Succinct Non-Interactive Zero-Knowledge Proofs with Preprocessing for LOGSNP

Yael Tauman Kalai*
M.I.T
yael@csail.mit.edu

Ran Raz†
Weizmann Institute of Science
ran.raz@weizmann.ac.il

August 2, 2006

Abstract

Let $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a Boolean formula of size d, or more generally, an arithmetic circuit of degree d, known to both Alice and Bob, and let $y \in \{0,1\}^m$ be an input known only to Alice. Assume that Alice and Bob interacted in the past in a preamble phase (that is, applied a preamble protocol that depends only on the parameters, and not on Λ, y). We show that Alice can (non-interactively) commit to y, by a message of size $\text{poly}(m, \log d)$, and later on prove to Bob any N statements of the form $\Lambda(x_1, y) = z_1, \ldots, \Lambda(x_N, y) = z_N$ by a (computationally sound) non-interactive zero-knowledge proof of size $\text{poly}(d, \log N)$. (Note the logarithmic dependence on N). We give many applications and motivations for this result. In particular, assuming that Alice and Bob applied in the past the (poly-logarithmic size) preamble protocol:

1. Given a CNF formula $\Psi(w_1, \ldots, w_m)$ of size N, Alice can prove the satisfiability of Ψ by a (computationally sound) non-interactive zero-knowledge proof of size $\text{poly}(m)$. That is, the size of the proof depends only on the size of the witness and not on the size of the formula.
2. Given a language L in the class $\mathcal{LOGSNP}$ and an input $x \in \{0,1\}^n$, Alice can prove the membership $x \in L$ by a (computationally sound) non-interactive zero-knowledge proof of size polylog$n$.
3. Alice can commit to a Boolean formula y of size m, by a message of size $\text{poly}(m)$, and later on prove to Bob any N statements of the form $y(x_1) = z_1, \ldots, y(x_N) = z_N$ by a (computationally sound) non-interactive zero-knowledge proof of size $\text{poly}(m, \log N)$.

Our cryptographic assumptions include the existence of a poly-logarithmic Symmetric-Private-Information-Retrieval (SPIR) scheme, as defined in [CMS99], and the existence of commitment schemes, secure against circuits of size exponential in the security parameter.

*Supported in part by NSF CyberTrust grant CNS-0430450. Part of this work was carried out while the author was at IBM T.J.Watson Research, New York.

†Part of this work was carried out while the author was at Microsoft Research, Redmond.

1 Introduction

The notion of *zero-knowledge proofs*, first introduced by Goldwasser, Micali and Rackoff [GMR89], has become one of the central notions of modern cryptography. Goldreich, Micali and Wigderson showed that for any language $L \in \mathcal{NP}$, the membership $x \in L$ can be proved by an interactive zero-knowledge proof of polynomial size [GMW86]. Kilian showed that for any language $L \in \mathcal{NP}$, the membership $x \in L$ can actually be proved by a succinct interactive zero-knowledge argument of poly-logarithmic size [K92] (see also [M94]).[1] Can the same be done non-interactively ?

Two models for non-interactive zero-knowledge proofs were suggested in the literature. The first model, introduced by Blum, Feldman and Micali [BFM88], is the *common random string* model, where one assumes that the prover and the verifier share a common random string. The second model, introduced by De Santis, Micali and Persiano [DMP88], is the *non-interactive zero-knowledge with preprocessing* model, where the prover and the verifier are allowed to interact in a preamble phase that depends only on the parameters for the problem (before the prover and the verifier see the actual statement that the prover needs to prove). We note that the second model is a stronger one, since the prover and the verifier can use the preamble phase to generate a common random string.

Most works in the literature consider the common random string model. In particular, Blum, Feldman and Micali showed that if the prover and the verifier share a common random string, then for any language $L \in \mathcal{NP}$, the membership $x \in L$ can be proved by a non-interactive zero-knowledge proof of polynomial size [BFM88]. An outstanding open problem is to show that the membership $x \in L$ can be proved by shorter non-interactive zero-knowledge arguments.

In this paper, we show that if the prover and the verifier interacted in a (poly-logarithmic size) preamble phase, then some statements can be proved by relatively short non-interactive zero-knowledge arguments. In particular, we show that in this model the satisfiability of a CNF formula $\Psi(w_1, \ldots, w_m)$ of any size can be proved by a non-interactive zero-knowledge argument of size $\mathrm{poly}(m)$, and we show that for any language $L \in \mathcal{LOGSNP}$ the membership $x \in L$ can be proved by a non-interactive zero-knowledge argument of size $\mathrm{polylog}n$.

1.1 Main Result

Let $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a Boolean formula of size d, or more generally, an arithmetic circuit that computes a polynomial of total degree at most d over $GF[2]$. Assume that Λ is known to both the prover and the verifier. Let $y \in \{0,1\}^m$ be an input known only to the prover. Let k be a security parameter and let N be an additional parameter. Our main result is a protocol that works in 3 phases:

1. **Preamble phase:** This phase depends only on the parameters d, k, N, and not on Λ or y. The exchanged messages are of size $\mathrm{poly}(k, \log d, \log N)$ (and so is the complexity of the parties in this phase).

[1] An argument is a *computationally sound* proof.

2. **Commitment phase:** In this phase the prover commits to the input y by sending one message of size $\text{poly}(m, k, \log d)$ (and so is the complexity of the prover in this phase).

3. **Proof phase:** In this phase the prover and the verifier are given $x_1, \ldots, x_N \in \{0,1\}^n$ and $z_1, \ldots, z_N \in \{0,1\}$. The prover proves that $\forall i : \Lambda(x_i, y) = z_i$ by sending one message of size $\text{poly}(d, k, \log N)$. The proof is zero-knowledge and is computationally sound with soundness exponentially small in k. The proof can then be verified by the verifier. The complexity of both parties in this phase is $\text{poly}(d, k, N, |\Lambda|)$, where $|\Lambda|$ is the size of the arithmetic circuit (or Boolean formula) Λ.

Our protocol works as well in the adaptive case, where y is generated by the (possibly cheating) prover after the preamble phase, and $\Lambda, x_1, \ldots, x_N, z_1, \ldots, z_N$ are generated by the (possibly cheating) prover after the commitment phase.

Note that the size of all messages is at most $\text{poly}(m, d, k, \log N)$. The main point of the protocol is the logarithmic dependence of the size of messages on N, and the fact that after the preamble phase (that doesn't depend on Λ, y), the protocol is non-interactive. Note that the complexity of both parties is $\text{poly}(d, k, N, |\Lambda|)$. That is, the complexity of both parties is polynomial in N and not logarithmic. This is necessary because the parties have to read $x_1, \ldots, x_N$ and $z_1, \ldots, z_N$.

We note that the protocol works as well for arithmetic circuits over any finite field. For simplicity of the presentation, we present our results over $GF[2]$. We also note that the protocol works as well when the polynomial computed by the arithmetic circuit Λ is of total degree at most d only in the second set of variables (i.e., in the y variables). The total degree in the first set of variables (i.e., in the x variables) can be arbitrarily large. Since any Boolean formula of size d may be translated into an arithmetic formula of degree d and size $O(d)$, the result for arithmetic circuits is more general than the one for Boolean formulas.

Our results (including all applications discussed below) rely on several cryptographic hardness assumptions of "exponential nature", including the existence of a poly-logarithmic *Symmetric-Private-Information Retrieval* (SPIR) scheme, and the existence of *commitment* schemes, secure against circuits of size exponential in the security parameter. The "exponential nature" of our cryptographic assumptions is necessary, because of the "poly-logarithmic nature" of the communication between the prover and the verifier in our protocol. For a detailed description of the assumptions that we rely on, see Section 3.2.

In the rest of the introduction we describe several applications and motivations of the main result. For simplicity of the presentation, we ignore in the rest of the introduction the security parameter k. The size of all messages and the complexity of the prover and the verifier depend polynomially on k, and the soundness obtained is always exponentially small in k.

1.2 Applications

1.2.1 Proofs for Satisfiability of CNF Formulas

Let $\Psi(y_1, \ldots, y_m)$ be a CNF formula of size N, and think of N as significantly larger than m. It is well known that if the prover and the verifier interacted in a preamble phase, or

if the prover and the verifier share a common random string, the satisfiability of Ψ can be proved by a non-interactive zero-knowledge proof of size poly(N) [BFM88]. It follows easily from our main result that if the prover and the verifier interacted in a preamble phase, the satisfiability of Ψ can be proved by a non-interactive zero-knowledge argument of size poly(m), which may be significantly smaller than poly(N).

More generally, if Ψ is of the form

$$\Psi(y_1, \ldots, y_m) = \bigwedge_{i=1}^{N} \Psi_i(y_1, \ldots, y_m),$$

where each Ψ_i is an arbitrary Boolean formula of size poly(m), the satisfiability of Ψ can be proved by a non-interactive zero-knowledge argument of size poly($m, \log N$). To prove this result, we take Λ to be an arithmetic circuit that applies a Boolean formula x on an input y (it is not hard to construct such a circuit of size and degree poly(m, n)), and we take $x_1, \ldots, x_N$ to be the N formulas $\Psi_1, \ldots, \Psi_N$, and $z_1 = \ldots = z_N = 1$. We take y to be a witness for the satisfiability of Ψ. Both, the commitment phase and the proof phase of our protocol are unified to be the non-interactive zero-knowledge argument for the satisfiability of Ψ.

1.2.2 Proofs for Membership in $\mathcal{LOGSNP}$ Languages

A related result that we obtain is that there are very short non-interactive zero-knowledge arguments for membership in $\mathcal{LOGSNP}$ languages:

As mentioned before, (if the prover and the verifier interacted in a preamble phase or if the prover and the verifier share a common random string) for any language $L \in \mathcal{NP}$, the membership $x \in L$ can be proved by a polynomial-size non-interactive zero-knowledge proof [BFM88]. An interesting class that lays in between $\mathcal{P}$ and $\mathcal{NP}$ is the class $\mathcal{LOGSNP}$, first defined by Papadimitriou and Yannakakis [PY96]. The class $\mathcal{LOGSNP}$ contains languages such as LOG CLIQUE, RICH HYPERGRAPH COVER, DOMINATING TOURNAMENT SET, and many other languages in $\mathcal{NP}$ with a relatively short witness-size. In particular, RICH HYPERGRAPH COVER, and DOMINATING TOURNAMENT SET, as well as several other languages, are known to be complete languages for $\mathcal{LOGSNP}$ [PY96].

In this paper, we show that (if the prover and the verifier interacted in a preamble phase), for any language $L \in \mathcal{LOGSNP}$, the membership $x \in L$ can be proved by a non-interactive zero-knowledge argument of poly-logarithmic size. Interestingly, to prove this result we need to use our protocol for an arithmetic circuit Λ with degree substantially smaller than its size.

1.2.3 How to Commit to a Formula

An interesting subcase of our main result is obtained when Λ is an arithmetic circuit (of size and degree poly(m)) that applies a Boolean formula y of size m on an input x of size $n \leq m$. In the commitment phase of our protocol, the prover commits to a Boolean formula y, by a message of size poly(m). Later on, the prover can prove any N statements of the form $y(x_1) = z_1, \ldots, y(x_N) = z_N$ by a non-interactive zero-knowledge argument of size poly($m, \log N$).

The following are examples for situations where such a protocol may be useful. The main drawback of our protocol, in these contexts, is that it works only for Boolean formulas and not for Boolean circuits.

- Alice claims that she found a short formula for factoring integers, but of course she doesn't want to reveal it. Bob sends Alice N integers $x_1, \ldots, x_N$ and indeed Alice menages to factor all of them correctly. But how can Bob be convinced that Alice really applied her formula, and not, say, her quantum computer ? We suggest that Alice commits to her formula, and then prove that she actually used that formula to factor $x_1, \ldots, x_N$.

- We want to run a chess contest between formulas. Obviously, the parties don't want to reveal their formulas (e.g., because they don't want their rival formulas to plan their next moves according to it). Of course we can just ask the parties to send their next move at each step. But how can we make sure that the parties actually use their formulas, and don't have teams of grand-masters working for them ? We suggest that the parties commit to their formulas before the contest starts and after the contest ends prove that they actually played according to the formula that they committed to.

1.2.4 The Fall of the Malicious Formulas

The main task of cryptography is to protect honest parties from malicious parties in interactive protocols. Assume that in an interaction between Alice and Bob, Alice is supposed to follow a certain protocol Λ. That is, on an input x, she is supposed to output $\Lambda(x, y)$, where y is her secret key. How can Bob make sure that Alice really follows the protocol Λ ? A standard solution, effective for many applications, is to add a commitment phase and a proof phase as follows: Before the interactive protocol starts, Alice is required to commit to her secret key y. After the interactive protocol ends, Alice is required to prove that she actually acted according to Λ, that is, on inputs $x_1, \ldots, x_N$, her outputs were $\Lambda(x_1, y), \ldots, \Lambda(x_N, y)$. In other words, Alice is required to prove N statements of the form $\Lambda(x_i, y) = z_i$. Typically, we want the proof to be zero-knowledge, since Alice doesn't want to reveal her secret key.

Suppose that we want the final proof phase to be non-interactive. Thus, Alice has to prove (non-interactively) N statements of the form $\Lambda(x_i, y) = z_i$. The only known ways to do that is by a proof of length $N \cdot q$, where q is the size of proof needed for proving one statement of the form $\Lambda(x_i, y) = z_i$. Note that $N \cdot q$ may be significantly larger than the total size of all other messages communicated between Alice and Bob.

Our main result shows that if Λ is a Boolean formula, there is a much more efficient way. Before the interactive protocol starts, Alice and Bob can apply the preamble phase and the commitment phase of our protocol. After the interactive protocol ends, Alice can apply the proof phase of our protocol, and prove to Bob N statements of the form $\Lambda(x_i, y) = z_i$, by sending one message of size $\mathrm{poly}(|\Lambda|, \log N)$.

1.2.5 Proofs of Knowledge of a Witness

Assume that both Alice and Bob have access to a very large database $[(x_1, z_1), \ldots, (x_N, z_N)]$. They face a learning problem: Their goal is to learn a small Boolean formula y (or a small

formula y in some subclass of formulas $\mathcal{C}$, such as DNF formulas) that explains the database. That is, the goal is to learn y such that $y(x_1) = z_1, \ldots, y(x_N) = z_N$. Alice claims that she managed to learn such a formula y, but she doesn't want to reveal it. Assume that Alice and Bob have already applied in the past the preamble phase of our protocol. It follows easily from our main result that Alice can prove to Bob the existence of such a y by a non-interactive zero-knowledge argument of size $\text{poly}(m, \log N)$, where m is the size of y.

To see this, take Λ to be, as before, an arithmetic circuit (of size and degree $\text{poly}(m)$) that applies a Boolean formula y of size m on an input x of size $n \leq m$. Both, the commitment phase and the proof phase of our protocol are unified to be the non-interactive zero-knowledge argument that Alice has a formula y that satisfies all equations.

2 Preliminaries

2.1 Notations and Definitions

We denote the security parameter by k. A function $\mu(k)$ is negligible in k (or just negligible) if for every polynomial $p(k)$ there exists a value K such that for all $k > K$ it holds that $\mu(k) < 1/p(k)$.

Throughout this paper, we formulate all the algorithms as (probabilistic) circuits. We note that the non-uniform model is chosen for convenience, and these algorithms could have been modeled as (probabilistic) Turing machines as well. For the sake of simplicity, we slightly abuse notations, and we let a circuit refer both to a single circuit and to a family of circuits: one for each input length. Also, we let a circuit refer both to a deterministic circuit and to a randomized circuit. We also let a circuit refer to an interactive (probabilistic) family of circuits. For example, we denote the prover of our protocol by a single circuit P, where actually $P = \{P_n\}$ and each P_n is a probabilistic interactive circuit that takes n bit strings as input.

2.2 Tools

Our protocol uses a number of different tools and primitives, which are briefly described in this section.

2.2.1 Private Information Retrieval (PIR)

A *Private Information Retrieval* (PIR) scheme, a concept introduced by Chor, Goldreich, Kushilevitz, and Sudan [CGKS98], allows a user to retrieve information from a database in a private manner. More formally, the database is modeled as an N bit string $x = (x_1, \ldots, x_N)$, out of which the user retrieves the i'th bit x_i, without revealing any information about the index i. A trivial PIR scheme consists of sending the entire database to the user, thus satisfying the PIR privacy requirement in the information-theoretic sense. A PIR scheme with communication complexity smaller than N is said to be *non-trivial*.

