'_i, and otherwise $\mathbf{y}_i[x]$ is set to be $*$. The simulator outputs $(\mathbf{y}_i)_{c_i \in Z}$.

- **Decoder.** The algorithm Dec takes as an input a circuit-SAT instance f and an authorized set $S \in \mathscr{A}_F$ together with partial assignments $(\mathbf{y}_i)_{i \in S}$. For every $i \in S$ the decoder parses $\mathbf{y}_i$ as the extended view of c_i, and sets view_i to be the view of c_i. The decoder Dec computes $(\pi', \mathsf{Sim}', \mathsf{Ext}') = C_\chi(\mathcal{F}_f)$ and $Z = [n] \setminus S$, and $(\mathbf{x}_i)_{i \in [n]} = \mathsf{Ext}'(Z, (\mathsf{view}_i)_{i \in S})$, and outputs $\mathbf{x} = \mathbf{x}_1 \oplus \ldots \oplus \mathbf{x}_n$.

Fix any circuit-SAT instance f, let $R(f) = (f_1, \ldots, f_n)$, and let $(\pi', \mathsf{Sim}', \mathsf{Ext}') = C_\chi(\mathcal{F}_f)$. It is not hard to verify that since the number of statements in π' is $|f| \cdot \mathrm{poly}(n \cdot 2^d)$ (see Theorem 2.7), then the size of each circuit f_i is also $|f| \cdot \mathrm{poly}(n \cdot 2^d)$. We continue with an analysis.

Correctness. Let $\mathbf{x}$ be a satisfying assignment of f. The global assignment $\mathbf{y}$ generated by $W(f, \mathbf{x})$ corresponds to an honest execution of π' in which the outputs of all the clients is 1, and every $\mathbf{y}_i$ corresponds to the extended view of the i-th client. Therefore $\mathbf{y}_i$ satisfies f_i. This concludes the correctness analysis.

Privacy. Let $\mathbf{x}$ be a satisfying assignment of f and fix any unauthorized set $Z \notin \mathscr{A}_\chi$, i.e., $\chi(Z) = 0$. Observe that in an honest execution of π' by $W(f, \mathbf{x})$ the inputs of the clients in Z are uniformly distributed, and that the output of every client is 1. Since π' provides χ-passive security, the distribution of $(\mathsf{view}_i)_{c_i \in Z} \leftarrow \mathsf{Sim}'(Z, (\mathbf{x}_i)_{c_i \in Z}, (y_i)_{c_i \in Z})$ is the same as the distribution of the views of the clients in Z in the execution of π' by $W(f, \mathbf{x})$. Fix those views and observe that the extended views can be computed from the views by the same procedure that Sim performs to extend the views. This concludes the privacy analysis.

Recovery. Fix any authorized set $S \in \mathscr{A}_\chi$, and any partial assignments $(\mathbf{y}_i)_{i \in S}$ that are pairwise consistent on Boolean values, and assume that $\mathbf{y}_i$ satisfies f_i for all $i \in S$. This implies that (1) for every variable x in the view of c_i it holds that $\mathbf{y}_i[x] \in \{0, 1\}$, (2) the view defined by $\mathbf{y}_i$ corresponds to an honest execution of π' by c_i, (3) the private channels and the OLE channel worked correctly between every pair of parties in S, and (4) the outputs of all the parties in S is 1, and no party raised the abort flag. Let $(\mathsf{view}_i)_{i \in S}$ correspond to the views of the parties in S according to $(\mathbf{y}_i)_{i \in S}$.

We note that there exists an active adversarial behavior $\tilde{\pi}'$ that can result in $(\mathsf{view}_i)_{i \in S}$. Indeed, for every statement $\mathsf{transmit}(c_i, c_j, x_1, x_2)$ where $i \notin S$ and $j \in S$, mark the statement by the value $\mathbf{y}_j[x_2]$, and for every statement $\mathsf{func}(\mathsf{OLE}, p_i, p_j, (x_1, x_2), x_3, x_4)$ where $i \notin S$ and $j \in S$, mark the statement by the value $\mathbf{y}_j[x_4]$. Now, if the parties in S execute $\tilde{\pi}'$ with the same randomness as in $(\mathsf{view}_i)_{i \in S}$ then they obtain the same views. In addition, since $S \in \mathscr{A}_\chi$ then $\chi(S) = 1$, and let $Z = [n] \setminus S$. This implies that $\mathsf{Dec}(f, S, (\mathsf{view}_i)_{i \in S}) = \mathsf{Ext}'(Z, (\mathsf{view}_i)_{i \in S})$ outputs $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ such that $\mathcal{F}_f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 1$, i.e., $f(\mathbf{x}) = 1$, where $\mathbf{x} = \mathbf{x}_1 \oplus \ldots \oplus \mathbf{x}_n$. This concludes the recovery analysis and the proof of the theorem. □

Constructing k-out-of-n NPSS. For $1 \leq k \leq n$, let Th^n_k be the threshold formula that takes n input bits, outputs 1 if the Hamming weight of the input is at least k, and outputs 0 otherwise. To construct k-out-of-n NPSS, it only remains to show that Th^n_k applies a logarithmic-depth formula that uses no constants. An explicit construction of such a formula follows from the classical constructions of logarithmic-depth sorting networks [AKS83, Pat90, Sei09]. At a high level, sorting networks are Boolean circuits with *comparators* as gates, where each comparator takes two bits and outputs them in sorted order, i.e., $\mathsf{comp}(x,y) = (\min(x,y), \max(x,y))$. Since every comparator can be implemented by a constant-size monotone circuit that uses no constants $\mathsf{comp}(x,y) = ((x \wedge y), (x \vee y))$, each output bit of the sorting network can be described as a logarithmic-depth monotone circuit that uses no constants. We can now take Th_k^n to be the $(n-k+1)$-output bit of the network. The following corollary follows.

Corollary 3.5 (Theorem 1.1 restated)**.** *There is a deterministic compiler C_{Th} that takes as an input two integers k and n, runs in time $\mathrm{poly}(n)$ and outputs the description of a k-out-of-n NPSS $(R, W, \mathsf{Sim}, \mathsf{Dec})$. The running time of all algorithms in $(R, W, \mathsf{Sim}, \mathsf{Dec})$ is $\mathrm{poly}(n \cdot |f|)$ where f is the circuit-SAT instance that the algorithms take as an input, and $|f|$ is the circuit-size of f measured by the number of wires.*

4 Application: Multi-String NIZK

In this section we present our results regarding multi-string NIZK. In Section 4.1 we present the basic definitions of multi-string NIZK, in Section 4.2 we formally prove Theorem 1.2, and in Section 4.3 we present some extensions to the theorem.

4.1 Basic Definitions

Syntactically, *non-interactive proof system* (NIP) Π for an NP-relation R consists of three PPT algorithms $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ that receive a security parameter 1^κ as their first input. The algorithm Gen samples a CRS, the algorithm $\mathcal{P}$ is a prover algorithm that takes an instance/witness pair $(f, \mathbf{x})$ and a CRS crs and generates a proof π, and the algorithm $\mathcal{V}$ is a verification algorithm that decides whether to accept a proof π with respect to an instance f and a CRS crs. By default, we assume that the NP-relation is taken to be circuit-SAT and so f is a circuit and $\mathbf{x}$ is a satisfying assignment.

We begin with the definition of multi-string NIZK [GO14]. (As we will later see the standard NIZK definitions can be derived as a degenerate version of the multi-string model.) At a high level, a NIP $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ satisfies t_c-completeness if completeness is satisfied as long as t_c out of the n strings are honestly generated, while the rest of the strings can be chosen by the adversary. The definition of soundness and zero-knowledge is similar, with respect to thresholds t_s and t_z. (Note that unlike the case of MPC/secret-sharing definitions here the thresholds indicate the number of *honest* entities.) For ease of reading, we follow the formalization of [GO14], and treat the thresholds t_c, t_z, t_s and the number of CRS n as constants that do not grow with the security parameter. This setting is already well-motivated and non-trivial to achieve. We mention, however, that our results hold even when these parameters grow polynomially with the security parameter. In particular, the complexity of our constructions is polynomial in the number of strings n.

Following [GO14], we consider a strong adversarial model where the adversary is allowed to adaptively choose the corrupted strings and their location based on the honestly generated strings. This is formalized in the following definition.

Definition 4.1 ((t,n) adversaries)**.** *A* (t,n)-adversary *with query complexity $p > t$ is a circuit $\mathcal{A}$ that given a sequence of p CRS $\mathsf{crs} = (\mathsf{crs}_1, \ldots, \mathsf{crs}_p)$ outputs a vector $\mathsf{crs}' = (\mathsf{crs}'_1, \ldots, \mathsf{crs}'_n)$ and possibly auxiliary information z such that at least t of the entries of crs' are taken from crs. That is,*

$$\exists I \subseteq [p],\ |I| \geq t,\ \forall i \in I,\ \exists! j \in [n],\ \mathsf{crs}_i = \mathsf{crs}'_j,$$

where $\exists!$ stands for the unique existential quantifier. Jumping ahead, the input strings crs *will always be sampled independently at random by using the (honest) CRS-sampler* $\mathsf{Gen}(1^\kappa)$*. We refer to I as the set of honestly generated CRS and to its complement $[n] \setminus I$ as the set of corrupted CRS. We also consider a* non-adaptive (t, n)-adversary *that, for some predefined subset $I \subset [n]$ of size at least t and for some predefined sequence of strings $(\mathsf{crs}'_i)_{i \notin I}$, takes as in input a vector of (honestly generated) CRS $(\mathsf{crs}_i)_{i\in[n]}$ and outputs $(\mathsf{crs}'_i)_{i\in[n]}$ where $\mathsf{crs}'_i = \mathsf{crs}_i, \forall i \in I$. Note that, by definition, the query complexity p is always smaller than the size of the circuit and so polynomial-sized circuits always have polynomial query complexity.*

We can now define completeness, soundness and zero-knowledge. Throughout this section *efficient adversaries* are modeled by a family of non-uniform circuits of size polynomial in the security parameter κ.

Definition 4.2 (Completeness)**.** *A triple of PPT algorithms* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *is* (t_c, n)-complete *if for every efficient (t_c, n)-adversary $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ with query complexity $p(\kappa) \geq t_c$, there exists a negligible function μ, so that for all sufficiently large κ it holds that*

$$\Pr_{\substack{\forall i\in[p(\kappa)]:\ \mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\kappa)\\ ((\mathsf{crs}'_i)_{i\in[n]}, f, \mathbf{x}) \leftarrow \mathcal{A}_\kappa((\mathsf{crs}_i)_{i\in[p(\kappa)]})\\ \pi \leftarrow \mathcal{P}(1^\kappa, (\mathsf{crs}'_i)_{i\in[n]}, f, \mathbf{x})}} \left[\ f(\mathbf{x}) = 1 \bigwedge \mathcal{V}(1^\kappa, (\mathsf{crs}'_i)_{i\in[n]}, f, \pi) = 0\ \right] \leq \mu(\kappa).$$

We obtain strong completeness *if the above holds for any choice of* $\mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\kappa)$, $((\mathsf{crs}'_i)_{i\in[n]}, f, \mathbf{x}) \leftarrow \mathcal{A}_\kappa((\mathsf{crs}_i)_{i\in[p(\kappa)]})$ *and the error probability is taken only over the coins of the prover and verifier. Completeness is* perfect *if* $\mu(\kappa) = 0$.

Definition 4.3 (Soundness)**.** *A triple of PPT algorithms* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *is* (t_s, n)-sound *if for every efficient (t_s, n)-adversary $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ with query complexity $p(\kappa) \geq t_s$, there exists a negligible function μ, so that for all sufficiently large κ it holds that*

$$\Pr_{\substack{\forall i\in[p(\kappa)]:\ \mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\kappa)\\ ((\mathsf{crs}'_i)_{i\in[n]}, f, \pi) \leftarrow \mathcal{A}_\kappa((\mathsf{crs}_i)_{i\in[p(\kappa)]})}} \left[\ f \text{ is not satisfiable} \bigwedge \mathcal{V}(1^\kappa, (\mathsf{crs}'_i)_{i\in[n]}, f, \pi) = 1\ \right] \leq \mu(\kappa).$$

The triple satisfies statistical soundness, if soundness is satisfied even with respect to circuit families of unbounded size $\mathcal{A}$ whose output-length and query complexity are polynomially bounded.

We continue with the definition of zero knowledge. Recall that in standard NIZK one typically distinguishes between non-adaptive zero-knowledge and adaptive zero-knowledge (See [Gol01, Chapter 4.10.3]). Roughly, in the former case the instance/witness pair (f, x) are selected independently of the CRS and in the latter case the pair may depend on the CRS. (Technically, this means that the simulator operates in two phases where the CRS is sampled in the first phase, and the proof is sampled in the second phase.) We consider both variants but for non-adaptive zero-knowledge assume that the (t_z, n)-adversary that samples the CRS is also non-adaptive. (This alignment between the CRS adversary and the instance adversary simplifies things.)

Definition 4.4 (Non-adaptive zero knowledge)**.** *A triple of PPT algorithms* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *is* (t_z, n) non-adaptive zero-knowledge *if there exists a PPT simulator* Sim*, such that for every efficient non-adaptive (t_z, n)-adversary given by $\left\{I_\kappa, (\mathsf{crs}_{\kappa,i})_{i\in[n]\setminus I_\kappa}\right\}_{\kappa\in\mathbb{N}}$ where $I_\kappa \subseteq [n]$ is of size at least t_z, and every sequence of polynomially-bounded valid instance/witness pairs $\{(f_\kappa, \mathbf{x}_\kappa)\}_{\kappa\in\mathbb{N}}$,*[12] *and every efficient distinguisher*

[12]That is, for every κ, the assignment $\mathbf{x}_\kappa$ satisfies f_κ and the length $|f_\kappa| + |\mathbf{x}_\kappa|$ is at most $p(\kappa)$ for some polynomial $p(\cdot)$.

$\mathcal{D} = \{\mathcal{D}_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function μ, so that for all sufficiently large κ it holds that

$$\left| \Pr_{\substack{\forall i\in I_\kappa:\ \mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\kappa) \\ \forall i\in[n]\setminus I_\kappa:\ \mathsf{crs}_i := \mathsf{crs}_{\kappa,i} \\ \pi \leftarrow \mathcal{P}(1^\kappa, (\mathsf{crs}_1,\ldots,\mathsf{crs}_n), f, \mathbf{x})}} [\mathcal{D}_\kappa((\mathsf{crs}_i)_{i\in[n]}, \pi) = 1] - \Pr[\mathcal{D}_\kappa(\mathsf{Sim}(1^\kappa, f_\kappa, I_\kappa, (\mathsf{crs}_{\kappa,i})_{i\in[n]\setminus I_\kappa})) = 1] \right| \le \mu(\kappa).$$

The triple satisfies statistical zero knowledge if zero knowledge is satisfied with respect to circuit families $\mathcal{D}$ of unbounded size.

Definition 4.5 (Adaptive zero-knowledge). *A triple of PPT algorithms* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *is* (t_z, n)-adaptive zero-knowledge *if there exist efficient simulators* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$*, such that*

- (CRS indistinguishability) *For every efficient distinguisher $\mathcal{D} = \{\mathcal{D}_\kappa\}_{\kappa\in\mathbb{N}}$ there exists a negligible function $\mu(\kappa)$ such that for all sufficiently large κ,*

$$\left| \Pr_{\mathsf{crs}\leftarrow\mathsf{Gen}(1^\kappa)} [\mathcal{D}_\kappa(\mathsf{crs}) = 1] - \Pr_{(\mathsf{crs},\tau)\leftarrow\mathsf{Sim}_1(1^\kappa)} [\mathcal{D}_\kappa(\mathsf{crs}) = 1] \right| \le \mu(\kappa).$$

- (Simulation indistinguishability) *For every efficient (t_z, n)-adversary $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ with query complexity $p(\kappa) \ge t_s$ and every efficient distinguisher $\{\mathcal{B}_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function μ, so that for all sufficiently large κ it holds that*

$$\left| \Pr_{\substack{\forall i\in[p(\kappa)]:\ (\mathsf{crs}_i,\tau_i)\leftarrow\mathsf{Sim}_1(1^\kappa) \\ ((\mathsf{crs}'_i,\tau'_i)_{i\in[n]}, f, \mathbf{x}, st)\leftarrow\mathcal{A}_\kappa((\mathsf{crs}_i,\tau_i)_{i\in[p(\kappa)]}) \\ \pi\leftarrow\mathcal{P}(1^\kappa, (\mathsf{crs}'_i)_{i\in[n]}, f, \mathbf{x})}} [\mathcal{B}_\kappa(st, \pi) = 1] \right.$$
$$\left. - \Pr_{\substack{\forall i\in[p(\kappa)]:\ (\mathsf{crs}_i,\tau_i)\leftarrow\mathsf{Sim}_1(1^\kappa) \\ ((\mathsf{crs}'_i,\tau'_i)_{i\in[n]}, f, \mathbf{x}, st)\leftarrow\mathcal{A}_\kappa((\mathsf{crs}_i,\tau_i)_{i\in[p(\kappa)]}) \\ \pi\leftarrow\mathsf{Sim}_2(1^\kappa, (\mathsf{crs}'_i,\tau'_i)_{i\in[n]}, f)}} [\mathcal{B}_\kappa(st, \pi) = 1] \right| \le \mu(\kappa),$$

  where we assume that $\mathcal{A}_\kappa$ always outputs $\mathbf{x}$ that satisfies f, and sets the trapdoor τ'_i of the corrupted CRS's to $\perp$.

The triple satisfies statistical adaptive zero-knowledge if the above holds even when $\mathcal{D}$ and $\mathcal{A}$ are circuit families of arbitrary (unbounded) size.

Finally, a triple of PPT algorithms $\Pi = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ is (t_c, t_s, t_z, n)-*NIZK* if it satisfies (t_c, n)-completeness, (t_s, n)-soundness and (t_z, n)-adaptive zero-knowledge. If the zero-knowledge property is relaxed to (t_z, n) non-adaptive zero-knowledge then Π is referred to as (t_c, t_s, t_z, n)-*NIZK with non-adaptive ZK*. We refer to the special case of $(1,1,1,1)$-NIZK as NIZK (with adaptive completeness, adaptive soundness and adaptive zero-knowledge). Similarly, we refer to $(1,1,1,1)$-NIZK with non-adaptive zero-knowledge as NIZK with non-adaptive ZK.[13]

[13]It is not hard to show that these definitions are indeed equivalent to the standard definitions of NIZK. One minor syntactic difference is the ability of a $(1,1)$-adversary to sample polynomially-many CRS's and select the preferred one, however, if such an adversary can violate completeness (resp., soundness, zero-knowledge) with inverse polynomial probability then a standard adversary that gets a single honestly generated CRS can apply the same attack with an inverse polynomial probability of success.

4.1.1 Non-Interactive Commitments in the MS-model

We will need the following non-standard notion of non-interactive commitments in the Multi-String model where there are n CRS and where hiding holds even if all the CRS are adversarially chosen and binding holds as long as one CRS is sampled honestly. (In fact, weaker notions suffice for our applications, and we chose this strong formulation for simplicity and due to the fact that we can achieve it based on weak assumptions.) We denote the NICOM CRS by $(\mathsf{pp}_i)_{i\in[n]}$ to distinguish them from the CRS of the NIZK. We will assume that honestly generated CRS pp is sampled uniformly from $\{0,1\}^\kappa$ since all our constructions satisfy this property anyway.

Definition 4.6 (Non-interactive commitment). *A* Multi-String non-interactive commitment *(MS-NICOM in short) is a pair of probabilistic algorithms* $(\mathsf{commit}, \mathsf{open})$ *that take as a common input a vector of public parameters* (pp_i) *each of length* κ*. The scheme should satisfy the following requirements:*

- **Syntax:** *The algorithm* commit *takes as an input a bit* $x \in \{0,1\}$ *and random string* r *and outputs a commitment/opening pair* (C, o)*. The algorithm* open *takes as an input a commitment/opening pair* (C, o) *and outputs a message* $x' \in \{0,1\} \cup \{\perp\}$.

- **Perfect correctness:** *For every security parameter* κ*, integer* n*, vector of public parameters* $\mathsf{pp} = (\mathsf{pp}_i)_{i\in[n]}$*, bit* x*, and random string* r*, it holds that* $\mathsf{open}_{\mathsf{pp}}(\mathsf{commit}_{\mathsf{pp}}(x; r)) = x$.

- **Binding against** $(1, n)$**-adversaries:** *For every polynomial* $n(\kappa)$ *and every efficient* $(1, n(\kappa))$*-adversary* $\mathcal{A} = \{\mathcal{A}_\kappa\}$ *with query complexity* $p(\kappa) > 1$*, there exists a negligible function* μ *such that for every security parameter* κ*,*

$$\Pr_{\substack{\forall i\in[p(\kappa)]:\ \mathsf{pp}_i \leftarrow \{0,1\}^\kappa \\ (\mathsf{pp}'=(\mathsf{pp}'_i)_{i\in[n]}, (C,o,o')) \leftarrow \mathcal{A}_\kappa((\mathsf{pp}_i)_{i\in[p(\kappa)]})}} \left[\ \mathsf{open}_{\mathsf{pp}'}(C, o) = 0 \bigwedge \mathsf{open}_{\mathsf{pp}'}(C, o') = 1\ \right]$$

  is upper-bounded by $\mu(\kappa)$*. Recall that being* $(1, n(\kappa))$*-adversary* $\mathcal{A}$ *must select some* i *and* j *such that* $\mathsf{pp}'_i = \mathsf{pp}_j$ *and can set all the other entries of* pp' *arbitrarily. The scheme is* statistically binding*, if the above holds even for inefficient adversaries with polynomial query complexity, and* perfectly binding *if, in addition,* $\mu = 0$.

- **Hiding against** $(0, n)$**-adversaries:** *For every polynomial* $n = n(\kappa)$*, every (adversarially chosen) vector of CRS* $\mathsf{pp} = (\mathsf{pp}_i)_{i\in[n]} \in (\{0,1\}^\kappa)^n$*, and every efficient distinguisher* $\mathcal{A} = \{\mathcal{A}_\kappa\}$*, there exists a negligible function* μ*, such that for every security parameter* κ*, it holds that*

$$\Big| \Pr_{(C,o)\leftarrow \mathsf{commit}_{\mathsf{pp}}(0)} [\mathcal{A}_\kappa(\mathsf{pp}, C) = 1] - \Pr_{(C,o)\leftarrow \mathsf{commit}_{\mathsf{pp}}(1)} [\mathcal{A}_\kappa(\mathsf{pp}, C) = 1] \Big| \le \mu(\kappa)$$

  The scheme is statistically hiding *if the above holds even for inefficient adversaries.*

For ease of reading, we typically omit the the public parameters from the algorithms. By default, the length of these parameters is set according to the global security parameter that is being used by the system. We also mention that the definition naturally generalizes to larger message domains.

Lemma 4.7 (MS-NICOM from OWF/CRH). *Assuming one-way functions (resp., collision-resistance hash functions), there exists MS-NICOM with computational hiding and statistical binding (resp., computational binding and statistical hiding).*

Proof sketch. We begin with a non-interactive commitment that employs a single (uniform) CRS. We require perfect correctness and computational-hiding (resp., statistical-hiding) for *every* CRS pp, and statistical-binding (resp., computational binding) holds except with negligible probability over a uniform choice of the CRS pp. (Put differently, such a scheme satisfies Definition 4.6 for $n = 1$.) Such a scheme can be constructed based on one-way functions using Naor's scheme [Nao91] (resp., based on collision-resistance hash functions via the construction of [DPP98b, HM96b]).

By using simple repetition we can lift both variants to the Multi-String model. That is, to commit to a bit b we duplicate the bit n times and commit to the ith copy independently with respect to pp_i, and to open the commitment we send all the corresponding openings, and verify that the opening algorithm accept the ith opening with respect to pp_i and b. Perfect correctness and hiding follow immediately (since they hold for every choice of the CRS pp).

