QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field

Kang Yang
State Key Laboratory of Cryptology
yangk@sklc.org

Pratik Sarkar
Boston University
pratik93@bu.edu

Chenkai Weng
Northwestern University
ckweng@u.northwestern.edu

Xiao Wang
Northwestern University
wangxiao@cs.northwestern.edu

Abstract

Zero-knowledge (ZK) proofs with an optimal memory footprint have attracted a lot of attention, because such protocols can easily prove very large computation with a small memory requirement. Such ZK protocol only needs $O(M)$ memory for both parties, where M is the memory required to verify the statement in the clear. In this paper, we propose several new ZK protocols in this setting, which improve the concrete efficiency and, at the same time, enable sublinear amortized communication for circuits with some notion of relaxed uniformity.

1. In the circuit-based model, where the computation is represented as a circuit over a field, our ZK protocol achieves a communication complexity of 1 field element per non-linear gate for any field size while keeping the computation very cheap.

   We implemented our protocol, which shows extremely high efficiency and affordability. Compared to the previous best-known implementation, we achieve $6\times$–$7\times$ improvement in computation and $3\times$–$7\times$ improvement in communication. When running on intro-level AWS instances, our protocol only needs one US dollar to prove one trillion AND gates (or 2.5 US dollars for one trillion multiplication gates over a 61-bit field).

2. In the setting where part of the computation can be represented as a set of polynomials, we can achieve communication sublinear to the polynomial size: the communication only depends on the input size and the highest degree of all polynomials, *independent* of the number of polynomials and the number of multiplications in the polynomials.

   Using the improved ZK protocol, we can prove matrix multiplication with communication proportional to the input size, rather than the number of multiplications. Proving the multiplication of two 1024×1024 matrices, our implementation, with one thread and 1 GB of memory, only needs 10 seconds and communicates 25 MB, $35\times$ faster than the state-of-the-art protocol Virgo that would need more than 140 GB of memory for the same task.

1 Introduction

Zero-knowledge (ZK) proofs allow a prover in possession of a witness w to prove to a verifier that $\mathcal{C}(w) = 1$ for some public circuit $\mathcal{C}$. Until recently, most works focus on non-interactive ZK proofs with a small proof size [e.g., BCC+16, BBB+18, WTs+18, BCR+19, BBHR19, Set20, GKR08, BFS20, ZXZS20, AHIV17, BFH+20], most of which suffer from a scalability problem. Specifically, to prove a circuit with t gates, these protocols require memory of size proportional to t. As a result, the memory constraint quickly becomes the bottleneck on the scale of computation that can be efficiently processed by a ZK proof.

Recently, ZK protocols with essentially unlimited scalability gained a lot of attention. Here, the protocol should use the minimal amount of memory possible, which is the memory required to *evaluate* the circuit in the clear. For example, ZK protocols based on privacy-free garbled circuits [JKO13, FNO15, HK20] satisfy this requirement and is also concretely efficient. However, due to the lower bound on garbled circuits, the best communication complexity one can hope for is κ bits per AND gate in the gate-by-gate paradigm, where κ is the computational security parameter. Some ZK proofs [GMO16, CDG+17, KKW18] using the "MPC-in-the-head" paradigm [IKOS07] are also streamable, thus achieving a small memory overhead, but the overall computation and communication complexities are still high. For *zero-knowledge succinct non-interactive arguments of knowledge* (zk-SNARKs), the only exception is to use recursive composition [BCCT13]. However, compared to the state-of-the-art memory-heavy zk-SNARKs [Set20, SL20], those using recursive composition are still orders of magnitude slower [COS20, BDFG20].

A recent promising line of work based on (subfield) Vector Oblivious Linear Evaluation [BCGI18, BCG+19b, SGRR19, BCG+19a, YWL+20], or (s)VOLE for short, provides a new direction to accomplish this goal. Weng, Yang, Katz and Wang [WYKW20] proposed a ZK protocol (Wolverine) that can prove circuits over any field with the communication complexity roughly $(3\rho/\log t + 1)$ field elements per multiplication gate where t is the circuit size and ρ is the statistical security parameter; in the case of large fields[1], the communication can be reduced to 4 field elements (or 2 field elements with much higher computational cost) per multiplication gate. Another recent work by Baum, Malozemoff, Rosen and Scholl (Mac′n′Cheese) [BMRS20] proposed a similar ZK protocol for large fields with the communication cost of 3 field elements per multiplication gate. Line-Point ZK (LPZK) by Dittmer, Ishai, and Ostrovsky [DIO20] achieves seemingly optimal communication cost (i.e., one field element per multiplication gate) in the gate-by-gate paradigm, but their protocol is only for large fields and has not been studied for implementation yet. See Table 1 for details.

Challenges. The recent line of work has significantly improved the efficiency of ZK proofs with essentially unlimited scalability. However, some challenges remain that we would like to solve in this paper.

1. LPZK is great in communication but it only supports large fields. For small fields (e.g., binary fields), the best known ZK protocol Wolverine is still based on the cut-and-choose technique and thus incurs a high overhead.

2. All of the above memory-efficient protocols require communication linear to the number of multiplication gates in the circuits (except for zk-SNARKs with recursive composition). One would naturally desire concretely efficient protocols with the same memory efficiency while only requiring sublinear communication cost. Mac′n′Cheese partially addressed this issue by incorporating the "stacking" technique [HK20], but its applicability is rather limited to circuits with many branches.

1.1 Our Contributions

In this paper, we present a set of new ZK protocols that achieve unprecedented practical performance while being able to scale to billions and even trillions of gates without any memory issue. Our circuit-based protocol only needs to send one field element per multiplication gate (and free for addition gates) for any field size with very efficient computational efficiency. Our polynomial-based protocol only needs to send $n + d$ field elements to prove a set of degree-d polynomials on n input values, independent of the number of polynomials and the number of multiplications in each polynomial. Combining both protocols, we are able to achieve sublinear communication as long as the circuit satisfies a weak notion of uniformity. We elaborate our results below.