A PIR scheme consists of three algorithms: Q^{PIR}, D^{PIR} and R^{PIR}. The query algorithm Q^{PIR} takes as input a security parameter k, the database size N, and an index $i \in [N]$

(that the user wishes to retrieve from the database). It outputs a query q, which should reveal no information about the index i, together with an additional output s, which is kept secret by the user and will later assist the user in retrieving the desired element from the database. The database algorithm D^{PIR} takes as input a security parameter k, the database $(x_1, \ldots, x_N)$ and a query q, and outputs an answer a. This answer enables the user to retrieve x_i, by applying the retrieval algorithm R^{PIR}, which takes as input a security parameter k, the database size N, an index $i \in [N]$, a corresponding pair (q, s) obtained from the query algorithm, and the database answer a corresponding to the query q. It outputs a value which is supposed to be the i'th value of the database.

In this paper we are interested in *poly-logarithmic* PIR schemes, formally defined by Cachin *et al.* [CMS99], as follows.

Definition 1. [CMS99] *Let Q^{PIR} and D^{PIR} be probabilistic polynomial size circuits, and let R^{PIR} be a deterministic polynomial size circuit. We say that $(Q^{PIR}, D^{PIR}, R^{PIR})$ is a* poly-logarithmic private information retrieval scheme *if there exist constants $a', b', c', d' > 0$, such that:*

1. (Correctness:) $\forall N$, $\forall$*database* $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, $\forall i \in [N]$, *and* $\forall k$,
$\Pr[R^{PIR}(1^k, N, i, (q,s), a) = x_i \mid (q,s) \leftarrow Q^{PIR}(1^k, N, i), a \leftarrow D^{PIR}(1^k, x, q)] \geq 1 - 2^{-a'k}$,
and the output of Q^{PIR} and D^{PIR} is of size $\leq \text{poly}(k, \log N)$.

2. (User Privacy:) $\forall N$, $\forall i, j \in [N]$, $\forall k$ *such that* $2^k \geq N^{b'}$, *and* $\forall$*adversary* $\mathcal{A}$ *of size* $2^{c'k}$,
$$\Big|\Pr[\mathcal{A}(1^k, N, q) = 1 \mid (q,s) \leftarrow Q^{PIR}(1^k, N, i)] - \Pr[\mathcal{A}(1^k, N, q) = 1 \mid (q,s) \leftarrow Q^{PIR}(1^k, N, j)]\Big| \leq 2^{-d'k}.$$

In this paper, we use the following equivalent definition for a poly-logarithmic PIR.

Definition 2. *Let k be the security parameter and N be the database size. Let Q^{PIR} and D^{PIR} be probabilistic circuits, and let R^{PIR} be a deterministic circuit. We say that $(Q^{PIR}, D^{PIR}, R^{PIR})$ is a* poly-logarithmic private information retrieval scheme *if the following conditions are satisfied:*

1. (Size Restriction:) *Q^{PIR} and R^{PIR} are of size $\leq \text{poly}(k, \log N)$, and D^{PIR} is of size $\leq \text{poly}(k, N)$. The output of Q^{PIR} and D^{PIR} is of size $\leq \text{poly}(k, \log N)$.*

2. (Correctness:) $\forall N$, $\forall k$, $\forall$*database* $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, *and* $\forall i \in [N]$,
$\Pr[R^{PIR}(k, N, i, (q,s), a) = x_i \mid (q,s) \leftarrow Q^{PIR}(k, N, i), a \leftarrow D^{PIR}(k, x, q)] \geq 1 - 2^{-k^3}$.

3. (User Privacy:) $\forall N$, $\forall k$, $\forall i, j \in [N]$, *and* $\forall$*adversary* $\mathcal{A}$ *of size at most* 2^{k^3},
$$\Big|\Pr[\mathcal{A}(k, N, q) = 1 \mid (q,s) \leftarrow Q^{PIR}(k, N, i)] - \Pr[\mathcal{A}(k, N, q) = 1 \mid (q,s) \leftarrow Q^{PIR}(k, N, j)]\Big| \leq 2^{-k^3}.$$

For the purpose of this paper, it suffices to prove the following claim.

Claim 1. *The existence of a poly-logarithmic PIR scheme according to Definition 1 implies the existence of a poly-logarithmic PIR scheme according to Definition 2.*

Proof of Claim 1. Let $(Q^{PIR}, D^{PIR}, R^{PIR})$ be any poly-logarithmic PIR scheme according to Definition 1, with constants $a', b', c', d' > 0$. Define $(\tilde{Q}^{PIR}, \tilde{D}^{PIR}, \tilde{R}^{PIR})$ as follows:

1. $\tilde{Q}^{PIR}(k, N, i) = Q^{PIR}(1^{k'}, N, i)$
2. $\tilde{D}^{PIR}(k, x, q) = D^{PIR}(1^{k'}, x, q)$
3. $\tilde{R}^{PIR}(k, N, i, (q, s), a) = R^{PIR}(1^{k'}, N, i, (q, s), a)$,

where $k' = \max\left\{b' \log N, \frac{k^3}{a'}, \frac{k^3}{c'}, \frac{k^3}{d'}\right\}$. We next argue that $(\tilde{Q}^{PIR}, \tilde{D}^{PIR}, \tilde{R}^{PIR})$ is a poly-logarithmic PIR scheme according to Definition 2. It is easy to see that the *size restriction* property holds. The *correctness* property follows from the correctness property of Definition 1 and from the fact that $k' \geq \frac{k^3}{a'}$. The *user privacy* property follows from the user privacy property Definition 1 and from the fact that $k' \geq \max\left\{b' \log N, \frac{k^3}{c'}, \frac{k^3}{d'}\right\}$. ■

2.2.2 Symmetric Private Information Retrieval

In the standard formulation of PIR, there is no concern about how many entries of the database the user learns. If one makes an additional requirement that the user must learn only one entry of the database, then this is called a symmetric private information retrieval (SPIR) scheme.

Definition 3. (Symmetric Private Information Retrieval Scheme): *Let $(Q^{PIR}, D^{PIR}, R^{PIR})$ be a poly-logarithmic PIR scheme according to Definition 2. We say that $(Q^{PIR}, D^{PIR}, R^{PIR})$ is a* symmetric private information retrieval (SPIR) scheme *if the additional following condition holds.*

- Data Privacy: $\forall N$, $\forall k$, $\forall q$, $\exists$*index* $i \in [N]$, *such that* $\forall$*databases* $x = (x_1, \ldots, x_N) \in \{0,1\}^N$ *and* $y = (y_1, \ldots, y_N) \in \{0,1\}^N$ *with* $x_i = y_i$, *and for every deterministic (not necessarily polynomial size) function* $\hat{R}$ *(thought of as the adversary)*

$$\Big|\Pr[\hat{R}(k, N, q, a) = 1 \mid a \leftarrow D^{PIR}(k, x, q)] - \Pr[\hat{R}(k, N, q, a) = 1 \mid a \leftarrow D^{PIR}(k, y, q)]\Big| \leq 2^{-k^3}.$$

In other words, $D^{PIR}(k, x, q)$ and $D^{PIR}(k, y, q)$ are statistically indistinguishable (with error $\leq 2^{-k^3}$). Moreover, if there exists an index $i' \in [N]$ such that $(q, s) \in \text{Support}(Q^{PIR}(k, N, i'))$ for some string s (i.e., $\Pr[(q, s) = Q^{PIR}(k, N, i')] > 0$ for some string s), then the data privacy condition holds with respect to $i = i'$.[2]

[2]We note that if the correctness condition of the PIR scheme was error-free then this statement would have been a direct consequence of the previous statement. The justification for adding this as a condition, is that [NP99] gave a general method for converting PIR schemes into SPIR schemes, and every SPIR scheme obtained from this methodology satisfies this condition.

Remark 1. Definitions 1, 2 and 3 consider only PIR and SPIR schemes where the database $(x_1, \ldots, x_N)$ consists of bits (i.e., $x_i \in \{0,1\}$). In our protocol we use a SPIR scheme where each x_i is a string in $\{0,1\}^t$. We can adjust our definition of a poly-logarithmic SPIR scheme to deal with strings in the straightforward manner. Also, any poly-logarithmic SPIR scheme $(Q^{SPIR}, D^{SPIR}, R^{SPIR})$ can be easily converted into a poly-logarithmic SPIR scheme $(\vec{Q}^{SPIR}, \vec{D}^{SPIR}, \vec{R}^{SPIR})$ in which the database consists of strings (rather than bits). This is done by thinking of the database d as t databases $d^1, \ldots, d^t$, where d^j is the database in which each entry consists of the j'th bit of the corresponding entry in d. The query algorithm $\vec{Q}^{SPIR}$ is identical to Q^{SPIR}. The database algorithm $\vec{D}^{SPIR}$ runs D^{SPIR} t times, once for each d^j. The retrieval algorithm $\vec{R}^{SPIR}$ runs R^{SPIR} t times, once for each answer of D^{SPIR}. Throughout this paper, when we refer to a SPIR scheme, we think of one in which the database consists of strings (rather than bits). For the sake of simplicity of notations, throughout the rest of this section (and in particular in Claim 2, Claim 3, and Corollary 1) we consider databases that consist only of bits. We note that when considering databases whose entries are strings, the proofs of Claim 2, Claim 3, and Corollary 1 remain essentially the same.

Remark 2. A poly-logarithmic SPIR scheme is similar to what is known as $\binom{N}{1}$-OT scheme. The difference being in the communication complexity. Namely, a $\binom{N}{1}$-OT scheme has communication complexity $\mathrm{poly}(N,k)$, whereas a poly-logarithmic SPIR scheme has communication complexity $\mathrm{poly}(\log N, k)$.

Claim 2. *Let $(Q^{PIR}, D^{PIR}, R^{PIR})$ be a poly-logarithmic PIR scheme according to Definition 2. Then $\forall N$, $\forall k \geq \log N$, and $\forall$adversary $\mathcal{B}$ of size $\leq 2^{k^3}$,*

$$\Pr[\mathcal{B}(k, N, q) = i] \leq \frac{2}{N},$$

where the probability is over $i \in_R [N]$, over q which is distributed according to $(q,s) \leftarrow Q^{PIR}(k, N, i)$, and over the random coin tosses of $\mathcal{B}$.

Proof of Claim 2. Assume for the sake of contradiction that there exists N, there exists $k \geq \log N$, and there exists an adversary $\mathcal{B}$ of size $\leq 2^{k^3}$ such that

$$\Pr[\mathcal{B}(k, N, q) = i \mid (q,s) \leftarrow Q^{PIR}(k, N, i)] > \frac{2}{N}$$

(where the probability is over $i \in_R [N]$, over the randomness of Q^{PIR} and $\mathcal{B}$). Notice that for every fixed $j \in [N]$,

$$\Pr[\mathcal{B}(k, N, q) = i \mid (q,s) \leftarrow Q^{PIR}(k, N, j)] \leq \frac{1}{N}$$

(where the probability is over $i \in_R [N]$ and over the randomness of Q^{PIR} and $\mathcal{B}$). This is the case since i is a random variable which is independent of q and since $i \in_R [N]$. Thus, for

every fixed $j \in [N]$,

$$\Pr[\mathcal{B}(k,N,q) = i \mid (q,s) \leftarrow Q^{PIR}(k,N,i)] -$$
$$\Pr[\mathcal{B}(k,N,q) = i \mid (q,s) \leftarrow Q^{PIR}(k,N,j)] > \frac{1}{N}$$

(where the probabilities are over $i \in_R [N]$ and over the randomness of Q^{PIR} and $\mathcal{B}$). This implies that for every $j \in [N]$ there exists $i \in [N]$ such that

$$\Pr[\mathcal{B}(k,N,q) = i \mid (q,s) \leftarrow Q^{PIR}(k,N,i)] -$$
$$\Pr[\mathcal{B}(k,N,q) = i \mid (q,s) \leftarrow Q^{PIR}(k,N,j)] > \frac{1}{N}$$

(where the probabilities are over the randomness of Q^{PIR} and $\mathcal{B}$). This contradicts the *user privacy* condition of the underlying PIR scheme, since $k \geq \log N$, which implies that $\frac{1}{N} \geq 2^{-k^3}$. ■

Claim 3. *Let $(Q^{SPIR}, D^{SPIR}, R^{SPIR})$ be a poly-logarithmic SPIR scheme according to Definition 3. Then $\forall N$, $\forall$large enough $k \geq \log N$, and $\forall$adversary $\mathcal{B}$ of size $\leq 2^{k^2}$,*

$$\Pr[\mathcal{B}(k,N,q_1,\ldots,q_k) = (i_1,\ldots,i_k)] \leq \left(\frac{3}{N}\right)^k,$$

where the probability is over $i_1,\ldots,i_k \in_R [N]$, over $(q_1,\ldots,q_k)$ where each q_j is distributed according to $(q_j,s_j) \leftarrow Q^{SPIR}(k,N,i_j)$, and over the random coin tosses of $\mathcal{B}$.

In the proof of Claim 3 we make use of the following Lemma. The lemma is taken from a Tech Report by Oded Goldreich, that can be found on his homepage [G05].

Lemma 1. [Goldreich] *Fix any two functions $F_1, F_2 : \{0,1\}^* \to \{0,1\}^*$, and fix any two independent probability ensembles $\{Y_m\}$ and $\{Z_m\}$ such that $Y_m, Z_m \in \{0,1\}^m$. Let $\rho_1(\cdot)$ be an upper-bound on the success probability of $s_1(\cdot)$-size circuits in computing F_1 over $\{Y_m\}$. That is, for every family of circuits $\{C_m\}$ of size $s_1(\cdot)$,*

$$\Pr[C_m(Y_m) = F_1(Y_m)] \leq \rho_1(m).$$

Likewise, let $\rho_2(\cdot)$ be an upper-bound on the probability that $s_2(\cdot)$-size circuits compute F_2 over $\{Z_m\}$. For any function $\ell : \mathbb{N} \to \mathbb{N}$, define $\{X_n\}$ to be the probability ensemble such that $X_n = (Y_{\ell(n)}, Z_{n-\ell(n)})$, and let F be the direct product function defined by $F(y,z) \stackrel{\text{def}}{=} (F_1(y), F_2(z))$, where $|y| = \ell(|yz|)$. Then, for every function $\epsilon : \mathbb{N} \to \mathbb{R}$, the function $\rho(\cdot)$ defined as

$$\rho(n) \stackrel{\text{def}}{=} \rho_1(\ell(n)) \cdot \rho_2(n - \ell(n)) + \epsilon(n)$$

is an upper-bound on the probability that families of $s(\cdot)$-size circuits correctly compute F over $\{X_n\}$, where

$$s(n) \stackrel{\text{def}}{=} \min\left\{ s_1(\ell(n)), \frac{s_2(n-\ell(n))}{\text{poly}(n/\epsilon(n))} \right\}.$$

Notice that the statement of Lemma 1 is not symmetric with respect to F_1 and F_2. In the proof of Claim 3, we make use of Lemma 1, and in particular, we use a careful induction that capitalizes on the asymmetry of Lemma 1.

Proof of Claim 3. Assume for the sake of contradiction that there exists N, there exists a large enough $k \geq \log N$, and there exists an adversary $\mathcal{B}$ of size $\leq 2^{k^2}$, such that

$$\Pr[\mathcal{B}(k, N, q_1, \ldots, q_k) = (i_1, \ldots, i_k)] > \left(\frac{3}{N}\right)^k \tag{1}$$

(where the probability is over $i_1, \ldots, i_k \in_R [N]$, over $(q_1, \ldots, q_k)$, where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$, and over the random coin tosses of $\mathcal{B}$).

We assume for simplicity, and without loss of generality, that $\mathcal{B}$ is a deterministic circuit. Notice that the parameters k, N can be hardwired into $\mathcal{B}$, and thus we can assume without loss of generality that

$$\Pr[\mathcal{B}(q_1, \ldots, q_k) = (i_1, \ldots, i_k)] > \left(\frac{3}{N}\right)^k$$

(where the probability is over $i_1, \ldots, i_k \in_R [N]$ and over $(q_1, \ldots, q_k)$, where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$).