To prove binding against $(1, n)$-adversaries, assume, towards a contradiction, that an adversary $\mathcal{A}$ violates binding with non-negligible probability $\mu(\kappa)$ and polynomial query complexity $p(\kappa)$. We translate this adversary into an adversary against the binding of the original scheme as follow. Given $\mathsf{pp} \leftarrow \{0,1\}^\kappa$ we generate a vector of public parameters $(\mathsf{pp}_j)_{j\in[p(\kappa)]}$ by placing pp in a randomly chosen location $i \in [p(\kappa)]$ and by sampling all other entries uniformly at random from $\{0,1\}^\kappa$. We feed the adversary $\mathcal{A}$ with the vector of public parameters, and get back the tuples $(\mathsf{pp}'_j)_{j\in[n]}$ and $(C_j, o_j, o'_j)_{j\in[n]}$. If there exists an i' such that $\mathsf{pp}'_{i'} = \mathsf{pp}_i$ we output $(C_{i'}, o_{i'}, o'_{i'})$. It is not hard to show that this attack succeeds with probability $\mu(\kappa)/p(\kappa)$, in contradiction to the binding of the original scheme. □

Succinct NICOM with local opening. We say that an MS-NICOM is *succinct* $(\mathsf{commit}, \mathsf{open})$ if supports messages x of arbitrary length ℓ, and generates commitments whose length grows linearly with the security parameter and number of CRS's, but is independent of ℓ. (Of course, computational hiding/binding are required only for messages of length polynomial in the security parameter.) We say that an MS-NICOM provides *local opening* if the opening/decommitment information o generated by $\mathsf{commit}_{\mathsf{pp}}(x)$ is composed of ℓ local commitments $(o_i)_{i\in[\ell]}$ each of length $O(n\kappa\log(\ell))$. Correctness should hold separately for each bit, i.e., $\mathsf{open}_{\mathsf{pp}}(o_i) = x[i]$, and the opening algorithm should run in time $n\text{poly}(\kappa)\cdot\log(\ell)$. Hiding should hold even when some of the bits were "opened". Formally, for every polynomials $n = n(\kappa), \ell = \ell(\kappa)$, every vector of public parameters $\mathsf{pp} \in (\{0,1\}^\kappa)^n$, every set $I \subset [\ell]$ and every messages $x_0, x_1 \in \{0,1\}^\ell$ for which $x_0[I] = x_1[I]$, it holds that

$$(\mathsf{pp}, C^0, (o_i^0)_{i\in I}) \approx (\mathsf{pp}, C^1, (o_i^1)_{i\in I})$$

where $(C^b, (o_j^b)_{j\in[\ell]}) = \mathsf{commit}_{\mathsf{pp}}(x_b)$ for $b \in \{0,1\}$. The above requirement can be stated for statistically-hiding commitments by replacing computational indistinguishability with statistical indistinguishability. (In contrast, note that succinct commitments can only be computationally binding.)

We note that the construction of [DPP98b, HM96b]) and its MS-extension (Lemma 4.7) yield succinct NICOM. The additional local-opening property can then be achieved by using the standard Tree-construction where the message bits appear on the leafs, each node is associated with a commitment to its children, and the root serves as the final commitment. (The same public parameters are used in all the calls to the base commitment.) Overall, we get computationally-binding statistically-hiding succinct MS-NICOM with local openning based on collision-resistance hash functions.

4.2 From NIZK to Multi-String NIZK

We transform NIZK to MS-NIZK in Figure 3.

The transform C

Given parameters n, t_z, let $(R, W, \mathsf{Sim}_{\mathsf{NPSS}}, \mathsf{Dec}_{\mathsf{NPSS}})$ be the $(n - t_z + 1)$-out-of-n NPSS promised in Corollary 3.5. Let $(\mathsf{commit}, \mathsf{open})$ be a statistically-binding MS-NICOM. Let $\Pi' = (\mathsf{Gen}', \mathcal{P}', \mathcal{V}')$ be a NIZK (in the standard single-CRS model). We define an MS-NIZK $\Pi = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ as follows.

- **CRS generator.** $\mathsf{Gen}(1^\kappa)$ takes as an input the security parameter 1^κ, samples $\mathsf{crs} \leftarrow \mathsf{Gen}'(1^\kappa)$ and pp for the NICOM, and outputs $(\mathsf{crs}, \mathsf{pp})$.
- **Prover.** The prover $\mathcal{P}(1^\kappa, (\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[n]}, f, \mathbf{x})$ takes as an input the security parameter 1^κ, n strings $(\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[n]}$, a circuit f and an assignment $\mathbf{x}$. The prover generates a proof π as follows.
  - The prover generates n circuit-SAT instances $(f_1, \dots, f_n)$ by calling $R(f)$, and samples a global assignment $\mathbf{y}$ together with n partial assignments $(\mathbf{y}_1, \dots, \mathbf{y}_n)$ by calling $W(f, \mathbf{x})$. The prover commits to the bits of $\mathbf{y} \in \{0,1\}^m$ by sampling $(C_i, o_i) \leftarrow \mathsf{commit}(\mathbf{y}[i])$.[a] She sets $\mathbf{C} := (C_i)_{i\in[m]}$ and $\mathbf{O} := (o_i)_{i\in[m]}$.
  - Based on the commitments $\mathbf{C}$ and the commitments CRS's, we define, for every $i \in [n]$, a circuit $\hat{f}_i$ that takes as an input a subset $J_i \subseteq [m]$ and openings $(o_j)_{j\in J_i}$, and outputs 1 if the following conditions hold: (1) for every $j \in J_i$, the decommitted value $w_j := \mathsf{open}(C_j, o_j)$ is not equal to $\perp$; (2) the partial assignment $(w_j)_{j\in J_i}$ (whose-non J_i entries are taken to be $*$) satisfies f_i. For every $i \in [n]$, the prover computes the circuit $\hat{f}_i$ and the assignment $\hat{\mathbf{y}}_i = (J_i, (o_j)_{j\in J_i})$, where $J_i := \{j \in [m] : \mathbf{y}_i[j] \neq *\}$.
  - For every $i \in [n]$ the prover samples $\pi_i \leftarrow \mathcal{P}'(1^\kappa, \mathsf{crs}_i, \hat{f}_i, \hat{\mathbf{y}}_i)$, and outputs $\pi = (\mathbf{C}, \pi_1, \dots, \pi_n)$.
- **Verifier.** The verifier $\mathcal{V}(1^\kappa, (\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[n]}, f, \pi)$ takes as an input the security parameter 1^κ, n strings $(\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[n]}$, a circuit f, and a proof $\pi = (\mathbf{C}, \pi_1, \dots, \pi_n)$. She generates the circuit-SAT instances $(f_i)_{i\in[n]} = R(f)$, and transforms them into the instances $(\hat{f}_i)_{i\in[n]}$ based on $\mathbf{C}$ and $(\mathsf{pp}_i)_{i\in[n]}$. The verifier accepts the proof if all the proofs π_i are accepted, i.e., if $\mathcal{V}'(1^\kappa, \mathsf{crs}_i, \hat{f}_i, \pi_i) = 1$ for every $i \in [n]$. If at least one of the proofs is rejected the verifier rejects.

[a](Here and throughout this section we omit the commitments CRS's and write $\mathsf{commit}(\cdot)$ and $\mathsf{open}(\cdot)$ instead of $\mathsf{commit}_{(\mathsf{pp}_i)_{i\in[n]}}$ and $\mathsf{open}_{(\mathsf{pp}_i)_{i\in[n]}}$.)

Figure 3: The transform C

In Section 4.2.1 we prove the following theorem.

Theorem 4.8 (Theorem 1.2 restated). *Consider the transform C defined in Figure 3 instantiated with an integer n, threshold t_z and some MS-NICOM with computational hiding and statistical binding. Then, C maps a NIP Π' into a NIP Π such that: (1) if Π' is strongly-complete (resp., perfectly complete) then Π is $(0, n)$ strongly-complete (resp., $(0, n)$ perfectly-complete); (2) if Π' is computationally-sound (resp., statistically-sound) then Π is (t_s, n) computationally-sound (resp., statistically-sound) for any $t_s > n - t_z$; (3) if Π' is non-adaptive zero-knowledge (resp., adaptive zero-knowledge) then Π is (t_z, n) non-adaptive zero-knowledge (resp., adaptive zero-knowledge); (4) if the CRS generator of Π' samples uniform strings of length $p(\kappa)$ then the CRS generator of Π samples uniform strings of length $p'(\kappa) = p(\kappa) + \kappa$. Moreover, the running time of the transform is* $\mathrm{poly}(n, \kappa)$.

Some comments are in place:

- (Underlying assumption): Recall that MS-NICOM with computational hiding and statistical binding can be based on one-way functions (Lemma 4.7), and so the theorem implies Theorem 1.2.

- (Tightness): It is shown in [GO14, Theorem 5] that, unless circuit-SAT is in $\mathbf{BPP}$, MS-NIZK exists only if $t_s + t_z > n$. Hence the parameters in the theorem are tight. (The original proof of [GO14, Theorem 5] is stated for the case of adaptive zero-knowledge but the argument holds for non-adaptive zero-knowledge as well.)

- (Imperfect completeness): The theorem assumes that the completeness property of the underlying NIZK is either perfect or strong[14] but does not apply to NIZK with general imperfect completeness. Nevertheless, we show, in Claim D.1 (Section D), that every NIZK with imperfect completeness can be efficiently transformed into a NIZK with strong completeness. Also, as observed by [GO14], whenever the verifier is deterministic (like in most standard NIZKs), an even simpler transformation yields perfect completeness as well.

- (Succinctness): Instead of using bit commitments, it is possible to use succinct NICOM with local opening. In this case, the new proof contains the commitment string $\mathbf{C}$ whose is length is $O(n, \kappa)$ and n proofs under Π' of circuit-SAT instances each of size is at most $|f| \cdot \log(|f|) \cdot \text{poly}(n, \kappa)$. Indeed, f_i is larger than f by a factor of $p(n)$ for some fixed polynomial $p(\cdot)$ and the verification of the opened commitments (per each bit of f_i) adds an additional factor of $\text{poly}(n, \kappa) \cdot \log(|\mathbf{y}|)$ where $|\mathbf{y}| \leq n \cdot p(n)|f|$. Suppose that the original proof system is *somewhat succinct* (resp., *succinct*), i.e., the length of a proof for S-size circuit is S^ϵ for some $\epsilon < 1$ (resp., $\text{poly}(\log(S), \kappa)$). Then assuming that n is polynomial in κ and that the size of f is sufficiently large compared to κ, the new system is also somewhat succinct (resp., succinct).

- (Online dependency on n): Note that the CRS-sampler of Π is independent of n and the given thresholds. Consequently, an authority can sample and publish a CRS without knowing how many authorities will eventually participate. In fact, assuming that N CRS were published the prover can decide which subset to use and how to set the threshold in an online manner while generating the proof itself.

- (Extensions): Theorem 4.8 extends in several ways. In particular, it also preserves proof of knowledge (see Section 4.3.1) and with the aid of statistically-hiding MS-NICOM (that can be based on collision-resistance hash functions) can also preserve statistical zero-knowledge (see Section 4.3). We mention (without a proof) that by simple modifications of the transformation we can also preserve additional useful properties, such as simulation-soundness, whenever $t_s \leq t_z$ (see [GO14] for formal definitions).

It is possible to use the above theorem to derive a combiner for NIZK.

Theorem 4.9. *Assuming the existence of one-way functions, there exists a transformation that takes n NIZK candidates $\Pi_i = (\mathsf{Gen}_i, \mathcal{P}_i, \mathcal{V}_i), i \in [n]$ and integer $t_z \leq n$ and defines a NIP Π^* such that if (1) all the candidate are perfectly complete and (2) at least t_z of the candidates satisfy the zero-knowledge property/adaptive zero-knowledge and (3) at least $t_s > n - t_z$ are computationally/statistically sound, then Π^**

[14]Recall that this essentially means that for every crs and every instance/witness pair $(f, \mathbf{x})$ a completeness error happens with at most negligible probability over the randomness of the verifier and prover.

is NIZK with zero-knowledge property/adaptive zero-knowledge and computational/statistical soundness. Moreover, if majority of the instances are NIZK (possibly with imperfect correctness) then the transformation works even if the other instances are not complete.

As before, if we use succinct MS-NICOM with local opening (based on collision resistance hash functions), we can preserve succinctness.

Proof. We show how to reduce the theorem to the statement in the MS-model (though a direct proof is also quite straightforward). The transformation is identical to the one described in Figure 3 with the following modifications. For every $i \in [n]$, the CRS-generator Gen^* samples $\mathsf{crs}_i \leftarrow \mathsf{Gen}_i(1^\kappa)$ and CRS $\mathsf{pp}_i \leftarrow \{0,1\}^\kappa$ for the computationally-hiding statistically-binding MS-NICOM (Lemma 4.7). Set $\mathsf{pp} = (\mathsf{pp}_i)_{i\in[n]}$.[15] The prover and verifier, $\mathcal{P}^*$ and $\mathcal{V}^*$ of Π^* are defined like in Figure 3 except that the generation and verification of the ith sub-proof π_i are obtained by calling $\mathcal{P}_i$ and $\mathcal{V}_i$, respectively.

Analysis. We reduce the security to the security in the MS-model as follows. For a set $S \subset [n]$, consider the NIP $\Pi'_S = (\mathsf{Gen}'_S, \mathcal{P}', \mathcal{V}')$ defined by a CRS-generator Gen' who samples $i \leftarrow S$ and $\mathsf{crs}_i \leftarrow \mathsf{Gen}_i(1^\kappa)$ and outputs the pair (i, crs_i), a prover $\mathcal{P}'(1^\kappa, (i, \mathsf{crs}_i), \cdot)$ that calls $\mathcal{P}_i(1^\kappa, \mathsf{crs}_i, \cdot)$ and a verifier $\mathcal{V}'(1^\kappa, (i, \mathsf{crs}_i), \cdot)$ that calls $\mathcal{V}_i(1^\kappa, \mathsf{crs}_i, \cdot)$. (Note that the prover/verifier are well defined even for $i \notin S$.)

First, let us take S to be the set of candidates that satisfies computational soundness (resp., statistical soundness). Then, Π'_S is a NIP that satisfies perfect completeness and soundness. Now consider a (t_s, n) adversary $\mathcal{A}_S$ against Π'_S that given $n\kappa$ CRS samples outputs a CRS vector $(i, \mathsf{crs}'_i)_{i\in[n]}$ where entries $i \in S$ are taken from its input and for $i \notin S$ the string crs'_i is sampled from $\mathsf{Gen}_i(1^\kappa)$. (With all but negligible probability the vector of inputs contains entries of the form $(i, \mathsf{crs}'_i)_{i\in[n]}$). Then, under the attack $\mathcal{A}_S$, the scheme $\Pi = C(\Pi')$ from Figure 3 behaves identically to our construction Π^*. (The same CRS distribution and an identical input/output behavior of the prover and verifier.) By Theorem 4.8, it follows that Π^* is perfectly complete and computationally (resp., statistically) sound.

Next, to establish non-adaptive/adaptive zero-knowledge take S to be the set of candidates that satisfies zero-knowledge (resp., adaptive zero-knowledge), and consider again the (t_z, n) adversary $\mathcal{A}_S$. Again, the system Π' satisfies zero-knowledge (resp., adaptive zero-knowledge) and, under the attack $\mathcal{A}_S$, the scheme $\Pi = C(\Pi')$ from Figure 3 behaves identically to our construction Π^*, and, by Theorem 4.8, it satisfies zero-knowledge (resp., adaptive zero-knowledge).

To prove the "Moreover" part, first apply Claim D.1 to each candidate and transform all the instances to instances with strong completeness without violating the zero-knowledge and soundness properties of the good instances. Then, apply the transformation with $t_s = t_z = \lceil (n+1)/2 \rceil$ and use the above analysis. □

4.2.1 Proof of Theorem 4.8

Soundness. We show that statistical/computational soundness is preserved. Consider a (t_s, n) adversary $\mathcal{P}^*$ given by a (possibly inefficient) non-uniform circuit family $\{\mathcal{P}^*_\kappa\}_{\kappa\in\mathbb{N}}$ and let q denote its output length and $p(\kappa) \geq t_s$ denote its query complexity. Suppose that $\mathcal{P}^*$ violates (t_s, n)-

[15]In fact, a single CRS for NICOM suffices, however the use of multiple CRS will allow us to reduce the current theorem to Theorem 1.2.

soundness with inverse polynomial probability of $\alpha(\kappa)$. We construct an adversary $\mathcal{A}$ with similar complexity that violates the soundness of Π' with inverse polynomial probability as follows.

Given a random $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa)$, sample $j \in [p(\kappa)]$, set $\mathsf{crs}_j = \mathsf{crs}$ and sample $\mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\kappa)$ for every $i \neq j$. In addition, sample, for each $i \in [p(\kappa)]$, honest CRS $\mathsf{pp}_i \leftarrow \{0,1\}^\kappa$ for the NICOM. Call the adversary $\mathcal{P}^*_\kappa((\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[p(\kappa)]})$ and denote its output by $((\mathsf{crs}'_i, \mathsf{pp}'_i)_{i\in[n]}, f, \pi)$. Let $I \subseteq [p(\kappa)]$ and $I' \subset [n]$ be sets of size at least t_s such that for all $i \in I$ there is $i' \in I'$ such that $\mathsf{crs}_i = \mathsf{crs}'_{i'}$ and $\mathsf{pp}_i = \mathsf{pp}'_{i'}$. If $j \in I$, parse $\pi = (\mathbf{C}, \pi_1, \ldots, \pi_n)$, compute the circuit $\hat{f}_{j'}$ (based on $f, \mathbf{C}, (\mathsf{pp}'_i)$) and output $(\hat{f}_{j'}, \pi_{j'})$. Otherwise, abort.

Let us condition on the event that $\mathcal{P}^*$ succeeds which happens with probability $\alpha(\kappa)$. This means that f is unsatisfiable but every proof π_i is accepted with respect to $\hat{f}_i$ and crs'_i. We claim that, except with negligible probability, at least one statement $\hat{f}_\ell, \ell \in I'$ is not satisfiable. Indeed, if this is not the case, there exists, for every $i \in I'$, an assignment $\hat{\mathbf{y}}_i$ that satisfies $\hat{f}_i$. Let $\mathbf{y}_i$ be the partial assignment defined by $\hat{\mathbf{y}}_i$. Since $\hat{f}_i(\hat{\mathbf{y}}_i) = 1$ we conclude that $\mathbf{y}_i$ is well defined and $f_i(\mathbf{y}_i) = 1$. Assuming that the statistical binding property of the commitment holds (which happens except with negligible probability), the assignments $(\mathbf{y}_i)_{i\in I'}$ are pairwise consistent on Boolean values. Since $t_s > n - t_z$, the perfect recovery of the NPSS scheme implies that $(\mathbf{y}_i)_{i\in I'}$ fully define an assignment $\mathbf{x}$ such that $f(\mathbf{x}) = 1$, in contradiction to our hypothesis.

We complete the argument by noting that from the point of view of the adversary the location j of the input CRS is distributed uniformly at random and so the event $j \in I$ happens with inverse polynomial probability of $1/p(\kappa)$. Let us condition on this event as well, and denote by $j' \in [n]$ the location for which $\mathsf{crs}'_{j'} = \mathsf{crs}_j$. The probability that $\hat{f}_{j'}$ is a false statement is at least $1/n$ (actually $1/t_s$) and so the new adversary violates soundness with inverse polynomial probability. Since the complexity of the new adversary is polynomial in the complexity of the original one, we preserve soundness both in the statistical and computational setting.

Zero knowledge. We begin with the case of non-adaptive ZK. Let Sim' be the simulator of Π'. The simulator $\mathsf{Sim}(1^\kappa, f, I, (\mathsf{crs}_i, \mathsf{pp}_i)_{i\in[n]\setminus I})$ takes as an input a security parameter 1^κ, a circuit f, a set $I \subseteq [n]$ of size at least t_z that indicates the location of honestly generated CRS's (hereafter referred to as "good") and a vector of "bad" CRS's $(\mathsf{crs}_i, \mathsf{pp}_i)_{i\in\bar{I}}$ where $\bar{I} = [n] \setminus I$ is the set of corrupted CRS's (hereafter referred to as "bad"). The simulator does as follows.

1. For every "good" index $i \in I$ the simulator samples pp_i. Therefore the simulator holds $\mathsf{pp}_1, \ldots, \mathsf{pp}_n$ that fully defines the commitments scheme.

2. The simulator computes $(f_1, \ldots, f_n) \leftarrow R(f)$ and samples witnesses for the "bad indices" $(\mathbf{y}_i)_{i\in\bar{I}} \leftarrow \mathsf{Sim}_{\mathsf{NPSS}}(1^\kappa, f, \bar{I})$, where we observe that $|\bar{I}| \leq n - t_z$.

3. The simulator computes the assignment $\mathbf{y} \in \{0,1\}^m$ in the following way: for every $i \in [m]$ for which there exists $j \in \bar{I}$ such that $\mathbf{y}_j[i] \neq *$, the simulator sets $\mathbf{y}[i] := \mathbf{y}_j[i]$ and otherwise the simulator sets $\mathbf{y}[i] := 0$. For every $i \in [m]$ the simulator samples $(C_i, o_i) \leftarrow \mathsf{commit}(\mathbf{y}[i])$ and sets $\mathbf{C}' = (C_i)_{i\in[n]}$ and $\mathbf{O}' = (o_i)_{i\in[n]}$. This fully defines $\hat{f}_1, \ldots, \hat{f}_n$ and $(\hat{\mathbf{y}}_i)_{i\in\bar{I}}$.

4. For every "bad" index $i \in \bar{I}$ the simulator computes $\pi'_i \leftarrow \mathcal{P}'(1^\kappa, \mathsf{crs}_i, \hat{f}_i, \hat{\mathbf{y}}_i)$. For every "good" index $i \in I$ the simulator samples $(\mathsf{crs}'_i, \pi'_i) \leftarrow \mathsf{Sim}'(1^\kappa, \hat{f}_i)$.

5. For "bad" index $i \in \bar{I}$ we denote $\mathsf{crs}'_i := \mathsf{crs}_i$, and for $i \in [n]$ we denote $\mathsf{pp}'_i := \mathsf{pp}_i$. The simulator outputs $((\mathsf{crs}'_i, \mathsf{pp}'_i, \pi'_i)_{i\in[n]}, \mathbf{C}')$.

We continue with an analysis. Let us fix $f_\kappa, \mathbf{x}_\kappa$, a set I_κ and strings $(\mathsf{crs}_i, \mathsf{pp}_i)_{i\in\bar{I}_\kappa}$. Observe that the simulated random variables $\mathsf{pp}'_1, \ldots, \mathsf{pp}'_n$ have the same distribution as in a real execution and fix them. It remains to compare the real-world random variables $(\mathbf{C}, (\mathsf{crs}_i, \pi_i)_{i\in[n]})$ to the simulated random variables $(\mathbf{C}', (\mathsf{crs}'_i, \pi'_i)_{i\in[n]})$.

Since $|\bar{I}_\kappa| \le n - t_z$, the privacy of the NPSS implies that $(\mathbf{y}_i)_{i\in\bar{I}_\kappa}$ have the same distribution as in a real execution and we fix them as well. Let us denote by $J \subseteq [m]$ the set of indices $i \in [m]$ such that there exists $j \in \bar{I}_\kappa$ with $\mathbf{y}_i[j] \ne *$. Let us denote $\mathbf{C}_J = (C_i)_{i\in J}$ and $\mathbf{O}_J = (o_i)_{i\in J}$ and we observe that these real-world random variables have the same distribution as the corresponding simulated random variables, and we fix them as well. By the privacy of the NIZK,

$$(\mathbf{C}_{[n]\setminus J}, (\mathsf{crs}_i, \pi_i)_{i\in I_\kappa}) \approx (\mathbf{C}_{[n]\setminus J}, (\mathsf{Sim}'(1^\kappa, \hat{f}_i))_{i\in I_\kappa}),$$

where $\approx$ stands for computational indistinguishability, the LHS corresponds to a real-world execution, and where we note that $\hat{f}_1, \ldots, \hat{f}_n$ are fully defined given $\mathbf{C}_{[n]\setminus J}$. By the hiding property of the commitment scheme (that holds for every fixing of $\mathsf{pp}_1, \ldots, \mathsf{pp}_n$),

$$(\mathbf{C}_{[n]\setminus J}, (\mathsf{Sim}'(1^\kappa, \hat{f}_i))_{i\in I_\kappa}) \approx (\mathbf{C}'_{[n]\setminus J}, (\mathsf{Sim}(1^\kappa, \hat{f}_i))_{i\in I_\kappa}),$$

where on the LHS $(\hat{f}_i)_{i\in I_\kappa}$ are defined with respect to $\mathbf{C}_{[n]\setminus J}$ and on the RHS $(\hat{f}_i)_{i\in I_\kappa}$ are defined with respect to $\mathbf{C}'_{[n]\setminus J}$. We therefore conclude that

$$(\mathbf{C}_{[n]\setminus J}, (\mathsf{crs}_i, \pi_i)_{i\in I}) \approx (\mathbf{C}'_{[n]\setminus J}, (\mathsf{Sim}'(1^\kappa, \hat{f}_i))_{i\in I}) = (\mathbf{C}'_{[n]\setminus J}, (\mathsf{crs}'_i, \pi'_i)_{i\in I}).$$

Finally, the proofs $(\pi)_{i\in\bar{I}}$ can be obtained from the LHS and the RHS by the same efficient randomized procedure. We conclude that $((\mathsf{crs}_i, \mathsf{pp}_i, \pi_i)_{i\in[n]}, \mathbf{C}) \approx ((\mathsf{crs}'_i, \mathsf{pp}'_i, \pi'_i)_{i\in[n]}, \mathbf{C}')$.