[1] In this paper, we say that a "large field" means that the size is at least 2^ρ.

Protocol	Boolean Circuit		Arithmetic Circuit	
	Size	Speed	Size	Speed
Wolverine [WYKW20]	7	1.25 M/sec	4	0.66 M/sec
Mac′n′Cheese [BMRS20]	–	–	3	0.4 M/sec
LPZK [DIO20]	–	–	1	–
QuickSilver	1	7.7 M/sec	1	4.8 M/sec

Table 1: **Comparing our work** (QuickSilver) **with prior related work.** Size represents the number of field elements to send for each multiplication gate, which is also the number of (s)VOLE correlations needed. Speed represents the number of multiplication gates that can be executed per second with unlimited bandwidth and a single thread.

Optimally efficient ZK proof for any field in the gate-by-gate paradigm. As summarized in Table 1, the recent LPZK protocol achieves the best communication cost in the gate-by-gate paradigm for large fields. However, when it comes to small fields like Boolean circuits, the cost of the best-known protocol [WYKW20] is still much high due to cut-and-choose. Our ZK protocol is developed on top of Wolverine and LPZK to achieve the best of both protocols: it needs to send one field element and a single sVOLE correlation per non-linear gate for any field (including the case of Boolean circuits). See Section 3.1 for a detailed explanation of the protocol.

Our implementation can prove Boolean circuits at a speed of 7.7 million AND gates per second (with one thread) and can prove arithmetic circuits over a 61-bit field at a speed of 4.8 million multiplication gates per second (with one thread). The performance could be doubled using four threads (See Section 6).

Efficient ZK proof beyond the gate-by-gate paradigm. The above protocol reminds us of the story in garbled circuits. After a long history of optimizations [BMR90, NPS99, KS08, KMR14], the half-gates garbling scheme [ZRE15] achieves free for XOR gates and 2κ-bit communication per AND gate, which is also proven to be optimal when the garbling algorithm takes one gate as input at a time (a.k.a., the gate-by-gate paradigm). The lower bound soon spawned new research to bypass it by garbling not in the gate-by-gate framework. For example, Ball et al. [BMR16] observed that garbling Boolean formulas as a whole can further improve communication efficiency.

Our optimized protocol puts us in a similar situation, where the gate-by-gate paradigm limits further improvement. Inspired by the above success story in garbling, we study the case of proving polynomial satisfiability as a whole. As we discuss in Section 3.2, we find that it is possible to prove a set of degree-d polynomials all on n variables with communication cost of $n + d$ field elements, independent of the number of polynomials or the number of multiplications in the polynomials.

This ZK proof for polynomial satisfiability has a direct application to prove knowledge of a solution of the *short-integer-solution* (SIS) problem, where verifying that all secret inputs are bounded in a small range can be viewed as low-degree polynomials. Our ZK protocol could prove knowledge of a solution of the SIS problem with $8\times$ improvement in proof size and $110\times$ improvement in execution time, compared with the state-of-the-art implementation [WYKW20].

Sublinear ZK proof for circuits with relaxed uniformity. The above efficient polynomial-based ZK protocol is sublinear when we compare the communication cost with the number of multiplications in the polynomial set. Using this result, we can improve the efficiency of many applications and reach communication sublinear to the circuit size as long as the circuit satisfies some weak notion of uniformity. More specifically, as long as the circuit contains many sub-circuits such that their polynomial representations are all bounded by some degree d, our technique can reduce the communication cost for proving these sub-circuits. For example, we can prove matrix multiplication in communication complexity of $O(n^2)$ where two matrices are of dimension $n \times n$, and a gate-by-gate protocol needs $O(n^3)$ communication. We discuss more complicated examples, as well as the general applicability in Section 3.3, which opens a new space of improvement for real-world applications.

For matrix multiplication, our implementation can prove an instance with $n = 1024$, which contains about one billion multiplications over a 61-bit field, in about 25 MB of communication and 10 seconds using just 1 GB of memory. This is $35\times$ faster than the prior best-known implementation [ZXZS20], which needs 148 GB of memory to accomplish this task.

2 Preliminaries

2.1 Notation

We use κ and ρ to denote the computational and statistical security parameters, respectively. We use $x \leftarrow S$ to denote that sampling x uniformly at random from a finite set S. For $n \in \mathbb{N}$, we denote by $[n]$ a set $\{1, \ldots, n\}$. For $a, b \in \mathbb{Z}$ with $a \leq b$, we write $[a, b] = \{a, \ldots, b\}$. We use bold lower-case letters like $\boldsymbol{x}$ for column vectors, and denote by x_i the i-th component of vector $\boldsymbol{x}$ where x_1 is the first entry.

For an extension field $\mathbb{F}_{p^r}$ of a finite field $\mathbb{F}_p$, where $p \geq 2$ is a prime or a power of a prime and $r \geq 1$ is an integer, we fix some monic, irreducible polynomial $f(X)$ of degree r and write $\mathbb{F}_{p^r} \cong \mathbb{F}_p[X]/f(X)$. Therefore, every field element $w \in \mathbb{F}_{p^r}$ can be denoted uniquely as $w = \sum_{h \in [r]} w_h \cdot X^{h-1}$ with $w_h \in \mathbb{F}_p$ for all $h \in [r]$. We could view the elements over $\mathbb{F}_{p^r}$ equivalently as the vectors in $(\mathbb{F}_p)^r$. When we write arithmetic expressions involving both elements of $\mathbb{F}_p$ and elements of $\mathbb{F}_{p^r}$, it is understood that field elements in $\mathbb{F}_p$ are viewed as the polynomials lying in $\mathbb{F}_{p^r}$ that have only constant terms.