Fix N and k as above. According to the correctness condition of the SPIR scheme, for every database $x = (x_1, \ldots, x_N)$ and for every $i \in [N]$,

$$\Pr[R^{SPIR}(k, N, i, (q, s), a) = x_i \mid (q, s) \leftarrow Q^{SPIR}(k, N, i), a \leftarrow D^{SPIR}(k, x, q)] \geq 1 - 2^{-k^3}.$$

For every $i \in [N]$, we denote by E_i the set of all queries corresponding to the i'th database entry, for which the correctness condition holds with overwhelming probability (over a uniformly chosen database). Precisely,

$$\begin{aligned} E_i \stackrel{\text{def}}{=} \{q : \quad & \exists s \in \{0,1\}^* \text{ s.t. } (q, s) \in \text{Support}(Q^{SPIR}(k, N, i)) \text{ and} \\ & \Pr[R^{SPIR}(k, N, i, (q, s), a) = x_i \mid x \in_R \{0,1\}^N, a \leftarrow D^{SPIR}(k, x, q)] \geq 1 - 2^{-k^2}\}. \end{aligned}$$

The correctness condition of the SPIR scheme implies that the following holds.

Claim 4. *For every $i \in [N]$,*

$$\Pr[q \in E_i \mid (q, s) \leftarrow Q^{SPIR}(k, N, i)] \geq 1 - 2^{-k^2}. \tag{2}$$

Proof of Claim 4. Assume for the sake of contradiction that there exists an $i \in [N]$ for which Inequality (2) does not hold. For convenience, denote by

$$p \stackrel{\text{def}}{=} \Pr[q \in E_i \mid (q, s) \leftarrow Q^{SPIR}(k, N, i)].$$

Then for a uniformly chosen database $x \in \{0,1\}^N$,

$$\Pr[R^{SPIR}(k,N,i,(q,s),a) = x_i \mid (q,s) \leftarrow Q^{SPIR}(k,N,i), a \leftarrow D^{SPIR}(k,x,q)] =$$
$$\Pr[R^{SPIR}(k,N,i,(q,s),a) = x_i \mid (q,s) \leftarrow Q^{SPIR}(k,N,i), a \leftarrow D^{SPIR}(k,x,q), q \in E_i] \cdot p +$$
$$\Pr[R^{SPIR}(k,N,i,(q,s),a) = x_i \mid (q,s) \leftarrow Q^{SPIR}(k,N,i), a \leftarrow D^{SPIR}(k,x,q), q \notin E_i] \cdot (1-p) <$$
$$p + (1-2^{-k^2})(1-p) =$$
$$(1-2^{-k^2}) + 2^{-k^2}p <$$
$$(1-2^{-k^2}) + 2^{-k^2}(1-2^{-k^2}) =$$
$$1 - 2^{-k^2} \cdot 2^{-k^2} \leq$$
$$1 - 2^{-k^3}.$$

This contradicts the correctness condition of the SPIR scheme. ■

The database privacy condition of the SPIR scheme implies that the following holds.

Claim 5. *For every distinct $i, j \in [N]$, $E_i \cap E_j = \emptyset$.*

Proof of Claim 5. Assume for the sake of contradiction that there exist distinct $i, j \in [N]$ and there exists a query q such that $q \in E_i \cap E_j$. This implies that there exist $s_i, s_j \in \{0,1\}^*$ such that

1. $(q, s_i) \in \text{Support}(Q^{SPIR}(k,N,i))$,
2. $(q, s_j) \in \text{Support}(Q^{SPIR}(k,N,j))$,
3. $\Pr[R^{SPIR}(k,N,i,(q,s_i),a) = x_i \mid x \in_R \{0,1\}^N, a \leftarrow D^{SPIR}(k,x,q)] \geq 1 - 2^{-k^2}$,
4. $\Pr[R^{SPIR}(k,N,j,(q,s_j),a) = x_j \mid x \in_R \{0,1\}^N, a \leftarrow D^{SPIR}(k,x,q)] \geq 1 - 2^{-k^2}$.

Consider the adversary $\hat{R}$ (that has the values s_i and s_j hardwired into it), that on input (k,N,q,a) outputs $R^{SPIR}(k,N,i,(q,s_i),a) \oplus R^{SPIR}(k,N,j,(q,s_j),a)$. In order to contradict the data privacy condition of the SPIR scheme, it remains to notice that items (3) and (4) above imply that for every index $\ell \in [N]$ there exist databases $x = (x_1, \ldots, x_N)$ and $y = (y_1, \ldots, y_N)$ such that $x_\ell = y_\ell$, $x_i \oplus x_j \neq y_i \oplus y_j$, and such that

1. $\Pr[R^{SPIR}(k,N,i,(q,s_i),a) = x_i \mid a \leftarrow D^{SPIR}(k,x,q)] \geq 1 - 2^{-k}$,
2. $\Pr[R^{SPIR}(k,N,j,(q,s_j),a) = x_j \mid a \leftarrow D^{SPIR}(k,x,q)] \geq 1 - 2^{-k}$,
3. $\Pr[R^{SPIR}(k,N,i,(q,s_i),a) = y_i \mid a \leftarrow D^{SPIR}(k,y,q)] \geq 1 - 2^{-k}$,
4. $\Pr[R^{SPIR}(k,N,j,(q,s_j),a) = y_j \mid a \leftarrow D^{SPIR}(k,y,q)] \geq 1 - 2^{-k}$.

■

Let $H^{(t)}$ be the function that on input $(q_1, \ldots, q_t)$ outputs $(i_1, \ldots, i_t)$, such that for every $j \in [t]$, if $q_j \in E_\ell$ (for some $\ell \in [N]$) then $i_j = \ell$, and if $q_j \notin E_1 \cup \ldots \cup E_N$ then $i_j = \perp$. Note that Claim 5 implies that $H^{(t)}$ is well defined. Recall that according to Claim 4,

$$\Pr[q \in E_1 \cup \ldots \cup E_N] \geq 1 - 2^{-k^2} \tag{3}$$

(where the probability is over $(q, s) \leftarrow Q^{SPIR}(k, N, i)$ where $i \in_R [N]$).

Inequality (3) together with Claim 2, imply that that the following holds.

Claim 6. *For every circuit C of size $\leq 2^{k^3}$,*

$$\Pr[C(q) = H^{(1)}(q)] \leq \frac{2.6}{N}$$

(where the probability is over $(q, s) \leftarrow Q^{SPIR}(k, N, i)$ where $i \in_R [N]$).

Proof of Claim 6. For $(q, s) \leftarrow Q^{SPIR}(k, N, i)$, where $i \in_R [N]$, the following holds:

$$\begin{aligned}
&\Pr[C(q) = H^{(1)}(q)] = \\
&\Pr[C(q) = H^{(1)}(q) \mid q \in E_1 \cup \ldots \cup E_N] \cdot \Pr[q \in E_1 \cup \ldots \cup E_N] + \\
&\Pr[C(q) = H^{(1)}(q) \mid q \notin E_1 \cup \ldots \cup E_N] \cdot \Pr[q \notin E_1 \cup \ldots \cup E_N] \ \leq \\
&\Pr[C(q) = H^{(1)}(q) \mid q \in E_1 \cup \ldots \cup E_N] + \Pr[q \notin E_1 \cup \ldots \cup E_N] \leq \\
&\Pr[C(q) = H^{(1)}(q) \mid q \in E_1 \cup \ldots \cup E_N] + 2^{-k^2} = \\
&\Pr[C(q) = i \mid q \in E_1 \cup \ldots \cup E_N] + 2^{-k^2} \leq \\
&\Pr[C(q) = i] \left(\Pr[q \in E_1 \cup \ldots \cup E_N]\right)^{-1} + 2^{-k^2} \leq \\
&\frac{2}{N}(1 - 2^{-k^2})^{-1} + 2^{-k^2} < \\
&\frac{2.2}{N} + \left(\frac{1}{N}\right)^k \leq \\
&\frac{1}{N}\left(2.2 + \frac{1}{3}\right) < \\
&\frac{2.6}{N},
\end{aligned}$$

where the last two inequalities follow from the fact that $N \geq 3$ (and $k \geq 2$), which in turn follows from the contradiction assumption (Inequality (1)). ■

Let $m \stackrel{\text{def}}{=} |q|$, and set $p(m) \stackrel{\text{def}}{=} \frac{2.6}{N}$ and $\epsilon(m) \stackrel{\text{def}}{=} 2^{-k^2}$. We first prove by induction on t that circuits of size 2^{k^2} cannot compute $H^{(t)}(q_1, \ldots, q_t)$ with success probability greater than $p(m)^t + \frac{\epsilon(m)}{1-p(m)}$. Notice that the induction basis is guaranteed by Claim 6. The induction step is proved using Lemma 1, with $F_1 = H^{(t-1)}$ and $F_2 = H^{(1)}$, along with $\rho_1((t-1)m) = p(m)^{t-1} + \frac{\epsilon(m)}{1-p(m)}$, $s_1((t-1)m) = 2^{k^2}$, and $\rho_2(m) = p(m)$, $s_2(m) = 2^{k^3}$, and with $\ell(n) = \frac{t-1}{t}n$. The fact that we can use Lemma 1 with F_2, ρ_2, s_2, follows from Claim 2, and the fact that

we can use Lemma 1 with F_1, ρ_1, s_1, follows from the induction hypothesis. Thus, we get that $\rho(tm)$ is an upper-bound on the probability that $s(tm)$-size circuit families correctly compute $H^{(t)}(q_1, \ldots, q_t)$, where

$$\begin{aligned}
\rho(tm) &= \rho_1((t-1)m) \cdot \rho_2(m) + \epsilon(m) \\
&= \left(p(m)^{t-1} + \frac{\epsilon(m)}{1-p(m)} \right) \cdot p(m) + \epsilon(m) \\
&= p(m)^t + \epsilon(m) \left(1 + \frac{p(m)}{1-p(m)} \right) \\
&= p(m)^t + \frac{\epsilon(m)}{1-p(m)},
\end{aligned}$$

and for every large enough k and $t \leq k$,

$$\begin{aligned}
s(tm) &= \min \left\{ s_1((t-1)m), \frac{s_2(m)}{\mathrm{poly}(tm/\epsilon(tm))} \right\} \\
&= \min \left\{ 2^{k^2}, \frac{2^{k^3}}{\mathrm{poly}(tm/\epsilon(tm))} \right\} \\
&= \min \left\{ 2^{k^2}, \frac{2^{k^3}}{\mathrm{poly}(k2^{k^2})} \right\} \\
&= 2^{k^2},
\end{aligned}$$

as desired.

Notice that for $t = k$ we have

$$\begin{aligned}
\rho(tm) = p(m)^k + \frac{\epsilon(m)}{1-p(m)} &= \left(\frac{2.6}{N} \right)^k + \frac{2^{-k^2}}{1 - \frac{2.6}{N}} \\
&= \left(\frac{2.6}{N} \right)^k + \frac{1}{(2^k)^k \left(1 - \frac{2.6}{N}\right)} \\
&\leq \left(\frac{2.6}{N} \right)^k + \frac{1}{N^k \left(1 - \frac{2.6}{N}\right)} \\
&= \frac{1}{N^k} \left(2.6^k + \frac{1}{1 - \frac{2.6}{N}} \right) \\
&\leq \left(\frac{2.7}{N} \right)^k,
\end{aligned}$$

where the last inequality holds for large enough k. Therefore, for N and k as above, we get that for every 2^{k^2}-size circuit C, it holds that

$$\Pr[C(q_1, \ldots, q_k) = H^{(k)}(q_1, \ldots, q_k)] \leq \left(\frac{2.7}{N} \right)^k \tag{4}$$

(where the probability is over $(q_1, \ldots, q_k)$, where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$ and $i_1, \ldots, i_k \in_R [N]$).

Thus, for every 2^{k^2}-size circuit C and for $(q_1, \ldots, q_k)$, such that $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$ and $i_1, \ldots, i_k \in_R [N]$, the following holds:

$$\begin{aligned}
&\Pr[C(q_1, \ldots, q_k) = (i_1, \ldots, i_k)] \leq \\
&\Pr[C(q_1, \ldots, q_k) = H^{(k)}(q_1, \ldots, q_k)] + \Pr[H^{(k)}(q_1, \ldots, q_k) \neq (i_1, \ldots, i_k)] \leq \\
&\left(\frac{2.7}{N}\right)^k + k \cdot 2^{-k^2} \leq \\
&\left(\frac{3}{N}\right)^k,
\end{aligned}$$

where the first inequality is a basic probability inequality, the second inequality follows from Inequality (4) and from Claim 4, and the last inequality follows from the fact that $k \geq \log N$ and is sufficiently large. This contradicts our assumption (Inequality (1)). ■

Corollary 1. *Let $(Q^{SPIR}, D^{SPIR}, R^{SPIR})$ be a poly-logarithmic SPIR scheme according to Definition 3. Then $\forall N$, $\forall$ large enough $k \geq 2 \log N$, and $\forall$adversary $\mathcal{B}$ of size $\leq 2^{k^{1.5}}$,*

$$\Pr\left[\mathcal{B}(k, N, q_1, \ldots, q_k) = (i'_1, \ldots, i'_k) \text{ s.t. } |\{j \in [k] : i'_j = i_j\}| \geq \frac{k}{2}\right] < \left(\frac{12}{N}\right)^{\frac{k}{2}},$$

where the probability is over $i_1, \ldots, i_k \in_R [N]$, over $(q_1, \ldots, q_k)$ where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$, and over the random coin tosses of $\mathcal{B}$.

Proof of Corollary 1. Assume for the sake of contradiction that there exists N, there exists a large enough $k \geq 2 \log N$, and there exists an adversary $\mathcal{B}$ of size $\leq 2^{k^{1.5}}$, such that

$$\Pr\left[\mathcal{B}(k, N, q_1, \ldots, q_k) = (i'_1, \ldots, i'_k) \text{ s.t. } |\{j \in [k] : i'_j = i_j\}| \geq \frac{k}{2}\right] \geq \left(\frac{12}{N}\right)^{\frac{k}{2}} \tag{5}$$

(where the probability is over $i_1, \ldots, i_k \in_R [N]$, over $(q_1, \ldots, q_k)$ where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$, and over the random coin tosses of $\mathcal{B}$).

We show that this contradicts Claim 3, by constructing an adversary $\mathcal{B}'$ of size $\text{poly}(2^{k^{1.5}}) < 2^{(\frac{k}{2})^2}$, such that

$$\Pr[\mathcal{B}'(k, N, q_1, \ldots, q_{\frac{k}{2}}) = (i_1, \ldots, i_{\frac{k}{2}})] > \left(\frac{3}{N}\right)^{\frac{k}{2}}$$

(where the probability is over $i_1, \ldots, i_{\frac{k}{2}} \in_R [N]$, over $(q_1, \ldots, q_{\frac{k}{2}})$ where each q_j is distributed according to $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$, and over the random coin tosses of $\mathcal{B}'$).[3]

[3] We assume for simplicity that k is even.

Algorithm $\mathcal{B}'$: On input $(k, N, q_1, \ldots, q_{\frac{k}{2}})$, Algorithm $\mathcal{B}'$ operates as follows:

1. Choose at random $i_{\frac{k}{2}+1}, \ldots, i_k \in_R [N]$.
2. For each $j \in [\frac{k}{2}+1, k]$, choose $(q_j, s_j) \leftarrow Q^{SPIR}(k, N, i_j)$.
3. Choose a random permutation $\pi : [k] \to [k]$.
4. Compute $(i'_{\pi(1)}, \ldots, i'_{\pi(k)}) \stackrel{\text{def}}{=} \mathcal{B}(k, N, q_{\pi(1)}, \ldots, q_{\pi(k)})$.
5. Output $(i'_1, \ldots, i'_{\frac{k}{2}})$.

Let E denote the event that

$$\left|\{j \in [k] : i'_j = i_j\}\right| \geq \frac{k}{2}.$$

Then,

$$\begin{aligned}
&\Pr[\mathcal{B}'(k, N, q_1, \ldots, q_{\frac{k}{2}}) = (i_1, \ldots, i_{\frac{k}{2}})] \geq \\
&\Pr[\mathcal{B}'(k, N, q_1, \ldots, q_{\frac{k}{2}}) = (i_1, \ldots, i_{\frac{k}{2}}) \mid E] \cdot \Pr[E] \geq \\
&\frac{1}{\binom{k}{k/2}} \cdot \Pr[E] > \frac{1}{2^k} \cdot \left(\frac{12}{N}\right)^{\frac{k}{2}} = \left(\frac{3}{N}\right)^{\frac{k}{2}},
\end{aligned}$$

as desired. We not that the first inequality follows from Bayes' law. The second inequality follows from the definition of event E. The third inequality follows from Inequality (5) and from the definition of event E. ■

2.2.3 Other Useful Primitives

Definition 4. *A (s_1, s_2)-bit commitment scheme $com = \{com_k\}$, where s_1 and s_2 are functions of the security parameter k, satisfies the following properties:*

1. *For every $k \in \mathbb{N}$, com_k is a probabilistic circuit of size* $\text{poly}(k)$*, that takes as input a bit $b \in \{0,1\}$ and outputs a string of length at most* $\text{poly}(k)$ *(corresponding to a commitment to the bit b).*
2. s_1-hiding: *For every probabilistic circuit family $\mathcal{D}$ (called a distinguisher) of size at most* $\text{poly}(s_1(k))$*,*

   $$|\Pr[\mathcal{D}(com_k(0)) = 1] - \Pr[\mathcal{D}(com_k(1)) = 1]| = \text{negl}(s_1(k))$$

   (where the probabilities are over the random coin tosses of com_k and $\mathcal{D}$).
3. *For every k there exists a deterministic circuit C_k of size at most $s_2(k)$ such that for every $b \in \{0,1\}$,*

   $$\Pr[C_k(com_k(b)) = b] = 1$$

   (where the probabilities are over the random coin tosses of com_k). Moreover, $C_k(y) = \perp$, for every y that is not a commitment string (i.e., for every $y \notin \text{Support}(com_k(0)) \cup \text{Support}(com_k(1))$).