We sketch the proof for the case of adaptive Zero-Knowledge. Denoting by $(\mathsf{Sim}'_1, \mathsf{Sim}'_2)$ the simulators of the NIZK, we define a CRS-simulator $\mathsf{Sim}_1(1^\kappa)$ as follows: Sample $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}'_1(1^\kappa)$ and $\mathsf{pp} \leftarrow \{0,1\}^\kappa$, and output $((\mathsf{crs}, \mathsf{pp}), \tau)$. The second "proof-simulator" $\mathsf{Sim}_2(1^\kappa, (\mathsf{crs}_i, \mathsf{pp}_i, \tau_i)_{i\in[n]}, f)$ sets $I \subseteq [n]$ to be the set of all $i \in [n]$ with $\tau_i \ne \perp$, computes $(f_1, \ldots, f_n) \leftarrow R(f)$, samples $(\mathbf{y}_i)_{i\in\bar{I}} \leftarrow \mathsf{Sim}_{\mathsf{NPSS}}(f, \bar{I})$, and computes the assignment $\mathbf{y} \in \{0,1\}^m$, the commitments $\mathbf{C}' = (C_i)_{i\in[n]}$ and the openings $\mathbf{O}' = (o_i)_{i\in[n]}$ like in the original simulator in the non-adaptive simulator defined above. This fully defines $\hat{f}_1, \ldots, \hat{f}_n$ and $(\hat{\mathbf{y}}_i)_{i\in\bar{I}}$. For every $i \in \bar{I}$ the simulator computes $\pi'_i \leftarrow \mathcal{P}'(1^\kappa, \mathsf{crs}_i, \hat{f}_i, \hat{\mathbf{y}}_i)$, and for every $i \in I$ the simulator samples $\pi_i \leftarrow \mathsf{Sim}'_2(1^\kappa, (\mathsf{crs}_i, \tau_i), \hat{f}_i)$. The simulator outputs $((\mathsf{crs}_i, \mathsf{pp}_i, \pi_i)_{i\in[n]}, \mathbf{C}')$. The analysis of the simulators is similar to the analysis of the above non-adaptive case.

This concludes the proof of the theorem. □

4.3 Extensions

In this section we present extensions of Theorem 4.8 to the setting of proof of knowledge and to the case of statistical zero-knowledge.

4.3.1 Proof of Knowledge

We extend the standard definition of proof-of-knowledge to the MS-setting.

Definition 4.10 (Proof of Knowledge). *A triple of PPT algorithms* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *is* (t_s, n)*-proof of knowledge (PoK) if there exist a PPT algorithm* Gen' *and an efficient deterministic algorithm* Ext *that satisfy the following properties:*

- (CRS indistinguishability) $\mathsf{Gen}'(1^\kappa)$ *takes a security parameter* 1^κ *and outputs* (crs', s') *such that* crs' *has the same distribution as* $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa)$.
- (Extraction) *For every efficient* (t_s, n)*-adversary* $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ *with query complexity* $p(\kappa) \geq t_s$*, there exists a negligible function* μ*, so that for all sufficiently large* κ *it holds that*

$$\Pr_{\substack{\forall i\in[p(\kappa)]:\ (\mathsf{crs}_i, s_i)\leftarrow\mathsf{Gen}'(1^\kappa)\\ ((\mathsf{crs}'_i)_{i\in[n]}, f, \pi)\leftarrow\mathcal{A}_\kappa((\mathsf{crs}_i)_{i\in[p(\kappa)]})}}\left[\begin{array}{c}\mathcal{V}(1^\kappa, (\mathsf{crs}'_1,\ldots,\mathsf{crs}'_n), f, \pi) = 1 \text{ and}\\ \mathsf{Ext}(1^\kappa, I, (\mathsf{crs}_i, s_i)_{i\in I}, (\mathsf{crs}'_i)_{i\in[n]}, f, \pi) = \mathbf{x} \text{ but } f(\mathbf{x}) = 0\end{array}\right] \leq \mu(\kappa),$$

where $I \subseteq [p(\kappa)]$ *is the set of honestly generated CRS (i.e.,* $|I| \geq t_s$, $\forall i \in I$, $\exists j \in [n]$, $\mathsf{crs}_i = \mathsf{crs}'_j$*).*

The special case of $(1,1)$*-PoK corresponds to the standard definition of NIZK-POK.*

It is not hard to see that (t_s, n)-PoK implies (t_s, n)-Soundness. The following lemma shows that every NIZK that satisfies proof of knowledge implies an MS-NIZK with proof of knowledge.

Lemma 4.11. *The transformation described in Figure 3 preserves proof of knowledge. That is, if* Π' *is NIZK-PoK then* Π *is* $(0, t_s, t_z, n)$*-NIZK-PoK. Moreover, this holds even if the transformation is instantiated with MS-NICOM whose binding property is only computational.*

Proof sketch. Denote by $(\mathsf{Gen}'', \mathsf{Ext}')$ the CRS-generator and the extractor of the underlying NIZK. We define the CRS-generator and extractor $(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Ext}})$ of the MS-NIZK in the following way. The CRS generator $\widetilde{\mathsf{Gen}}(1^\kappa)$ samples $(\mathsf{crs}, s) \leftarrow \mathsf{Gen}''(1^\kappa)$, samples public parameters pp and outputs $((\mathsf{crs}, \mathsf{pp}), s)$. The extractor $\widetilde{\mathsf{Ext}}(1^\kappa, I, (\mathsf{crs}_i, \mathsf{pp}_i, s_i)_{i\in I}, (\mathsf{crs}'_i, \mathsf{pp}'_i)_{i\in[n]}, f, (\mathbf{C}, (\pi_i)_{i\in[n]}))$ first computes a mapping $\phi : I \to [n]$ mapping each index $i \in I$ to the index $j \in [n]$ such that $\mathsf{crs}_i = \mathsf{crs}'_j$. The extractor also computes $\hat{f}_1, \ldots, \hat{f}_n$, and for every $i \in I$ computes $\hat{\mathbf{y}}_{\phi(i)} = \mathsf{Ext}'(1^\kappa, (\mathsf{crs}_i, s_i), \hat{f}_{\phi(i)}, \pi_{\phi(i)})$. Given $\hat{\mathbf{y}}_{\phi(i)}$ the extractor computes the corresponding partial assignment $\mathbf{y}_{\phi(i)}$ and applies the NPSS decoder $\mathsf{Dec}_{\mathsf{NPSS}}(f, (\mathbf{y}_{\phi(i)})_{i\in I})$ to obtain the assignment $\mathbf{x}$. The extractor outputs $\mathbf{x}$. To analyze the extractor, assume that the proof is accepted and so all the sub-proofs pass verification. If the partial assignments $(\mathbf{y}_{\phi(i)})_{i\in I}$ are pair-wise consistent on Boolean values then the recovery property of the NPSS guarantees that $\mathbf{x}$ satisfies f (since $t_s > n - t_z$). If the partial assignments $(\mathbf{y}_{\phi(i)})_{i\in I}$ are not pair-wise consistent, then we violate the computational binding of the MS-NICOM as we obtain two different valid openings for the same commitment. □

4.3.2 Statistical Zero Knowledge

We show that if the original NIZK has statistical zero knowledge and proof of knowledge, then we can obtain an MS-NIZK with statistical zero-knowledge and proof of knowledge. This is done by instantiating our transformation with MS-NICOM with computational binding and statistical hiding whose existence follows from the existence of collision-resistance hash functions as shown in Lemma 4.7.

Theorem 4.12. *Consider the transform C defined in Figure 3 instantiated with an integer* n*, threshold* t_z *and with computationally-binding statistically-hiding MS-NICOM. Then, C maps a NIP* Π' *into a NIP*

Π *such that: (1) if* Π' *is strongly-complete (resp., perfectly complete) then* Π *is* $(0,n)$ *strongly-complete (resp.,* $(0,n)$ *perfectly-complete); (2) if* Π' *is PoK then* Π *is* (t_s,n) *PoK for any* $t_s > n - t_z$*; (3) if* Π' *is non-adaptive statistical zero-knowledge (resp., adaptive statistical zero-knowledge) then* Π *is* (t_z,n) *non-adaptive statistical zero-knowledge (resp., adaptive statistical zero-knowledge); (4) if the CRS generator of* Π' *samples uniform strings of length* $p(\kappa)$ *then the CRS generator of* Π *samples uniform strings of length* $p'(\kappa) = p(\kappa) + \kappa$*. Moreover, the running time of the transform is* $\mathrm{poly}(n,\kappa)$*.*

Proof sketch. Proof of knowledge follows from Lemma 4.11, and statistical adaptive/non-adaptive zero-knowledge follows in a similar way to the original proof of Theorem 4.8, by replacing computational indistinguishability with statistical indistinguishability. □

5 Application: Designated-Prover NIZK and Round-Optimal Honest-Majority MPC in Minicrypt

In this section we present our applications for designated-prover NIZK and MPC. In Section 5.1 we present the notion of designated-prover NIZK (DP-NIZK) together with two constructions: the first construction (Section 5.1.1) achieves perfect zero knowledge, and the second construction (Section 5.1.1) achieves statistical-soundness, and both constructions we only require the minimal assumption of one-way functions. In Section 5.2 we use DP-NIZK to construct an honest majority protocols with one offline round and one online round for public single input functionalities (Section 5.2.1), and general single input functionalities (Section 5.2.2). Finally, in Section 5.3 we present our 3-round honest-majority protocol for general MPC.

5.1 Designated-Prover NIZK

Designated-prover NIZK (DP-NIZK) is a proof system similar to NIZK with a small modification: In addition to generating the CRS, the *trusted party* also generates a secret key that is given to the prover. The secret key is then used by the prover to compute the proof, and has to remain secret to maintain zero knowledge. We continue with a formal definition, given with respect to circuit-SAT, and modified from [BGT20]. All our definitions apply to adaptive adversaries (that depend on the CRS) but are limited to a single-theorem setting (this suffices for our MPC applications). We note that, unlike the standard NIZK setting, the only known transform from single-theorem DP-NIZK to the multiple-theorem setting requires strong assumptions like leveled homomorphic encryption [BGT20].

Definition 5.1. *A* designated prover non-interactive zero-knowledge *(DP-NIZK) proof* Π *is a tuple of PPT algorithms* $(\mathsf{Gen},\mathcal{P},\mathcal{V})$ *with the following syntax:*

- $(\mathsf{crs},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$*: Given the instance length* κ*, the randomized set-up algorithm* Gen *outputs a CRS* crs *and a secret key for the prover* $\mathsf{sk}_\mathcal{P}$*.*

- $\pi \leftarrow \mathcal{P}(1^\kappa,\mathsf{crs},\mathsf{sk}_\mathcal{P},f,\mathbf{x})$*: Given the instance length* κ*, a CRS* crs*, a secret key* $\mathsf{sk}_\mathcal{P}$*, a circuit-SAT instance* f *with description of length at most* κ*, and an assignment* $\mathbf{x}$*, the randomized prover outputs a proof* π*.*

- $b = \mathcal{V}(1^\kappa,\mathsf{crs},f,\pi)$ *Given the instance length* κ*, a CRS* crs*, a circuit-SAT instance* f *with description of length at most* κ*, and a proof* π*, the deterministic verifier returns a bit* b*, representing accept or reject.*

The algorithms satisfy the following properties:

Perfect completeness. *For every $\kappa \in \mathbb{N}$, for every circuit-SAT instance f with description of length κ, and for every satisfying assignment $\mathbf{x}$, it holds that*

$$\Pr_{\substack{(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})\leftarrow\mathsf{Gen}(1^\kappa)\\ \pi\leftarrow\mathcal{P}(1^\kappa,\mathsf{crs},\mathsf{sk}_{\mathcal{P}},f,\mathbf{x})}}[\mathcal{V}(1^\kappa,\mathsf{crs},f,\pi)=1]=1.$$

Adaptive Proof of knowledge. *There exist a PPT algorithm* Gen' *and an efficient deterministic algorithm* Ext *that satisfy the following properties:*

- (CRS indistinguishability) $\mathsf{Gen}'(1^\kappa)$ *takes as an input the instance length* 1^κ *and outputs* $(\mathsf{crs}',\mathsf{sk}'_{\mathcal{P}},\tau')$ *such that* $(\mathsf{crs}',\mathsf{sk}'_{\mathcal{P}})$ *has the same distribution as* $(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})\leftarrow\mathsf{Gen}(1^\kappa)$.
- (Extraction) *For every polynomially-bounded non-uniform family of malicious provers* $\mathcal{P}^*=\{\mathcal{P}^*_\kappa\}_{\kappa\in\mathbb{N}}$ *there is a negligible function* $\mu(\kappa)$ *such that for all sufficiently large* κ *it holds that*

$$\Pr_{\substack{(\mathsf{crs},\mathsf{sk}_{\mathcal{P}},\tau)\leftarrow\mathsf{Gen}'(1^\kappa)\\ (f,\pi)\leftarrow\mathcal{P}^*_\kappa(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})}}[\mathsf{Ext}(1^\kappa,\mathsf{crs},\tau,f,\pi)=\mathbf{x}\wedge f(\mathbf{x})=0\wedge\mathcal{V}(1^\kappa,\mathsf{crs},f,\pi)=1]\le\mu(\kappa),$$

  where $\mathcal{P}^*_\kappa$ *outputs a circuit-SAT instance* f *of size at most* κ.

When extraction holds with respect to unbounded families of malicious provers, we say that we have statistical *proof of knowledge.*

Adaptive Zero knowledge. *There exist PPT algorithms* $(\mathsf{Sim}_1,\mathsf{Sim}_2)$ *such that for every polynomially-bounded non-uniform family of circuits* $\{\mathcal{A}_\kappa,\mathcal{B}_\kappa\}_{\kappa\in\mathbb{N}}$ *there exists a negligible function* μ *such that for all sufficiently large* κ *it holds that*

$$\left|\Pr_{\substack{(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})\leftarrow\mathsf{Gen}(1^\kappa)\\ (f,\mathbf{x})\leftarrow\mathcal{A}_\kappa(\mathsf{crs})\\ \pi\leftarrow\mathcal{P}(1^\kappa,\mathsf{crs},\mathsf{sk}_{\mathcal{P}},f,\mathbf{x})}}[\mathcal{B}_\kappa(\mathsf{crs},\pi)=1]-\Pr_{\substack{(\mathsf{crs},\tau)\leftarrow\mathsf{Sim}_1(1^\kappa)\\ (f,\mathbf{x})\leftarrow\mathcal{A}_\kappa(\mathsf{crs})\\ \pi\leftarrow\mathsf{Sim}_2(1^\kappa,\mathsf{crs},\tau,f)}}[\mathcal{B}_\kappa(\mathsf{crs},\pi)=1]\right|\le\mu(\kappa),$$

where $\mathcal{A}_\kappa$ *always outputs a circuit* f *with description of length* κ *and a satisfying assignment* $\mathbf{x}$. *When zero knowledge holds with respect to unbounded non-uniform families of circuits, we say that we have* statistical zero knowledge. *If, in addition,* $\mu(\kappa)=0$ *then we say that we have* perfect zero knowledge.

In the following sections we provide two constructions of DP-NIZK: The first construction provides *perfect zero knowledge* (Section 5.1.1), and the second construction provides *statistical proof of knowledge* (Section 5.1.2), and both constructions only assume the existence of one-way functions.

5.1.1 DP-NIZK with Perfect Zero Knowledge

We begin with a construction of DP-NIZK that provides perfect zero knowledge from one-way functions. Before presenting an overview of our construction, we recall the notion of private simultaneous messages protocol (PSM) [FKN94] also known as fully-decomposable randomized encoding [IK00, AIK04].

Building block: PSM. Let f be a circuit with k inputs and ℓ outputs, and recall that in a PSM for f there are k honest senders $Q_1, \ldots, Q_k$ and a single receiver R. Each Q_i holds a single input bit $x_i \in \{0,1\}$ of an assignment $\mathbf{x} = (x_1, \ldots, x_k)$ for f, and the senders also share a random string ρ that is not known to R. The protocol allows each Q_i to send to R a single message $m_i := \mathsf{psm}_i(x_i; \rho)$ that depends only on its private input bit x_i and shared randomness ρ, such that the messages $(m_1, \ldots, m_k)$ reveal the output $f(\mathbf{x})$, but no other information about the assignment $\mathbf{x}$.

Definition 5.2 (PSM Protocols). *Let $X_1, \ldots, X_\ell, Z$ be finite sets, and let $X = X_1 \times \ldots \times X_\ell$. An ℓ-party PSM protocol* psm*, computing a ℓ-argument function $f : X \to Z$ consists of:*

- *A message computation function $\mathsf{psm}_i : X_i \times R \to M_i$, for every party $i \in \{1, \ldots, \ell\}$, where R is a finite set of common random inputs and M_i is a finite message domain.*
- *A reconstruction function $\mathsf{rec} : M_1 \times \ldots \times M_\ell \to Z$ that will be computed by the receiver.*

The protocol $\mathsf{psm} = \big(\mathsf{psm}_1, \ldots, \mathsf{psm}_\ell, \mathsf{rec}\big)$ should satisfy the following properties.

1. **(Correctness)** *For every $(x_1, \ldots, x_\ell) \in X$ and $r \in R$, $\mathsf{rec}(\mathsf{psm}_1(x_1; r), \ldots, \mathsf{psm}_m(x_\ell; r)) = f(x_1, \ldots, x_\ell)$.*
2. **(Privacy)** *There exists a simulator $\mathsf{Sim}_{\mathsf{psm}}$, such that for every $(x_1, \ldots, x_\ell) \in X$,*

$$\mathsf{Sim}_{\mathsf{psm}}\big(f(x_1, \ldots, x_\ell)\big) \equiv (\mathsf{psm}_1(x_1; r), \ldots, \mathsf{psm}_m(x_\ell; r)),$$

   for $r \leftarrow R$.

Note that PSM security addresses only the case where the parties $Q_1, \ldots, Q_\ell$ are honest.

Lemma 5.3 (Polynomial-time PSM Protocols [IK02]). *For every ℓ-argument functionality f that admits a Boolean $\mathbf{NC}^1$ circuit of size s, there exists a PSM protocol with complexity of $\mathrm{poly}(s)$. In particular, if $s = \mathrm{poly}(\ell)$, then there exists a PSM protocol with complexity $\mathrm{poly}(\ell)$.*

Moving to 3SAT. It will be convenient to replace circuit-SAT with an $\mathbf{NP}$-hard problem whose $\mathbf{NP}$ relation can be evaluated in $\mathbf{NC}^1$. For example, we can use 3SAT, the language of satisfiable 3CNF formulas. Assuming that the 3CNF φ is represented via the natural representation[16] and the witness $\mathbf{w}$ is an assignment, the evaluation $\varphi(\mathbf{w})$ can be computed by an $\mathbf{NC}^1$ *universal evaluator circuit* $U(\varphi, \mathbf{w})$. (Here we abuse notation and do not distinguish between φ to its representation.) Further note that circuit-SAT can be efficiently reduced to 3SAT via a *witness-preserving* reduction and so DP-NIZK for the latter implies DP-NIZK for the former up to some polynomial loss in efficiency. (The overhead of the reduction can be minimized by using a generalized notion of 3SAT where each constraint is replaced with a general degree-2 equation of the form $xy - z = 0$; circuit-SAT can easily reduced to this language by augmenting the witness with the values of intermediate wires and by adding a consistency constraint for each gate.)

[16]Take φ to be a vector of clauses where each clause contains three literals, represented by the indices and the "polarity" bits of the corresponding variables.

Construction overview. The construction relies on information-theoretic PSM protocol and a digital signature scheme whose existence follows from any one-way function [Lam79, GMR84, Gol87, NY89, Rom90]. (See [Gol04] for a formal definition.) Let κ denote the length of φ. By padding the assignment, we can assume that it is of length κ as well. Let $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa \times \{0,1\}$ be an extension of the universal evaluator that given $(\varphi, \mathbf{w})$ outputs the instance φ and the bit $\varphi(\mathbf{w})$ indicating whether $\mathbf{w}$ satisfies φ. Since the universal evaluator is in $\mathbf{NC}^1$ so is F. Let psm be a PSM protocol for F.

Roughly, the trusted party samples a signing/verification keys for the signature and publishes the verification key as the CRS. In addition, she samples PSM randomness r, computes the PSM messages for all possible inputs, and privately sends a signed version of these messages to the prover. Given an instance φ and a witness $\mathbf{w}$, the prover releases the corresponding PSM messages, and the verifier checks that the signature is valid and that the PSM recovery algorithm outputs $(\varphi, 1)$. We proceed with a formal description.

DP-NIZK Π_{pzk}

- **Primitives.**
  - A perfectly-secure PSM $\mathsf{psm} = (\mathsf{psm}_1, \ldots, \mathsf{psm}_{2\kappa}, \mathsf{rec})$ for F, as promised in Lemma 5.3.
  - A digital signature scheme $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$.
- **Set-up algorithm** $\mathsf{Gen}(1^\kappa)$**.** On input 1^κ the algorithm Gen does as follows.
  1. Samples randomness r for psm, signature keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(1^\kappa)$, and sets $\mathsf{crs} := \mathsf{vk}$.
  2. For every $i \in [\kappa]$ and $b \in \{0,1\}$ computes
  $$\hat{\varphi}_{i,b} := \mathsf{psm}_i(b; r), \hat{w}_{i,b} := \mathsf{psm}_{\kappa+i}(b; r), \alpha_{i,b} \leftarrow \mathsf{Sign}_{\mathsf{sk}}((i, \hat{\varphi}_{i,b})), \quad \text{and } \beta_{i,b} \leftarrow \mathsf{Sign}_{\mathsf{sk}}((i, \hat{w}_{i,b})). \tag{1}$$
  3. Sets $\mathsf{sk}_{\mathcal{P}} := (r, (\alpha_{i,b}, \beta_{i,b})_{i\in[\kappa], b\in\{0,1\}})$ and outputs $(\mathsf{crs}, \mathsf{sk}_{\mathcal{P}})$.
- **Prover** $\mathcal{P}(1^\kappa, \mathsf{crs}, \mathsf{sk}_{\mathcal{P}}, \varphi, \mathbf{w})$**.** The prover on input 1^κ, CRS $\mathsf{crs} = \mathsf{vk}$, secret key $\mathsf{sk}_{\mathcal{P}} := (r, (\alpha_{i,b}, \beta_{i,b})_{i\in[\kappa], b\in\{0,1\}})$, a 3CNF instance $\varphi = (\varphi_i)_{i\in[\kappa]}$ with description of length κ, and a satisfying assignment $\mathbf{w} = (\mathbf{w}_i)_{i\in[\kappa]}$ does as follows.
  1. The prover computes, for every $i \in [\kappa]$, the PSM messages $\hat{\varphi}_{i,b} := \mathsf{psm}_i(\varphi_i; r), \hat{w}_{i,b} := \mathsf{psm}_{\kappa+i}(\mathbf{w}_i; r)$ and outputs
  $$\pi := ((\hat{\varphi}_i, \alpha_i), (\hat{w}_i, \beta_i))_{i\in[\kappa]}. \tag{2}$$
- **Verifier** $\mathcal{V}(1^\kappa, \mathsf{crs}, \varphi, \pi)$**.** The verifier on input 1^κ, CRS $\mathsf{crs} = \mathsf{vk}$, 3CNF instance φ with description of size κ, and a proof π parses π according to Eq. (2) and verifies that (a) the PSM messages evaluates to 1 and are consistent with φ, i.e.,
  $$\mathsf{rec}((\hat{\varphi}_i)_{i\in[\kappa]}, (\hat{w}_i)_{i\in[\kappa]}) = (\varphi, 1)$$
  and (b) that all the signatures pass verification; That is, for every $i \in [\kappa]$ it holds that $\mathsf{Verify}_{\mathsf{vk}}((i, \hat{\varphi}_i), \alpha_i) = 1$ and $\mathsf{Verify}_{\mathsf{vk}}((i, \hat{w}_i), \beta_i) = 1$. If all these conditions hold, the verifier accepts the proof and outputs 1. Otherwise, the verifier rejects the proof and outputs 0.

Figure 4: DP-NIZK Π_{pzk}

Theorem 5.4. *Assuming the existence of one-way functions, Π_{pzk} is a DP-NIZK with perfect zero knowledge.*

Proof sketch. Completeness follows from the perfect correctness of the digital signature scheme, and the perfect correctness of the PSM. We continue with proof of knowledge and perfect zero knowledge.