A circuit $\mathcal{C}$ over a field $\mathbb{F}_p$ is defined by a set of input wires $\mathcal{I}_{\mathsf{in}}$, along with a list of gates of the form $(\alpha, \beta, \gamma, T)$, where α, β are the indices of the input wires of the gate, γ is the index of the output wire of the gate, and $T \in \{\mathsf{Add}, \mathsf{Mult}\}$ is the type of the gate. For proving the satisfiability of a circuit $\mathcal{C}$, we consider that there is only one output wire in circuit $\mathcal{C}$. If $p = 2$, then $\mathcal{C}$ is a Boolean circuit with $\mathsf{Add} = \oplus$ and $\mathsf{Mult} = \wedge$. If $p > 2$ is a prime or a power of a prime, then $\mathcal{C}$ is an arithmetic circuit where $\mathsf{Add}/\mathsf{Mult}$ corresponds to addition/multiplication in $\mathbb{F}_p$. We let t denote the number of multiplication gates in the circuit, and let $n = |\mathcal{I}_{\mathsf{in}}|$.

A set of t *multivariable* polynomials over $\mathbb{F}_p$ is denoted as $\{f_1, \ldots, f_t\}$, where each polynomial has degree at most d and takes the same n variables $(x_1, \ldots, x_n)$ as input. Every polynomial f_i can be denoted as $\sum_{h \in [0,d]} f_{i,h}$, where all terms in $f_{i,h}$ have degree exactly h. For a vector $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathbb{F}_p^n$, we define $f_i(\boldsymbol{z}) = f_i(z_1, \ldots, z_n)$ as a value obtained by computing the polynomial f_i on $x_j = z_j$ for $j \in [n]$. Given a polynomial f over $\mathbb{F}_p$, we could naturally define a polynomial over $\mathbb{F}_{p^r}$ by interpreting operations over $\mathbb{F}_p$ (i.e., additions and multiplications) to be operations over $\mathbb{F}_{p^r}$.

2.2 MACs and Functionalities

Information-Theoretic MACs. We use *information-theoretic message authentication codes* (IT-MACs) [BDOZ11, NNOB12] to authenticate values over $\mathbb{F}_p$ or $\mathbb{F}_{p^r}$. Specifically, let $\Delta \in \mathbb{F}_{p^r}$ be a *global key*, which is sampled uniformly at random and known only by one party $\mathcal{V}$. A value $x \in \mathbb{F}_p$ or $\mathbb{F}_{p^r}$ known by the other party $\mathcal{P}$ can be authenticated by giving $\mathcal{V}$ a uniform key $k \in \mathbb{F}_{p^r}$ and giving $\mathcal{P}$ the corresponding MAC tag $m = k - \Delta \cdot x \in \mathbb{F}_{p^r}$. We denote such an authenticated value by $[x]$. Note that authenticated values are additively homomorphic. In particular, given the public coefficients $c_1, \ldots, c_\ell, c \in \mathbb{F}_p$ or $\mathbb{F}_{p^r}$, the parties can compute $[y] = \sum_{i=1}^{\ell} c_i \cdot [x_i] + c$ locally, where $\mathcal{P}$ compute $y := \sum_{i=1}^{\ell} c_i \cdot x_i + c$, $m_y := \sum_{i=1}^{\ell} c_i \cdot m_{x_i}$, and $\mathcal{V}$ computes $k_y := \sum_{i=1}^{\ell} c_i \cdot k_{x_i} + c \cdot \Delta$. We extend the above notation to vectors of authenticated

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

Input. Upon receiving (input, id, w) from a prover $\mathcal{P}$ and (input, id) from a verifier $\mathcal{V}$, with id a fresh identifier, store (id, w).

Prove circuit satisfiability. Upon receiving $(\mathsf{prove}, \mathcal{C}, id_1, \ldots, id_n)$ from $\mathcal{P}$ and $(\mathsf{verify}, \mathcal{C}, id_1, \ldots, id_n)$ from $\mathcal{V}$ where $id_1, \ldots, id_n$ are present in memory, retrieve (id_i, w_i) for $i \in [n]$ and define a vector $\boldsymbol{w} = (w_1, \ldots, w_n)$. Send true to $\mathcal{V}$ if $\mathcal{C}(\boldsymbol{w}) = 1$ and false otherwise.

Prove polynomial satisfiability. Upon receiving $(\mathsf{prove}, \{f_i\}_{i\in[t]}, id_1, \ldots, id_n)$ from $\mathcal{P}$ and $(\mathsf{verify}, \{f_i\}_{i\in[t]}, id_1, \ldots, id_n)$ from $\mathcal{V}$ where $id_1, \ldots, id_n$ are present in memory, retrieve (id_i, w_i) for $i \in [n]$ and define a vector $\boldsymbol{w} = (w_1, \ldots, w_n)$. Then send true to $\mathcal{V}$ if $f_i(\boldsymbol{w}) = 0$ for all $i \in [t]$ and false to $\mathcal{V}$ otherwise.

Figure 1: Zero-knowledge functionality for circuit and polynomial satisfiability.

values as well. In this case, $[\boldsymbol{x}]$ means that $\mathcal{P}$ holds $\boldsymbol{x} \in \mathbb{F}_p^n$ and $\boldsymbol{m} \in \mathbb{F}_{p^r}^n$, while $\mathcal{V}$ holds $\boldsymbol{k} \in \mathbb{F}_{p^r}^n$ with $\boldsymbol{k} = \boldsymbol{m} + \Delta \cdot \boldsymbol{x}$.

Security model and functionalities. All our protocols are proven in the universal composability (UC) framework [Can01] in the presence of a malicious, static adversary. We provide a brief overview of the UC framework in the Appendix A. Our ZK functionality is shown in Figure 1. As we provide the protocols for both circuits and polynomial sets, we extend the ZK functionality accordingly.