Remark. Throughout this paper, we use a (s_1, s_2)-bit commitment scheme to commit to strings rather than to single bits. This is done in a bit by bit manner. Namely, in order to commit to a string $x = (x_1, \ldots, x_t) \in \{0,1\}^t$, we apply our (s_1, s_2)-bit commitment scheme to each x_i separately. For simplicity, we abuse notations, and let $com_k(x)$ denote the random variable $(com_k(x_1), \ldots, com_k(x_t))$.

In the following definitions it may help the reader to think of the security parameter k as a function of the input length n, and to think of $s(k)$ as at least polynomial in n.

Definition 5. *A protocol (P, V) for proving membership in L is said to be s-zero-knowledge, where s is a function of the security parameter k, if the following holds: For every deterministic (interactive) circuit family V^* (thought of as a possibly cheating verifier) of size at most $\mathrm{poly}(s(k))$, there exists a probabilistic circuit family S (known as the simulator) of size $\mathrm{poly}(|V^*|, |P|)$, such that for every probabilistic circuit family $\mathcal{D}$ (called a distinguisher) of size at most $\mathrm{poly}(s(k))$, for every $x \in L$, every $z \in \{0,1\}^*$, and every k,*

$$|\Pr[\mathcal{D}((P, V^*(z))(x)) = 1] - \Pr[\mathcal{D}(S(x, z)) = 1]| = \mathrm{negl}(s(k)).$$

Remark. This definition deviates from the standard definition in three ways.

1. In the standard definition $s(k) = n$, where n is the input length, thus ensuring that any (possibly cheating) verifier, of size at most polynomial in the input length, does not gain any knowledge from the interaction. Our definition ensures that also (possibly larger) verifiers of size $\mathrm{poly}(s(k))$ do not gain any knowledge from the interaction.

2. In the standard definition V^* and S are modeled as probabilistic Turing machines rather than circuits. As we mentioned in Section 2.1, we model these algorithms as circuits only for the sake of convenience.

3. Our definition only considers *deterministic* verifiers V^*. This is without loss of generality. The definition ensures that the zero-knowledge property holds also for randomized verifiers. This is the case, since the randomness can be thought of as part of the auxiliary input.

Definition 6. *A protocol (P, V) for proving membership in L is said to be s-sound, where s is a function of the security parameter k, if the following holds: For every probabilistic (interactive) circuit family P^* (thought of as a possibly cheating prover) of size at most $\mathrm{poly}(s(k))$ and for every $x \notin L$,*

$$\Pr[(P^*, V)(x) = 1] = \mathrm{negl}(s(k)).$$

Definition 7. *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. Then a protocol (P, V) is said to be a s-strong proof-of-knowledge for the relation R, if there exists a negligible function μ and a probabilistic (strict) polynomial time oracle machine K (called the knowledge extractor), such that for every unbounded (interactive) circuit family P^* (thought of as a possibly cheating prover) and every input x, if $\Pr[(P^*, V)(x) = 1] \geq \mu(s(k))$ then the machine K, with input x and oracle access to P^*, outputs a witness w such that $(x, w) \in R$ with probability at least $1 - \mu(s(k))$.*

Remark 1. Notice that the s-strong proof-of-knowledge property implies the s-soundness property. Namely, if (P, V) is a s-strong proof-of-knowledge protocol for a relation R then it is a s-sound protocol for proving membership in the corresponding language

$$L_R \stackrel{\text{def}}{=} \{x : (x, w) \in R\}.$$

When we say that (P, V) is a s-strong proof-of-knowledge protocol for an $\mathcal{NP}$ language $L = L_R$, we mean that it is a s-strong proof-of-knowledge protocol for the corresponding relation R, which is in P.

Remark 2. In our protocol we use a s-strong proof-of-knowledge for the relation

$$R = \{(\vec{q}, (\vec{s}, \vec{w}, \vec{r})) : \forall j,\ (q^j, s^j) = Q^{SPIR}(k, N, w^j; r^j)\}, \tag{6}$$

where $\vec{q} = (q^1, \ldots, q^k)$, $\vec{s} = (s^1, \ldots, s^k)$, $\vec{w} = (w^1, \ldots, w^k)$, and $\vec{r} = (r^1, \ldots, r^k)$ is the randomness of Q^{SPIR}, and k, N are fixed. We think of this as a s-strong proof-of-knowledge of $\vec{w}$. We disregard the rest of the witness since we do not use it.

3 Our Protocol

In this section we prove our main technical result. Let k be the security parameter. Let $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be an arithmetic circuit of degree d in the y variables over $GF[2]$, and let N be an additional parameter. We construct a protocol between a prover and a verifier, that after interacting in a preamble stage (that has communication complexity $\text{poly}(k, \log d, \log N)$ and depends only on the parameters k, N, d, and not on Λ), allows the prover to commit noninteractively to any element $y \in \{0,1\}^m$, by a message of size $\text{poly}(k, m, \log d)$ that does not depend on Λ, and later to prove any N statements of the form $\Lambda(x_1, y) = z_1, \ldots, \Lambda(x_N, y) = z_N$, by a non-interactive zero-knowledge argument of size $\text{poly}(k, d, \log N)$.

3.1 Overview

We begin by giving a high level overview of our protocol. For simplicity, in this overview we combine the preamble stage and the (noninteractive) commitment stage, into a single (interactive) commitment stage. Let $\mathbb{F}$ be a field of size $\text{poly}(k, d)$, say $|\mathbb{F}| > kd$, such that $\mathbb{F}$ is an extension of $GF[2]$. We can view Λ as an arithmetic circuit over $\mathbb{F}$ (rather than over $GF[2]$). Namely, we let $\Lambda : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}$.

In order to commit to $y = (y_1, \ldots, y_m)$ the prover does the following: For each y_i, the prover chooses a random element $r_i \in_R \mathbb{F}$ and defines $A_i : \mathbb{F} \to \mathbb{F}$ by $A_i(t) = r_i t + y_i$. Then he sends to the verifier the values of $A_1, \ldots, A_m$ on a single element $\omega \in \mathbb{F} \setminus \{0\}$ chosen by the verifier.[4] This is done without the prover knowing the value of ω, by using a *symmetric private information retrieval* scheme. That is, the verifier privately retrieves the element

[4]The reason that we restrict ω to be different than 0 is that $A_i(0) = y_i$ and y_i should remain hidden from the verifier.

$A(\omega) = (A_1(\omega), \ldots, A_m(\omega))$ from the database $\{A(t)\}_{t\in\mathbb{F}\setminus\{0\}} = \{A_1(t), \ldots, A_m(t)\}_{t\in\mathbb{F}\setminus\{0\}}$ managed by the prover.[5] The reason that it is important to hide ω from the prover is that if the prover knew ω then he could have later changed y to be any y', by choosing $r'_i \in \mathbb{F}$ such that $r'_i\omega + y'_i = r_i\omega + y_i$. The fact that $A(0) = y$ and that the verifier holds the value $A(\omega)$, where ω is hidden from the prover, implies that $A(\omega)$ can be thought of as a (perfectly hiding) commitment to y. Note that for any $x \in \{0,1\}^n$ and $A = (A_1, \ldots, A_m)$, the function $\Lambda(x, A(t))$ (as a function of t) is a polynomial from $\mathbb{F}$ to $\mathbb{F}$ of degree at most d. Also, note that $\Lambda(x, A(0)) = \Lambda(x, y)$, and that given x the verifier can compute the value $\Lambda(x, A(\omega))$ by himself. Thus, loosely speaking, $\Lambda(x, A(\omega))$ can be thought of as a commitment to $\Lambda(x, y)$.

In the reveal phase the prover needs to prove non-interactively, and in a zero-knowledge manner, N statements of the form $\Lambda(x_i, y) = z_i$ (or equivalently, N statements of the form $\Lambda(x_i, A(0)) = z_i$). The first idea that comes to mind, is to have the prover simply reveal all the polynomials $\Lambda(x_i, A(t))$ for $i = 1, \ldots, N$. The verifier will then accept a proof $\{v_i\}_{i=1}^N$ if and only if for every $i \in [N]$, it holds that v_i is a polynomial of degree at most d, $v_i(\omega) = \Lambda(x_i, A(\omega))$, and $v_i(0) = z_i$.

This naive protocol is computationally sound, with cheating probability being at most $2/k$. Intuitively, the reason is that if a cheating prover P^* can prove for some x that both $\Lambda(x, y) = z$ and $\Lambda(x, y) = z'$, then it means that P^* can find two *distinct* polynomials v and v' of degree at most d, such that $v(\omega) = v'(\omega)$. Since v and v' agree on at most d values, P^* can be used to predict ω with success probability $1/d$. If P^* succeeds in doing this with probability greater than $2/k$ then ω can be guessed with probability greater than $2/kd$, contradicting Claim 2 (assuming $|\mathbb{F}| > kd$).

Despite the above, this protocol does not have the desired properties. Firstly, it is not zero-knowledge and may actually reveal information about y. This can be easily fixed as follows: Instead of simply revealing the entire polynomial $\Lambda(x_i, A(t))$, the prover will send a commitment to $\Lambda(x_i, A(t))$, and prove in a non-interactive zero-knowledge manner that the committed value is a polynomial of degree at most d, that on input ω outputs $\Lambda(x_i, A(\omega))$, and that on input 0 outputs z_i.[6] Secondly, the communication complexity of this protocol is of size $\mathrm{poly}(k, d, N)$, whereas we seek a much shorter proof of size $\mathrm{poly}(k, d, \log N)$.

Instead, we will use a linear error-correcting-code, to obtain a single polynomial which in some sense combines all these N polynomials. The idea is the following: Let ECC be any linear error-correcting-code that maps elements in $\mathbb{F}^N$ to codewords in $\mathbb{F}^M$, where M is polynomially related to N.

Notice that $C(t) \stackrel{\text{def}}{=} ECC(\Lambda(x_1, A(t)), \ldots, \Lambda(x_N, A(t)))$ is a polynomial from $\mathbb{F}$ to $\mathbb{F}^M$ of degree at most d. This follows from the fact that ECC is a *linear* code, which implies that

[5] Actually, we could have used here a $\binom{|\mathbb{F}|-1}{1}$-OT scheme, in which case the resulting communication complexity would have been $\mathrm{poly}(k, d, m)$, instead of $\mathrm{poly}(k, \log d, m)$. We note that we are not concerned with this increase in communication complexity, since in the reveal phase the communication complexity is anyway polynomial in k, d, m. The main reason that we use a SPIR scheme is that we haven't defined a $\binom{n}{1}$-OT scheme.

[6] Recall that the prover does not know ω, and thus in order to prove that $v_i(\omega) = \Lambda(x_i, A(\omega))$, the prover will actually need to use a SPIR scheme. Throughout this high-level overview, we ignore this technicality.

there is an $N \times M$ matrix $E = (e_{i,j})_{i\in[N],j\in[M]}$ (over $\mathbb{F}$) such that

$$C(t) = (\Lambda(x_1, A(t)), \ldots, \Lambda(x_N, A(t)))\, E = \left(\sum_{i=1}^{N} \Lambda(x_i, A(t)) e_{i,1}, \ldots, \sum_{i=1}^{N} \Lambda(x_i, A(t)) e_{i,M} \right),$$

which in turn implies that

$$C_\ell(t) \stackrel{\text{def}}{=} (C(t))_\ell = \sum_{i=1}^{N} \Lambda(x_i, A(t)) e_{i,\ell}$$

is a polynomial of degree at most d (since it is a sum of polynomials of degree at most d). Moreover, notice that given $(x_1, z_1), \ldots, (x_N, z_N)$, such that $\Lambda(x_i, y) = z_i$, and given $A(\omega)$, the verifier can compute by himself the values $C(0) = ECC(z_1, \ldots, z_N)$ and $C(\omega) = ECC(\Lambda(x_1, A(\omega)), \ldots, \Lambda(x_N, A(\omega)))$.

The idea is to prove that $\forall i$: $\Lambda(x_i, y) = z_i$, by revealing a single polynomial $C_\ell(t)$, where the index $\ell \in [M]$ is chosen by the verifier and kept secret from the prover. Thus, the verifier will privately retrieve the polynomial $C_\ell(t)$, via a PIR protocol. The verifier will then check that the function v that he retrieved is a polynomial of degree at most d, and will check that $v(\omega) = C_\ell(\omega)$ and that $v(0) = C_\ell(0)$.

The proof that this protocol is (computationally) sound is similar to that of the naive protocol. Let $(1 - \delta)$ be the relative distance of the code ECC. Namely, for every $z \neq z'$, $ECC(z)$ and $ECC(z')$ differ in at least a $(1 - \delta)$-fraction of their coordinates. If a cheating prover P^* can prove, with probability greater than $\frac{2}{(1-\delta)k}$, that both $\forall i$: $\Lambda(x_i, y) = z_i$, and $\forall i$: $\Lambda(x_i, y) = z'_i$, where $(z'_1, \ldots, z'_N) \neq (z_1, \ldots, z_N)$, then P^* can be used to find (with probability greater than $\frac{2}{k}$) two polynomials v and v' of degree at most d, such that $v(\omega) = v'(\omega)$ and $v(0) \neq v'(0)$. To show this we use the fact that if $(z'_1, \ldots, z'_N) \neq (z_1, \ldots, z_N)$ then $ECC(z'_1, \ldots, z'_N)$ and $ECC(z_1, \ldots, z_N)$ differ in at least $1 - \delta$ of the coordinates, and thus $v(0) \neq v'(0)$ with probability at least $1 - \delta$ (over $\ell \in_R [M]$). Since the prover does not know the index ℓ chosen by the verifier, we are able to prove that P^* can be used to find (with probability greater than $(1 - \delta)\frac{2}{(1-\delta)k} = \frac{2}{k}$) two *distinct* polynomials v and v' of degree at most d, such that $v(\omega) = v'(\omega)$. Since v and v' agree on at most d values, P^* can be used to predict ω with success probability $\frac{2}{kd}$, thus contradicting Claim 2 (assuming $|F| > kd$). We conclude that the resulting protocol is (computationally) sound, with cheating probability at most $\frac{2}{(1-\delta)k}$.

However, this protocol is not zero-knowledge. As was done with the naive protocol, we fix this by, instead of having the prover reveal the entire polynomial C_ℓ, the prover will commit to C_ℓ and will give a non-interactive zero-knowledge proof that the committed value is a degree d polynomial from $\mathbb{F}$ to $\mathbb{F}$, such that on input 0 outputs the ℓ'th coordinate of $ECC(z_1, \ldots, z_N)$, and on input ω outputs the ℓ'th coordinate of $ECC(\Lambda(x_1, A(\omega)), \ldots, \Lambda(x_N, A(\omega)))$.

The resulting protocol has $\text{poly}(k, d, m, \log N)$ communication complexity, is zero-knowledge, and is computationally sound. However the cheating probability is quite high ($\frac{1}{\text{poly}(k)}$). We boost the soundness by repeating the protocol in parallel several times.

3.2 Tools and Assumptions

Let k be the security parameter. We think of k as a function of all the other parameters, and we assume that $k \geq K_0 \log N$ for a large enough constant K_0. Let $s_1(k)$ and $s_2(k)$ be two functions such that $k \leq s_1(k) < s_2(k)$ and such that $\text{poly}(s_2(k)) < 2^k$ for every large enough k. Our protocol makes use of several primitives. In what follows, we first list all the primitives that are used in our protocol, and we then show under which assumptions these primitives exist.