Proof of knowledge. On an input 1^κ the algorithm $\mathsf{Gen}'(1^\kappa)$ samples $(\mathsf{crs}, \mathsf{sk}_\mathcal{P}) \leftarrow \mathsf{Gen}(1^\kappa)$, sets $\tau := \mathsf{sk}_\mathcal{P}$ and outputs $(\mathsf{crs}, \mathsf{sk}_\mathcal{P}, \tau)$. The knowledge extractor Ext takes as an input the security parameter 1^κ, a CRS $\mathsf{crs} = \mathsf{vk}$, a trapdoor $\tau = (r, (\alpha_{i,b}, \beta_{i,b})_{i\in[\kappa], b\in\{0,1\}})$, a 3CNF φ whose description length is κ and a proof $\pi = ((\hat{\varphi}_i, \alpha_i), (\hat{w}_i, \beta_i))_{i\in[\kappa]}$. The extractor outputs an assignment $\mathbf{x} = (\mathbf{x}_i)_{i\in[\kappa]}$ defined as follows: For every $i \in [\kappa]$, the extractor computes the PSM messages $\hat{w}_{i,0}, \hat{w}_{i,1}$ as in Eq. (1) and sets the bit $\mathbf{x}_i$ such that $\hat{w}_i = \hat{w}_{i,x_i}$. If $\hat{w}_i$ is not one of $\hat{w}_{i,0}, \hat{w}_{i,1}$ the extractor fails.

Analysis: CRS indistinguishability is straightforward. In addition, it is not hard to verify that the extraction property is violated only if the adversary forges a signature, and therefore only with negligible probability. This concludes the analysis of proof of knowledge.

Perfect zero knowledge. We define the simulators in the following way. The simulator Sim_1 on input 1^κ does as follows: It samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(1^\kappa)$, sets $\mathsf{crs} := \mathsf{vk}$ and $\tau := \mathsf{sk}$, and outputs (crs, τ). The simulator Sim_2 on input 1^κ, a CRS $\mathsf{crs} = \mathsf{vk}$, a trapdoor $\tau = \mathsf{sk}$, and a 3CNF φ does as follows: It executes the PSM simulator $\mathsf{Sim}_{\mathsf{psm}}$ on $(\varphi, 1)$ to obtain the PSM messages $(\hat{\varphi}_i)_{i\in[\kappa]}, (\hat{w}_i)_{i\in[\kappa]}$, and signs each message with $\alpha_i \leftarrow \mathsf{Sign}_{\mathsf{sk}}((i, \hat{\varphi}_i))$, and $\beta_i \leftarrow \mathsf{Sign}_{\mathsf{sk}}((i, \hat{w}_i))$. Finally, the simulator outputs $\pi := ((\hat{\varphi}_i, \alpha_i), (\hat{w}_i, \beta_i))_{i\in[\kappa]}$.

We observe that the simulation is perfect. Indeed, fix any unbounded non-uniform family of circuits $\{\mathcal{A}_\kappa, \mathcal{B}_\kappa\}_{\kappa\in\mathbb{N}}$. Observe that the simulated $(\mathsf{sk}, \mathsf{vk})$ have the same distribution as the corresponding random variables generated by $\mathsf{Gen}(1^\kappa)$, and fix those random variables. Then $\mathsf{crs} = \mathsf{sk}$ is fixed, and so the tuple $(f, \mathbf{x})$ that $\mathcal{A}_\kappa(\mathsf{crs})$ outputs is fixed. The PSM simulator $\mathsf{Sim}_{\mathsf{psm}}$ perfectly simulates the messages $(\hat{\varphi}_i)_{i\in[\kappa]}, (\hat{w}_i)_{i\in[\kappa]}$, in the proof, and let us fix those messages as well. Finally, since $\tau = \mathsf{sk}$, the signatures of those messages in the simulation are sampled in the same way they are sampled by the trusted party $\mathsf{Gen}(1^\kappa)$. This concludes the analysis of zero knowledge and the proof of the theorem. $\square$

Remark 5.5 (Comparison to previous constructions)**.** *Previous works [BGT20, AKP22b] have used closely-related constructions, except that instead of using signatures the trusted party published commitments to the PSM messages. This commitment-based variant does not seem to achieve adaptive zero-knowledge due to a selective-opening issue, i.e., the set of opened commitments depends on the statement that is chosen after the commitments are published. This issue can be solved either by using statistically-hiding commitments or by complexity leveraging. The signature-based solution avoids this problem.*

5.1.2 DP-NIZK with Statistical Proof of Knowledge

We continue with a construction of DP-NIZK Π_{statPoK} that provides statistical proof of knowledge, assuming the existence of one-way functions. The construction is based on NIZK in the hidden-bits model [FLS90]. This approach is mentioned in [BGT20, Remark 1.1] without a proof and is attributed to anonymous referee. We continue with an overview of NIZK in the hidden-bits model.

NIZK in the hidden-bits model. In the hidden-bits model, the prover is first given access to a random string $\mathbf{r}$ of length m. The prover sends to the verifier a certificate π together with a subset $I \subseteq [m]$. The verifier is then given access only to the sub-string $\mathbf{r}_I$, and has to decide whether to accept or reject based on $(I, \mathbf{r}_I, \pi)$. See Section F.1 for a formal definition of the hidden-bit model, and some special properties that the construction of [FLS90] satisfies.

Construction overview. At a high level, we instantiate the hidden-bits model using non-interactive commitments. That is, we let the trusted party sample the CRS string $\mathbf{r} = (r_1, \ldots, r_m)$ for the NIZK in the hidden-bits model, and commit to every bit r_i with a commitment C_i, using the statistically-binding commitments of Naor [Nao89], where we also let the trusted party generate the public parameters pp of the commitment scheme. The CRS is set to be $(\mathsf{pp}, C_1, \ldots, C_m)$, and the openings $(o_1, \ldots, o_m)$ are the secret-key of the prover. To prove that f is satisfiable, the prover samples the proof (I, π) of the prover in the hidden-bits model, and appends the openings that correspond to indices in I, i.e., it sets its own proof to be $(I, \pi, (o_i)_{i \in I})$. Finally, the verifier simply verifies that the openings are valid, and that the hidden-bits verifier is accepting the proof. We continue with a formal description of the protocol.

DP-NIZK Π_{statPoK}

- **Primitives.**
  - A statistically-binding non-interactive commitment scheme $(\mathsf{commit}, \mathsf{open})$.
  - Let $\Pi_{\mathsf{hidBits}} = (\mathsf{Gen}_{\mathsf{hidBits}}, \mathcal{P}_{\mathsf{hidBits}}, \mathcal{V}_{\mathsf{hidBits}})$ be the NIZK protocol in the hidden-bits model of [FLS90] (see Section F.1).
- **Set-up algorithm $\mathsf{Gen}(1^\kappa)$.** On input 1^κ the algorithm Gen does as follows.
  1. Samples pp for the non-interactive commitment scheme.
  2. Samples a random string $\mathbf{r} \leftarrow \mathsf{Gen}_{\mathsf{hidBits}}(1^\kappa)$. We denote $\mathbf{r} = (r_1, \ldots, r_m)$.
  3. For every $i \in [m]$ commits $(C_i, o_i) \leftarrow \mathsf{commit}(r_i)$.
  4. Sets $\mathsf{crs} := (\mathsf{pp}, C_1, \ldots, C_m)$ and $\mathsf{sk}_{\mathcal{P}} := (o_1, \ldots, o_m)$.
  5. Outputs $(\mathsf{crs}, \mathsf{sk}_{\mathcal{P}})$.
- **Prover $\mathcal{P}(1^\kappa, \mathsf{crs}, \mathsf{sk}_{\mathcal{P}}, f, \mathbf{x})$.** The prover on input 1^κ, CRS $\mathsf{crs} = (\mathsf{pp}, C_1, \ldots, C_m)$, secret ket $\mathsf{sk}_{\mathcal{P}} = (o_1, \ldots, o_m)$, a circuit-SAT instance f with description of length κ, and a satisfying assignment $\mathbf{x}$ does as follows.
  1. The prover computes $r_i := \mathsf{open}(C_i, o_i)$ for every $i \in [m]$, and sets $\mathbf{r} := (r_1, \ldots, r_m)$.
  2. The prover samples $(I, \pi_{\mathsf{hidBits}}) \leftarrow \mathcal{P}_{\mathsf{hidBits}}(f, \mathbf{x}, \mathbf{r})$.
  3. The prover outputs $\pi := (I, \pi_{\mathsf{hidBits}}, (o_i)_{i \in I})$.
- **Verifier $\mathcal{V}(1^\kappa, \mathsf{crs}, f, \pi)$.** The verifier on input 1^κ, CRS $\mathsf{crs} = (\mathsf{pp}, C_1, \ldots, C_m)$, a circuit-SAT instance f with description of length κ, and a proof $\pi = (I, \pi_{\mathsf{hidBits}}, (o'_i)_{i \in I})$ does as follows.
  1. For every $i \in I$, the verifiers computes $r'_i := \mathsf{open}(C_i, o'_i)$, and verifies that $r'_i \neq \perp$. If the verification fails then the verifier outputs 0. Otherwise, the verifier sets $\mathbf{r}'_I := (r'_i)_{i \in I}$.
  2. The verifier computes the bit $b := \mathcal{V}_{\mathsf{hidBits}}(f, I, \mathbf{r}'_I, \pi_{\mathsf{hidBits}})$ and outputs b.

Figure 5: DP-NIZK Π_{statPoK}

Theorem 5.6. *Assuming the existence of one-way functions, Π_{statPoK} is a DP-NIZK with statistical zero knowledge.*

The proof follows the same lines as the transformation from the construction of [FLS90] in the hidden-bits model to adaptive-NIZK, as described in [Gol01, Chapter 4.10.3.2] (see also [Gol04, Chapter C.4.3]). A full proof appears in Section E.1.

5.2 Single Input Functionality

In Section 5.2 and Section 5.3, the distinction between clients and servers will not be useful, and therefore we simply use "parties". In addition, we use the standard security notions of MPC, instead of the definitions from Section 2 that were tailored for the construction of NPSS. In more details, we consider secure computation with n parties, denoted $P_1, \ldots, P_n$, and assume that the adversary corrupts a minority t of the parties, i.e., $n \geq 2t + 1$. We denote by C the set of corrupt parties, and by H the set of honest parties. Throughout, we assume that the parties can communicate via secure point-to-point channels, and that they also have access to a broadcast channel. We consider a static, active rushing adversary and require full security, including guaranteed output delivery. In fact, all the results in this section are proved to be UC-secure in the plain model, i.e., they require no trusted setup. We also mention that some of our protocols achieve a strong notion of security called *everlasting security* [MU10] where the protocol is secure against an adversary that is computationally bounded during the execution, but becomes computationally unbounded afterward. For more information about the framework of universal composability, and about everlasting security, the reader is referred to Section F.2. We also refer the reader to [AKP22b] for further discussion about everlasting security in our context.

A *single input functionality* (SIF) is a functionality that takes an input from a single party, and returns the outputs to all the parties. In Section 5.2.1 we present a 2-round offline/online construction of *public SIF*, where the same output is given to all the parties. Then, in Section 5.2.2 we explain how to transform public SIF into general SIF.

5.2.1 Public Single Input Functionality

A public single input functionality is a single input functionality where all the parties receive the same output. This is formalized in the following functionality.

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

There is a distinguished party D. The functionality is parameterized by a circuit f on k input bits and ℓ output bits.

- **Inputs.** The dealer inputs $\mathbf{x} \in \{0,1\}^k$.
- **Outputs.** The functionality computes $\mathbf{y} := f(\mathbf{x})$ and returns $\mathbf{y}$ to all the parties.

Figure 6: Functionality $\mathcal{F}_{\mathsf{psif}}$

Throughout, we assume that D is one of the n parties $P_1, \ldots, P_n$, say $D = P_1$. Our goal is to construct a 2-round offline/online protocol for $\mathcal{F}_{\mathsf{psif}}$. We begin with a description of a simple 3-round protocol.

Basic protocol. The main idea is to let D use a $(t+1)$-out-of-n NPSS $(R, W, \mathsf{Sim}, \mathsf{Ext})$ to share the $\mathbf{NP}$ statement "There is $\mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$" among the parties, and let the i-th party verify that the i-th share is valid. Formally, we define the function $h_{\mathbf{y}}(\mathbf{x})$ that returns 1 if $f(\mathbf{x}) = \mathbf{y}$ and 0 otherwise. We let D broadcast $\mathbf{y}$, and sample a global assignment $\mathbf{z}$ as well as partial assignments $\mathbf{z}_1, \ldots, \mathbf{z}_n$ from $W(h_{\mathbf{y}}, \mathbf{x})$. D now commits to $\mathbf{z}$ and sends to the i-th party P_i openings to the partial assignment $\mathbf{z}_i$. In the second round, every P_i can locally compute $(\hat{h}_1, \ldots, \hat{h}_n) = R(h_{\mathbf{y}})$ and verifies that the committed partial assignment $\mathbf{z}_i$ satisfies $\hat{h}_i$. If the verification fails then P_i raises a public complaint. Finally, in the third round D opens the commitments that correspond to the shares $\mathbf{z}_i$ of every complaining party P_i so that the parties could verify that $\hat{h}(\mathbf{z}_i) = 1$. The parties accept $\mathbf{y}$ only if the verification succeeds for every P_i that raised a complaint.

We observe that an honest P_i always accepts the shares of an honest D, and therefore it provides both *completeness* and *privacy* for the partial assignment $\mathbf{z}_i$. In addition, it provides *soundness* for a corrupt D, i.e., we are guaranteed that $\hat{h}_i(\mathbf{z}_i) = 1$. We also notice that *completeness* is guaranteed even when P_i is corrupt, as an honest D always opens the correct shares for every complaint.

Two-round protocol. To reduce the number of rounds, we note that instead of letting every P_i verify the i-th statement, we can simply let every P_i act as the trusted party in a DP-NIZK instance where D proves that the committed $\mathbf{z}_i$ satisfies $\hat{h}_i$. For an honest P_i the protocol provides completeness, soundness and privacy, just like in the basic protocol. To make sure that the protocol provides completeness even when P_i is corrupted by the adversary, we simply let D verify that the i-th proof will be accepted, and publicly send the openings to the witness $\mathbf{z}_i$ if not. The perfect completeness of the DP-NIZK now guarantees that $\mathbf{z}_i$ will be publicly revealed only if P_i is corrupt. In this sense, our protocol can be seen as a transformation from DP-NIZK into "multi-string DP-NIZK", similar to the NIZK–to–MS-NIZK transformation in Theorem 4.8.

When we instantiate the protocol with statistically-binding commitments (that can be based on one-way functions), and with Π_{statPoK}, we obtain a protocol with *statistical soundness*.[17] If we instantiate the protocol with statistically-hiding commitments (that can be based on collision-resistance hash functions) and with Π_{pzk}, we obtain a protocol with *everlasting security*. In both cases, the protocol has *perfect completeness*, i.e., when the dealer plays honestly with input $\mathbf{x}$ the output is $f(\mathbf{x})$ with probability 1. We continue with a formal description of the protocol.

Protocol psif

There is a distinguished party D. The functionality is parameterized by a circuit f on k input bits and ℓ

[17]That is, even when the dealer is computationally unbounded, with all but negligible probability the output of the protocol is in the image of f.

output bits. We denote the security parameter by κ.

- **Primitives.**
  - An MS-NICOM $(\mathsf{commit}, \mathsf{open})$ (see further details in Section 4.1.1).
  - A $(t+1)$-out-of-n NPSS $(R, W, \mathsf{Sim}, \mathsf{Dec})$ from Corollary 3.5.
  - For every $\mathbf{y} \in \{0,1\}^\ell$ we define a circuit $h_\mathbf{y}$ that takes $\mathbf{x} \in \{0,1\}^k$ and returns 1 if $f(\mathbf{x}) = \mathbf{y}$.
  - For a circuit $\hat{h}$ on m input bits and a single output bit, for public parameters pp and for a set of commitments $\mathbf{C} = (C_i)_{i\in[m]}$, we define by $G_{\hat{h},\mathsf{pp},\mathbf{C}}$ the circuit that takes as an input a set $S \subseteq [m]$ and openings $(o_i)_{i\in S}$, and does as follows. For every $i \in S$ the circuit $G_{\hat{h},\mathsf{pp},\mathbf{C}}$ computes $\alpha_i = \mathsf{open}_\mathsf{pp}(C_i, o_i)$ and for every $i \notin S$ the circuit sets $\alpha_i = *$. The circuit outputs a bit b that is set to 1 if both (1) $\alpha_i \neq \bot$ for every $i \in S$, and (2) the partial assignment $(\alpha_1, \ldots, \alpha_m)$ satisfies $\hat{h}$. Otherwise, b is set to be 0.

    In the protocol, pp will be the public parameters generated by the non-interactive initialization of the commitment scheme, and the function $\hat{h}$ will be one of the functions $(\hat{h}_1, \ldots, \hat{h}_n) = R(h_\mathbf{y})$ for some $\mathbf{y} \in \{0,1\}^\ell$. We denote by B an upper bound on the length of the binary representation of $G_{\hat{h},\mathsf{pp},\mathbf{C}}$ and the number of inputs m. We observe that B is polynomial in the number of parties n, the circuit size of f, and the security parameter κ, and can be computed efficiently given n, the circuit size of f and the complexity of R.
  - A DP-NIZK $\Pi = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$.
  - We set $\kappa' := \max(\kappa, B)$, and we assume that all cryptographic primitives are executed with security parameter κ'.
- **Offline round.** In the offline round, every P_i samples $(\mathsf{crs}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^{\kappa'})$ and CRS $\mathsf{pp}_i \leftarrow \{0,1\}^\kappa$ for the NICOM, broadcasts $(\mathsf{crs}_i, \mathsf{pp}_i)$ and sends sk_i to D. We let $\mathsf{pp} = (\mathsf{pp}_i)_{i\in[n]}$ and write $\mathsf{commit} = \mathsf{commit}_\mathsf{pp}$ and $\mathsf{open} = \mathsf{open}_\mathsf{pp}$.
- **Inputs.** D receives the input $\mathbf{x} \in \{0,1\}^k$.
- **Online round.** D does as follows.
  1. Computes $\mathbf{y} := f(\mathbf{x})$ and broadcasts $\mathbf{y}$.
  2. Computes $(\hat{h}_1, \ldots, \hat{h}_n) = R(h_\mathbf{y})$.
  3. Samples $(\mathbf{z}, \mathbf{z}_1, \ldots, \mathbf{z}_n) \leftarrow W(h_\mathbf{y}, \mathbf{x})$. Let us denote $\mathbf{z} = (\alpha_1, \ldots, \alpha_m)$. In addition, for every $i \in [n]$ we denote by S_i the set of all indices $j \in [m]$ such that $\mathbf{z}_i[j] \neq *$.
  4. Samples $(C_i, o_i) \leftarrow \mathsf{commit}(\alpha_i)$ for every $i \in [m]$ and broadcasts $\mathbf{C} := (C_i)_{i\in[m]}$.
  5. For every $i \in [n]$, the dealer samples $\pi_i \leftarrow \mathcal{P}(1^{\kappa'}, \mathsf{crs}_i, \mathsf{sk}_i, G_{\hat{h}_i,\mathsf{pp},\mathbf{C}}, (S_i, (o_j)_{j\in S_i}))$. The dealer verifies that $\mathcal{V}(1^{\kappa'}, \mathsf{crs}_i, G_{\hat{h}_i,\mathsf{pp},\mathbf{C}}, \pi_i) = 1$. If the verification succeeds the dealer broadcasts π_i, and otherwise the dealer broadcasts $(\mathsf{fail}, S_i, (o_j)_{j\in S_i})$.
- **Local computation.** Given the broadcasts $\mathbf{y}$ and $\mathbf{C}$ of D, every party P computes $(\hat{h}_1, \ldots, \hat{h}_n) = R(h_\mathbf{y})$, and the functions $G_{\hat{h}_i,\mathsf{pp},\mathbf{C}}$. For every $i \in [n]$, if D broadcasts $(\mathsf{fail}, S_i, (o_j)_{j\in S_i})$ then P verifies that $G_{\hat{h}_i,\mathsf{pp},\mathbf{C}}(S_i, (o_j)_{j\in S_i}) = 1$, and if D broadcast π_i then P verifies that $\mathcal{V}(1^{\kappa'}, \mathsf{crs}_i, G_{\hat{h}_i,\mathsf{pp},\mathbf{C}}, \pi_i) = 1$. If the verification succeeds for every $i \in [n]$ then P outputs $\mathbf{y}$, and otherwise P outputs $f(0, \ldots, 0)$.

Figure 7: Protocol psif

Theorem 5.7. *Let κ be a security parameter, let n be the number of parties, and let $t < n/2$ be the number of corrupt parties. We obtain the following results:*

- (Statistical soundness from OWFs) *Assuming the existence of one-way functions, protocol* psif*, when instantiated with statistically-binding MS-NICOM (that are implied by one-way functions) and* $\Pi = \Pi_{\mathsf{statPoK}}$ *(Protocol 5), is a UC-secure implementation of* $\mathcal{F}_{\mathsf{psif}}$*, against a static, active, rushing adversary that corrupts at most* t *of the parties. In addition, the protocol provides perfect completeness and statistical soundness.*

- (Everlasting security from CRH) *Assuming the existence of collision-resistant hash functions, protocol* psif*, when statistically-hiding commitments (that are implied by collision-resistance hash functions) and* $\Pi = \Pi_{\mathsf{pzk}}$ *(Protocol 4) is a UC-secure implementation with everlasting security of* $\mathcal{F}_{\mathsf{psif}}$*, against a static, active, rushing adversary that corrupts at most* t *of the parties. In addition, the protocol provides perfect completeness.*

In all cases, the complexity of the protocol is $\mathrm{poly}(\kappa, n, s)$*, where* s *is the circuit size of* f*. In addition, in the offline phase every party communicates by sending a private message to* D *and sending a public broadcast message, and in the online phase only* D *communicates by sending a public broadcast message, and the decision whether to accept or reject depends only on the broadcast messages.*

Proof sketch. The proof follows the same lines as the proof of Theorem 4.8 and its extensions, and we therefore only provide a proof sketch. Denote by $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ the simulators of Π, and by $(\mathsf{Gen}', \mathsf{Ext})$ the set-up generator and extractor promised by the proof of knowledge of Π. Recall that Sim and Dec are the simulator and decoder of the NPSS. We split into cases.

Honest D**.** In the offline phase, the simulator performs the non-interactive initialization, by sampling public parameters pp_i on behalf of each honest P_i and broadcasting pp_i. In addition, for every honest P_i, the simulator samples $(\mathsf{crs}_i, \tau_i) \leftarrow \mathsf{Sim}_1(1^{\kappa'})$, and broadcasts crs_i on behalf of P_i. At this stage the simulator receives the broadcast messages $(\mathsf{pp}_i, \mathsf{crs}_i)$ and the private message sk_i of every corrupt P_i. This concludes the simulation of the offline round.

In the online round the simulator receives the output $\mathbf{y}$. The simulator computes $(\hat{h}_1, \dots, \hat{h}_n) = R(h_{\mathbf{y}})$ and also executes the NPSS simulator $\mathsf{Sim}_{\mathsf{NPSS}}$ on $h_{\mathbf{y}}$ and C to obtain the (at most t) partial assignments $(\mathbf{z}_i)_{i \in \mathsf{C}}$. For every $i \in \mathsf{C}$ the simulator computes the set S_i to be the set of all indices $j \in [m]$ such that $\mathbf{z}_i[j] \neq *$. In addition, for every $j \in [m]$, if there is some $i \in \mathsf{C}$ such that $j \in S_i$, the simulator commits to $(C_i, o_i) \leftarrow \mathsf{commit}(\mathbf{z}_i[j])$, and otherwise the simulator commits to $(C_i, o_i) \leftarrow \mathsf{commit}(0)$. The simulator broadcasts $\mathbf{y}$ and $\mathbf{C} := (C_1, \dots, C_m)$ on behalf of D.

In addition, for every $i \in \mathsf{C}$, the simulator samples $\pi \leftarrow \mathcal{P}(1^{\kappa'}, \mathsf{crs}_i, \mathsf{sk}_i, G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}}, (S_i, (o_j)_{j \in S_i}))$. If $\mathcal{V}(1^{\kappa'}, \mathsf{crs}_i, G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}}, \pi_i) = 0$ then the simulator broadcasts $(\mathsf{fail}, S_i, (o_j)_{j \in S_i})$ on behalf of D. Otherwise, the simulator broadcasts π_i on behalf of D. Finally, for every $i \in \mathsf{H}$ the simulator samples $\pi_i \leftarrow \mathsf{Sim}_2(1^{\kappa'}, \mathsf{crs}_i, \tau_i, G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}})$ and broadcasts π_i on behalf of D. This concludes the simulation.

It is not hard to verify that the real-world output is $\mathbf{y}$ with probability 1. In addition, a standard argument shows that the simulation is statistically-close to the real world if the underlying commitment scheme is statistically-hiding and $\Pi = \Pi_{\mathsf{pzk}}$, and the simulation is computationally-indistinguishable from the real world if the underlying commitment scheme is computationally hiding and $\Pi = \Pi_{\mathsf{statPoK}}$.