We use the standard sVOLE functionality [BCG+19b, BCG+19a] shown in Figure 6 of Appendix A to generate authenticated values, where $\mathcal{P}$ obtains $\boldsymbol{x} \in \mathbb{F}_p^\ell$ and $\boldsymbol{m} \in \mathbb{F}_{p^r}^\ell$, and $\mathcal{V}$ obtains $\Delta \in \mathbb{F}_{p^r}$ and $\boldsymbol{k} \in \mathbb{F}_{p^r}^\ell$, such that $\boldsymbol{m} = \boldsymbol{k} - \Delta \cdot \boldsymbol{x}$. This sVOLE functionality can be efficiently realized using the recent LPN-based protocols [SGRR19, BCG+19a, YWL+20, WYKW20]. These protocols have the communication complexity sublinear to the number of resulting sVOLE correlations.

3 Technical Overview

In this section, we provide the intuition of our ZK protocols, and leave the full protocol description, as well as the proofs of security, to later sections.

3.1 Zero-Knowledge Proof for Circuit Satisfiability over Any Sized Field

In the previous section, we introduced authenticated values, which can also be viewed as a way for the prover to commit values to the verifier. It is a non-interactive commitment with an interactive preprocessing phase. Given this tool, recent work [WYKW20, DIO20, BMRS20] designed efficient zero-knowledge proofs for circuit satisfiability with high scalability and communication linear to the circuit size. These works all follow the same paradigm below.

1. The prover first commits to all wire values in the circuit to the verifier, which takes $n+t$ field elements of communication and $n+t$ number of (s)VOLE correlations, where n is the input size and t is the number of multiplication gates. Because the underlying commitment is additively homomorphic, addition gates can be processed for free.

2. The prover proves that the committed values on multiplication gates are correct by executing a checking procedure with the verifier. This is where prior works differ. The first approach used in ZK proofs Wolverine [WYKW20] and Mac'n'Cheese [BMRS20] uses the VOLE-based commitment as a black box and follow the ideas in *secure multi-party computation* (MPC) to construct many authenticated/committed

Beaver triples. These Beaver triples can then be used to check the correctness of the first step. The second approach by LPZK [DIO20] uses the fact that IT-MACs are linear relationships. Their checking procedure needs only one extra field element per multiplication gate, which can be further eliminated using a random oracle. The resulting protocol appears to be optimal for large fields in this gate-by-gate paradigm.

Our ZK protocol in the gate-by-gate paradigm. Our protocol in the circuit-based setting can be viewed as a hybrid approach of Wolverine, which uses subfield VOLE to support any field size, and LPZK, which crucially relies on the fact that the VOLE-based commitments are linear relationships.

Similar to prior work, we use a linear scan on the circuit to compute the authenticated values on all the wires in the circuit. In particular, for each multiplication gate, the prover $\mathcal{P}$ has (w_α, m_α), (w_β, m_β), $(w_\gamma, m_\gamma) \in \mathbb{F}_p \times \mathbb{F}_{p^r}$, and the verifier $\mathcal{V}$ holds $k_\alpha, k_\beta, k_\gamma, \Delta \in \mathbb{F}_{p^r}$ such that the following four equations hold:

$$w_\gamma = w_\alpha \cdot w_\beta \quad \text{and} \quad m_i = k_i - w_i \cdot \Delta \ \text{ for } i \in \{\alpha, \beta, \gamma\}.$$

If $\mathcal{P}$ is malicious, the first equation could potentially be incorrect and our task is to check that this relationship holds for all gates. Although the last three equations are linear equations from the perspective of the verifier, the first equation is not linear. The crucial observation is that it is possible to convert the non-linear checking to a linear checking. Specifically, we observe that for the i-th multiplication gate with wire values $(w_\alpha, w_\beta, w_\gamma)$, if it is computed correctly (i.e., $w_\gamma = w_\alpha \cdot w_\beta$), then we have the following relation:

$$\begin{aligned}
&\overbrace{B_i = k_\alpha \cdot k_\beta - k_\gamma \cdot \Delta}^{\text{known to } \mathcal{V}} \\
&= (m_\alpha + w_\alpha \cdot \Delta) \cdot (m_\beta + w_\beta \cdot \Delta) - (m_\gamma + w_\gamma \cdot \Delta) \cdot \Delta \\
&= m_\alpha \cdot m_\beta + (w_\beta \cdot m_\alpha + w_\alpha \cdot m_\beta - m_\gamma) \cdot \Delta + (w_\alpha \cdot w_\beta - w_\gamma) \cdot \Delta^2 \\
&= \underbrace{m_\alpha \cdot m_\beta}_{\substack{\text{known to } \mathcal{P} \\ \text{denoted as } A_{0,i}}} + \underbrace{(w_\beta \cdot m_\alpha + w_\alpha \cdot m_\beta - m_\gamma)}_{\substack{\text{known to } \mathcal{P} \\ \text{denoted as } A_{1,i}}} \cdot \underbrace{\Delta}_{\substack{\text{known to } \mathcal{V} \\ \text{global key}}}
\end{aligned}$$

We can see that the above relationship is now linear and very similar to the IT-MAC relationship. What's more, we also show in Section 4 that if the underlying wire values (i.e., $w_\alpha, w_\beta, w_\gamma$) are not computed correctly, then the above relationship can hold only with probability $2/p^r$: now it becomes a quadratic equation of Δ, where there are at most two values of Δ that satisfy the equation.