1. A poly-logarithmic SPIR scheme $(Q^{SPIR}, D^{SPIR}, R^{SPIR})$, as defined in Definition 3.
2. A (s_1, s_2)-bit-commitment scheme *com* as defined in Definition 4
3. A s_1-strong proof-of-knowledge protocol for $\mathcal{NP}$ which is s_2-zero-knowledge, as defined in Definition 7 and Definition 5, respectively.
4. A non-interactive proof for $\mathcal{NP}$ which is s_1-zero-knowledge and s_2-sound, as defined in Definition 5 and Definition 6, respectively.[7]
5. A two party protocol for generating a random string, that is secure (in the sense of [GMW87]) against adversaries of size at most $\text{poly}(s_2(k))$.
6. A linear error correcting code $ECC : \mathbb{F}^N \to \mathbb{F}^M$ with relative distance $1 - \delta$, with $M = O(N)$ and $\delta \leq \frac{1}{48}$.

These primitives exist under the following assumptions:

1. Cachin *et al.* [CMS99] showed that a poly-logarithmic PIR scheme exists under the Extended Reimann Hypothesis and the Φ-Hiding Assumption. The Φ-Hiding Assumption essentially asserts that on input n and p, it is hard to decide whether p divides $\phi(n)$, where n is a product of two random primes and p is a prime chosen at random either from the set of primes that divide $\phi(n)$ or from the set of primes that do not divide $\phi(n)$. We refer the reader to [CMS99] for the precise formulation of this assumption.

   Naor and Pinkas [NP99] showed a general reduction transforming any PIR scheme into a SPIR scheme. In particular, their reduction can be used to construct a poly-logarithmic SPIR scheme from any poly-logarithmic PIR scheme and any "strong" two-message *oblivious transfer* (OT) scheme.[8] Such an OT scheme is known to exist under each of the following assumptions [AIR01, NP01, K05]:

   (a) The DDH Assumption against exponential adversaries.

   (b) the N'th Residuosity Assumption against exponential adversaries.

   (c) The Quadratic Residuosity Assumption against exponential adversaries, together with the Extended Reimann Hypothesis.

[7] We refer the reader to Definition 4.10.15 in [G01] for the definition of a non-interactive zero-knowledge proof.

[8] By a "strong" OT scheme, we mean an OT scheme that is secure against adversaries of size exponential in the security parameter (rather than adversaries of size polynomial in the security parameter).

2. A (s_1, s_2)-bit-commitment scheme *com* exists assuming the existence of a one-way permutation π, that *cannot* be inverted in time $\text{poly}(s_1(k))$, but *can* be inverted in time $s_2(k)$. Namely, π should satisfy the following two properties:

   (a) For every circuit $\mathcal{A}$ of size $\text{poly}(s_1(k))$,

   $$\Pr[\mathcal{A}(y) = \pi^{-1}(y)] < \frac{1}{2}$$

   (where the probability is over $y \in_R \{0,1\}^k$).[9]

   (b) There exists a circuit C_k of size $s_2(k)$ such that for every $y \in \{0,1\}^k$, it holds that $C_k(y) = \pi^{-1}(y)$.

   For further details, see Section 4.4.1 and Theorem 2.6.2 in [G01].

3. The existence of a s_2-zero-knowledge proof system is based on the existence of a one-way function f that cannot be inverted by circuits of size $\text{poly}(s_2(k))$. Namely, f should satisfy that for every circuit $\mathcal{A}$ of size $\text{poly}(s_2(k))$,

   $$\Pr[\mathcal{A}(y) \in f^{-1}(y)] < \frac{1}{2}$$

   (where the probability is over $y \in_R \{0,1\}^k$).

   The existence of a s_2-zero-knowledge, s_1-strong proof-of-knowledge system is based on the same assumption, and essentially involves $(\log s_1(k))^2$ sequential compositions of a s_2-zero-knowledge proof-of-knowledge system. For further details, see Section 4.4 and Section 4.7.6 in [G01].

4. The existence of a non-interactive proof for $\mathcal{NP}$ which is s_1-zero-knowledge and s_2-sound is based on the existence of a family of trapdoor permutations that cannot be inverted by circuits of size $\text{poly}(s_1(k))$ (as above). For further details, see Section 4.10 in [G01].

5. The existence of a two party protocol for generating a random string, that is secure (in the sense of [GMW87]) against adversaries of size at most $\text{poly}(s_2(k))$, is based on the existence of a one-way function f that cannot be inverted by circuits of size $\text{poly}(s_2(k))$ (as above). This follows from the coin tossing protocol due to Lindell [L03], which is in turn based on the protocol due to Blum [B82].

3.3 Protocol Description

Our protocol is associated with the functions s_1 and s_2, with the parameters $k, N, M, \mathbb{F}, n, m, d$, and with the inputs $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, $y \in \{0,1\}^m$, $(x_1, z_1), \ldots, (x_N, z_N) \in \{0,1\}^n \times \{0,1\}$. We think of the functions s_1 and s_2 as fixed, and we think of the parameters

[9] We note that due to amplification results for one-way permutations (and due to the fact that $s_1(k) \geq k$), the above probability can be amplified to be $\text{negl}(s_2(k))$.

as inputs to the protocol. We think of $\mathbb{F}$ and M as fixed functions of all the other parameters. For example, take $M = 1000N$ and take $\mathbb{F}$ to be the smallest extension of $GF[2]$ larger than kd. The input Λ is an arithmetic circuit of syntactic degree d (in the y variables),[10] and we assume for simplicity that $|\Lambda| \geq d, n, m$.

Recall that we think of k as a function of all the other parameters, and we assume that $k \geq K_0 \log N$ for a large enough constant K_0. Recall that $s_1(k)$ and $s_2(k)$ are such that $k \leq s_1(k) < s_2(k)$ and such that $\text{poly}(s_2(k)) < 2^k$ for every large enough k.

We formally describe our protocol in three phases: The preamble phase is described in Figure 1, the commitment phase is described in Figure 2, and the reveal phase is described in Figure 3.

Preamble Phase

Parameters: $k, \mathbb{F}, M$.

$V \to P$:

1. For $j = 1, \ldots, k$:
   (a) Choose $\omega^j \in_R \mathbb{F} \setminus \{0\}$, and let $(q_1^j, s_1^j) = Q^{SPIR}(k, |\mathbb{F}| - 1, \omega^j)$.
   (b) Choose $\ell^j \in_R [M]$, and let $(q_2^j, s_2^j) = Q^{SPIR}(k, M, \ell^j)$.
2. Send (q_1, q_2), where $q_1 = (q_1^1, \ldots, q_1^k)$ and $q_2 = (q_2^1, \ldots, q_2^k)$.

$V \rightleftarrows P$:

V gives a s_1-strong proof-of-knowledge of $(\omega^1, \ldots, \omega^k)$, which is s_2-zero-knowledge.

$V \rightleftarrows P$:

V and P generate three random strings R_1, R_2, R_3 (that will be used for the NIZK's in the commitment stage and the reveal stage). These strings are generated by a protocol that is secure (in the sense of [GMW87]) against adversaries of size at most $\text{poly}(s_2(k))$.

Figure 1: Preamble Phase

Remarks:

1. For simplicity, in our proofs we think of the strings R_1, R_2, R_3, generated in the preamble phase, as truly random strings that are given both to the prover and to the verifier by a trusted party. This is without loss of generality, since these strings are generated by a protocol which is secure (in the sense of [GMW87]) against adversaries of size $\text{poly}(s_2(k))$, and in this paper we only consider adversaries of size at most $\text{poly}(s_2(k))$.

[10] The reason we took d to be the syntactic degree of Λ, rather than the degree itself, is that the syntactic degree can be computed from Λ in *deterministic* polynomial time, whereas this is not known to be the case for the degree itself. The syntactic degree is always an upper bound on the degree. Any other upper bound, given to both parties, can be used as well.

2. Notice that the preamble phase consists of $O(\log s_2(k))^2$ rounds. This is due to the s_2-strong proof-of-knowledge protocol. Our result still holds if we replace the s_2-strong proof-of-knowledge protocol with a s_2-proof-of-knowledge protocol (defined analogously), which results with a constant-round preamble phase. We chose to use a strong proof-of-knowledge rather than a (standard) proof-of-knowledge for the sake of simplicity of the proofs.

3.4 Proof of Security

Assuming the existence of the primitives stated above, we prove two theorems: the first concerning the zero-knowledge property of our protocol, and the second concerning the soundness of our protocol.

Theorem 1. *Our protocol is s_1-zero-knowledge in the following sense: For every deterministic (interactive) circuit family V^* (thought of as a possibly cheating verifier) of size at most $\mathrm{poly}(s_1(k))$ there exists a (probabilistic) circuit family S (known as the simulator) of size $\mathrm{poly}(|V^*|, |P|)$, that receives the same input as V^* (including V^*'s auxiliary input Z), such that for every probabilistic circuit family $\mathcal{D}$ (called a distinguisher) of size at most $\mathrm{poly}(s_1(k))$, for every N, every $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, every k such that $s_1(k) \geq \max\{N, |\Lambda|\}$, every $y \in \{0,1\}^m$, every N pairs $(x_1, z_1), \ldots, (x_N, z_N)$, and every $Z \in \{0,1\}^*$,*

$$\Big| \Pr\left[\mathcal{D}((P(y), V^*(Z))(k, \Lambda, (x_1, z_1), \ldots, (x_N, z_N))) = 1\right] - \Pr\left[\mathcal{D}(S(Z, k, \Lambda, (x_1, z_1), \ldots, (x_N, z_N))) = 1\right] \Big| = \mathrm{negl}(s_1(k)).$$

Remark: We note that Theorem 1 holds also if Λ and $(x_1, z_1), \ldots, (x_N, z_N)$ are chosen adaptively (by the verifier) after the commitment phase. One can verify that the proof of Theorem 1 goes through in this adaptive setting. Intuitively, this is the case since the simulator receives Λ and $(x_1, z_1), \ldots, (x_N, z_N)$ only in the reveal phase (and these values may as well depend on the messages exchanged in the preamble phase and the commitment phase).

In the second theorem we make use of the following notations. For simplicity (and without loss of generality), we consider only deterministic provers. We denote the (possibly cheating) deterministic prover by P^*. We denote by h the random bits of V. We denote by V_h the deterministic verifier with random bits fixed to h. We denote by COMM_h the set of messages exchanged between P^* and V in the preamble and commitment phases. Note that COMM_h is a random variable (depending only on the randomness h of the verifier). Recall that in the commitment phase the prover sends k commitments $\{c^j\}_{j=1}^k$,[11] and that these commitments are transparent to circuits of size $\mathrm{poly}(s_2(k))$. For every h we define $y_h \in \{0,1\}^m \cup \{\perp\}$ as follows: If the decommitment of c^1 is a linear function from $\mathbb{F}$ to $\mathbb{F}^m$ that on input 0 outputs a value $y' \in \{0,1\}^m$, then we define $y_h \stackrel{\mathrm{def}}{=} y'$. Otherwise, we define $y_h = \perp$. We think of y_h as the value that the prover committed to. After the commitment phase, P^* generates a circuit Λ and values $x_1, \ldots, x_N \in \{0,1\}^n$ and $\tilde{z}_1, \ldots, \tilde{z}_N \in \{0,1\}$,

[11] These should all be commitments to random linear functions from $\mathbb{F}$ to $\mathbb{F}^m$ that on input 0 output y, where y is the prover's private input.

and his goal is to convince the verifier V that $(\tilde{z}_1, \ldots, \tilde{z}_N) = (\Lambda(x_1, y_h), \ldots, \Lambda(x_N, y_h))$, even though this may not be the case.[12] Note that these values may depend on h, a dependency which is ignored in our notations. Also note that in the reveal phase the interaction between P^* and V is deterministic.

Theorem 2. *Our protocol is s_2-sound in the following sense: For every deterministic (interactive) circuit family P^* (thought of as a possibly cheating prover) of size at most $\mathrm{poly}(s_2(k))$, for every N, and every k such that $s_2(k) \geq \max\{N, |\Lambda|\}$, let E_{cheat} denote the event that in the reveal phase P^* generates $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ and $x_1, \ldots, x_N \in \{0,1\}^n$ and $\tilde{z}_1, \ldots, \tilde{z}_N \in \{0,1\}$ such that:*

1. $(\tilde{z}_1, \ldots, \tilde{z}_N) \neq (\Lambda(x_1, y_h), \ldots, \Lambda(x_N, y_h))$ or $y_h = \perp$
2. $(P^*, V_h)(k, \Lambda, (x_1, \tilde{z}_1), \ldots, (x_N, \tilde{z}_N)) = 1$

(where h, y_h and V_h are as above). Then

$$\Pr[E_{\mathrm{cheat}}] = \mathrm{negl}(s_2(k)).$$

Proof of Theorem 1. For every (possibly cheating) verifier V^*, we construct a (black-box) simulator S that simulates the interaction between P and V^*.

Simulator S: The simulator S simulates each phase of the protocol, as follows.

Preamble phase. In this phase S gets as input the parameters $k, \mathbb{F}, M$. It runs V^*, while simulating the honest prover. Also, S uses the knowledge extractor K to extract $(w^1, \ldots, w^k)$ from the s_1-strong proof-of-knowledge given by V^*. Namely, S runs K with oracle access to V^* and with input $(q_1^1, \ldots, q_1^k)$, corresponding to the first set of queries generated by V^*. This can be done by a circuit of size $\mathrm{poly}(|V^*|, |P|, k) = \mathrm{poly}(|V^*|, |P|)$. Notice that if K fails in the extraction with non-negligible probability (in $s_1(k)$), then with probability $1 - \mathrm{negl}(s_1(k))$ the prover P will reject the proof given by V^* and abort the protocol, in which case the simulation is trivial. Therefore, from now on, we can assume that K successfully extracts $(w^1, \ldots, w^k)$ corresponding to $(q_1^1, \ldots, q_1^k)$.

Commitment phase. Recall that in this phase P uses its private input $y = (y_1, \ldots, y_m)$, whereas S does not know this private input. The simulator S only gets as input the parameters $k, \mathbb{F}, m$. In this phase, S arbitrarily chooses $s_1, \ldots, s_m \in \{0,1\}$ (for example, $s_1 = \ldots = s_m = 0$) and simulates the honest prover, while using $s = (s_1, \ldots, s_m)$ as the private input. Namely, for each $j = 1, \ldots, k$, the simulator S defines $\tilde{A}^j$ as follows:

1. Choose at random $r_1^j, \ldots, r_m^j \in_R \mathbb{F}$.

[12] We allow the prover to choose the circuit Λ and values $(x_1, \ldots, x_N)$ and $(\tilde{z}_1, \ldots, \tilde{z}_N)$ after the commitment phase. In general our soundness property holds whenever these values are chosen by a probabilistic circuit of size $\mathrm{poly}(s_2(k))$ that may depend on the interaction between the prover and the verifier in the preamble and commitment phases.

2. For each $i \in [m]$, let $\tilde{A}_i^j : \mathbb{F} \to \mathbb{F}$ be the linear function defined by $\tilde{A}_i^j(t) = r_i^j t + s_i$.
3. Let $\tilde{A}^j : \mathbb{F} \to \mathbb{F}^m$ be the linear function defined by $\tilde{A}^j(t) = (\tilde{A}_1^j(t), \ldots, \tilde{A}_m^j(t))$.

Then the simulator S simulates the honest prover, while using $\tilde{A}^j$ instead of A^j. This requires a circuit of size poly($|P|$).

Reveal phase. In this phase the simulator S is given as input an arithmetic circuit $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ of degree d, and N pairs $(x_1, z_1), \ldots, (x_N, z_N)$. The simulator S needs to generate $(proof^1, \ldots, proof^k)$. Recall that the honest prover uses the function $ECC(\Lambda(x_1, A^j(t)), \ldots, \Lambda(x_N, A^j(t)))$ when generating $proof^j$. The simulator does not know this function (as it depends on A^j, which was generated using the private input y). For each $j \in [k]$, instead of using the function $ECC(\Lambda(x_1, A^j(t)), \ldots, \Lambda(x_N, A^j(t)))$, the simulator will use a different function $\tilde{C}^j : \mathbb{F} \to \mathbb{F}^M$, chosen as follows: $\tilde{C}^j$ is an arbitrary degree $\leq d$ polynomial that satisfies:

1. $\tilde{C}^j(0) = ECC(z_1, \ldots, z_N)$.
2. $\tilde{C}^j(\omega^j) = ECC(\Lambda(x_1, \tilde{A}^j(\omega^j)), \ldots, \Lambda(x_N, \tilde{A}^j(\omega^j)))$.