Corrupt D**.** In the offline phase, the simulator performs the non-interactive initialization, by sampling public parameters pp_i on behalf of each honest P_i and broadcasting pp_i. In addition, for every honest P_i, the simulator samples $(\mathsf{crs}_i, \mathsf{sk}_i, \tau_i) \leftarrow \mathsf{Gen}'(1^{\kappa'})$, broadcasts crs_i and sends sk_i to

D on behalf of P_i. The simulator then receives the messages of the corrupt parties in the offline round, and the messages of the corrupt dealer in the online round. Therefore, the offline-round messages and the online round messages have the same distribution as in a real-world execution of the protocol. It remains to explain how to extract the input of the corrupt dealer.

In the online phase the simulator receives the broadcasts of the corrupt dealer: $\mathbf{y}$, $\mathbf{C} := (C_1, \ldots, C_m)$, and for every $i \in [n]$ either π_i or $(\mathsf{fail}, S_i, (o_j)_{j \in S_i})$. The simulator computes $(\hat{h}_1, \ldots, \hat{h}_n) = R(h_{\mathbf{y}})$. For every $i \in [n]$, if the dealer broadcasts $(\mathsf{fail}, S_i, (o_j)_{j \in S_i})$ then the simulator verifies that $G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}}(S_i, (o_j)_{j \in S_i}) = 1$, and if the dealer broadcasts π_i then the simulator verifies that $\mathcal{V}(1^{\kappa'}, \mathsf{crs}_i, G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}}, \pi_i) = 1$. If the verification fails the simulator inputs $(0, \ldots, 0)$ to $\mathcal{F}_{\mathsf{psif}}$ and terminates.

Otherwise, for every $i \in \mathsf{H}$ the simulator defines the partial assignment $\mathbf{z}_i$ in the following way: (1) if the dealer broadcasts $(\mathsf{fail}, S_i, (o_j)_{j \in S_i})$ then for every $j \in S_i$ the simulator sets $\mathbf{z}_i[j] := \mathsf{open}(C_j, o_j)$, and for every $j \notin S_i$ the simulator sets $\mathbf{z}_i[j] := *$, and (2) if the dealer broadcasts π_i, the simulator computes $(S_i, (o_j)_{j \in S_i}) = \mathsf{Ext}(1^{\kappa'}, \mathsf{crs}_i, \tau_i, G_{\hat{h}_i, \mathsf{pp}, \mathbf{C}}, \pi_i)$, and defines $\mathbf{z}_i$ like in the first case. The simulator executes the NPSS decoder Dec on $h_{\mathbf{y}}$, the set H, and the partial shares $(\mathbf{z}_i)_{i \in \mathsf{H}}$ to obtain an assignment $\mathbf{x}$, and inputs $\mathbf{x}$ to $\mathcal{F}_{\mathsf{psif}}$. This concludes the simulation.

We continue with a short analysis. Assume that the local verification of the parties succeeds and the output is set to be $\mathbf{y}$. Observe that if the adversary does not violate the binding property of the commitment scheme, and the extraction of the DP-NIZK, then the partial assignments of the $t+1$ honest parties $(\mathbf{z}_i)_{i \in \mathsf{H}}$ are pairwise consistent on Boolean values, and $\mathbf{z}_i$ satisfies $\hat{h}_i$. Therefore, the soundness of the NPSS scheme guarantees that $h_{\mathbf{y}}(\mathbf{x}) = 1$ for $\mathbf{x} = \mathsf{Dec}(h_{\mathbf{y}}, \mathsf{H}, (\mathbf{z}_i)_{i \in \mathsf{H}})$, i.e., $h(\mathbf{x}) = \mathbf{y}$. Furthermore, if the underlying commitment scheme is statistically-binding, and $\Pi = \Pi_{\mathsf{statPoK}}$ then we obtain soundness even when D is computationally unbounded. This concludes the proof sketch. $\square$

5.2.2 Single Input Functionality

First, we give a formal definition of single input functionalities.

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

There is a distinguished party D. The functionality is parameterized by n circuits $f_1, \ldots, f_n$, where f_i has k input bits and ℓ_i output bits. The functionality receives the set of corrupt parties C.

- **Inputs.** The dealer D inputs $\mathbf{x} \in \{0,1\}^k$.
- **Outputs.** The functionality computes $\mathbf{y}_i := f_i(\mathbf{x})$ for every $i \in [n]$ and returns $\mathbf{y}_i$ to P_i.

Figure 8: Functionality $\mathcal{F}_{\mathsf{sif}}$

From public SIF to SIF. The work of [AKP22b] presented a round-preserving transformation from public SIF to SIF, based only on non-interactive commitments. However, for non-interactive commitments that require public parameters (such as Naor's commitments or CRH-based commitments), their construction required a trusted set-up: a CRS that picks the public parameters. The reason their protocol requires a CRS is that it uses the commitment scheme already in the first (offline) round, so the public parameters need to be available at the beginning of the protocol. We show that a simple modification of their protocol allows us to obtain a protocol in the plain model, without any trusted set-up, from any non-interactive commitment scheme.

To do so, we first present a simplification of their protocol, where we still assume that the public parameters are part of the CRS. In the offline round, let each each party P_i broadcast a public commitment C_i to a one-time pad $\mathbf{r}_i$, and send the corresponding opening o_i to D. In the online phase, let D compute the public single input functionality, that takes as input $\mathbf{x}$, as well as all the commitments and openings. For every P_i the functionality recovers $\mathbf{r}_i = \mathsf{open}(C_i, o_i)$, computes the output $\mathbf{y}_i := f_i(\mathbf{x})$ and publicly outputs $(C_i, \mathbf{y}_i + \mathbf{r}_i)$. If the opening of C_i fails, then the functionality publicly outputs $(C_i, \mathbf{y}_i, \mathsf{fail})$. The parties verify that D used the correct commitments, and therefore the binding property of the commitments implies that a corrupt D cannot violate the soundness. In addition, for an honest D the outputs of all honest parties are encrypted with a one-time pad, and therefore the adversary learns no information about them. It is not hard to verify that when the underlying commitment scheme is statistically hiding the transformation preserves everlasting security, and when the underlying commitment scheme is statistically binding the transformation preserves statistical soundness.[18]

To get rid of the trusted set-up, we observe that we only care that a corrupt D cannot violate the binding property of the commitments that belong to honest parties.[19] Therefore, we can simply let P_i pick the public parameters that are used for the commitment C_i of $\mathbf{r}_i$. This guarantees that for every honest P_i the binding property holds. Full details of our protocol sif are postponed to Section E.2. We conclude with the following theorem, whose proof also appears in Section E.2.

Theorem 5.8. *Let κ be a security parameter, let n be the number of parties, and let $t < n/2$ be the number of corrupt parties. We obtain the following results:*

- (Statistical soundness from OWFs) *Assuming the existence of one-way functions, there exists a 2-round offline/online protocol* sif *that is a UC-secure implementation of $\mathcal{F}_{\mathsf{sif}}$, against a static, active, rushing adversary that corrupts at most t of the parties. In addition, the protocol provides perfect completeness and statistical soundness.*

- (Everlasting security from CRH) *Assuming the existence of collision-resistant hash functions, there exists a 2-round offline/online protocol* sif *that is a UC-secure implementation with everlasting security of $\mathcal{F}_{\mathsf{sif}}$, against a static, active, rushing adversary that corrupts at most t of the parties. In addition, the protocol provides perfect completeness.*

In all cases, the complexity of the protocol is $\mathrm{poly}(\kappa, n, s_1, \ldots, s_n)$, where s_i is the circuit size of f_i. In addition, in the offline phase every party communicates by sending a private message to D and sending a public broadcast message, and in the online phase only D communicates by sending a public broadcast message.

5.3 Secure Multiparty Computation

We continue with our results for round-optimal honest-majority general computation. We consider a functionality $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ that takes an input $\mathbf{x}_i$ from each P_i, computes the outputs

[18]That is, even when the dealer is computationally unbounded, with all but negligible probability there exists some input $\mathbf{x}$ such that the output of the honest parties is consistent with $\mathbf{x}$, i.e., $\mathbf{y}_i = f_i(\mathbf{x})$ for all $i \in \mathsf{H}$.

[19]More formally, we need a weak binding property: an *external party* who is given an honestly-generated commitment and its opening, should not be able to de-commit to a different value.

$(\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and returns $\mathbf{y}_i$ to P_i. Our starting point is the transformation of [AKP22a] from a 2-round offline/online SIF protocol to a 3-round general computation protocol. Like in the transformation of [AKP22b] from public SIF to SIF, the transformation of [AKP22a] requires a CRS that picks the public parameters of the non-interactive commitment scheme, due to the use of commitments in the first round of the protocol. In fact, just like in [AKP22a], the first-round commitments that an honest P_i generates are only used to guarantee that *other parties* will not violate the binding property, while for a corrupt P_i we have no requirements.[20] Therefore, the public parameters for P_i's commitments can be picked by P_i, and the CRS is eliminated in the same way as in Section 5.2.2.

We mention that when the transformation is instantiated with statistically-binding commitments, and when the underlying SIF protocol provides statistical soundness, then the general MPC protocol provides *statistical correctness*, i.e., for every inputs $(\mathbf{x}_i)_{i\in\mathsf{H}}$ to the honest parties, and for every computationally-unbounded adversary that corrupts a minority of the parties, with all but negligible probability the output is $\mathcal{F}(\mathbf{x}'_1, \ldots, \mathbf{x}'_n)$ where $\mathbf{x}'_i = \mathbf{x}_i$ for all $i \in \mathsf{H}$. Similarly, when the transformation is instantiated with statistically-hiding commitments, and when the underlying SIF protocol provides everlasting security, then the general MPC protocol provides everlasting security. We omit the full details of the transformation from this version of the paper, and conclude with the following theorem.

Theorem 5.9. *Let κ be a security parameter, let n be the number of parties, and let $t < n/2$ be the number of corrupt parties. Let $\mathcal{F}$ be an n-party functionality. We obtain the following results:*

- (Statistical correctness from OWFs) *Assuming the existence of one-way functions, there exists a 3-round protocol* mpc *that is a UC-secure implementation of $\mathcal{F}$, against a static, active, rushing adversary that corrupts at most t of the parties. In addition, the protocol provides statistical correctness.*

- (Everlasting security from CRH) *Assuming the existence of collision-resistant hash functions, and assuming that $\mathcal{F}$ is in $\mathbf{NC}^1$, there exists a 3-round protocol* mpc *that is a UC-secure implementation with everlasting security of $\mathcal{F}$, against a static, active, rushing adversary that corrupts at most t of the parties.*

In all cases, the complexity of the protocol is $\mathrm{poly}(\kappa, n, s)$, where s is the circuit size of $\mathcal{F}$.

[20] In the current description of the protocol of [AKP22a], in the first round every P_i also commits to its private randomness, and in the second round P_i proves in zero-knowledge (using the SIF protocol) that its behavior is consistent with the committed randomness. Since those commitments are only used in the second round, and only by P_i, we can postpone their generation to the second round.

References

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. pages 395–424, 2018.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175, 2004.

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 483–501, 2012.

[AK25] Benny Applebaum and Eliran Kachlon. NIZK amplification via leakage-resilient secure computation, 2025. Unpublished manuscript.

[AKP22a] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Round-optimal honest-majority MPC in minicrypt and with everlasting security - (extended abstract). In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 103–120. Springer, 2022.

[AKP22b] Benny Applebaum, Eliran Kachlon, and Arpita Patra. Verifiable relation sharing and multi-verifier zero-knowledge in two rounds: Trading nizks with honest majority - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 33–56. Springer, 2022.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9, 1983.

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptology*, 30(1):58–151, 2017.

[BCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[BD91] Mike Burmester and Yvo Desmedt. Broadcast interactive proofs (extended abstract). pages 81–95, 1991.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112, 1988.

[BG24] Nir Bitansky and Nathan Geier. Amplification of non-interactive zero knowledge, revisited. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 361–390. Springer, 2024.

[BGT20] Zvika Brakerski, Sanjam Garg, and Rotem Tsabary. FHE-based bootstrapping of designated-prover NIZK. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 657–683. Springer, 2020.

[BJMS20] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 120–150. Springer, 2020.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 60–79. Springer, 2006.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. *IACR Cryptol. ePrint Arch.*, page 258, 2020.

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO '88, 8th Annual International Cryptology Conference*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge, MARK 1979, New York, NY, USA, June 4-7, 1979*, pages 313–318. IEEE, 1979.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology—EUROCRYPT 2003: International*

Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22, pages 614–629. Springer, 2003.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[CCH$^+$19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37*, pages 91–122. Springer, 2018.

[CDI$^+$13] Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 185–202. Springer, 2013.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CK89] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 62–72, New York, NY, USA, 1989. Association for Computing Machinery.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 542–552, 1991.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 532–550. Springer, 2014.

[DFK$^+$92] Cynthia Dwork, Uri Feige, Joe Kilian, Moni Naor, and Muli Safra. Low communication 2-prover zero-knowledge proofs for np: Preliminary version. In *Annual International Cryptology Conference*, pages 215–227. Springer, 1992.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DPP98a] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Trans. Inf. Theory*, 44(3):1143–1151, 1998.

[DPP98b] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Trans. Inf. Theory*, 44(3):1143–1151, 1998.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). pages 554–563, 1994.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE, 1990.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984.

[GJS19] Vipul Goyal, Aayush Jain, and Amit Sahai. Simultaneous amplification: The case of non-interactive zero-knowledge. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 608–637. Springer, 2019. Crypto talk avaialable in https://www.youtube.com/watch?v=DLK079q6aWM&ab_channel=IACR.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 63–82, 2015.

[GMR84] S Goldwasser, S Micali, and RL Rivest. A ”paradoxical” solution to the signature problem. In *25th Annual Symposium onFoundations of Computer Science, 1984.*, pages 441–448. IEEE, 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

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

[Gol87] Oded Goldreich. Two remarks concerning the goldwasser-micali-rivest signature scheme. In *Proceedings on Advances in cryptology—CRYPTO'86*, pages 104–110, 1987.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In *2018 IEEE 59th annual symposium on Foundations of Computer Science (FOCS)*, pages 850–858. IEEE, 2018.

[HM96a] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 1996.

[HM96b] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. pages 201–215, 1996.

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptol.*, 13(1):31–60, 2000.

[HN24] Shuichi Hirahara and Mikito Nanashima. One-way functions and zero knowledge. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1731–1738. ACM, 2024.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. pages 244–256, 2002.

[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.

[Ish20] Yuval Ishai. Zero-knowledge proofs from information-theoretic proof systems, 2020. See also the Simons talk `https://simons.berkeley.edu/talks/efficient-zero-knowledge-proofs-modular-approach`.

[Jai24] Aayush Jain. Aayush jain's website, 2024. `https://sites.google.com/view/aayushjain/home`.

[KNY14] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for np. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 254–273. Springer, 2014.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 496–505, 1997.

[KRR17] Yael Tauman Kalai, Guy N Rothblum, and Ron D Rothblum. From obfuscation to the security of fiat-shamir for proofs. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*, pages 224–251. Springer, 2017.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. 1979.

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. New constructions of statistical nizks: Dual-mode dv-nizks and more. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 410–441. Springer, 2020.

[MU10] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. *J. Cryptol.*, 23(4):594–671, 2010.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 128–136. Springer, 1989.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[Nao06] Moni Naor. Secret sharing for access structures beyond P, 2006. Slides: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/minicrypt.html`.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[OW07] Ryan O'Donnell and Karl Wimmer. Approximation by DNF: examples and counterexamples. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 195–206. Springer, 2007.

[Pat90] Michael S Paterson. Improved sorting networks with o (log n) depth. *Algorithmica*, 5(1):75–92, 1990.

[PHGR16] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: nearly practical verifiable computation. *Commun. ACM*, 59(2):103–112, 2016.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, 1990.

[SCG$^+$14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[Sei09] Joel Seiferas. Sorting networks of logarithmic depth, further simplified. *Algorithmica*, 53:374–384, 2009.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SMP87] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer, 1987.

[Spi71] Philip Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971.

[SV97] Amit Sahai and Salil P. Vadhan. A complete promise problem for statistical zero-knowledge. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 448–457. IEEE Computer Society, 1997.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484, 2014.

[Weg83] Ingo Wegener. Relating monotone formula size and monotone depth of boolean functions. *Inf. Process. Lett.*, 16(1):41–42, 1983.

A Missing Proofs: MPC

A.1 Proof of Claim 2.4

Passive security. We begin by proving that $\sigma(\pi)$ computes $\mathcal{F}$ with χ_σ-passive security. Perfect correctness follows immediately from the perfect correctness of π. For perfect privacy, we note that for every set $Z' \subseteq \mathcal{C}$ that satisfies $\chi_\sigma(Z') = 0$, and for every choice of inputs, the view of Z' in an execution of π' is the same as the view of $Z := Z' \cup \sigma^{-1}(Z')$ in an execution of π, with the same inputs to the clients. Since $0 = \chi_\sigma(Z') = \chi(Z' \cup \sigma^{-1}(Z')) = \chi(Z)$, we can simulate the views by executing Sim on Z together with the inputs and outputs of the clients in Z'.

Perfect active correctness with abort. We continue by proving that $\sigma(\pi)$ computes $\mathcal{F}$ with χ-perfect active correctness with abort. Fix any set $S' \subseteq \mathcal{C}$ that satisfies $\chi_\sigma(S') = 1$, and let $Z' := \mathcal{C} \backslash S'$. Let $S := S' \cup \sigma^{-1}(S')$ and $Z := Z' \cup \sigma^{-1}(Z')$, and observe that $\chi(S) = 1$ and that $Z = \mathcal{P} \setminus S$. We note that for every active adversarial behavior $\tilde{\pi}'$ against π' with respect to Z' there is an active adversarial behavior $\tilde{\pi}$ against π with respect to the set of parties Z, such that the distribution of the variables the belong to honest parties in $\tilde{\pi}'$ is the same as the distribution of the variables the belong to honest parties in $\tilde{\pi}$. Indeed, in $\tilde{\pi}$ every $\mathsf{transmit}$ or func statement where the sender is corrupt and the receiver is honest is marked in the same way as in $\tilde{\pi}'$. Therefore, given Z' and the views of the honest parties in $\tilde{\pi}'$, denoted $(\mathsf{view}'(p))_{p \in S'}$, we can locally compute the corresponding views $(\mathsf{view}(p))_{p \in S}$ of the honest parties in $\tilde{\pi}$ and execute Ext on Z and $(\mathsf{view}(p))_{p \in S}$ to extract the inputs of all the clients. This concludes the proof of the claim. □

A.2 Proof of Claim 2.8

We split into case.

Case 1. Assume that g_i is an OR gate. Then for every set $S \subseteq \mathcal{C} \cup \mathcal{W}_i$,

$$\begin{aligned}
\chi_i(S) &= \begin{cases} \chi_{i-1}(S) & \text{if } w_A, w_B \notin S, \\ \chi_{i-1}((S \setminus \{w_A, w_B\}) \cup \{w\}), & \text{otherwise,} \end{cases} \\
&= \begin{cases} \chi'_{i-1}(S \cap \mathcal{W}_{i-1}) & \text{if } w_A, w_B \notin S, \\ \chi'_{i-1}(((S \setminus \{w_A, w_B\}) \cup \{w\}) \cap \mathcal{W}_{i-1}), & \text{otherwise,} \end{cases} \\
&= \begin{cases} \chi'_{i-1}(S \cap \mathcal{W}_{i-1}) & \text{if } w_A, w_B \notin S, \\ \chi'_{i-1}((S \cap \mathcal{W}_{i-1}) \cup \{w\}), & \text{otherwise,} \end{cases} \\
&= \chi'_i(S \cap \mathcal{W}_i),
\end{aligned}$$

where the first equality follows from Lemma 2.5, and the second equality follows by the induction hypothesis. The last equality follows since the gate g_i is an OR gate, so the value of the wire w is 1 if and only if the value of w_A is 1 or the value of w_B is one (or both). In other words, for every $S' \subseteq \mathcal{W}_i$ it holds that $\chi'_i(S') = \chi'_{i-1}(S') = \chi'_{i-1}(S \cap \mathcal{W}_{i-1})$ if $w_A, w_B \notin S'$, and $\chi'_i(S') = \chi'_{i-1}((S' \setminus \{w_A, w_B\}) \cup \{w\}) = \chi'_{i-1}((S \cap \mathcal{W}_{i-1}) \cup \{w\})$ otherwise. This concludes the analysis of the first case.

Case 2. Assume that g_i is an AND gate. Then for every set $S \subseteq \mathcal{C} \cup \mathcal{W}_i$,

$$\begin{aligned}
\chi_i(S) &= \begin{cases} \chi_{i-1}((S \setminus \{w_A, w_B\}) \cup \{w\}) & \text{if } w_A, w_B \in S, \\ \chi_{i-1}(S \setminus \{w_A, w_B\}), & \text{otherwise,} \end{cases} \\
&= \begin{cases} \chi'_{i-1}(((S \setminus \{w_A, w_B\}) \cup \{w\}) \cap \mathcal{W}_{i-1}) & \text{if } w_A, w_B \in S, \\ \chi'_{i-1}((S \setminus \{w_A, w_B\}) \cap \mathcal{W}_{i-1}), & \text{otherwise,} \end{cases} \\
&= \begin{cases} \chi'_{i-1}((S \cap \mathcal{W}_{i-1}) \cup \{w\}) & \text{if } w_A, w_B \in S, \\ \chi'_{i-1}(S \cap \mathcal{W}_{i-1}), & \text{otherwise,} \end{cases} \\
&= \chi'_i(S \cap \mathcal{W}_i),
\end{aligned}$$

where the first equality follows from Lemma 2.6, and the second equality follows by the induction hypothesis. The last equality follows since the gate g_i is an AND gate, so the value of the wire w is 1 if and only if the value of w_A is 1 and the value of w_B is one. In other words, for every $S' \subseteq \mathcal{W}_i$ it holds that $\chi'_i(S') = \chi'_{i-1}((S' \setminus \{w_A, w_B\}) \cup \{w\}) = \chi'_{i-1}((S' \cap \mathcal{W}_{i-1}) \cup \{w\})$ if $w_A, w_B \in S'$, and $\chi'_i(S') = \chi'_{i-1}(S' \setminus \{w_A, w_B\}) = \chi'_{i-1}(S \cap \mathcal{W}_{i-1})$ otherwise. This concludes the analysis of the second case, and the proof of the lemma. □

B Proof of Lemma 2.5

Let $\mathcal{F}$ be an n-party functionality, let $\pi = (d_1, \ldots, d_L)$ be an n-client m-server protocol among $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ and over $\mathcal{X}$ that computes $\mathcal{F}$ with χ-dual security, and let $\tau \in \mathcal{S}$ be a server. Consider the protocol $\pi' = G_\vee(\pi, \tau)$ among $\mathcal{P}' = (\mathcal{P} \setminus \{\tau\}) \cup \{A, B\}$ and over a variable space $\mathcal{X}'$. Our goal is to prove that π' computes $\mathcal{F}$ with $\chi_\vee$-dual security, where

$$\chi_\vee(S) = \begin{cases} \chi(S), & \text{if } A, B \notin S \\ \chi((S \setminus \{A, B\}) \cup \{\tau\}), & \text{otherwise,} \end{cases}$$

for every set $S \subseteq \mathcal{P}'$. We begin with the following two lemmas, that prove the first part of Lemma 2.5.

Lemma B.1. *Protocol π' computes $\mathcal{F}$ with $\chi_\vee$-passive security.*

Lemma B.2. *Protocol π' computes $\mathcal{F}$ with $\chi_\vee$-perfect active correctness with abort.*

We prove Lemma B.1 in Section B.1 and Lemma B.2 in Section B.2. For the "moreover" part of Lemma 2.5 we first make the following observations. First, we note that in a single server substitution $\pi' = G_\vee(\pi, \tau)$, if a statement d_i of π does not involve τ then it remains unchanged, while if it involves τ then it is replaced with a constant number of statements. Second, we also note that the simulator Sim' that corresponds to π' (see Section B.1 for full details) is *local* in the following sense. The simulator generates the view $\mathcal{V}'$ of the corrupt parties by first executing the original simulator Sim to obtain a view $\mathcal{V}$, and then iterates over the original instructions $d_1, \ldots, d_L$, and for every instruction d_i,

- If d_i does not involve τ then it remains unchanged in π' and the variables that are updated in d_i are assigned the same value in $\mathcal{V}'$ as in $\mathcal{V}$.

- If d_i involves τ then d_i is replaced with a constant number of statements in π'. To simulate the execution of those statements, Sim' performs a constant number of operations that depend only on d_i and the values assigned to variables in d_i according to $\mathcal{V}$.