Now when we look at a circuit with t multiplication gates, we can obtain one such relationship for each gate. Namely, for each $i \in [t]$, $\mathcal{P}$ has $A_{0,i}, A_{1,i} \in \mathbb{F}_{p^r}$ and $\mathcal{V}$ has $B_i \in \mathbb{F}_{p^r}$ such that $B_i = A_{0,i} + A_{1,i} \cdot \Delta$. We can check all t linear relations in a batch using a random linear combination. In particular, the verifier samples a uniform element $\chi \in \mathbb{F}_{p^r}$ *after* the above values have been defined, and then checks that the following relationship holds:

$$\underbrace{\sum_{i \in [t]} B_i \cdot \chi^i}_{\substack{\text{known to } \mathcal{V} \\ \text{denoted as } B}} = \underbrace{\sum_{i \in [t]} A_{0,i} \cdot \chi^i}_{\substack{\text{known to } \mathcal{P} \\ \text{denoted as } A_0}} + \underbrace{\Big(\sum_{i \in [t]} A_{1,i} \cdot \chi^i\Big)}_{\substack{\text{known to } \mathcal{P} \\ \text{denoted as } A_1}} \cdot \underbrace{\Delta}_{\substack{\text{known to } \mathcal{V} \\ \text{global key}}}$$

By the verifier sending just one field element (i.e., χ), we are able to reduce checking t equations in the circuit to checking the above single equation, that is $B = A_0 + A_1 \cdot \Delta$, where $\mathcal{V}$ has B and Δ, while $\mathcal{P}$ has A_0 and A_1. This could be easily checked by using a random linear relationship $B^* = A_0^* + A_1^* \cdot \Delta$ with

$B^*, A_0^*, A_1^* \in \mathbb{F}_{p^r}$ to mask field elements A_0 and A_1, and then opening the masked elements. In particular, $\mathcal{P}$ sends $U = A_0 + A_0^*$ and $V = A_1 + A_1^*$ to $\mathcal{V}$, who checks that $B + B^* = U + V \cdot \Delta$. Finally, this random linear relationship over $\mathbb{F}_{p^r}$ can be easily obtained by generating subfield VOLE correlations on $\mathbb{F}_p$ and packing them to $\mathbb{F}_{p^r}$.

Note that the online phase of the ZK protocol where the circuit and witness are known, can be made *non-interactive* by computing χ using a random oracle to hash the transcript up to that point. In Section 4, we provide the detailed description of our ZK protocol for circuit satisfiability and prove that it is UC-secure. We report the performance of the protocol in Section 6.

3.2 Zero-Knowledge Proof for Polynomial Sets

The above ZK protocol is great: for Boolean circuits, it sends only one bit per AND gate in the sVOLE-hybrid model and supports free-XOR. We believe that in the gate-by-gate paradigm, the communication cost is *optimal*: intuitively, each AND gate has to require some communication when we process each gate individually.

This reminds us of the successful story of garbling, where a progression of research reached the half-gates garbling scheme [ZRE15] proven to be optimal in the gate-by-gate paradigm. However, a lower bound is merely a way to rule out certain approaches and in this case, the garbling lower bound implies that a better garbling scheme either needs to use something beyond random oracle and linear operations, or needs to go beyond the gate-by-gate paradigm and look at more gates as a whole. Indeed, the follow-up works [MPas15, BMR16, KP17] have shown that it is indeed possible to garble a formula more efficiently. In our second protocol, we apply the same philosophy to our ZK protocol for circuit satisfiability.

Our ZK protocol: proving inner product with smaller cost. Recall that in the above idea when we prove a multiplication gate with wire values (x, y, z), two parties essentially prove $f(x, y, z) = x \cdot y - z = 0$. This could be viewed as a degree-2 polynomial on three variables. Let's first generalize it to a degree-2 polynomial with more than just one multiplication. Suppose f is a degree-2 polynomial such that $f(x_1, \ldots, x_n) = c_0 + \sum_{i\in[n/2]} c_i \cdot x_i \cdot x_{n/2+i}$. Two parties hold authenticated values $[w_1], \ldots, [w_n]$, and the prover wants to prove $f(w_1, \ldots, w_n) = 0$. We will follow the similar thinking process as above. In particular, observe that

$$\begin{aligned}
f(k_1, \ldots, k_n) &= c_0 + \sum_{i\in[n/2]} c_i \cdot k_i \cdot k_{n/2+i} \\
&= c_0 + \sum_{i\in[n/2]} c_i \cdot (m_i + w_i \cdot \Delta) \cdot (m_{n/2+i} + w_{n/2+i} \cdot \Delta) \\
&= \Big(c_0 + \sum_{i\in[n/2]} c_i \cdot m_i \cdot m_{n/2+i}\Big) + \Big(\sum_{i\in[n/2]} c_i \cdot m_i \cdot w_{n/2+i} + c_i \cdot m_{n/2+i} \cdot w_i\Big) \cdot \Delta + \\
&\quad \Big(\sum_{i\in[n/2]} c_i \cdot w_i \cdot w_{n/2+i}\Big) \cdot \Delta^2 \\
&= \Big(c_0 + \sum_{i\in[n/2]} c_i \cdot m_i \cdot m_{n/2+i}\Big) + \Big(\sum_{i\in[n/2]} c_i \cdot m_i \cdot w_{n/2+i} + c_i \cdot m_{n/2+i} \cdot w_i\Big) \cdot \Delta - c_0 \cdot \Delta^2.
\end{aligned}$$

The last equation is due to the fact that $f(w_1, \ldots, w_n) = c_0 + \sum_{i\in[n/2]} c_i \cdot w_i \cdot w_{n/2+i} = 0$. Reorganizing the above equation a bit, we can obtain the following equation:

$$\underbrace{f(k_1, \ldots, k_n) + c_0 \cdot \Delta^2}_{\text{known to } \mathcal{V}\text{, denoted as } B} = \underbrace{\Big(c_0 + \sum_{i\in[n/2]} c_i \cdot m_i \cdot m_{n/2+i}\Big)}_{\text{known to } \mathcal{P}\text{, denoted as } A_0} + \underbrace{\Big(\sum_{i\in[n/2]} c_i \cdot m_i \cdot w_{n/2+i} + c_i \cdot m_{n/2+i} \cdot w_i\Big)}_{\text{known to } \mathcal{P}\text{, denoted as } A_1} \cdot \Delta.$$

This is still a linear relationship $B = A_0 + A_1 \cdot \Delta$, which we could prove easily as in our first protocol. Essentially, we can prove a degree-2 polynomial with $n/2$ multiplications with a communication cost of just $O(1)$, in addition to the cost of committing the witness. This is independent of the number of multiplications in the polynomial, which could be as many as $O(n)$. One immediate observation is that if we have t such polynomials to be proven, the total communication cost is still $O(1)$ rather than $O(t)$, by using the same random-linear-combination idea to reduce all linear checks to a single check. This protocol immediately allows us to prove inner product faster than the circuit-based ZK protocol.