S generates $proof^j$ exactly as the honest prover does, while using the function $\tilde{C}^j$ instead of the function $ECC(\Lambda(x_1, A^j(t)), \ldots, \Lambda(x_N, A^j(t)))$.
This requires a circuit of size poly$(N, |P|)$ = poly($|P|$).

Proof of Simulation: First, notice that the simulator S is of size poly($|V^*|, |P|$). Next, notice that S follows the instructions of the honest prover, with the following exceptions:

1. In the commitment phase, for every $j \in [k]$, S uses $\tilde{A}_1^j, \ldots, \tilde{A}_m^j$ instead of $A_1^j, \ldots, A_m^j$.
2. In the reveal phase, for every $j \in [k]$, S uses $\tilde{C}^j$ instead of $ECC(\Lambda(x_1, A^j(t)), \ldots, \Lambda(x_N, A^j(t)))$.

Assume for the sake of contradiction that there exists a distinguisher $\mathcal{D}_1$ of size poly($s_1(k)$), an infinite sequence of $\{Z, k, \Lambda, (x_1, z_1), \ldots, (x_N, z_N), y\}$,[13] such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_1\left(S(Z, k, \Lambda, (x_1, z_1), \ldots, (x_N, z_N))\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_1\left((P(y), V^*(Z))(k, \Lambda, (x_1, z_1), \ldots, (x_N, z_N))\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

We note that S behaves exactly as the honest prover does during the preamble phase (the honest prover P does not use his secret input y during this phase). Let $\left(\{c^j\}_{j\in[k]}, NIZK, \{a^j\}_{j\in[k]}\right)$ be the messages sent by P during the commitment phase, and let $\{proof^j\}_{j\in[k]}$ be the messages sent by P during the reveal phase. Similarly, let $\left(\{\tilde{c}^j\}_{j\in[k]}, NI\tilde{Z}K, \{\tilde{a}^j\}_{j\in[k]}\right)$ and $\{\tilde{proof}^j\}_{j\in[k]}$ be the corresponding messages generated by S. Since V^* is *deterministic*, it

[13] We assume that for every k, $s_1(k) \geq \max\{N, |\Lambda|\}$.

is easy to see that that there exists a distinguisher $\mathcal{D}_2$ of size $\text{poly}(s_1(k))$ (that has V^* hard-wired into it), such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_2\left(Z,k,\Lambda,(x_1,z_1),\ldots,(x_N,z_N),R_1,R_2,R_3,\{c^j\}_{j\in[k]},NIZK,\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]}\right)=1\right]-\\&\Pr\left[\mathcal{D}_2\left(Z,k,\Lambda,(x_1,z_1),\ldots,(x_N,z_N),R_1,R_2,R_3,\{\tilde{c}^j\}_{j\in[k]},\tilde{NIZK},\{\tilde{a}^j\}_{j\in[k]},\{\tilde{proof}^j\}_{j\in[k]}\right)=1\right]\\&\geq\frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

Since $\mathcal{D}_2$ is *non-uniform*, we can assume without loss of generality that all the elements $Z,k,\Lambda,(x_1,z_1),\ldots,(x_N,z_N)$ are hard-wired into $\mathcal{D}_2$, and thus we get that

$$\begin{aligned}&\Pr\left[\mathcal{D}_2\left(R_1,R_2,R_3,\{c^j\}_{j\in[k]},NIZK,\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]}\right)=1\right]-\\&\Pr\left[\mathcal{D}_2\left(R_1,R_2,R_3,\{\tilde{c}^j\}_{j\in[k]},\tilde{NIZK},\{\tilde{a}^j\}_{j\in[k]},\{\tilde{proof}^j\}_{j\in[k]}\right)=1\right]\\&\geq\frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

The fact that $NIZK$ is s_1-zero-knowledge (using the random string R_1), and the fact that the definition of s_1-zero-knowledge is robust against auxiliary inputs, imply that given $z\stackrel{\text{def}}{=}(R_2,R_3,\{c^j\}_{j\in[k]},\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]})$, the pair $(R_1,NIZK)$ can be simulated by a probabilistic circuit of size $\text{poly}(s_1(k))$, so that every distinguisher of size at most $\text{poly}(s_1(k))$, that is given z, distinguishes between the simulated pair and the real pair with probability at most $\text{negl}(s_1(k))$. This implies the existence of a distinguisher $\mathcal{D}_3$ of size $\text{poly}(s_1(k))$ such that,

$$\begin{aligned}&\Pr\left[\mathcal{D}_2\left(R_1,R_2,R_3,\{c^j\}_{j\in[k]},NIZK,\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]}\right)=1\right]-\\&\Pr\left[\mathcal{D}_3\left(R_2,R_3,\{c^j\}_{j\in[k]},\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]}\right)=1\right]\\&=\text{negl}(s_1(k)).\end{aligned}$$

Note that the distinguisher $\mathcal{D}_3$ also satisfies

$$\begin{aligned}&\Pr\left[\mathcal{D}_2\left(R_1,R_2,R_3,\{\tilde{c}^j\}_{j\in[k]},\tilde{NIZK},\{\tilde{a}^j\}_{j\in[k]},\{\tilde{proof}^j\}_{j\in[k]}\right)=1\right]-\\&\Pr\left[\mathcal{D}_3\left(R_2,R_3,\{\tilde{c}^j\}_{j\in[k]},\{\tilde{a}^j\}_{j\in[k]},\{\tilde{proof}^j\}_{j\in[k]}\right)=1\right]\\&=\text{negl}(s_1(k))\end{aligned}$$

(since $(R_1,NIZK)$ and $(R_1,\tilde{NIZK})$ can be simulated by the same simulator). A simple triangle inequality implies that

$$\begin{aligned}&\Pr\left[\mathcal{D}_3\left(R_2,R_3,\{c^j\}_{j\in[k]},\{a^j\}_{j\in[k]},\{proof^j\}_{j\in[k]}\right)=1\right]-\\&\Pr\left[\mathcal{D}_3\left(R_2,R_3,\{\tilde{c}^j\}_{j\in[k]},\{\tilde{a}^j\}_{j\in[k]},\{\tilde{proof}^j\}_{j\in[k]}\right)=1\right]\\&\geq\frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

A standard hybrid argument shows that there exists an index $j \in [k]$ and a distinguisher $\mathcal{D}_4$ of size $\text{poly}(s_1(k))$ (that has the secret values $(y_1, \ldots, y_m)$ and the values $(s_1, \ldots, s_m)$ hard-wired into it), such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_4\left(R_2, R_3, c^j, a^j, proof^j\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_4\left(R_2, R_3, \tilde{c}^j, \tilde{a}^j, \tilde{proof}^j\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

(In order to derive the above inequality we use the fact that P is of size at most $\text{poly}(s_1(k))$, which follows from the fact that $s_1(k) \geq \max\{N, |\Lambda|\}$.)

For $j \in [k]$ as above, let $\{A^j(t), NIZK^j(t)\}_{t\in\mathbb{F}\setminus\{0\}}$, and $\{\tilde{A}^j(t), \tilde{NIZK}^j(t)\}_{t\in\mathbb{F}\setminus\{0\}}$ be the databases used by the prover and the simulator in the commitment phase. Namely,

$$\begin{aligned}a^j &= D^{SPIR}\left(k, \{A^j(t), NIZK^j(t)\}_{t\in\mathbb{F}\setminus\{0\}}, q_1^j\right) \\ \tilde{a}^j &= D^{SPIR}\left(k, \{\tilde{A}^j(t), \tilde{NIZK}^j(t)\}_{t\in\mathbb{F}\setminus\{0\}}, q_1^j\right)\end{aligned}$$

where q_1^j is sent by the verifier V^* in the preamble phase.

The *data privacy* condition of the SPIR scheme implies that the extracted element ω^j satisfies that for every database $d = \{d_t\}_{t\in\mathbb{F}\setminus\{0\}}$ such that $d_{\omega^j} = (A^j(\omega^j), NIZK^j(\omega^j))$,

$$\left|\Pr\left[\mathcal{D}_4\left(R_2, R_3, c^j, a^j, proof^j\right) = 1\right] - \Pr\left[\mathcal{D}_4\left(R_2, R_3, c^j, b^j, proof^j\right) = 1\right]\right| \leq 2^{-k^3},$$

where $b^j = D^{SPIR}(k, d, q_1^j)$. (In the above inequality we used the fact that when generating $proof^j$ the prover did not use a^j.) In particular, we can take $d = \{d_t\}_{t\in\mathbb{F}\setminus\{0\}}$ with $d_t = 0$ for $t \neq \omega^j$, and $d_t = (A^j(t), NIZK^j(t))$ for $t = \omega^j$.

Similarly, the *data privacy* condition of the SPIR scheme implies that

$$\left|\Pr\left[\mathcal{D}_4\left(R_2, R_3, \tilde{c}^j, \tilde{a}^j, \tilde{proof}^j\right) = 1\right] - \Pr\left[\mathcal{D}_4\left(R_2, R_3, \tilde{c}^j, \tilde{b}^j, \tilde{proof}^j\right) = 1\right]\right| \leq 2^{-k^3},$$

where $\tilde{b}^j = D^{SPIR}(k, \tilde{d}, q_1^j)$, and $\tilde{d} = \{\tilde{d}_t\}_{t\in\mathbb{F}\setminus\{0\}}$, where $\tilde{d}_t = 0$ for $t \neq \omega^j$, and $\tilde{d}_t = (\tilde{A}^j(t), \tilde{NIZK}^j(t))$ for $t = \omega^j$.

A simple triangle inequality implies that

$$\begin{aligned}&\Pr\left[\mathcal{D}_4\left(R_2, R_3, c^j, b^j, proof^j\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_4\left(R_2, R_3, \tilde{c}^j, \tilde{b}^j, \tilde{proof}^j\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

This implies that there exists a distinguisher $\mathcal{D}_5$ of size $\text{poly}(s_1(k))$ (that has the element ω^j hard-wired into it), such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_5\left(R_2, R_3, c^j, A^j(\omega^j), NIZK^j(\omega^j), proof^j\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_5\left(R_2, R_3, \tilde{c}^j, \tilde{A}^j(\omega^j), \tilde{NIZK}^j(\omega^j), \tilde{proof}^j\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

As before, the fact that $NIZK^j(\omega^j)$ and $\tilde{NIZK}^j(\omega^j)$ are s_1-zero-knowledge (using the random string R_2) implies that there exists a distinguisher $\mathcal{D}_6$ of size $\text{poly}(s_1(k))$ such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_6\left(R_3, c^j, A^j(\omega^j), proof^j\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_6\left(R_3, \tilde{c}^j, \tilde{A}^j(\omega^j), \tilde{proof}^j\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

Let $\left\{c_i^j, a_i^j\right\}_{i\in[M]}$ and $\left\{\tilde{c}_i^j, \tilde{a}_i^j\right\}_{i\in[M]}$ be the databases used by the prover and the simulator in the reveal phase. Namely,

$$\begin{aligned}proof^j &= D^{SPIR}\left(k, \left\{c_i^j, a_i^j\right\}_{i\in[M]}, q_2^j\right) \\ \tilde{proof}^j &= D^{SPIR}\left(k, \left\{\tilde{c}_i^j, \tilde{a}_i^j\right\}_{i\in[M]}, q_2^j\right)\end{aligned}$$

where q_2^j is sent by the verifier V^* in the commitment phase.

As before, the *data privacy* condition of the SPIR-scheme implies that there exists an index $\ell^j \in [M]$, and there exists a distinguisher $\mathcal{D}_7$ of size $\text{poly}(s_1(k))$ (that has the index ℓ^j hardwired into it), such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_7\left(R_3, c^j, A^j(\omega^j), c_{\ell^j}^j, a_{\ell^j}^j\right) = 1\right] - \\ &\Pr\left[\mathcal{D}_7\left(R_3, \tilde{c}^j, \tilde{A}^j(\omega^j), \tilde{c}_{\ell^j}^j, \tilde{a}_{\ell^j}^j\right) = 1\right] \\ &\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

Let $\left\{f_{\ell^j}^j(t), f_{\ell^j}^j(0), NIZK_{\ell^j}^j(t)\right\}_{t\in\mathbb{F}\setminus\{0\}}$ and $\left\{\tilde{f}_{\ell^j}^j(t), \tilde{f}_{\ell^j}^j(0), \tilde{NIZK}_{\ell^j}^j(t)\right\}_{t\in\mathbb{F}\setminus\{0\}}$ be the databases used by the prover and the simulator in the reveal phase. Namely,

$$\begin{aligned}a_{\ell^j}^j &= D^{SPIR}\left(k, \left\{f_{\ell^j}^j(t), f_{\ell^j}^j(0), NIZK_{\ell^j}^j(t)\right\}_{t\in\mathbb{F}\setminus\{0\}}, q_1^j\right) \\ \tilde{a}_{\ell^j}^j &= D^{SPIR}\left(k, \left\{\tilde{f}_{\ell^j}^j(t), \tilde{f}_{\ell^j}^j(0), \tilde{NIZK}_{\ell^j}^j(t)\right\}_{t\in\mathbb{F}\setminus\{0\}}, q_1^j\right)\end{aligned}$$

Again, the *data privacy* condition of the SPIR-scheme implies that there exists a distinguisher $\mathcal{D}_8$ of size $\text{poly}(s_1(k))$, such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_8\left(R_3, c^j, A^j(\omega^j), c^j_{\ell j}, f^j_{\ell j}(\omega^j), f^j_{\ell j}(0), NIZK^j_{\ell j}(\omega^j)\right)=1\right]-\\&\Pr\left[\mathcal{D}_8\left(R_3, \tilde{c}^j, \tilde{A}^j(\omega^j), \tilde{c}^j_{\ell j}, \tilde{f}^j_{\ell j}(\omega^j), \tilde{f}^j_{\ell j}(0), \tilde{NIZK}^j_{\ell j}(\omega^j)\right)=1\right]\\&\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

As before, the fact that $NIZK^j_{\ell j}(\omega^j)$ and $\tilde{NIZK}^j_{\ell j}(\omega^j)$ are s_1-zero-knowledge (using the random string R_3), implies that there exists a distinguisher $\mathcal{D}_9$ of size $\text{poly}(s_1(k))$, such that

$$\begin{aligned}&\Pr\left[\mathcal{D}_9\left(c^j, A^j(\omega^j), c^j_{\ell j}, f^j_{\ell j}(\omega^j), f^j_{\ell j}(0)\right)=1\right]-\\&\Pr\left[\mathcal{D}_9\left(\tilde{c}^j, \tilde{A}^j(\omega^j), \tilde{c}^j_{\ell j}, \tilde{f}^j_{\ell j}(\omega^j), \tilde{f}^j_{\ell j}(0)\right)=1\right]\\&\geq \frac{1}{\text{poly}(s_1(k))}.\end{aligned}$$

Recall that according to the definition of the protocol and according to the definition of the simulator S, it holds that

$$\begin{aligned}f^j_{\ell j}(\omega^j) &= \ \ell^j\text{'th coordinate of } ECC\left(\Lambda(x_1, A^j(\omega^j)), \ldots, \Lambda(x_N, A^j(\omega^j))\right)\\f^j_{\ell j}(0) &= \ \ell^j\text{'th coordinate of } ECC(z_1, \ldots, z_N)\\\tilde{f}^j_{\ell j}(\omega^j) &= \ \ell^j\text{'th coordinate of } ECC(\Lambda(x_1, \tilde{A}^j(\omega^j)), \ldots, \Lambda(x_N, \tilde{A}^j(\omega^j)))\\\tilde{f}^j_{\ell j}(0) &= \ \ell^j\text{'th coordinate of } ECC(z_1, \ldots, z_N).\end{aligned}$$

Let

$$\begin{aligned}e_k &= (A^j(\omega^j), f^j_{\ell j}(\omega^j), f^j_{\ell j}(0))\\\tilde{e}_k &= (\tilde{A}^j(\omega^j), \tilde{f}^j_{\ell j}(\omega^j), \tilde{f}^j_{\ell j}(0)),\end{aligned}$$

and let

$$\begin{aligned}c_k &= (c^j, c^j_{\ell j})\\\tilde{c}_k &= (\tilde{c}^j, \tilde{c}^j_{\ell j}).\end{aligned}$$

Then,

$$\Pr\left[\mathcal{D}_9\left(e_k, c_k\right)=1\right]-\Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \tilde{c}_k\right)=1\right]\geq\frac{1}{\text{poly}(s_1(k))}. \tag{7}$$

Notice that the ensembles $\{e_k\}$ and $\{\tilde{e}_k\}$ are identically distributed. Furthermore, let $\hat{c}_k = (com_k(A'), com_k(A''))$, where A' is some fixed polynomial of degree at most 1 from $\mathbb{F}$ to $\mathbb{F}^M$, and A'' is some fixed polynomial of degree at most d from $\mathbb{F}$ to $\mathbb{F}$ (for example, A' and A'' can be taken to be the zero functions). Notice that

$$\begin{aligned}
&\Pr\left[\mathcal{D}_9\left(e_k, c_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \tilde{c}_k\right) = 1\right] = \\
&\left(\Pr\left[\mathcal{D}_9\left(e_k, c_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(e_k, \hat{c}_k\right) = 1\right]\right) + \\
&\left(\Pr\left[\mathcal{D}_9\left(e_k, \hat{c}_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \hat{c}_k\right) = 1\right]\right) + \\
&\left(\Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \hat{c}_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \tilde{c}_k\right) = 1\right]\right)
\end{aligned}$$

Finally, the following three inequalities contradict Inequality (7).

$$\Pr\left[\mathcal{D}_9\left(e_k, c_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(e_k, \hat{c}_k\right) = 1\right] \leq \mathrm{negl}(s_1(k)) \tag{8}$$

$$\Pr\left[\mathcal{D}_9\left(e_k, \hat{c}_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \hat{c}_k\right)\right] = 1 \leq \mathrm{negl}(s_1(k)) \tag{9}$$

$$\Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \hat{c}_k\right) = 1\right] - \Pr\left[\mathcal{D}_9\left(\tilde{e}_k, \tilde{c}_k\right) = 1\right] \leq \mathrm{negl}(s_1(k)). \tag{10}$$

Inequality (8) follows from the fact that *com* is s_1-hiding, together with the observation that we can assume without loss of generality that e_k is fixed and thus can be hardwired into $\mathcal{D}_9$. (The reason we can assume that e_k is fixed is that if Inequality (8) holds for every fixed e_k then it also holds for the random variable e_k.)