Parallel applications. Moving to the case of parallel applications, fix a list $(s_1, \ldots, s_k)$ of servers in $\mathcal{S}$, and denote $\pi_0 = \pi$, $\pi_i = G_\vee(\pi_{i-1}, s_i)$ for $i = 1, \ldots, k$, and $\pi'' = \pi_k$. We note that the simulator Sim'' that corresponds to π'' can be described in the following way. The simulator generates the view $\mathcal{V}''$ of the corrupt parties by first executing the original simulator Sim to obtain a view $\mathcal{V}$, and then iterating over the original instructions $d_1, \ldots, d_L$, and for every instruction d_i,

- If d_i involves at most a single party s_j from the list, then Sim'' performs the same operations that Sim' would perform to update the view.
- Otherwise d_i involves two parties s_j and s_ℓ from the list, where $j < \ell$. Let us denote by $(e_1, \ldots, e_q)$ the statements obtained from the compilation of d_i in π_j, and let use denote by $(e_{i,1}, \ldots, e_{i,v_i})$ the statements obtained from the compilation of e_i in π_ℓ.

  The the simulator Sim'' first performs the same operations that Sim' would perform when d_i is transformed into $(e_1, \ldots, e_q)$ to obtain the simulated view of those operations. Given the simulated views of those operations, Sim'' executes Sim' again, with respect to the compilation of the statements $(e_1, \ldots, e_q)$ into $(e_{1,1}, \ldots, e_{q,v_q})$ to obtain the simulated view of those operations, and updates the view $\mathcal{V}''$ accordingly. Since q and $v_1, \ldots, v_q$ are constants, the simulator performs only a constant number of steps.

We conclude that Sim'' performs a constant number of steps for each step of Sim. A similar argument shows that the same is true for the extractor. This concludes the proof of Lemma 2.5. $\square$

B.1 Proof of Lemma B.1

We first prove correctness and then privacy.

B.1.1 Correctness

Fix inputs $(x_1, \ldots, x_n)$ to the clients. Our goal is to prove that in every execution of π' with inputs $(x_1, \ldots, x_n)$, the outputs of the clients is according to $\mathcal{F}(x_1, \ldots, x_n)$. Fix any execution E' of π' with the inputs $x_1, \ldots, x_n$ to the clients.

Claim B.3. *For every $i = 0, \ldots, L$ it holds that after the execution of the compilation of the i-th statement of π in π', there exists an execution E of π on the same inputs such that (1) for every variable $x \in \mathcal{X}$ that does not belong to τ, the value of x in E is the same as the value of $x \in \mathcal{X}'$ in E', and (2) for every variable $x \in \mathcal{X}$ that belongs to τ, the value of x is the same as the values of x^A and x^B in E'.*

The claim follows by an inductive argument, using a simple case analysis according to the i-th statement. We omit the proof of Claim B.3. The claim now implies that the abort flag is never raised in E', and correctness follows because every statement $\mathtt{output}(c, i, x)$ in π' remains the same in π and the value of x in both cases is the same.

B.1.2 Privacy

We define the simulator Sim' for π'. Let Sim be the simulator of π, let us denote $\mathcal{P} = \mathcal{C} \cup \mathcal{S} = \{p_1, \dots, p_{n+m-1}, \tau\}$, and let us assume that the i-th client is p_i, for $i \in [n]$. Let $(Z', (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ be the inputs of Sim', where $\chi_\vee(Z') = 0$. We split into cases.

Case 1. Assume that $A \in Z'$ or $B \in Z'$ (or both). Then $0 = \chi_\vee(Z') = \chi((Z' \setminus \{A, B\}) \cup \{\tau\})$, and we define $Z := (Z' \setminus \{A, B\}) \cup \{\tau\}$. In this case, we let the simulator Sim' simulate the views of the parties in $Z' \cup \{A, B\}$, even if A or B are not in Z' (note that we can always ignore the views of non-corrupted parties). To do so, Sim' first execute $\mathsf{Sim}(Z, (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ to sample the views $\mathcal{V} := (\mathsf{view}(\tau)) \cup (\mathsf{view}(p_i))_{p_i \in Z \setminus \{\tau\}}$.

Let use denote the view of a party p in π' by $\mathsf{view}'(p)$. For every $i = 1, \dots, L$, and for the i-th statement d_i of π, the simulator Sim' updates the views $\mathcal{V}' := (\mathsf{view}'(p))_{p \in Z'} \cup (\mathsf{view}'(A), \mathsf{view}'(B))$ in the following way.

- If d_i does not involve τ, then Sim' assigns the variables that are updated in d_i the same values as in $\mathcal{V}$. If d_i is an abort statement, then Sim' ignores it.
- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$ then Sim' holds $v = \mathsf{val}(x_1) = \mathsf{val}(x_2)$. The simulator updates the views of A and B by setting the values of x_2^A and x_2^B to v. In addition, the simulator simulates the message v that A sends to B, as well as the local computation of B, and updates the view of B accordingly.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ then Sim' holds $v = \mathsf{val}(x_4)$. The simulator updates the views of A and B by setting the values of x_4^A and x_4^B to v. In addition, the simulator simulates the message v that A sends to B, as well as the local computation of B, and updates the view of B accordingly.
- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ (resp., $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$) and p is corrupt, then Sim' holds $v = \mathsf{val}(x_2)$ (resp., $v = \mathsf{val}(x_4)$). The simulator updates the message that p receives from A and B (resp., the output of p in the OLE with A and the OLE with B) to be v, and in addition simulates the local computation of p, and updates the view of p accordingly.
- If d_i is a computation statement $\mathsf{comp}(\tau, \mathsf{op}, (x_1, x_2), x)$ and $\mathsf{op} \in \{+, \times\}$, then Sim' holds $v = \mathsf{val}(x)$ and assigns x^A and x^B the value v, and updates the views of A and B accordingly.
- If d_i is a computation statement $\mathsf{comp}(\tau, \mathsf{rand}, \emptyset, x)$ then Sim' computes $v = \mathsf{val}(x)$ and assigns x^A and x^B the value v, and updates the views of A and B accordingly.

Case 2. Assume that $A, B \notin Z'$, so $\chi_\vee(Z') = \chi(Z') = 0$. We define set $Z := Z'$. In this case the simulator Sim' simulates the views of the parties in Z'. To do so, we first let Sim' execute $\mathsf{Sim}(Z, (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ to sample the views $\mathcal{V} := (\mathsf{view}(p_i))_{p_i \in Z}$.

Let use denote the view of a party p in π' by $\mathsf{view}'(p)$. For every $i = 1, \dots, L$, and for the i-th statement d_i of π, the simulator Sim' updates the views $\mathcal{V}' := (\mathsf{view}'(p))_{p \in Z'}$ in the following way.

- If d_i does not involve τ, then Sim' assigns the variables that are updated in d_i the same values as in $\mathcal{V}$. If d_i is an abort statement, then Sim' ignores it.

- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ (resp., $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$) and p is corrupt, then Sim' holds $v = \mathsf{val}(x_2)$ (resp., $v = \mathsf{val}(x_4)$). The simulator updates the message that p receives from A and B (resp., the output of p in the OLE with A and the OLE with B) to be v, and in addition simulates the local computation of p, and updates the view of p accordingly.

At the end, the simulator outputs $\mathcal{V}'$. This concludes the description of Sim'. Privacy now follows from the following claim.

Claim B.4. *For all inputs $(x_1, \ldots, x_n)$ to the clients, and for every $Z' \subseteq \mathcal{P}'$ such that $\chi_\vee(Z') = 0$, it holds that*

$$(\mathsf{view}'(p))_{p \in Z'} \equiv \mathsf{Sim}'(Z', (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}}),$$

where $(y_1, \ldots, y_n) = \mathcal{F}(x_1, \ldots, x_n)$.

To prove the claim, we note that the simulated views $\mathcal{V}'$ are generated by applying the update procedure of Sim' on the simulated views $\mathcal{V}$ that Sim generates. We also note that if we apply the same update procedure on the real-world views $\mathcal{V}_{\mathsf{Real}}$ in an execution of π, then we obtain the same distribution as the real-world views $\mathcal{V}'_{\mathsf{Real}}$ in an execution of π'. Privacy now follows from the privacy of π, as $\mathcal{V}$ has the same distribution as $\mathcal{V}_{\mathsf{Real}}$, since both in Case 1 and in Case 2 it holds that $\chi(Z) = 0$. We omit a full proof of the claim. This concludes the proof of Lemma B.1. $\square$

B.2 Proof of Lemma B.2

We begin with the definition of the extractor Ext'. Let us denote $\mathcal{P} = \mathcal{C} \cup \mathcal{S} = \{p_1, \ldots, p_{n+m-1}, \tau\}$, and let us assume that the i-th client is p_i, for $i \in [n]$. Fix a set $S' \subseteq \mathcal{P}'$ such that $\chi_\vee(S') = 1$, and let $Z' := \mathcal{P}' \setminus S'$. The extractor takes as input Z' and views $\mathcal{V}' = (\mathsf{view}'(p))_{p \in S'}$ of parties from an execution of an active adversarial behavior $\tilde{\pi}'$ with respect to Z', where the abort flag was not raised. The extractor Ext' generates a set $S \subseteq \mathcal{P}$ with $\chi(S) = 1$, a set $Z := \mathcal{P} \setminus S$ and views $\mathcal{V} = (\mathsf{view}(p))_{p \in S}$, and applies Ext on $(Z, \mathcal{V})$ to obtain the inputs $\mathbf{x}^*$. We split into cases.

Case 1. Assume that $A \in S'$ or $B \in S'$ (or both). Define $S := (S' \setminus \{A, B\}) \cup \{\tau\}$ and $Z := \mathcal{P} \setminus S$, and observe that $1 = \chi_\vee(S') = \chi((S' \setminus \{A, B\}) \cup \{\tau\}) = \chi(S)$. We assume without loss of generality that $A \in S'$ and therefore $A \notin Z'$ (the case where $B \in S'$ is symmetric). The extractor generates views $\mathcal{V} := (\mathsf{view}(p))_{p \in S}$ that correspond to an execution of an active adversarial behavior against π. (We note that $\mathcal{V}$ includes the view of τ.) To do so, for every $i = 1, \ldots, L$, and for the i-th statement d_i of π, the extractor Ext' updates the views $\mathcal{V}$ in the following way.

- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$ let v_2^A be the value of x_2^A according to the view of A in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v_2^A to x_2 in the view of τ.

- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ for $p \notin Z$, let v^A be the message that A sent p according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v^A to x_2 in the view of p.

- If d_i is $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ for $p_2 \notin Z$, let v_2 be the value of x_2 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v_2 to x_2 in the view of p_2.

- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ let v_4^A be the value of x_4^A according to the view of A in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v_4^A to x_4 in the view of τ.

- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$ for $p \notin Z$, let v_4^A be the output of the OLE instance that corresponds to A, according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v_4^A to x_4 in the view of p.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p_1, p_2, (x_1, x_2), (x_3), (x_4))$ for $p_2 \notin Z$, let v_4 be the value of x_4 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v_4 to x_4 in the view of p_2.
- If d_i is $\mathsf{comp}(\tau, \mathsf{op}, X, x)$ for $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$, then let v^A be the value of x^A according to the view of A in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v^A to x in the view of τ.
- If d_i is $\mathsf{comp}(p, \mathsf{op}, X, x)$ for $p \notin Z$ and $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$ then let v be the value of x according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of p.
- d_i is an input statement $\mathsf{input}(c, i, x)$ for $c \notin Z$, then let v be the value of x according to the view of c in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of c.
- Otherwise, do nothing.

Case 2. Otherwise, assume that $A, B \notin S'$. Define $S := S'$ and $Z := \mathcal{P} \backslash S$, and observe that $\tau \in Z$, and that $1 = \chi_\vee(S') = \chi(S)$. The extractor generates views $\mathcal{V} := (\mathsf{view}(p))_{p \in S}$ that correspond to an execution of an adversarial behaviour in π. To do so, for every $i = 1, \ldots, L$, and for the i-th statement d_i of π, the extractor Ext' updates the views $\mathcal{V}$ in the following way.

- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ for $p \notin Z$, let v^A be the message that A sends to p according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v^A to x_2 in the view of p.
- If d_i is $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ for $p_2 \notin Z$, let v be the message that p_1 sends to p_2 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_2 in the view of p_2.
- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$ for $p \notin Z$, let v^A be the output of the OLE instance of p with A according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v^A to x_4 in the view of p.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p_1, p_2, (x_1, x_2), (x_3), (x_4))$ for $p_2 \notin Z$, let v be the value of x_4 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_4 in the view of p_2.
- If d_i is $\mathsf{comp}(p, \mathsf{op}, X, x)$ for $p \notin Z$ and $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$ then let v be the value of x according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of p.
- d_i is an input statement $\mathsf{input}(c, i, x)$ for $c \notin Z$, then let v be the value of x according to the view of c in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of c.
- Otherwise, do nothing.

At the end, Ext' computes $\mathbf{x}^* = \mathsf{Ext}(Z, \mathcal{V})$ and outputs $\mathbf{x}^*$. This concludes the description of Ext'. Perfect correctness with abort now follows from the following claim.

Claim B.5. *For all inputs $(x_1, \ldots, x_n)$ to the clients, for every $S' \subseteq \mathcal{P}'$ that satisfies $\chi_\vee(S') = 1$, for $Z' := \mathcal{P}' \setminus S'$, for every active adversarial behavior $\tilde{\pi}'$ of π' with respect to Z', and for every execution of $\tilde{\pi}'$ in which the abort flag is not raised and the honest parties have view $\mathcal{V}'$ the following holds. There is an input $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ such that (1) $x_i^* = x_i$ for all honest clients $p_i \in \mathcal{C} \setminus Z'$, (2) the outputs of the honest clients is consistent with $\mathcal{F}(\mathbf{x}^*)$, and (3) $\mathsf{Ext}'(Z', \mathcal{V}') = \mathbf{x}^*$.*

To prove the claim, we note that the views $\mathcal{V}$ are generated by applying the update procedure of Ext' on the views $\mathcal{V}'$ from the execution of $\tilde{\pi}'$, and that the clients have the same outputs in $\mathcal{V}$ as in $\mathcal{V}'$. In addition we note that there is a natural adversarial behavior $\tilde{\pi}$ against π with respect to the set Z, where the views $\mathcal{V}$ are in the support. Correctness now follows from the correctness of Ext, since both in Case 1 and in Case 2 it holds that $\chi(S) = 1$. We omit a full proof of the claim. This concludes the proof of Lemma B.2. □

C Proof of Lemma 2.6

Let $\mathcal{F}$ be an n-party functionality, let $\pi = (d_1, \dots, d_L)$ be an n-client m-server protocol among $\mathcal{P} = \mathcal{C} \cup \mathcal{S}$ and over $\mathcal{X}$ that computes $\mathcal{F}$ with χ-dual security, and let $\tau \in \mathcal{S}$ be a server. Consider the protocol $\pi' = G_\wedge(\pi, \tau)$ among $\mathcal{P}' = (\mathcal{P} \setminus \{\tau\}) \cup \{A, B\}$ and over a variable space $\mathcal{X}'$. Our goal is to prove that π' computes $\mathcal{F}$ with $\chi_\wedge$-dual security, where

$$\chi_\wedge(S) = \begin{cases} \chi((S \setminus \{A, B\}) \cup \{\tau\}), & \text{if } A, B \in S \\ \chi(S \setminus \{A, B\}), & \text{otherwise,} \end{cases}$$

We first prove the following two lemmas.

Lemma C.1. *Protocol π' computes $\mathcal{F}$ with $\chi_\wedge$-passive security.*

Lemma C.2. *Protocol π' computes $\mathcal{F}$ with $\chi_\wedge$-perfect active correctness with abort.*

The proof of Lemma C.1 appears in Section C.1, and the proof of Lemma C.2 appears in Section C.2. The "moreover" part of the lemma follows in the same way as in the proof of Lemma 2.5 (see Section B). This concludes the proof of Lemma 2.6. □

C.1 Proof of Lemma C.1

We first prove correctness and then privacy.

C.1.1 Correctness

Fix inputs $(x_1, \dots, x_n)$ to the clients. Our goal is to prove that in every execution of π' with inputs $(x_1, \dots, x_n)$, the outputs of the clients is according to $\mathcal{F}(x_1, \dots, x_n)$. Fix any execution E' of π' with the inputs $x_1, \dots, x_n$ to the clients.

Claim C.3. *For every $i = 0, \dots, L$ it holds that after the execution of the compilation of the i-th statement of π in π', there exists an execution E of π with the same inputs such that (1) for every variable $x \in \mathcal{X}$ that does not belong to τ, the value of x in E is the same as the value of $x \in \mathcal{X}'$ in E', and (2) for every variable $x \in \mathcal{X}$ that belongs to τ, it holds that $\mathsf{val}(x) = \mathsf{val}(x^A) \oplus \mathsf{val}(x^B)$.*

The claim follows by an inductive argument, using a simple case analysis according to the i-th statement. We omit the proof of Claim B.3. The claim now implies that the abort flag is never raised in E', and correctness follows because every statement $\mathsf{output}(c, i, x)$ in π' remains the same in π and the value of x in both cases is the same.

C.1.2 Privacy

We define the simulator Sim' for π'. Let Sim be the simulator of π, let us denote $\mathcal{P} = \mathcal{C} \cup \mathcal{S} = \{p_1, \dots, p_{n+m-1}, \tau\}$, and let us assume that the i-th client is p_i, for $i \in [n]$. let $(Z', (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ be the inputs of Sim'. We split into cases.

Case 1. First, assume that $A, B \in Z'$ and therefore $\chi_\wedge(Z') = \chi((Z' \setminus \{A, B\}) \cup \{\tau\}) = 0$. Let $Z := (Z' \setminus \{A, B\}) \cup \{\tau\}$. We first let Sim' execute $\mathsf{Sim}(Z, (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ to sample the views $\mathcal{V} := (\mathsf{view}(\tau)) \cup (\mathsf{view}(p_i))_{p_i \in Z \setminus \{\tau\}}$. Let use denote the view of a party p in π' by $\mathsf{view}'(p)$. For every $i = 1, \dots, L$, and for the i-th statement d_i of π, the simulator Sim' updates the views $\mathcal{V}' := (\mathsf{view}'(p))_{p \in Z'}$ in the following way.

- If d_i does not involve τ, then Sim' assigns the variables that are updated in d_i the same values as in $\mathcal{V}$.

- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$ then Sim' holds $v = \mathsf{val}(x_1) = \mathsf{val}(x_2)$. The simulator samples a secret sharing of v on behalf of p, by sampling a random bit $v^A \leftarrow \{0, 1\}$ and setting $v^B := v \oplus v^A$. If $p \in Z'$ then the simulator updates the view of p accordingly. In addition, the simulator assigns v^A to x_2^A and v^B to x_2^B, and updates the views of A and B accordingly.

- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ then the simulator holds $v_4 = \mathsf{val}(x_4)$. We split into cases. If $p \in Z'$ then the simulator also holds $v_i := \mathsf{val}(x_i)$ for all $i \in [4]$, as well as $v_3^A := \mathsf{val}(x_3^A)$ and $v_3^B := \mathsf{val}(x_3^B)$. In this case the simulator updates the views of p, A and B according to the compilation of d_i. In more details, it takes the role of p and samples $r \leftarrow \{0, 1\}$, inputs $(v_1, v_2 \oplus r)$ to the first OLE instance and (v_1, r) to the second OLE instance. On behalf of A it inputs v_3^A to the first OLE instance and sets the output to be $v_1 \cdot v_3^A \oplus v_2 \oplus r$. Similarly, on behalf of B it inputs v_3^B to the second OLE instance and sets the output to be $v_1 \cdot v_3^B \oplus r$. The simulator updates the views of p, A and B accordingly.

  Otherwise, $p \notin Z'$. In this case, the simulator samples $r \leftarrow \{0, 1\}$, sets the value of x_4^A to be r and the value of x_4^B to be $r \oplus v_4$. The simulator updates the views of A and B accordingly.

- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$, then the simulator holds v_1^A, v_1^B, that are the values of x_1^A and x_1^B, respectively. The simulator samples $r \leftarrow \{0, 1\}$ on behalf of A and sends r to B. The simulator then sends $y^A := v_1^A \oplus r$ to p on behalf of A, and $y^B := v_1^B \oplus r$ on behalf of B. It also computes $y^A \oplus y^B$ on behalf of p and assigns it to x_2. The simulator updates the views of the corrupt parties in $\{p, A, B\} \cap Z'$ accordingly.

- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$, the Sim holds $v_1^A := \mathsf{val}(x_1^A), v_2^A := \mathsf{val}(x_2^A)$ and $v_1^B := \mathsf{val}(x_1^B), v_2^B := \mathsf{val}(x_2^B)$. We split into cases. If $p \in Z'$, then Sim' also holds $v_3 = \mathsf{val}(x_3)$. In this case the simulator updates the views of p, A and B according to the compilation of d_i. In more details, it takes the role of A and B, samples $r \leftarrow \{0, 1\}$ on behalf of A and sends r to B. It inputs $(v_1^A, v_2^A \oplus r)$ on behalf of A to the first OLE instance, and $(v_1^B, v_2^B \oplus r)$ on behalf of B to the second OLE instance. In addition it inputs v_3 on behalf of p. The outputs of the OLE instances are set to be $y^A := v_1^A \cdot v_3 \oplus v_2^A \oplus r$ and $y^B := v_1^B \cdot v_3 \oplus v_3^A \oplus r$, respectively. Finally it computes $v_4 = y^A \oplus y^B$ and assigns v_4 to x_4. The simulator updates the view of p, A and B according to the above simulation.

If $p \notin Z'$ then the simulator samples $r \leftarrow \{0,1\}$ on behalf of A and sends r to B. It computes the inputs $(v_1^A, v_2^A \oplus r)$ on behalf of A and $(v_1^B, v_2^B \oplus r)$ on behalf of B, and updates the views of A and B accordingly.

- If d_i is a computation statement $\mathsf{comp}(\tau, +, \{x_1, x_2\}, x)$ then the simulator updates the views of A and B according to the compilation of d_i. In more details, it sets the value of x^A to be $v_1^A \oplus v_2^A$ and the value of x^B to be $v_1^B \oplus v_2^B$, where $v_1^A, v_2^A, v_1^B, v_2^B$ are the values of $x_1^A, x_2^A, x_1^B, x_2^B$, respectively.
- If d_i is a computation statement $\mathsf{comp}(\tau, \times, (x_1, x_2), x)$, then Sim' computes $v_1^A = \mathsf{val}(x_1^A), v_2^A = \mathsf{val}(x_2^A)$ and $v_1^B = \mathsf{val}(x_1^B), v_2^B = \mathsf{val}(x_2^B)$. It then updates the views of A and B according to the compilation of d_i. In more details, it samples two random bits $v_3^A \leftarrow \{0,1\}$ and $r \leftarrow \{0,1\}$ on behalf of A. In the first OLE execution A inputs $(v_1^A, v_1^A \cdot v_2^A \oplus v_3^A \oplus r)$, B inputs v_2^B and receives the output y_1. In the second OLE execution A inputs (v_2^A, r), B inputs v_1^B and receives the output y_2. Finally, A assigns v_3^A to x_3^A, and B assigns $y_1 \oplus y_2 \oplus v_1^B \cdot v_2^B$ to x_3^B. The simulator updates the views of A and B according to the above simulation.
- If d_i is a computation statement $\mathsf{comp}(\tau, \mathsf{rand}, \emptyset, x)$ then Sim' holds $v = \mathsf{val}(x)$, samples $r \leftarrow \{0,1\}$, assigns r to x^A and $v \oplus r$ to x^B. The simulator updates the views of A and B accordingly.
- If d_i is an abort statement $\mathsf{abort}(\tau, x)$, then the simulator updates the views of A and B according to the compilation of d_i. In more details, it takes the roles of A and B, sends $v^A := \mathsf{val}(x^A)$ from A to B, and recovers $v = v^A \oplus v^B$ on behalf of B. The simulator updates the views of A and B accordingly.

Case 2. Assume that $A \notin Z'$ or $B \notin Z'$ (or both), and therefore $\chi_\wedge(Z') = \chi(Z' \setminus \{A, B\}) = 0$. Let $Z := Z' \setminus \{A, B\}$. We first let Sim' execute $\mathsf{Sim}(Z, (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}})$ to sample the views $\mathcal{V} := (\mathsf{view}(p))_{p \in Z}$. Let use denote the view of a party p in π' by $\mathsf{view}'(p)$. For every $i = 1, \dots, L$, and for the i-th statement d_i of π, the simulator Sim' updates the views $\mathcal{V}' := (\mathsf{view}'(p))_{p \in Z'}$ in the following way.