Our ZK protocol: proving arbitrary degree-d polynomials. The above protocol is not as generalized as we want in two aspects: 1) it only handles degree-2 polynomials, not ones with a higher degree; 2) when being executed for t times, we are assuming the polynomials are the same. Now we generalize it to the most extensive format. We assume that the witness is $(w_1, \ldots, w_n) \in \mathbb{F}_p^n$; there are totally t polynomials to be proven and each multivariable polynomial $f_i(x_1, \ldots, x_n)$ over $\mathbb{F}_p$ has a degree at most d. The prover wants to prove that $f_i(w_1, \ldots, w_n) = 0$ for all $i \in [t]$. Note that a polynomial can only take a subset of witness as input, and the witnesses that are not used will be not involved in the evaluation of the polynomial. Below, we show how to prove such polynomial set in communication of d field elements over $\mathbb{F}_{p^r}$, in addition to the n field elements over $\mathbb{F}_p$ to commit the witness.

When there are n variables and degree up to d, there are surprisingly high number of terms, making it difficult to even write it out. Note that in practice commonly used polynomials only have a small number of terms, but we do not want to limit in any way for our ZK protocol. As a result, for every n-variable d-degree polynomial $f \in \{f_1, \ldots, f_t\}$, we will represent it as $f(x_1, \ldots, x_n) = \sum_{h \in [0,d]} g_h(x_1, \ldots, x_n)$, where g_h is a degree-h polynomial such that all terms in g_h have exactly degree h. This turns out to help in our calculation. Now we apply the same idea as follows:

$$\begin{aligned}
&f(k_1, \ldots, k_n) = f(m_1 + w_1 \cdot \Delta, \ldots, m_n + w_n \cdot \Delta) \\
&= \sum_{h \in [0,d]} g_h(m_1 + w_1 \cdot \Delta, \ldots, m_n + w_n \cdot \Delta) \\
&= \sum_{h \in [0,d]} \Big(g_h(w_1, \ldots, w_n) \cdot \Delta^h + \sum_{j \in [0,h-1]} A_h^j \cdot \Delta^j \Big) \\
&= \sum_{h \in [0,d]} g_h(w_1, \ldots, w_n) \cdot \Delta^h + \sum_{h \in [0,d]} \sum_{j \in [0,h-1]} A_h^j \cdot \Delta^j.
\end{aligned}$$

In the above equation, A_h^j is the coefficient in front of x^j for the univariate polynomial $q_h(x) = g_h(m_1 + w_1 \cdot x, \ldots, m_n + w_n \cdot x)$. Therefore, $q_h(\Delta) = g_h(w_1, \ldots, w_n) \cdot \Delta^h + \sum_{j \in [0,h-1]} A_h^j \cdot \Delta^j$.

At this point, we are stuck on the first part of the above equation. Recall that the protocol works in the multiplication-gate case, as the polynomial $f(x, y, z) = x \cdot y - z$ is exactly the coefficient of Δ^2 and thus proving that the relationship is linear implies that this coefficient is zero. In the case of inner product, there is a constant term c_0, but we are able to get around with it since c_0 is public. Here, the polynomial is generic and there may be lower-order terms on Δ. Ideally, we want an equation with the term $\left(\sum_{h \in [0,d]} g_h(w_1, \ldots, w_n)\right) \cdot \Delta^d$, and then prove that the rest part of the equation is a $(d-1)$-degree polynomial. To sum up, we need a way to shift each sub-polynomial so that the terms align. It turns out to be possible. See below.

Now instead of evaluating $f(k_1, \ldots, k_n)$ as we always do, we will decompose this polynomial and shift each sub-polynomial. The verifier now computes

$$\begin{aligned}
\sum_{h\in[0,d]} g_h(k_1,\ldots,k_n)\cdot\Delta^{d-h} &= \sum_{h\in[0,d]} g_h(m_1+w_1\cdot\Delta,\ldots,m_n+w_n\cdot\Delta)\cdot\Delta^{d-h}\\
&= \sum_{h\in[0,d]}\Big(g_h(w_1,\ldots,w_n)\cdot\Delta^d + \sum_{j\in[0,h-1]} A_h^j\cdot\Delta^{j+d-h}\Big)\\
&= \sum_{h\in[0,d]} g_h(w_1,\ldots,w_n)\cdot\Delta^d + \sum_{h\in[0,d-1]} A_h\cdot\Delta^h\\
&= f(w_1,\ldots,w_n)\cdot\Delta^d + \sum_{h\in[0,d-1]} A_h\cdot\Delta^h\\
&= \sum_{h\in[0,d-1]} A_h\cdot\Delta^h.
\end{aligned}$$

Here A_h^j is defined as above, and A_h is the aggregated coefficient for all terms with Δ^h. Note that the prover with witnesses $\{w_i\}$ and MACs $\{m_i\}$ can compute all the coefficients locally. The coefficients $\{A_h\}$ are polynomial coefficients when we treat it as a single-variable polynomial on Δ. Therefore, we can compute $\{A_h\}$ efficiently by evaluating the polynomial on $d+1$ points and then use number theoretic transformation to interpolate it. In many practical applications, the polynomial is usually simple and thus the coefficients can be derived without using the above generic approach. This relationship can be viewed as an *oblivious polynomial evaluation* (OPE), where the verifier has Δ and the prover has a polynomial $P(x)=\sum_{h\in[0,d-1]} A_h\cdot x^h$ over $\mathbb{F}_{p^r}$. The verifier wants to check that the resulting evaluation in the above equation is the same as $P(\Delta)$. It is not hard to check the above polynomial relation, as sVOLE can be used to generate (V)OPE in an efficient way (see Section 4.1 for details). Similarly, we can perform the checks for all t polynomials in a batch using the random linear combination. This results in a total communication of $(n+dr)\log p$ bits in the sVOLE-hybrid model.