Inequality (9) follows from the fact that $\hat{c}_k$ is independent of e_k and $\tilde{e}_k$, and can be simulated internally by $\mathcal{D}_9$, together with the fact that the ensembles $\{e_k\}$ and $\{\tilde{e}_k\}$ are identically distributed.

Inequality (10) follows from the same reasoning used to obtain Inequality (8). ■

Proof of Theorem 2. Assume for the sake of contradiction that there exists a (possibly cheating) deterministic prover P^* of size at most $\mathrm{poly}(s_2(k))$, and there exists an infinite set $\mathcal{I} = \{(k, N)\}$,[14] such that for every $(k, N) \in \mathcal{I}$,

$$\Pr[E_{\mathrm{cheat}}] \geq \frac{1}{\mathrm{poly}(s_2(k))}. \tag{11}$$

The s_2-soundness property of the NIZK proofs used in the commitment phase, together with the second condition of the definition of E_{cheat}, implies that for every $(k, N) \in \mathcal{I}$,

$$\Pr[E_{\mathrm{cheat}} \wedge (y_h = \perp)] = \mathrm{negl}(s_2(k)).$$

This, together with Inequality (11), implies that for every $(k, N) \in \mathcal{I}$,

$$\Pr[E_{\mathrm{cheat}} \wedge (y_h \neq \perp)]] \geq \frac{1}{\mathrm{poly}(s_2(k))}.$$

We let E denote the event that E_{cheat} holds and $y_h \neq \perp$. Hence,

$$\Pr[E] \geq \frac{1}{\mathrm{poly}(s_2(k))}. \tag{12}$$

[14] We assume that for every k, $s_2(k) \geq \max\{N, |\Lambda|\}$.

For every $(k, N) \in \mathcal{I}$, every h, and for the values $(x_1, \ldots, x_N)$ generated by P^*, we denote by

$$(z_1, \ldots, z_N) \stackrel{\text{def}}{=} (\Lambda(x_1, y_h), \ldots, \Lambda(x_N, y_h))$$

Recall that E denotes the event that P^* generates $(x_1, \ldots, x_N)$ and $(\tilde{z}_1, \ldots, \tilde{z}_N)$, such that:

1. $(\tilde{z}_1, \ldots, \tilde{z}_N) \neq (z_1, \ldots, z_N)$.
2. $(P^*, V_h)(k, \Lambda, (x_1, \tilde{z}_1), \ldots, (x_N, \tilde{z}_N)) = 1$.
3. $y_h \neq \perp$.

We show that the existence of P^* as above contradicts the security of the SPIR scheme used in our protocol. Let

$$S \stackrel{\text{def}}{=} \{\ell : (ECC(z_1, \ldots, z_N))_\ell = (ECC(\tilde{z}_1, \ldots, \tilde{z}_N))_\ell\}.$$

Note that S is a random variable that depends on h. This is ignored in our notation. Notice that if $(z_1, \ldots, z_N) \neq (\tilde{z}_1, \ldots, \tilde{z}_N)$ then $|S| \leq \delta M$. This follows from the fact that $ECC : \mathbb{F}^N \to \mathbb{F}^M$ is a linear error correcting code with relative distance $1 - \delta$.

In the following claim, we denote by $\ell^1, \ldots, \ell^k \in [M]$ the database entries corresponding to the queries $(q_2^1, \ldots, q_2^k)$ generated by V in the preamble phase.

Claim 7. *For every $(k, N) \in \mathcal{I}$,*

$$\Pr\left[\left|\{\ell^j : \ell^j \in S\}\right| \leq \frac{k}{2} \mid E\right] = 1 - \text{negl}(s_2(k)).$$

Proof of Claim 7. Assume for the sake of contradiction that for infinitely many $(k, N) \in \mathcal{I}$,

$$\Pr\left[\left|\{\ell^j : \ell^j \in S\}\right| > \frac{k}{2} \mid E\right] \geq \frac{1}{\text{poly}(s_2(k))}. \tag{13}$$

Let E^* denote the event that E holds and $|\{\ell^j : \ell^j \in S\}| > \frac{k}{2}$. Then Inequalities (12) and (13) imply that for infinitely many $(k, N) \in \mathcal{I}$,

$$Pr[E^*] \geq \frac{1}{\text{poly}(s_2(k))}. \tag{14}$$

We show that this implies the existence of an algorithm $\mathcal{A}$ of size $\text{poly}(s_2(k))$ (that uses P^* as a black-box), that for infinitely many $(k, N) \in \mathcal{I}$, on input $q_2 = (q_2^1, \ldots, q_2^k)$, succeeds in predicting correctly the entries corresponding to $\frac{k}{2}$ of these queries, with probability at least $\left(\frac{12}{M}\right)^{\frac{k}{2}}$. This contradicts Corollary 1, since $\text{poly}(s_2(k)) < 2^{k^{1.5}}$.

Algorithm $\mathcal{A}$: For every $(k, N) \in \mathcal{I}$, on input $q_2 = (q_2^1, \ldots, q_2^k)$, algorithm $\mathcal{A}$ feeds P^* the input (k, N), and imitates the honest verifier. More specifically, $\mathcal{A}$ operates as follows.

1. Randomly choose $\omega^1, \ldots, \omega^k \in_R \mathbb{F} \setminus \{0\}$, and set $q_1 = (q_1^1, \ldots, q_1^k)$, where $(q_1^j, s_1^j) = Q^{SPIR}(k, |\mathbb{F}| - 1, \omega^j)$.

2. Feed P^* the message (q_1, q_2), where q_2 is the input to $\mathcal{A}$, and interactively feed P^* a s_1-strong proof-of-knowledge of $(\omega^1, \ldots, \omega^k)$ which is s_2-zero-knowledge. Continue imitating the honest verifier in the protocol for generating the random strings R_1, R_2, R_3.

   Denote by COMM_h the messages exchanged between P^* and V in the preamble and the commitment phases.

3. Retrieve y_h from COMM_h (Recall that since *com* is a (s_1, s_2)-bit-commitment scheme, this can be done by a circuit of size $\text{poly}(s_2(k))$). If $y_h = \perp$ then abort.

4. Upon receiving $(x_1, \tilde{z}_1), \ldots, (x_N, \tilde{z}_N)$ from P^* in the reveal phase, compute $(z_1, \ldots, z_N) \stackrel{\text{def}}{=} (\Lambda(x_1, y_h), \ldots, \Lambda(x_N, y_h))$. Compute

$$S \stackrel{\text{def}}{=} \{\ell : (ECC(z_1, \ldots, z_N))_\ell = (ECC(\tilde{z}_1, \ldots, \tilde{z}_N))_\ell\}.$$

5. $\forall j \in [k]$, choose $\hat{\ell}^j \in_R S$.

6. Output $(\hat{\ell}^1, \ldots, \hat{\ell}^k)$.

Notice that,

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ guesses correctly} \geq \tfrac{k}{2} \text{ of the entries}] \geq \\
&\Pr[\mathcal{A} \text{ guesses correctly} \geq \tfrac{k}{2} \text{ of the entries} \mid E^*] \cdot \Pr[E^*] \geq \\
&\left(\frac{1}{\delta M}\right)^{\frac{k}{2}} \cdot \Pr[E^*] \geq \left(\frac{1}{\delta M}\right)^{\frac{k}{2}} \cdot \frac{1}{2^k} = \left(\frac{1}{4\delta M}\right)^{\frac{k}{2}} \geq \left(\frac{12}{M}\right)^{\frac{k}{2}}
\end{aligned}$$

as desired. We note that the first inequality follows from Bayes' law. The second inequality follows from the fact that if E^* occurs then $|S| \leq \delta M$. The third inequality follows from Inequality (14) and from the fact that $\text{poly}(s_2(k)) < 2^k$ for every large enough k. The fourth inequality follows from the fact that $\delta \leq \frac{1}{48}$.

To conclude the proof of this claim it remains to notice that $\mathcal{A}$ is of size $\text{poly}(s_2(k)) + \text{poly}(|P^*|, |V|) = \text{poly}(s_2(k))$.

■

Next we use Claim 7 to construct an algorithm $\mathcal{B}$ of size $\text{poly}(s_2(k))$ (that uses P^* as a black-box), that for every $(k, N) \in \mathcal{I}$, takes as input a sequence of k queries $q_1 = (q_1^1, \ldots, q_1^k)$ to the SPIR scheme with security parameter k and database of size $|\mathbb{F}| - 1$, where $\mathbb{F}$ is the field used by our protocol. It uses P^* to predict the entries corresponding to these queries. We think of $(q_1^1, \ldots, q_1^k)$ as generated by $(q_1^j, s_1^j) = Q^{SPIR}(k, |\mathbb{F}| - 1, w^j)$, where $w^j \in_R \mathbb{F} \setminus \{0\}$ is generated by the honest verifier. Note that algorithm $\mathcal{B}$ does not know $w^1, \ldots, w^k$ and is trying to predict them. We prove that for every $(k, N) \in \mathcal{I}$, algorithm $\mathcal{B}$ predicts correctly

the entries corresponding to all these queries with probability greater than $\left(\frac{3}{|\mathbb{F}|-1}\right)^k$. This contradicts Claim 3, since $\text{poly}(s_2(k)) < 2^k < 2^{k^2}$.

Algorithm $\mathcal{B}$: For every $(k, N) \in \mathcal{I}$, on input $q_1 = (q_1^1, \ldots, q_1^k)$, algorithm $\mathcal{B}$ feeds P^* the input (k, N), and imitates the honest verifier. More specifically, $\mathcal{B}$ operates as follows.

1. Randomly choose $\ell^1, \ldots, \ell^k \in_R [M]$, and set $q_2 = (q_2^1, \ldots, q_2^k)$, where $(q_2^j, s_2^j) = Q^{SPIR}(k, M, \ell^j)$.
2. Feed P^* the message (q_1, q_2), where q_1 is the input to $\mathcal{B}$.
3. Imitate the s_1-strong proof-of-knowledge of $(\omega^1, \ldots, \omega^k)$. Since this proof is s_2-zero-knowledge, it can be simulated by a circuit of size $\text{poly}(s_2(k))$, and every distinguisher of size $\text{poly}(s_2(k))$ can distinguish between a real view and a simulated view only with probability $\text{negl}(s_2(k))$. Use this simulator in order to imitate the proof.
4. Continue imitating the honest verifier in the protocol for generating the random strings R_1, R_2, R_3.
5. Denote by COMM_h the messages exchanged between P^* and V during the preamble and commitment phases, and denote by $(\{c^j\}_{j=1}^k, NIZK, \{a^j\}_{j=1}^k)$ the message sent by P^* during the commitment phase.
6. For every $j \in [k]$, find the function A^j such that $c^j \in \text{Support}(com_k(A^j))$. If such a function does not exist then abort. Recall that this can be done by a circuit of size $\text{poly}(s_2(k))$ since com is a (s_1, s_2)-bit-commitment scheme.

   Let $y_h \stackrel{\text{def}}{=} A^1(0)$, as before.
7. Upon receiving $(x_1, \tilde{z}_1), \ldots, (x_N, \tilde{z}_N)$ and $proof = (proof^1, \ldots, proof^k)$ from P^* in the reveal phase, do the following for each $j = 1, \ldots, k$:
   (a) Use ℓ^j and the pair (q_2^j, s_2^j) to retrieve $\left(c_2^j, a_2^j\right)$ from $proof^j$. Namely, let $\left(c_2^j, a_2^j\right) = R^{SPIR}(k, M, \ell^j, (q_2^j, s_2^j), proof^j)$.
   (b) Find f^j such that $c_2^j \in \text{Support}(com_k(f^j))$. As before, this can be done by a circuit of size $\text{poly}(s_2(k))$ since com is a (s_1, s_2)-bit-commitment scheme. As before, if f^j doesn't exist then abort.
8. Notice that for every $j \in [k]$, if $proof^j$ is accepted by an honest verifier, then the s_2-soundness of the NIZK proofs used in the commitment and reveal phases, implies that the following holds with probability $1 - \text{negl}(s_2(k))$:
   (a) $f^j : \mathbb{F} \to \mathbb{F}$ is a polynomial of degree at most d,
   (b) $f^j(0)$ is equal to the ℓ^j'th coordinate of $ECC(\tilde{z}_1, \ldots, \tilde{z}_N)$,
   (c) $f^j(\omega^j)$ is equal to the ℓ^j'th coordinate of $ECC(\Lambda(x_1, A^j(\omega^j)), \ldots, \Lambda(x_N, A^j(\omega^j)))$, where w^j is the database entry (held by the honest verifier) corresponding to the query q_1^j.

(d) A^j is a linear function from $\mathbb{F}$ to $\mathbb{F}^m$.

We denote by E^* the event that E holds, and that for every $j \in [k]$: items (a),(b),(c) and (d) hold. Then, for every $(k, N) \in \mathcal{I}$,

$$\Pr[E^*] \geq \frac{1}{\text{poly}(s_2(k))} \tag{15}$$

9. $\forall j \in [k]$, choose $\hat{\omega}^j$ at random from the set

$$\Delta^j \stackrel{\text{def}}{=} \{u \in \mathbb{F} \setminus \{0\} : \tilde{f}^j(u) = (ECC(\Lambda(x_1, A^j(u)), \ldots, \Lambda(x_N, A^j(u))))_{\ell^j}\}.$$

Notice that by (c), if E^* holds then it is always the case that the query w^j (held by the honest verifier), corresponding to q_1^j, is an element in Δ^j. Moreover, if E^* holds and $\ell^j \notin S$ (i.e., $(ECC(z_1, \ldots, z_N))_{\ell^j} \neq (ECC(\tilde{z}_1, \ldots, \tilde{z}_N))_{\ell^j}$) then $|\Delta^j| \leq d$ (where d is the degree of Λ).

10. Output $(\hat{\omega}^1, \ldots, \hat{\omega}^k)$.

Claim 8. *For every $(k, N) \in \mathcal{I}$,*

$$\Pr[\mathcal{B}(q_1^1, \ldots, q_1^k) = (\omega^1, \ldots, \omega^k) \mid (q_1^j, s_1^j) \leftarrow Q^{SPIR}(k, |\mathbb{F}| - 1, \omega^j)] \geq \left(\frac{3}{|\mathbb{F}| - 1}\right)^k,$$

where the probability is over $\omega^1, \ldots, \omega^k \in_R \mathbb{F} \setminus \{0\}$, and over the random coin tosses of $\mathcal{B}$ and Q^{SPIR}.