- If d_i does not involve τ, then Sim' assigns the variables that are updated in d_i the same values as in $\mathcal{V}$.
- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$ and $p \in Z'$, then Sim' holds $v = \mathsf{val}(x_1) = \mathsf{val}(x_2)$. The simulator updates the views of the corrupt parties according to the compilation of d_i. That is, the simulator samples a secret sharing of v on behalf of p, by sampling a random bit $v^A \leftarrow \{0,1\}$ and setting $v^B := v \oplus v^A$, and then sending v^A to A And v^B to B on behalf of p. The simulator updates the view of the corrupt parties in $\{p, A, B\} \cap Z'$ accordingly.
- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$ and $p \notin Z'$, but $\{A, B\} \cap Z' \neq \emptyset$, then the simulator samples $r \leftarrow \{0,1\}$, and if $A \in Z'$ the simulator assigns r to x^A, and if $B \in Z'$ the simulator assigns r to x^B.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ and $p \in Z'$ then the simulator holds $v_i := \mathsf{val}(x_i)$ for $i \in [2]$. In this case the simulator updates the views of the corrupt parties according to the compilation of d_i. In more details, it takes the role of p and samples $r \leftarrow \{0,1\}$, inputs

$(v_1, v_2 \oplus r)$ to the first OLE instance and (v_1, r) to the second OLE instance. If $A \in Z'$ then the simulator inputs $v_3^A := \mathsf{val}(x_3^A)$ to the first OLE instance and outputs $v_1 \cdot v_3^A \oplus v_2 \oplus r$ to A. Similarly, if $B \in Z'$ then the simulator inputs $v_3^B := \mathsf{val}(x_3^B)$ to the second OLE instance and outputs $v_1 \cdot v_3^B \oplus r$ to A. The simulator updates the views of the corrupt parties in $\{p, A, B\} \cap Z'$ accordingly.

- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$ and $p \notin Z'$, but $\{A, B\} \cap Z' \neq \emptyset$, then the simulator samples $r \leftarrow \{0, 1\}$, and if $A \in Z'$ the simulator assigns r to x_4^A, and if $B \in Z'$ the simulator assigns r to x_4^B.
- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ then we split into cases. First, assume that $p \in Z'$, so the simulator holds $v_1 = v_2 = \mathsf{val}(x_2)$.
  - If A is corrupt then the simulator holds $v_1^A = \mathsf{val}(x_1^A)$. The simulator samples $r \leftarrow \{0, 1\}$ on behalf of A and sends it to B. The simulator sends $v_1^A \oplus r$ to p on behalf of A, and $v_2 \oplus v_1^A \oplus r$ to p on behalf of B.
  - If B is corrupt then the simulator holds $v_1^B = \mathsf{val}(x_1^B)$. The simulator samples $r \leftarrow \{0, 1\}$ on behalf of A and sends it to B. The simulator sends $v_2 \oplus v_1^B \oplus r$ to p on behalf of A, and $v_1^B \oplus r$ to p on behalf of B.
  - Otherwise, both A and B are honest. In this case the simulator samples $r \leftarrow \{0, 1\}$, and sets the message from A to p to be r, and the message from B to p to be $v_2 \oplus r$.

  The simulator updates the views of the corrupt parties in $\{p, A, B\} \cap Z'$ accordingly.

  Otherwise, assume that $p \notin Z'$. In this case, if A is corrupt then we let the simulator sample a random bit $r \leftarrow \{0, 1\}$ on behalf of A, and compute $v_1^A \oplus r$. If B is corrupt, then we let the simulator receive a random bit $r \leftarrow \{0, 1\}$ on behalf of B and compute $v_1^B \oplus r$. The simulator updates the views of the corrupt parties accordingly.
- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$ then we split into cases. First, assume that $p \in Z'$, so the simulator holds $v_3 = \mathsf{val}(x_3)$ and $v_4 = \mathsf{val}(x_4)$.
  - If A is corrupt then the simulator holds $v_1^A = \mathsf{val}(x_1^A)$ and $v_2^A = \mathsf{val}(x_2^A)$. The simulator samples $r \leftarrow \{0, 1\}$ on behalf of A, sends r to B and inputs $(v_1^A, v_2^A \oplus r)$ to the first OLE instance. The output of the first OLE instance is set to be $v_1^A \cdot v_3 \oplus v_2^A \oplus r$. The output of the second OLE instance is set to be $v_4 \oplus v_1^A \cdot v_3 \oplus v_2^A \oplus r$.
  - If B is corrupt then the simulator holds $v_1^B = \mathsf{val}(x_1^B)$ and $v_2^B = \mathsf{val}(x_2^B)$. The simulator samples $r \leftarrow \{0, 1\}$ on behalf of A, sends r to B and inputs $(v_1^B, v_2^B \oplus r)$ to the second OLE instance. The output of the first OLE instance is set to be $v_4 \oplus v_1^B \cdot v_3 \oplus v_2^B \oplus r$. The output of the second OLE instance is set to be $v_1^B \cdot v_3 \oplus v_2^B \oplus r$.
  - Otherwise, both A and B are honest. The simulator samples $r \leftarrow \{0, 1\}$, sets the output of the first OLE instance to r and the output of the second OLE instance to $r \oplus v_4$.

  The simulator updates the views of the corrupt parties in $\{p, A, B\} \cap Z'$ accordingly.

  Otherwise, assume that $p \notin Z'$. In this case, if A is corrupt then we let the simulator sample a random bit $r \leftarrow \{0, 1\}$ on behalf of A, and compute $v_2^A \oplus r$. If B is corrupt, then we let the simulator receive a random bit $r \leftarrow \{0, 1\}$ on behalf of B and compute $v_2^B \oplus r$. The simulator updates the views of the corrupt parties accordingly.

- If d_i is a computation statement $\mathsf{comp}(\tau, +, \{x_1, x_2\}, x)$ then the simulator updates the views of the corrupt parties among $\{A, B\}$ according to the compilation of d_i. In more details, if A (resp., B) is corrupt then it sets the value of x^A (resp., x^B) to be $v_1^A \oplus v_2^A$ (resp., $v_1^B \oplus v_2^B$) where $v_i^A = \mathsf{val}(x_i^A)$ (resp., $v_i^B = \mathsf{val}(x_i^B)$).

- If d_i is a computation statement $\mathsf{comp}(\tau, \times, \{x_1, x_2\}, x)$, then we split into cases. If $A \in Z'$ then the simulator takes the role of A, samples random bits v_3^A and r, and inputs $(v_1^A, v_1^A \cdot v_2^A \oplus v_3^A \oplus r)$ to the first OLE instance, and (v_2^A, r) to the second OLE instance, where v_1^A and v_2^A are the values of x_1^A and x_2^A, respectively. It updates the view of A accordingly, including assigning v_3^A to x_3^A.

  Otherwise, B is corrupt. In this case the simulator takes the role of B in the following way. It inputs v_2^B to the first OLE instance and v_1^B to the second OLE instance, where v_1^B and v_2^B are the values of x_1^B and x_2^B, respectively. It sets the outputs of the two OLE instances to be random bits $y_1 \leftarrow \{0,1\}$ and $y_2 \leftarrow \{0,1\}$, respectively. Finally, it assigns $y_1 \oplus y_2 \oplus v_1^B \cdot v_2^B$ to x_3^B. It updates the view of B accordingly.

- If d_i is a computation statement $\mathsf{comp}(\tau, \mathsf{rand}, \emptyset, x)$ then Sim' samples a random bit $r \leftarrow \{0,1\}$. If A is corrupt then it assigns r to x^A, and if B is corrupt then it assigns r to x^B.

- If d_i is an abort statement $\mathsf{abort}(\tau, x)$, then we split into cases. If A is corrupt then the simulator does nothing. If B is corrupt then the simulator computes $v^B := \mathsf{val}(x^B)$, sets $v^A := v^B$, and sets v^A to be the message from A to B. Now, on behalf of B, it recovers $v = v^A \oplus v^B = 0$. The simulator updates the view of B accordingly.

At the end, the simulator outputs $\mathcal{V}'$. This concludes the description of Sim'. Privacy now follows from the following claim.

Claim C.4. *For all inputs $(x_1, \dots, x_n)$ to the clients, and for every $Z' \subseteq \mathcal{P}'$ that satisfies $\chi_\wedge(Z') = 0$ it holds that*

$$(\mathsf{view}'(p))_{p \in Z'} \equiv \mathsf{Sim}'(Z', (x_i)_{p_i \in Z' \cap \mathcal{C}}, (y_i)_{p_i \in Z' \cap \mathcal{C}}),$$

where $(y_1, \dots, y_n) = \mathcal{F}(x_1, \dots, x_n)$.

To prove the claim, we note that the simulated views $\mathcal{V}'$ are generated by applying the update procedure of Sim' on the simulated views $\mathcal{V}$ that Sim generates. We also note that if we apply the same update procedure on the real-world views $\mathcal{V}_{\mathsf{Real}}$ in an execution of π, then we obtain the same distribution as the real-world views $\mathcal{V}'_{\mathsf{Real}}$ in an execution of π'. Privacy now follows from the privacy of π, as $\mathcal{V}$ has the same distribution as $\mathcal{V}_{\mathsf{Real}}$, since both in Case 1 and in Case 2 it holds that $\chi(Z) = 0$. We omit a full proof of the claim.

C.2 Proof of Lemma C.2

We begin with the definition of the extractor Ext'. Let us denote $\mathcal{P} = \mathcal{C} \cup \mathcal{S} = \{p_1, \dots, p_{n+m-1}, \tau\}$, and let us assume that the i-th client is p_i, for $i \in [n]$. Let $S' \subseteq \mathcal{P}'$ be a set that satisfies $\chi_\wedge(S') = 1$, and let $Z' := \mathcal{P}' \setminus S'$. The extractor takes as input Z' and views $\mathcal{V}' = (\mathsf{view}'(p))_{p \in S'}$ of parties from an execution of an active adversarial behavior $\tilde{\pi}'$ with respect to Z', where the abort flag was not raised. The extractor Ext' generates a set $S \subseteq \mathcal{P}$ that satisfies $\chi(S) = 1$, sets $Z := \mathcal{P} \setminus S$, generates views $\mathcal{V} = (\mathsf{view}(p))_{p \in S}$, and applies Ext on $(Z, \mathcal{V})$ to obtain the inputs $\mathbf{x}^*$. We split into cases.

Case 1. Assume that $A, B \in S'$ so $A, B \notin Z'$. Let $S := (S' \setminus \{A, B\}) \cup \{\tau\}$ and observe that $1 = \chi_\wedge(S') = \chi(S)$. Let $Z := \mathcal{P} \setminus S = Z'$. The extractor generates views $\mathcal{V} := (\mathsf{view}(p))_{p \in S}$ that correspond to an execution of an adversarial behavior in π. (We note that $\mathcal{V}$ includes the view of τ.) To do so, for every $i = 1, \ldots, L$, and for the i-th statement d_i of π, the extractor Ext' updates the views $\mathcal{V}$ in the following way.

- If d_i is $\mathsf{transmit}(p, \tau, x_1, x_2)$, let v_2^A (resp., v_2^B) be the value of x_2^A (resp., x_2^B), according to the view of A (resp., B) in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning $v_2^A \oplus v_2^B$ to x_2 in the view of τ.
- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ for $p \notin Z$, let $v = \mathsf{val}(x_2)$ according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_2 in the view of p.
- If d_i is $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ for $p_2 \notin Z$, let v_2 be the value of x_2 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_2 in the view of p_2.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p, \tau, (x_1, x_2), (x_3), (x_4))$, let v_4^A (resp., v_4^B) be the value of x_4^A (resp., x_4^B), according to the view of A (resp., B) in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning $v_4^A \oplus v_4^B$ to x_4 in the view of τ.
- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$, let v be the value of x_4 according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_4 in the view of p.
- If d_i is $\mathsf{func}(\mathsf{OLE}, p_1, p_2, (x_1, x_2), (x_3), (x_4))$ for $p_2 \notin Z$, let v be the value of x_4 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_4 in the view of p_2.
- If d_i is $\mathsf{comp}(\tau, \mathsf{op}, X, x)$ for $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$, then let v^A (resp., v^B) be the value of x^A (resp., v^B) according to the view of A (res., B) in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning $v^A \oplus v^B$ to x in the view of τ.
- If d_i is $\mathsf{comp}(p, \mathsf{op}, X, x)$ for $p \notin Z$ and $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$ then let v be the value of x according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of p.
- d_i is an input statement $\mathsf{input}(c, i, x)$ for $c \notin Z$, then let v be the value of x according to the view of c in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of c.
- Otherwise, do nothing.

Case 2. Otherwise, $A \notin S'$ or $B \notin S'$. Let $S := S' \setminus \{A, B\}$, and observe that $1 = \chi_\wedge(S') = \chi(S)$. Let $Z := \mathcal{P} \setminus S$ and observe that $\tau \in Z$. The extractor generates views $\mathcal{V} := (\mathsf{view}(p))_{p \in S}$ that correspond to an execution of an adversarial behavior in π. To do so, for every $i = 1, \ldots, L$, and for the i-th statement d_i of π, the extractor Ext' updates the views $\mathcal{V}$ in the following way.

- If d_i is $\mathsf{transmit}(\tau, p, x_1, x_2)$ for $p \notin Z$, let v be the value of x_2 according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_2 in the view of p.
- If d_i is $\mathsf{transmit}(p_1, p_2, x_1, x_2)$ for $p_2 \notin Z$, let v_2 be the value of x_2 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_2 in the view of p_2.
- If d_i is $\mathsf{func}(\mathsf{OLE}, \tau, p, (x_1, x_2), (x_3), (x_4))$, let v be the value of x_4 according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_4 in the view of p.

- If d_i is $\mathsf{func}(\mathsf{OLE}, p_1, p_2, (x_1, x_2), (x_3), (x_4))$ for $p_2 \notin Z$, let v be the value of x_4 according to the view of p_2 in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x_4 in the view of p_2.
- If d_i is $\mathsf{comp}(p, \mathsf{op}, X, x)$ for $p \notin Z$ and $\mathsf{op} \in \{+, \times, \mathsf{rand}\}$ then let v be the value of x according to the view of p in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of p.
- d_i is an input statement $\mathsf{input}(c, i, x)$ for $c \notin Z$, then let v be the value of x according to the view of c in the execution of $\tilde{\pi}'$. Update $\mathcal{V}$ by assigning v to x in the view of c.
- Otherwise, do nothing.

At the end, Ext' computes $\mathbf{x}^* = \mathsf{Ext}(Z, \mathcal{V})$ and outputs $\mathbf{x}^*$. This concludes the description of Ext'. Perfect correctness now follows from the following claim.

Claim C.5. *For all inputs $(x_1, \ldots, x_n)$ to the clients, for every $S \subseteq \mathcal{P}'$ that satisfies $\chi_\wedge(S') = 1$, for $Z' := \mathcal{P} \setminus S'$, for every adversarial behavior $\tilde{\pi}'$ of π' with respect to Z', and for every execution of $\tilde{\pi}'$ in which the abort flag is not raised and the honest parties have view $\mathcal{V}'$ the following holds. There is an input $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ such that (1) $x_i^* = x_i$ for all honest clients $p_i \in S'$, (2) the outputs of the honest clients is consistent with $\mathcal{F}(\mathbf{x}^*)$, and (3) $\mathsf{Ext}'(Z', \mathcal{V}') = \mathbf{x}^*$.*

To prove the claim, we note that the views $\mathcal{V}$ are generated by applying the update procedure of Ext' on the views $\mathcal{V}'$ from the execution of $\tilde{\pi}'$, and that the clients have the same outputs in $\mathcal{V}$ as in $\mathcal{V}'$. In addition we note that there is a natural adversarial behavior $\tilde{\pi}$ against π with respect to the set Z, where the views $\mathcal{V}$ are in the support. Correctness now follows from the correctness of Ext, since both in Case 1 and in Case 2 it holds that $\chi(S) = 1$. We omit a full proof of the claim.

D Missing proofs: MS-NIZK

We prove the following simple claim.

Claim D.1. *There exists a transformation that maps any (single-CRS) NIP into a (single-CRS) NIP with* strong completeness*. Furthermore, computational/statistical soundness and computational adaptive/non-adaptive zero-knowledge are preserved. If in the original scheme completeness holds against inefficient adversaries then the transformation preserves statistical adaptive/non-adaptive zero-knowledge.*

Sketch. First consider the case where the verifier is deterministic. In this case, we let the prover check that the generated proof passes verification. If the verification fails, the prover sends the witness (together with a special symbol). The verifier acts like the original verifier except that whenever a witness is given, she verifies that the witness is valid and accept the proof if this is indeed the case. It is not hard to see that the modified NIZK has perfect completeness (which implies strong completeness) and that soundness is not affected. Zero-knowledge is violated whenever the original prover generates an erroneous proof. However, since this event happens with negligible probability this increases the zero-knowledge deviation by a negligible quantity. This argument holds both for adaptive and non-adaptive zero-knowledge, and it extends to statistical adaptive/non-adaptive zero-knowledge assuming that completeness holds against inefficient adversaries.

If the verifier is randomized, we let the prover approximate the acceptance probability of the verifier with an additive error of, say, 0.1 with confidence error that is negligible in κ. (This can be done by sampling κ many random tapes and computing the empirical error.) If the approximated acceptance probability is smaller than, say, 0.9, the prover replaces the proof with its witness. We also modify the verifier by letting her approximate the acceptance probability of the original verifier (with similar confidence and additive error) and accept the proof if either the approximation is at least, say, 0.7, or if the proof contains a satisfying witness. A completeness error occurs only if the prover's approximation is larger than 0.9 but the verifier's approximation is smaller than 0.7, which happens only with negligible probability. Soundness is not affected and the adaptive/non-adaptive zero-knowledge error grows additively by α — the probability that a completeness error occurs in the original scheme plus the probability that the approximation fails — which is negligible by assumption. □

E Missing proofs: DP-NIZK and Round-Optimal MPC

In this section we provide formal security proofs for the protocols we present in Section 5. In all proof of security in the UC framework, we assume without loss of generality that the environment Env is deterministic, and we denote by view *the view of* Env, that consists of the messages that the corrupt parties sent and received, the inputs of the honest parties (which are picked by the environment), and the outputs of the honest parties. We always assume that the adversary is the dummy adversary (see Section F.2 for more information about the framework of universal composability).

E.1 Proof of Theorem 5.6

Completeness follows from the perfect completeness of the NIZK in the hidden-bits model. We therefore continue with the analysis of proof of knowledge and zero knowledge.

Proof of knowledge. Let us denote by $\mathsf{Ext}_{\mathsf{hidBits}}$ the extractor of Π_{hidBits}. We begin with the definition of Gen' and Ext.

$\mathsf{Gen}'(1^n)$**.** On input 1^n the algorithm samples $(\mathsf{crs}, \mathsf{sk}_{\mathcal{P}}) \leftarrow \mathsf{Gen}(1^n)$, sets $\tau := \mathsf{sk}_{\mathcal{P}}$ and outputs $(\mathsf{crs}, \mathsf{sk}_{\mathcal{P}}, \tau)$.

$\mathsf{Ext}(1^n, \mathsf{crs}, \tau, f, \pi)$**.** On input 1^n, $\mathsf{crs} = (\mathsf{pp}, C_1, \ldots, C_m)$, trapdoor $\tau = (o_1, \ldots, o_m)$, a circuit-SAT instance f with description of length n, and a proof $\pi = (I, \pi_{\mathsf{hidBits}}, (o'_i)_{i \in I})$. The extractor computes $r_i := \mathsf{open}(C_i, o_i)$, sets $\mathbf{r} := (r_1, \ldots, r_m)$, computes $\mathbf{x} := \mathsf{Ext}_{\mathsf{hidBits}}(f, I, \mathbf{r}, \pi_{\mathsf{hidBits}})$ and outputs $\mathbf{x}$.

Since the NIZK in the hidden-bits model has perfect proof of knowledge, every computationally-unbounded non-uniform family of malicious provers $\mathcal{P}^* = \{\mathcal{P}^*_n\}_{n \in \mathbb{N}}$ violates proof of knowledge only if it successfully violates the binding property of the NICOM. Since the NICOM is statistically-binding, this happens only with negligible probability.

Zero knowledge. Let Sim be the simulator of Π_{hidBits}. We continue with the definition of $(\mathsf{Sim}_1, \mathsf{Sim}_2)$.

$\mathsf{Sim}_1(1^n)$**.** On input 1^n the simulator samples pp and commits $(C_i, o_i) \leftarrow \mathsf{commit}(0)$ for every $i \in [m]$. The simulator sets $\mathsf{crs} = (\mathsf{pp}, C_1, \ldots, C_m)$ and $\tau = (o_1, \ldots, o_m)$. The simulator outputs (crs, τ).

$\mathsf{Sim}_2(1^n, \mathsf{crs}, \tau, f)$. On input 1^n, CRS $\mathsf{crs} = (\mathsf{pp}, C_1, \ldots, C_m)$, trapdoor $\tau = (o_1, \ldots, o_m)$, and circuit-SAT instance f, the simulator samples $(I, \mathbf{r}_I, \pi_{\mathsf{hidBits}}) \leftarrow \mathsf{Sim}(f)$ and outputs $\pi := (I, \pi_{\mathsf{hidBits}}, (o_i)_{i\in I})$.

Assume towards contradiction that there is a polynomially-bounded non-uniform family of circuits $\{\mathcal{A}_n, \mathcal{B}_n\}_{n\in\mathbb{N}}$ and a polynomial $q(n)$, such that for infinitely many n's it holds that

$$\left| \Pr_{\substack{(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})\leftarrow\mathsf{Gen}(1^n)\\(f,\mathbf{x})\leftarrow\mathcal{A}_n(\mathsf{crs})\\\pi\leftarrow\mathcal{P}(1^n,\mathsf{crs},\mathsf{sk}_{\mathcal{P}},f,\mathbf{x})}} [\mathcal{B}_n(\mathsf{crs}, \pi) = 1] - \Pr_{\substack{(\mathsf{crs},\tau)\leftarrow\mathsf{Sim}_1(1^n)\\(f,\mathbf{x})\leftarrow\mathcal{A}_n(\mathsf{crs})\\\pi\leftarrow\mathsf{Sim}_2(1^n,\mathsf{crs},\tau,f)}} [\mathcal{B}_n(\mathsf{crs}, \pi) = 1] \right| > 1/q(n), \tag{3}$$

where $\mathcal{A}_n$ always outputs a circuit f with description of length n and a satisfying assignment $\mathbf{x}$. We define the real-world experiment by sampling $(\mathsf{crs}, \mathsf{sk}_{\mathcal{P}}) \leftarrow \mathsf{Gen}(1^n)$, $(f, \mathbf{x}) \leftarrow \mathcal{A}_n(\mathsf{crs})$ $\pi \leftarrow \mathcal{P}(1^n, \mathsf{crs}, \mathsf{sk}_{\mathcal{P}}, f, \mathbf{x})$ and $b \leftarrow \mathcal{B}(\mathsf{crs}, \pi)$, and outputting b, and we observe that the left term in Equation (3) corresponds to the real-world experiment. Similarly, we define the simulation experiment by sampling $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^n)$, $(f, \mathbf{x}) \leftarrow \mathcal{A}_n(\mathsf{crs})$, $\pi \leftarrow \mathsf{Sim}_2(1^n, \mathsf{crs}, \tau, f)$ and $b \leftarrow \mathcal{B}(\mathsf{crs}, \pi)$, and outputting b, and we observe that the the right term in Equation (3) corresponds to the simulation experiment.

Recall that there is some integer $k = k(n)$ such that with probability 1 the CRS of Π_{hidBits} contains exactly k ones, and the remaining $m - k$ entries are zero. (See a discussion regarding the special properties of Π_{hidBits} in F.1). We show that there exists a polynomially-bounded non-uniform family of circuits $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ that distinguishes k commitments of 0 from k commitments of 1 with non-negligible advantage. This contradicts the hiding property of the non-interactive commitment scheme.

We define the distinguisher $\mathcal{D}_n$ in the following way. The distinguisher takes as an input public parameters pp, and $k(n)$ commitments $C_1, \ldots, C_k$. The distinguisher samples $\mathbf{r} \leftarrow \mathsf{Gen}_{\mathsf{hidBits}}(1^n)$ where $\mathbf{r} = (r_1, \ldots, r_m) \in \{0,1\}^m$, and sets $J \subseteq [m]$ to be the set of all indices $i \in [m]$ such that $r_i = 1$. Observe that $|J| = k$ and let us denote $J = \{i_1, \ldots, i_k\}$. For every $i \in [m]$ the adversary defines C'_i in the following way: if $i = i_j \in J$ the distinguisher sets $C'_i := C_j$, and otherwise the distinguisher samples $(C_i, o_i) \leftarrow \mathsf{commit}(0)$, and sets $C'_i := C_i$. The distinguisher sets $\mathsf{crs} := (\mathsf{pp}, C'_1, \ldots, C'_m)$, and executes $(f, \mathbf{x}) \leftarrow \mathcal{A}_n(\mathsf{crs})$ and $(I, \pi_{\mathsf{hidBits}}) \leftarrow \mathcal{P}_{\mathsf{hidBits}}(f, \mathbf{x}, \mathbf{r})$. Since the prover in Π_{hidBits} opens only zeros in the CRS with probability 1 (see Section F.1), it holds that $I \cap J = \emptyset$ and therefore the distinguisher holds an opening o_i for every $i \in I$. Finally, the distinguisher sets $\pi := (I, \pi_{\mathsf{hidBits}}, (o_i)_{i\in I})$, computes $b \leftarrow \mathcal{B}_n(\mathsf{crs}, \pi)$ and outputs the bit b.