3.3 Other Useful Results

By applying the above results in different settings, we can obtain the following interesting results.

Proving knowledge of solutions to lattice problems. Immediately, our ZK protocol shows a way to prove low-degree polynomials with very high efficiency. This could help us to prove knowledge of a solution of the *short integer solution* (SIS) problem. In particular, we assume that the prover knows a vector $\boldsymbol{s}$, such that $\mathbf{A}\cdot\boldsymbol{s}=\boldsymbol{t}$ and $\boldsymbol{s}\in[-B,B]^m$ for a small value $B\in\mathbb{N}$ (or $\boldsymbol{s}\in\{0,1\}^m$), where both parties know the public matrix $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$ and vector $\boldsymbol{t}\in\mathbb{Z}_q^n$ (here we assume that q is a prime). The main cost is to prove that all input values are bounded, which can be modeled as a set of low-degree polynomials, something our ZK protocol is very good at. As a result, our ZK protocol significantly outperforms all prior works for proving knowledge of solutions to SIS problems. In particular, for $\boldsymbol{s}\in\{-1,0,1\}^m$, our protocol uses $8\times$ less communication and $110\times$ less execution-time compared to the state-of-the-art protocol Wolverine [WYKW20]. See Section 5.2 for details of our solution and Section 6.2 for performance evaluation.

Proving matrix multiplication with sublinear communication. Here, the prover intends to prove that $\mathbf{A}\cdot\mathbf{B}=\mathbf{C}$, where $\mathbf{A},\mathbf{B}\in\mathbb{F}_p^{n\times n}$ are two secret matrices and $\mathbf{C}\in\mathbb{F}_p^n$ is a public matrix. Using the circuit-based ZK protocol, this will need $O(n^3)$ communication. In the previous section, we present a ZK protocol for inner product of two vectors with $O(1)$ communication (in addition to the cost of committing the witness), which is independent of the number of the inner-product executions. This immediately gives us a ZK protocol for proving matrix multiplication with $O(n^2)$ communication for committing two secret

Functionality $\mathcal{F}^{p,r}_{\text{ext-sVOLE}}$

Initialize: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$, sample $\Delta \leftarrow \mathbb{F}_{p^r}$ if $\mathcal{V}$ is honest, and receive $\Delta \in \mathbb{F}_{p^r}$ from the adversary otherwise. Store Δ and send it to $\mathcal{V}$, and ignore all subsequent (init) commands.

Extend: This procedure can be run multiple times. Upon receiving (extend, ℓ) from $\mathcal{P}$ and $\mathcal{V}$, do the following:

1. If $\mathcal{V}$ is honest, sample $\boldsymbol{k} \leftarrow \mathbb{F}^{\ell}_{p^r}$. Otherwise, receive $\boldsymbol{k} \in \mathbb{F}^{\ell}_{p^r}$ from the adversary.
2. If $\mathcal{P}$ is honest, sample $\boldsymbol{x} \leftarrow \mathbb{F}^{\ell}_{p}$ and compute $\boldsymbol{m} := \boldsymbol{k} - \Delta \cdot \boldsymbol{x} \in \mathbb{F}^{\ell}_{p^r}$. Otherwise, receive $\boldsymbol{x} \in \mathbb{F}^{\ell}_{p}$ and $\boldsymbol{m} \in \mathbb{F}^{\ell}_{p^r}$ from the adversary, and then recompute $\boldsymbol{k} := \boldsymbol{m} + \Delta \cdot \boldsymbol{x} \in \mathbb{F}^{\ell}_{p^r}$.
3. Send $(\boldsymbol{x}, \boldsymbol{m})$ to $\mathcal{P}$ and $\boldsymbol{k}$ to $\mathcal{V}$.

Vector Oblivious Polynomial Evaluation: Upon receiving (VOPE, d) from $\mathcal{P}$ and $\mathcal{V}$, do the following:

1. If $\mathcal{V}$ is honest, sample $B \leftarrow \mathbb{F}_{p^r}$. Otherwise, receive $B \in \mathbb{F}_{p^r}$ from the adversary.
2. If $\mathcal{P}$ is honest, sample $A_i \leftarrow \mathbb{F}_{p^r}$ for $i \in [d]$ and compute $A_0 := B - \sum_{i\in[d]} A_i \cdot \Delta^i$. Otherwise, receive $\{A_i\}_{i\in[0,d]}$ with $A_i \in \mathbb{F}_{p^r}$ from the adversary and re-compute $B := \sum_{i\in[0,d]} A_i \cdot \Delta^i$.
3. Send $\{A_i\}_{i\in[0,d]}$ to $\mathcal{P}$ and B to $\mathcal{V}$.

Figure 2: Functionality for extended subfield VOLE.

matrices and additional $O(1)$ communication for proving the matrix relationship ($\mathbf{A} \cdot \mathbf{B}$ has n^2 executions of inner product between two vectors of dimension n).

Proving integer multiplication over a ring with a linear amortized communication cost. The multiplication of two n-bit integers needs $O(n^2)$ AND gates. Although multiplication is more efficient over large fields, we may prefer computing over a ring $\mathbb{Z}_{2^n}$ (e.g., $n = 32$), for applications where matching cleartext computation is crucial. Now, we could actually prove the multiplication of two integers in communication of $O(n)$ bits. In particular, it is known that n-bit multiplier can be represented as a Boolean circuit of depth $2\log n + 3$ [BHWK16]. Therefore, the polynomial that represents each bit in the multiplication has degree at most $8n^2$. If there are t integer multiplications to be proven, the amortized communication cost of each multiplication will be $3n + \frac{8n^2\kappa}{t}$ bits, which becomes $4n$ bits when $t \approx 8n\kappa$. Here the communication of $3n$ bits for each integer multiplication is used to commit input and output values.