Note that in order to reach a contradiction it suffices to prove Claim 8. This follows from Claim 3 and from the fact that $\mathcal{B}$ is of size $\leq \text{poly}(s_2(k))$, which is smaller than 2^{k^2}. In the proof of Claim 8 we make use of Claim 7.

Proof of Claim 8. For every $(k, N) \in \mathcal{I}$,

$$\begin{aligned}
&\Pr[\mathcal{B}(q_1^1, \ldots, q_1^k) = (\omega^1, \ldots, \omega^k) \mid (q_1^j, s_1^j) \leftarrow Q^{SPIR}(k, |\mathbb{F}| - 1, \omega^j)] \geq \\
&\Pr[\mathcal{B}(q_1^1, \ldots, q_1^k) = (\omega^1, \ldots, \omega^k) \mid E^* \wedge (q_1^j, s_1^j) \leftarrow Q^{SPIR}(k, |\mathbb{F}| - 1, \omega^j)] \cdot \Pr[E^*] \geq \\
&\left(\frac{1}{d}\right)^{\frac{k}{2}} \left(\frac{1}{|\mathbb{F}| - 1}\right)^{\frac{k}{2}} \cdot \Pr[E^*] \geq \\
&\left(\frac{k}{|\mathbb{F}| - 1}\right)^{\frac{k}{2}} \left(\frac{1}{|\mathbb{F}| - 1}\right)^{\frac{k}{2}} \cdot \Pr[E^*] = \\
&\left(\frac{k^{\frac{1}{2}}}{|\mathbb{F}| - 1}\right)^{k} \cdot \Pr[E^*] \geq \\
&\left(\frac{k^{\frac{1}{2}}}{|\mathbb{F}| - 1}\right)^{k} \cdot \frac{1}{2^k} \geq \\
&\left(\frac{3}{|\mathbb{F}| - 1}\right)^{k},
\end{aligned}$$

as desired. We note that the first inequality follows from Bayes' law. The second equality follows from the definition of E^*, from the fact that if E^* holds then $\omega^j \in \Delta^j$ for every $j \in [k]$, and from the fact that for every $\ell^j \notin S$, $|\Delta^j| \leq d$ (assuming E^* holds). The third inequality follows from the fact that $|\mathbb{F}| > kd$. The forth inequality follows from Inequality (15). ■

4 Applications

Most of the applications described in the introduction follow easily by taking Λ to be an arithmetic circuit of size and degree $\text{poly}(m, n)$ that takes as input (the description of) a Boolean formula $y \in \{0,1\}^m$ and an element $x \in \{0,1\}^n$, and outputs the value of the formula y applied on x.

An easy way to see the existence of such a circuit Λ is as follows: First, assume without loss of generality that $n \leq m$, and that the formula y is of depth $O(\log m)$. (It is well known that every formula of size m is equivalent to a formula of depth $O(\log m)$ and the translation can be done efficiently, so we can assume for convenience of the presentation that our formula is given in this form). Let C be a (universal) Boolean circuit of size $\text{poly}(m)$ and depth $O(\log m)$ that applies a Boolean formula y of size m and depth $O(\log m)$ on an input x of length $n \leq m$. (The existence of such a circuit C is easy to prove directly, and follows from the fact that Boolean circuits are a universal model of computation). Without loss of generality, we can assume that all gates in C are in $\{\neg, \wedge\}$ and we can inductively translate $\neg v$ to $1 - v$ and $v_1 \wedge v_2$ to $v_1 \cdot v_2$ to obtain an equivalent arithmetic circuit Λ. Note that since the depth of C is $O(\log m)$, the degree of the polynomial computed by Λ is at most $\text{poly}(m)$.

The only application described in the introduction that doesn't follow easily from the existence of a circuit Λ as above is the application for proofs of membership in $\mathcal{LOGSNP}$ languages.

4.1 Proofs for Membership in $\mathcal{LOGSNP}$ Languages

A rich class that lays in between $\mathcal{P}$ and $\mathcal{NP}$ is the class $\mathcal{LOGSNP}$, defined by Papadimitriou and Yannakakis [PY96]. The class $\mathcal{LOGSNP}$ contains many languages in $\mathcal{NP}$, with a poly-logarithmic witness-size. In particular, the language DOMINATING TOURNAMENT SET is known to be a complete problem for $\mathcal{LOGSNP}$ [PY96].

A tournament is an $n \times n$ matrix T, such that for every i we have $T_{i,i} = 1$, and for every $i \neq j$ we have $T_{i,j} = 1 \iff T_{j,i} = 0$. A dominating set for a tournament T is a set of rows $y \subset [n]$, such that for every column $j \in [n]$ there exists a row $i \in y$ with $T_{i,j} = 1$. It is not hard to see, by applying a greedy algorithm, that any tournament has a dominating set of size at most $\log n$. The language $\mathcal{L} \stackrel{\text{def}}{=}$ DOMINATING TOURNAMENT SET is the set of all pairs (T, ℓ), such that T is a tournament that has a dominating set of size ℓ.

Given a tournament T of size $n \times n$ and an integer $\ell \leq \log n$, a witness for $(T, \ell) \in \mathcal{L}$ is a vector $y = (y_1, ..., y_\ell) \in [n]^\ell$, such that for every column $j \in [n]$,

$$\bigvee_{i=1}^{\ell} T_{y_i,j} = 1.$$

We will show that for every T of size $n \times n$ and every integer $\ell \leq \log n$, there is an arithmetic circuit $\Lambda = \Lambda_{T,\ell}$ (with T, ℓ hardwired to it) that takes as input an index $j \in [n]$ and a vector $y = (y_1, ..., y_\ell) \in [n]^\ell$ and outputs the value of $\bigvee_{i=1}^{\ell} T_{y_i,j}$, and such that Λ is of size polynomial in n and degree poly-logarithmic in n.

The existence of such a circuit $\Lambda_{T,\ell}$ implies that (if the prover and the verifier interacted in a preamble stage) the membership in $\mathcal{L} \stackrel{\text{def}}{=}$ DOMINATING TOURNAMENT SET can be proved by a non-interactive zero-knowledge argument of poly-logarithmic size. Given (T, ℓ), the prover and the verifier construct the circuit $\Lambda = \Lambda_{T,\ell}$ as above. They can then apply our protocol with $N = n$ and $x_1 = 1, \ldots, x_n = n$, and $z_1 = \ldots = z_n = 1$. Both, the commitment phase and the proof phase of our protocol are unified to be the non-interactive zero-knowledge argument for membership of (T, ℓ) in $\mathcal{L}$. In the commitment phase the prover commits to a witness $y = (y_1, ..., y_\ell) \in [n]^\ell$ and in the reveal phase the prover proves that for every j, $\Lambda(j, y) = 1$. Since the length of the witness y is $O(\log^2 n)$ and the degree of Λ is poly-logarithmic in n, the total length of the non-interactive zero-knowledge argument is poly-logarithmic in n.

Given (T, ℓ), we will now show how to construct the arithmetic circuit $\Lambda = \Lambda_{T,\ell}$ (with T, ℓ hardwired to it) that takes as input an index $j \in [n]$ and a vector $y = (y_1, ..., y_\ell) \in [n]^\ell$ and outputs the value of $\bigvee_{i=1}^{\ell} T_{y_i,j}$, (and such that Λ is of size polynomial in n and degree poly-logarithmic in n). This is done as follows.

1. For every y_i, define $Y_i \in \{0,1\}^n$ to be the vector that has 1 in coordinate y_i and 0 in every other coordinate. Since every entry of Y_i can be written as a term in the bits of y_i, every entry of Y_i can be written as a polynomial of degree $\log n$ in the bits of y_i.

2. Define an $\ell \times n$ matrix $\hat{T}$, by $\hat{T}_{i,j'} = T_{y_i,j'}$. Thus the i'th row of $\hat{T}$ is $(Y_i)^{tr} \cdot T$, where $(Y_i)^{tr}$ denotes the transpose of Y_i. Hence, every entry in the i'th row of $\hat{T}$ can be written as a polynomial of degree $\log n$ in the bits of y_i.

3. For every $j' \in [n]$, define $v_{j'} = \bigvee_{i=1}^{\ell} T_{y_i,j'} = \bigvee_{i=1}^{\ell} \hat{T}_{i,j'}$. Since this can be written as a polynomial of degree ℓ in the entries of $\hat{T}$, every $v_{j'}$ can be written as a polynomial of degree $\ell \cdot \log n$ in the bits of $y_1, ..., y_\ell$.

4. Define $I_j \in \{0,1\}^n$ to be the vector that has 1 in coordinate j and 0 in every other coordinate. As before, every entry of I_j can be written as a polynomial of degree $\log n$ in the bits of j.

5. Define $\Lambda(j, y) = (v_1, ..., v_n) \cdot I_j$. Since $v_1, ..., v_n$ can be written as polynomials of degree $\ell \cdot \log n$ in the bits of $y_1, ..., y_\ell$ and since the entries of I_j can be written as polynomials of degree $\log n$ in the bits of j, we conclude that $\Lambda(j, y)$ can be written as a polynomial of degree $O(\log^2 n)$ in all the input bits. Note that

$$\Lambda(j, y) = (v_1, ..., v_n) \cdot I_j = v_j = \bigvee_{i=1}^{\ell} T_{y_i,j}.$$

References

[AIR01] W. Aiello, Y. Ishai, and O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *EUROCRYPT 2001*, pages 119-135.

[B82] M. Blum. Coin Flipping by Phone. In *IEEE Spring COMPCOM 1982*, pages 133-137.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *STOC 1988*, pages 103-112.

[CMS99] C. Cachin, S. Micali, and M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *EUROCRYPT 1999*, pages 402-414.

[CGKS98] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private Information Retrieval. In *J. ACM 45(6), 1998*, pages 965-981.

[DMP88] A. De Santis, S. Micali, G. Persiano: Non-Interactive Zero-Knowledge with Preprocessing. In *CRYPTO 1988*, pages 269-282.

[G01] O. Goldreich. Foundations of Cryptography: Volume 1 – Basic Tools. *Cambridge University Press, 2001.*

[G05] O. Goldreich. Proving Yao's XOR Lemma via the Direct Product Lemma www.wisdom.weizmann.ac.il/ oded/cc-texts.html

[GMR89] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *SIAM Journal on Computing, 18(1), 1989*, pages 186-208.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In *CRYPTO 1986*, pages 171-185.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC 1987*, pages 218-229.

[K05] Y. T. Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In *EUROCRYPT 2005*, pages 78-95.

[K92] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC 1992*, pages 723-732.

[L03] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Journal of Cryptology 16(3), 2003*, pages 143-184.

[M94] S. Micali. CS Proofs (Extended Abstracts). In *FOCS 1994*, pages 436-453.

[NP99] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *proc. of 31st STOC 1999*, pages 245-254.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA 2001*, pages 448-457.

[PY96] C. H. Papadimitriou and M. Yannakakis. On Limited Nondeterminism and the Complexity of the V-C Dimension. In *J. Comput. Syst. Sci. 53(2), 1996*, pages 161-170.

Commitment Phase

Parameters: $k, \mathbb{F}, m$.

Private Input: $y = (y_1, \ldots, y_m) \in \{0,1\}^m$.

$P \to V$:

1. For $j = 1, \ldots, k$:
   (a) Choose $r_1^j, \ldots, r_m^j \in_R \mathbb{F}$.
   (b) For each $i \in [m]$, let $A_i^j : \mathbb{F} \to \mathbb{F}$ be the linear function defined by $A_i^j(t) = r_i^j t + y_i$.
   (c) Let $A^j : \mathbb{F} \to \mathbb{F}^m$ be the function defined by $A^j(t) = (A_1^j(t), \ldots, A_m^j(t))$.
   (d) Let $c^j = com_k(A^j)$, where $com = \{com_k\}$ is a (s_1, s_2)-bit-commitment scheme.
2. Let $NIZK$ be a proof that $c^1, \ldots, c^k$ are commitments to linear functions $B^1, \ldots, B^k$ from $\mathbb{F}$ to $\mathbb{F}^m$, such that $B^1(0) = \ldots = B^k(0) \in \{0,1\}^m$. This proof is non-interactive, s_1-zero-knowledge and s_2-sound (and uses the random string R_1 generated in the preamble phase).
3. For $j = 1, \ldots, k$, let $a^j = D^{SPIR}(k, \{A^j(t), NIZK^j(t)\}_{t \in \mathbb{F} \setminus \{0\}}, q_1^j)$, where $NIZK^j(t)$ is a proof for c^j being a commitment to a linear function from $\mathbb{F}$ to $\mathbb{F}^m$, that on input t outputs $A^j(t)$. As above, this proof is non-interactive, s_1-zero-knowledge and s_2-sound (and uses the random string R_2 generated in the preamble phase).
4. Send $\left(\{c^j\}_{j=1}^k, NIZK, \{a^j\}_{j=1}^k\right)$.

V :

1. Verify $NIZK$
2. For each $j \in [k]$,
   (a) Let $(v^j(\omega^j), NIZK^j(\omega^j)) = R^{SPIR}(k, |\mathbb{F}| - 1, w^j, (q_1^j, s_1^j), a^j)$.
   (b) Verify that $NIZK^j(\omega^j)$ is a proof for c^j being a commitment to a linear function from $\mathbb{F}$ to $\mathbb{F}^m$ that on input w^j outputs $v^j(w^j)$.

Figure 2: Commitment Phase

Reveal Phase

Parameters: $k, \mathbb{F}, M, d$

Common Input: An arithmetic circuit $\Lambda : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, and N pairs $(x_1, z_1), \ldots, (x_N, z_N) \in \{0,1\}^n \times \{0,1\}$.

$P \to V$:

1. For every $j \in [k]$, generate $proof^j$ (corresponding to (q_1^j, q_2^j)) as follows:
   - For $i = 1, \ldots, M$:
     (a) Let $f_i^j(t) = \left(ECC\left(\Lambda(x_1, A^j(t)), \ldots, \Lambda(x_N, A^j(t))\right)\right)_i$.
     (b) Let $c_i^j = com_k(f_i^j)$, where $com = \{com_k\}$ is a (s_1, s_2)-bit-commitment scheme.
     (c) Let $a_i^j = D^{SPIR}(k, \{f_i^j(t), f_i^j(0), NIZK_i^j(t)\}_{t \in \mathbb{F} \setminus \{0\}}, q_1^j)$, where $NIZK_i^j(t)$ is a proof that c_i^j is a commitment to a degree $\leq d$ polynomial from $\mathbb{F}$ to $\mathbb{F}$, that on input t outputs $f_i^j(t)$ and on input 0 outputs $f_i^j(0)$. This proof is non-interactive, s_1-zero-knowledge and s_2-sound (and uses the random string R_3 generated in the preamble phase).
   - Let $proof^j = D^{SPIR}(k, \{c_i^j, a_i^j\}_{i \in [M]}, q_2^j)$.
2. Send $(proof^1, \ldots, proof^k)$.

V:

For every $j \in [k]$, verify $proof^j$ as follows:

1. Retrieve $(c_{\ell^j}^j, a_{\ell^j}^j)$ from $proof^j$. That is, let $(c_{\ell^j}^j, a_{\ell^j}^j) = R^{SPIR}(k, M, \ell^j, (q_2^j, s_2^j), proof^j)$.
2. Retrieve $(v_{\ell^j}^j(\omega^j), v_{\ell^j}^j(0), NIZK_{\ell^j}^j(\omega^j))$ from $a_{\ell^j}^j$. That is, let $(v_{\ell^j}^j(\omega^j), v_{\ell^j}^j(0), NIZK_{\ell^j}^j(\omega^j)) = R^{SPIR}(k, |\mathbb{F}| - 1, \omega^j, (q_1^j, s_1^j), a_{\ell^j}^j)$.
3. Accept if and only if the following conditions hold:
   (a) $NIZK_{\ell^j}^j(\omega^j)$ is a proof that $c_{\ell^j}^j$ is a commitment to a degree $\leq d$ polynomial from $\mathbb{F}$ to $\mathbb{F}$ that on input w^j outputs $v_{\ell^j}^j(\omega^j)$ and on input 0 outputs $v_{\ell^j}^j(0)$.
   (b) $v_{\ell^j}^j(0)$ is equal to the ℓ^j'th coordinate of $ECC(z_1, \ldots, z_N)$.
   (c) $v_{\ell^j}^j(\omega^j)$ is equal to the ℓ^j'th coordinate of $ECC(\Lambda(x_1, v^j(\omega^j)), \ldots, \Lambda(x_N, v^j(\omega^j)))$.

Figure 3: Reveal Phase