It is not hard to see that when $C_1, \ldots, C_k$ are commitments of ones, we obtain the same distribution as in the real-world experiment, i.e.,

$$\Pr_{\substack{\mathsf{pp}\\\forall i\in[k],\ (C_i,o_i)\leftarrow\mathsf{commit}_{\mathsf{pp}}(1)}} [\mathcal{D}_n(\mathsf{pp}, C_1, \ldots, C_k) = 1] = \Pr_{\substack{(\mathsf{crs},\mathsf{sk}_{\mathcal{P}})\leftarrow\mathsf{Gen}(1^n)\\(f,\mathbf{x})\leftarrow\mathcal{A}_n(\mathsf{crs})\\\pi\leftarrow\mathcal{P}(1^n,\mathsf{crs},\mathsf{sk}_{\mathcal{P}},f,\mathbf{x})}} [\mathcal{B}_n(\mathsf{crs}, \pi) = 1].$$

The following claim shows that when $C_1, \ldots, C_k$ are commitments of zeros, we obtain the same distribution as in the simulation experiment.

Claim E.1. *It holds that*

$$\Pr_{\substack{\mathsf{pp}\\\forall i\in[k],\ (C_i,o_i)\leftarrow\mathsf{commit}_{\mathsf{pp}}(0)}} [\mathcal{D}_n(\mathsf{pp}, C_1, \ldots, C_k) = 1] = \Pr_{\substack{(\mathsf{crs},\tau)\leftarrow\mathsf{Sim}_1(1^n)\\(f,\mathbf{x})\leftarrow\mathcal{A}_n(\mathsf{crs})\\\pi\leftarrow\mathsf{Sim}_2(1^n,\mathsf{crs},\tau,f)}} [\mathcal{B}_n(\mathsf{crs}, \pi) = 1].$$

Given the claim, we conclude that for infinitely many n's, the distinguisher $\mathcal{D}_n$ distinguishes $k(n)$ commitments of ones from $k(n)$ commitments of zeros with advantage $1/q(n)$, in contradiction to the hiding property of the commitment scheme, which concludes the analysis of the zero-knowledge property. We continue with the proof of Claim E.1.

Proof of Claim E.1. Consider the random variables $(\mathsf{crs}, f, \mathbf{x}, I, \pi_{\mathsf{hidBits}}, (o_i)_{i \in I}, b)$ that are generated in the simulation experiment, and in an execution of $\mathcal{D}_n$ with commitments of zeros. Since all the commitments are commitments of zero, the random variable crs in both experiments has the same distribution, and we fix it. Moreover, in both experiments $\mathcal{A}_n$ chooses $(f, \mathbf{x})$ in the same way, and let use fix them as well. In addition, since all the committed values are zeros, the random variable crs is independent of the choice of $\mathbf{r}$ by $\mathcal{D}_n$. Therefore, the perfect zero knowledge of Π_{hidBits} implies that $(I, \pi_{\mathsf{hidBits}})$ have the same distribution in both cases, and let us fix them as well. Finally, the openings $(o_i)_{i \in I}$ have the same distribution in both experiments, as they are generated by sampling commitments of zeros, and let us fix them as well. We therefore conclude that the output b of $\mathcal{B}_n$ has the same distribution in both experiments. This concludes the proof of the claim. $\square$

This concludes the proof of Theorem 5.6. $\square$

E.2 The SIF Construction and Proof of Theorem 5.8

We begin with a formal definition of the protocol sif.

Protocol sif

There is a distinguished party D. The functionality is parameterized by circuits $f_1, \ldots, f_n$, where f_i has k input bits and ℓ_i output bits. We denote the security parameter by κ, and assume that all cryptographic primitives are executed with security parameter κ.

- **Primitives.**
  - A non-interactive commitment scheme $(\mathsf{commit}, \mathsf{open})$. In the following we will slightly abuse notation, and use the bit-commitment to commit to a bit-string, by simply committing to each bit of the string independently.
  - A public single input functionality $\mathcal{G}$ that (1) Takes from the dealer an input an assignment $\mathbf{x} \in \{0,1\}^k$, public parameters $\mathsf{pp}_1, \ldots, \mathsf{pp}_n$, commitments $C_1, \ldots, C_n$ and openings $o_1, \ldots, o_n$, where C_i is a commitment on ℓ_i bits, (2) Computes $\mathbf{y}_i := f_i(\mathbf{x})$ and $\mathbf{r}_i := \mathsf{open}_{\mathsf{pp}_i}(C_i, o_i)$ for every $i \in [n]$, (3) For every $i \in [n]$, if $\mathbf{r}_i = \perp$ (i.e., the opening failed), it sets $\mathsf{output}_i := (\mathsf{fail}, \mathbf{y}_i)$; otherwise, if $\mathbf{r}_i \in \{0,1\}^{\ell_i}$, it sets $\mathsf{output}_i := \mathbf{y}_i \oplus \mathbf{r}_i$, (4) It outputs $(\mathsf{pp}_1, \ldots, \mathsf{pp}_n, C_1, \ldots, C_n, \mathsf{output}_1, \ldots, \mathsf{output}_n)$.
  - A 2-round offline/online protocol psif for the secure computation of $\mathcal{G}$.
- **Offline round.** In the offline round the parties executes the offline round of psif with D as the dealer. In addition, every P_i does as follows.
  1. Samples $\mathbf{r}_i \leftarrow \{0,1\}^{\ell_i}$ and public parameters pp_i for the non-interactive commitment scheme.
  2. Commits $(C_i, o_i) \leftarrow \mathsf{commit}_{\mathsf{pp}_i}(\mathbf{r}_i)$.
  3. Broadcasts (pp_i, C_i) and sends o_i to D.

- **Inputs.** D receives the input $\mathbf{x} \in \{0,1\}^k$.
- **Online round.** The parties execute the online round of psif, where D inputs the tuple $(\mathbf{x}, \mathsf{pp}_1, \ldots, \mathsf{pp}_n, C_1, \ldots, C_n, o_1, \ldots, o_n)$
- **Local computation.** Every party P_i does as follows. Let $(\mathsf{pp}'_1, \ldots, \mathsf{pp}'_n, C'_1, \ldots, C'_n, \mathsf{output}_1, \ldots, \mathsf{output}_n)$ be the output of psif, and let (pp_j, C_j) be the broadcast of P_j in the offline round. If there exists $j \in [n]$ such that $(\mathsf{pp}_j, C_j) \neq (\mathsf{pp}'_j, C'_j)$ then P_i outputs $f_i(0, \ldots, 0)$ and terminates. Otherwise, if $\mathsf{output}_i = (\mathsf{fail}, \mathbf{y}_i)$ then P_i outputs $\mathbf{y}_i$ and terminates. Otherwise $\mathsf{output}_i = \mathbf{z}_i \in \{0,1\}^{\ell_i}$, and P_i computes $\mathbf{y}_i := \mathbf{z}_i \oplus \mathbf{r}_i$, outputs $\mathbf{y}_i$ and terminates.

Figure 9: Protocol sif

Theorem E.2 (Theorem 5.8 restated.). *Let κ be a security parameter, let n be the number of parties, and let $t < n/2$ be the number of corrupt parties. We obtain the following results:*

- (Statistical soundness from OWFs) *Assuming the existence of one-way functions, protocol* sif*, when instantiated with statistically-binding commitments (that are implied by one-way functions) and when* psif *is the statistically-sound protocol promised in Theorem 5.7, is a UC-secure implementation of $\mathcal{F}_{\mathsf{sif}}$, against a static, active, rushing adversary that corrupts at most t of the parties. In addition, the protocol provides statistical soundness against a corrupt dealer D.*

- (Everlasting security from CRH) *Assuming the existence of collision-resistant hash functions, protocol* sif*, when statistically-hiding commitments (that are implied by collision-resistance hash functions) and when* psif *is the protocol with everlasting security promised in Theorem 5.7, is a UC-secure implementation with everlasting security of $\mathcal{F}_{\mathsf{sif}}$, against a static, active, rushing adversary that corrupts at most t of the parties.*

In all cases, the complexity of the protocol is $\mathrm{poly}(\kappa, n, s_1, \ldots, s_n)$, where s_i is the circuit size of f_i. In addition, in the offline phase every party communicates by sending a private message to D and sending a public broadcast message, and in the online phase only D communicates by sending a public broadcast message.

Proof sketch. We prove security in the $\mathcal{F}_{\mathsf{psif}}$-hybrid model. We split into cases.

Honest D. In the $\mathcal{F}_{\mathsf{psif}}$-hybrid model, the offline round consists only of the broadcast messages of the parties. On behalf of every honest P_i, the simulator samples pp_i and $(C_i, o_i) \leftarrow \mathsf{commit}_{\mathsf{pp}_i}(0^{\ell_i})$ and broadcasts (pp_i, C_i). At the end of the round, the simulator receives the broadcast (pp_i, C_i) of every corrupt P_i, as well as the opening o_i that P_i sends to D.

In the online phase the simulator receives the output $\mathbf{y}_i \in \{0,1\}^{\ell_i}$ for every corrupt P_i. The online phase consists only of the output of $\mathcal{F}_{\mathsf{psif}}$. The simulator sets the output to be $(\mathsf{pp}_1, \ldots, \mathsf{pp}_n, C_1, \ldots, C_n, \mathsf{output}_1, \ldots, \mathsf{output}_n)$, where output_i is defined in the following way:

- If P_i is honest, then the simulator samples $\mathsf{output}_i \leftarrow \{0,1\}^{\ell_i}$.
- If P_i is corrupt, then the simulator computes $\mathbf{r}_i = \mathsf{open}_{\mathsf{pp}_i}(C_i, o_i)$. If $\mathbf{r}_i = \perp$ then the simulator sets $\mathsf{output}_i := (\mathsf{fail}, \mathbf{y}_i)$. Otherwise, if $\mathbf{r}_i \in \{0,1\}^{\ell_i}$, the simulator sets $\mathsf{output}_i := \mathbf{y}_i \oplus \mathbf{r}_i$.

This concludes the simulation. We observe that in a real-world execution in the $\mathcal{F}_{\mathsf{psif}}$-hybrid model, the output of every honest P_i is $\mathbf{y}_i = f_i(\mathbf{x})$ with probability 1. In addition, a standard argument shows that the simulation is statistically-close to the real world if the underlying commitment scheme is statistically-hiding, and otherwise, if the commitment scheme is only computationally-hiding, the simulation is only computationally-close to the real world.

Corrupt D. When D is corrupt, the simulator simply takes the role of the honest parties, and initiates an execution of sif. That is, in the offline phase, on behalf of each honest P_i the simulator samples pp_i, $\mathbf{r}_i \leftarrow \{0,1\}^{\ell_i}$ and $(C_i, o_i) \leftarrow \mathsf{commit}_{\mathsf{pp}_i}(\mathbf{r}_i)$, broadcasts (pp_i, C_i) on behalf of P_i, and sends o_i to the corrupt dealer. At this stage the simulator receives the broadcast messages (pp_i, C_i) of every corrupt P_i. In the online phase the simulator receives the inputs $(\mathbf{x}, \mathsf{pp}'_1, \ldots, \mathsf{pp}'_n, C'_1, \ldots, C'_n, o'_1, \ldots, o'_n)$ of D to $\mathcal{F}_{\mathsf{psif}}$, computes the output $(\mathsf{pp}'_1, \ldots, \mathsf{pp}'_n, C'_1, \ldots, C'_n, \mathsf{output}_1, \ldots, \mathsf{output}_n)$ of $\mathcal{F}_{\mathsf{psif}}$ and sets it to be the output of $\mathcal{F}_{\mathsf{psif}}$. In addition, the simulator verifies that for every $i \in [n]$ it holds that $(\mathsf{pp}_i, C_i) = (\mathsf{pp}'_i, C'_i)$. If the verification fails the simulator sets $\mathbf{x}' := (0, \ldots, 0)$ and otherwise the simulator sets $\mathbf{x}' := \mathbf{x}$. The simulator inputs $\mathbf{x}'$ to $\mathcal{F}_{\mathsf{sif}}$ and terminates. This concludes the simulation.

Observe that the view of the corrupt parties in the simulation has the same distribution as in the real world. It therefore remains to analyze the outputs of the honest parties. We observe that the simulation corresponds to an execution of sif in the $\mathcal{F}_{\mathsf{psif}}$-hybrid model. It is not hard to verify that in every such execution, the outputs of the honest parties in the real world differ from the outputs in the ideal world only if the adversary violates the binding property of the commitment scheme, which occurs only with negligible probability. In fact, if the commitment scheme is statistically-binding, and psif provides statistically soundness, then sif also provides statistical soundness. This concludes the proof of the theorem. $\square$

F Additional Background

F.1 The Hidden-Bits Model and the FLS protocol

In this section we present a formal definition of the hidden-bits model, due to [FLS90]. Our definition is tailored for our use of the hidden-bits model in the applications, and therefore it slightly deviates from that of [FLS90]. In particular, instead of requiring the CRS to be uniformly distributed, we allow it to be sampled by a randomized efficient generator Gen. As a consequence, we can obtain a NIZK in the hidden-bits model, that satisfies slightly stronger properties than the standard definition.

Definition F.1. *A* non-interactive zero-knowledge scheme in the hidden-bits model *is a triple* $(\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ *of PPT algorithms with the following syntax:*

- *The randomized generator* Gen *takes as an input* 1^n *and outputs a binary string* $\mathbf{r}$ *of polynomially-bounded length* $m = m(n) = \mathrm{poly}(n)$.
- *The randomized prover* $\mathcal{P}$ *takes as an input a circuit-SAT instance* f *with a description of length* n, *an assignment* $\mathbf{x}$ *and a string* $\mathbf{r} \in \{0,1\}^m$, *and outputs a pair* (I, π) *where* $I \subseteq [m]$ *and* π *is called the certificate.*

- *The deterministic verifier $\mathcal{V}$ takes as an input a circuit-SAT instance f, a subset $I \subseteq [m]$, a sub-string $\mathbf{r}_I$ of $\mathbf{r}$ restricted to I, and a certificate π. The verifier outputs a bit b representing accept or reject.*

The algorithms satisfy the following properties.

- (Perfect completeness) *For every $n \in \mathbb{N}$, every circuit-SAT instance f with a description of length n, and every satisfying assignment $\mathbf{x}$, it holds that*

$$\Pr_{\substack{\mathbf{r} \leftarrow \mathsf{Gen}(1^n) \\ (I,\pi) \leftarrow \mathcal{P}(f,\mathbf{x},\mathbf{r})}} [\mathcal{V}(f, I, \mathbf{r}_I, \pi) = 1] = 1.$$

- (Perfect proof of knowledge) *There exists an efficient deterministic algorithm* Ext *such that for every $n \in \mathbb{N}$, and every computationally-unbounded non-uniform family of malicious provers $\mathcal{P}^* = \{\mathcal{P}^*_n\}_{n \in \mathbb{N}}$,*

$$\Pr_{\substack{\mathbf{r} \leftarrow \mathsf{Gen}(1^n) \\ (f,I,\pi) \leftarrow \mathcal{P}^*_n(\mathbf{r})}} [\mathsf{Ext}(f, I, \mathbf{r}, \pi) = \mathbf{x} \wedge f(\mathbf{x}) = 0 \wedge \mathcal{V}(f, I, \mathbf{r}_I, \pi) = 1] = 0,$$

where $\mathcal{P}^*_n$ outputs a circuit f with a description of length n.

- (Perfect zero knowledge) *There exist an efficient simulator* Sim, *such that for every $n \in \mathbb{N}$, every circuit f with a description of length n, every satisfying assignment $\mathbf{x}$, and every computationally-unbounded non-uniform family of circuits $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$,*

$$\left| \Pr_{\substack{\mathbf{r} \leftarrow \mathsf{Gen}(1^n) \\ (I,\pi) \leftarrow \mathcal{P}(f,\mathbf{x},\mathbf{r})}} [\mathcal{B}_n(f, I, \mathbf{r}_I, \pi) = 1] - \Pr_{(I,\mathbf{r}_I,\pi) \leftarrow \mathsf{Sim}(f)} [\mathcal{B}_n(f, I, \mathbf{r}_I, \pi) = 1] \right| = 0,$$

where $\mathcal{A}_n(\mathbf{r})$ outputs a circuit f with a description of length n, and a satisfying assignment $\mathbf{x}$.

The Feige-Lapidot-Shamir construction. The first construction of NIZK in the hidden-bits model was presented in the celebrated work of [FLS90] (FLS), and we observe that their construction, with some simple modifications, satisfies our (slightly stronger) definition.

We recall that in the FLS construction the CRS $\mathbf{r}$ consisted of many block $\mathbf{r}_1, \ldots, \mathbf{r}_\ell$, where each block can be either *useful* or *not useful*. The existence of a single useful block in the CRS guaranteed perfect soundness (and, in fact, proof of knowledge), and since each block was useful with some inverse polynomial probability, a polynomial number of blocks ℓ was sampled, to guarantee that with probability at least (say) $2/3$ there is at least one useful block. Since we allow $\mathbf{r}$ to be generated by an efficient generator Gen, we simply assume that the generator samples only a single block that is good with probability 1.[21] In addition, as observed by [Gol01, Chapter 4.10.3.2] (see also [Gol04, Chapter C.4.3]), the original protocol satisfies *adaptive* soundness (and, in fact, proof of knowledge), as well as our notion of zero knowledge. We therefore obtain the following theorem.

Theorem F.2 ([FLS90])**.** *There exists a hidden-bits proof system.*

[21] We also recall that the original construction used uniformly random bits to simulate biased bits, that were required to guarantee that a block is useful with noticeable probability. We ignore this technicality, as our generator can sample a useful block with probability 1.

Special properties of FLS. We observe that, under the simplifications we've presented, the FLS scheme satisfies the following special properties, that will be important for the applications. First, there is some integer $k = k(n)$ such that with probability 1 the CRS $\mathbf{r}$ contains exactly k ones and the remaining $m - k$ entries are zero. In addition, with probability 1 the prover $\mathcal{P}(f, \mathbf{x}, \mathbf{r})$ generates a set $I \subseteq [m]$ such that $\mathbf{r}_I$ is the all-zero string. Finally, the simulator $\mathsf{Sim}(f)$ has the following structure: It sets $\mathbf{r}$ to be the all-zero string, samples I, π from some distribution that depends only in f and outputs $(I, \mathbf{r}_I, \pi)$.

F.2 UC Security

The following is taken, with changes, from [AKP22b]. In this section we give a high-level description of the UC-framework, due to [Can01]. For more details, the reader is referred to [Can01]. We begin with a short description of the standard model, and then explain how the UC-framework augments it. At a high level, in the standard model, security of a protocol is argued by comparing the real-world execution to an ideal-world execution. In an ideal-world execution, the inputs of the parties are transferred to a trusted party $\mathcal{F}$ (called the *ideal functionality*) over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol π securely implements $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary Sim (called the *simulator*), that controls the same parties as $\mathcal{A}$, so that the global output of an execution of π with $\mathcal{A}$ (consisting of the honest parties' outputs and the output of $\mathcal{A}$), is indistinguishable from the global output of the ideal-world execution with $\mathcal{F}$ and Sim (consisting of the honest parties' outputs and the output of Sim).

The UC-framework augments the standard model by adding an additional entity, called the *environment* Env. In the real-world, Env arbitrarily interacts with the adversary $\mathcal{A}$, and, in addition, Env generates the inputs of the honest parties at the beginning of the execution, and receives their outputs at the end of the execution. In the ideal world, *the same* environment Env arbitrarily interacts with the simulator Sim, and, in addition, Env communicates with dummy parties, that receive the honest parties' inputs from Env and immediately transfer them to $\mathcal{F}$, and later receive the honest parties' outputs from $\mathcal{F}$ and immediately transfer them to Env. In both worlds, at the end of the execution the environment Env outputs a single bit.

For a security parameter κ and input ζ to Env, we denote the distribution of the output bit of $\mathsf{Env}(\zeta)$ in a real-world execution of π with adversary $\mathcal{A}$ by $\text{REAL}_{\pi,\mathsf{Env}(\zeta),\mathcal{A}}(\kappa)$. We denote the distribution of the output bit of $\mathsf{Env}(\zeta)$ in an ideal-world execution with ideal-functionality $\mathcal{F}$, simulator Sim by $\text{IDEAL}_{\mathcal{F},\mathsf{Env}(\zeta),\mathsf{Sim}}(\kappa)$. Intuitively, we say that a protocol π *UC-emulates* an ideal-functionality $\mathcal{F}$ if for every real-world polynomial-time adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator Sim, so that for any environment Env and any input ζ to Env, it holds that $\left\{\text{REAL}_{\pi,\mathsf{Env}(\zeta),\mathcal{A}}(\kappa)\right\}_{\kappa}$ is computationally indistinguishable from $\left\{\text{IDEAL}_{\mathcal{F},\mathsf{Env}(\zeta),\mathsf{Sim}}(\kappa)\right\}_{\kappa}$.

The dummy-adversary. Since the above definition quantifies over all environments, we can merge the adversary $\mathcal{A}$ with the environment Env. That is, it is enough to require that the simulator Sim will be able to simulate, for any environment Env, the *dummy adversary* that simply delivers messages from Env to the protocol machines. For more information, see [Can01].

The hybrid model. The UC-framework is appealing because it has strong composability properties. Consider a protocol ρ that securely implements an ideal functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model (which means that the parties in ρ have access to an ideal functionality $\mathcal{F}$), and let π be a protocol that securely implements $\mathcal{F}$. The composition theorem guarantees that if we replace in ρ each call to $\mathcal{F}$ with an execution of π we obtain a secure protocol. This means that it is enough to prove the security of a protocol in the hybrid model, where the analysis is much simpler.

Corruption-aware functionalities. Throughout, we assume that our functionalities are *corruption-aware*, which means that they might depend on the identities of the corrupt parties C. The notion of corruption aware functionalities was first introduced by [Can01], and the reader is referred to [Can01] for more information (see also [AL17, Section 6.2]).

Reactive functionalities. In this work, we consider both single-phase functionalities and reactive (or multi-phase) functionalities. A single-phase functionality maps the inputs of the parties to the outputs in a single phase of computation. A multi-phase functionality consists of multiple phases of computation, where each phase depends on the internal state of the functionality. In each phase the functionality receives the inputs of the parties, computes the outputs based on the inputs and the internal state, and updates the internal state based on the inputs. We only consider functionalities with a single phase, or with two phases. For more information about reactive functionalities, see, e.g., [Gol04, Chapter 7.7.1.3].

Everlasting security. We also consider a hybrid version of statistical and computational security. Intuitively, *everlasting security* requires that an environment which is polynomially-bounded *during* the execution and is allowed to be unbounded *after* the execution, cannot distinguish the real-world from the ideal-world. Observe that this security notion lies between computational-security (where we consider only environments that are *always* polynomially-bounded) and statistical-security (where we also consider environments that are unbounded during the execution of the protocol).

The notion of everlasting security was formalized in the UC-framework by [MU10]. In a nutshell, instead of considering environments that are unbounded after the execution, it is enough to consider only environments that are always polynomially-bounded, but are not limited to a single bit output. In particular, such environments can output their whole view. Using the same notation as before, $\mathrm{REAL}_{\pi,\mathsf{Env}(\zeta),\mathcal{A}}(\kappa)$ and $\mathrm{IDEAL}_{\mathcal{F},\mathsf{Env}(\zeta),\mathsf{Sim}}(\kappa)$, to denote the output distribution of Env in the real-world and in the ideal-world (where now the output may contain more than one bit), we say that a protocol π UC-emulates an ideal functionality $\mathcal{F}$ with everlasting security, if for every polynomial-time real-world adversary $\mathcal{A}$ there exists an ideal-world polynomial-time simulator Sim such that for any polynomial-time environment Env and any input ζ to Env, the random variables $\left\{\mathrm{REAL}_{\pi,\mathsf{Env}(\zeta),\mathcal{A}}(\kappa)\right\}_{\kappa}$, and $\left\{\mathrm{IDEAL}_{\mathcal{F},\mathsf{Env}(\zeta),\mathsf{Sim}}(\kappa)\right\}_{\kappa}$ are *statistically* indistinguishable. Therefore, in general, in order to prove security it is enough to show that the view of the environment in the real-world is statistically-close to the view of the environment in the ideal-world.

We mention that the composition theorems of UC-security hold for protocols with everlasting security (i.e., the composition of two protocols with everlasting security results in a protocol with everlasting security). For a formal definition and statement of the composition theorem, the reader is referred to [MU10].