Generic sublinear ZK proof via amortization. If the amortization is considered, then one immediate result is that we can prove t number of circuits with each N multiplication gates on totally n variables in amortized communication sublinear to the circuit size. This is because the polynomial that represents an N-sized circuit can have a degree at most N. If the number $t = O(\sqrt{N})$ of circuits to be proven, then the amortized communication cost per circuit is $O(\frac{n+N}{t}) \approx O(\sqrt{N})$. If each circuit has a more compact polynomial representation (in terms of the highest degree), the communication saving will be even higher.

We could develop on top of this idea for general circuits with *weak* uniformity. Specifically, given a large circuit C, we can identify all sub-circuits denoted by $C_1, \ldots, C_t$, that have compact polynomial representations. Now, we can prove the evaluation of gates in $C \setminus \{C_1, \ldots, C_t\}$ using our ZK protocol for circuit satisfiability, which provides us with the commitments on all input wires and the wires related to the gates. Then we can prove all the sub-circuits $C_1, \ldots, C_t$ in a communication cost linear to the highest degree of all the circuits represented as polynomials. Note that the input and output values of all the sub-circuits have been committed in the first step, and thus no extra cost is needed. Here we only need a reasonably *weak* uniformity: we *do not* require all sub-circuits to be of the same size or the same topology; instead we only need that the polynomial representations of sub-circuits are all bounded by some degree d.

4 Zero-Knowledge Proof for Circuit Satisfiability over Any Field

In this section, we present our ZK protocol for circuit satisfiability over any field with communication of only one field element per multiplication gate using sVOLE as a subroutine. First of all, we introduce a functionality (and the corresponding protocol) that extends sVOLE to additionally support *vector oblivious polynomial evaluation* (VOPE), which is crucial for our ZK protocols in this section and the next section. Then, we provide the details of our ZK protocol, and give the formal security proof.

4.1 Extended Subfield Vector Oblivious Linear Evaluation

Extended sVOLE functionality. To accommodate our efficient ZK protocols for circuits and polynomial sets (described in Section 4.2 and Section 5), we propose an extended sVOLE functionality $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ defined in Figure 2 to generate authenticated values and special correlations related to random polynomials. This functionality is the same as that shown in Figure 6 of Appendix A, except that it additionally allows two parties to obtain *vector oblivious polynomial evaluation* (VOPE) correlations over $\mathbb{F}_{p^r}$ with the guarantee that the same global key Δ is used between sVOLE and VOPE. In particular, given a polynomial-degree d input by both parties, this functionality will sample $d+1$ uniform coefficients over extension field $\mathbb{F}_{p^r}$ to define a random polynomial g, and then output the coefficients to a party $\mathcal{P}$ and $g(\Delta)$ to the other party $\mathcal{V}$.

Protocol for realizing extended sVOLE functionality. We construct the protocol to UC-realize functionality $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ by extending the sVOLE protocol. Recall that our extended functionality can be viewed as adding the support to output VOPE correlations over extension field $\mathbb{F}_{p^r}$. Our protocol to accomplish it takes two steps: 1) packing subfield VOLE correlations between $\mathbb{F}_p$ and $\mathbb{F}_{p^r}$ into VOLE correlations over $\mathbb{F}_{p^r}$; 2) multiplying independent VOLE correlations to obtain a VOPE correlation. We note that a malicious party $\mathcal{V}$ could cause the outputting coefficients $A_1, \ldots, A_{d-1}$ of honest party $\mathcal{P}$ to be always 0 by setting $\Delta = 0$ and all its keys as 0. To prevent the attack, we *iteratively* multiply the VOLE correlations over $\mathbb{F}_{p^r}$, and use an extra independent VOLE correlation to randomize the product of VOLE correlations after multiplication is computed in every iteration. Details of the protocol are described in Figure 3.

The security of this protocol is proved in the following theorem, where the proof of this theorem is postponed to Appendix B.

Theorem 1. *Protocol $\Pi^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ shown in Figure 3 UC-realizes functionality $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ with statistical error $(d-1)/p^r$ and the information-theoretic security in the $\mathcal{F}^{p,r}_{\mathsf{sVOLE}}$-hybrid model.*

4.2 Our ZK Protocol using extended sVOLE

We have already discussed the intuition of our ZK protocol for circuit satisfiability over any field in Section 3.1, and thus here directly describe the details of the protocol in Figure 4. In the following, we prove the security of our ZK protocol, and show that the online phase can be made *non-interactive* in the random oracle model.

Proof of security. When both parties are honest, we easily see that the verifier will output true with probability 1. In particular, according to the description in Section 3.1, we have that the check in protocol $\Pi^{p,r}_{\mathsf{ZK}}$ always passes for an honest execution. For an honest protocol execution, we always have that $w_h = 1$ (and thus $k_h = m_h + \Delta$) for the single output wire h. Overall, our ZK protocol $\Pi^{p,r}_{\mathsf{ZK}}$ shown in Figure 4 has the perfect completeness.

Theorem 2. *Protocol $\Pi^{p,r}_{\mathsf{ZK}}$ UC-realizes functionality $\mathcal{F}_{\mathsf{ZK}}$ that proves the circuit satisfiability in the $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$-hybrid model with soundness error $(t+3)/p^r$ and the information-theoretic security.*

Protocol $\Pi^{p,r}_{\mathsf{ext\text{-}sVOLE}}$

Initialize. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}^{p,r}_{\mathsf{sVOLE}}$, which returns a uniform $\Delta \in \mathbb{F}_{p^r}$ to $\mathcal{V}$.

Generate sVOLE correlations. On input (extend, ℓ), two parties $\mathcal{P}$ and $\mathcal{V}$ call $\mathcal{F}^{p,r}_{\mathsf{sVOLE}}$ to directly generate ℓ sVOLE correlations.

Generate VOPE correlations. On input (VOPE, d), two parties $\mathcal{P}$ and $\mathcal{V}$ execute the following:

1. For each $i \in [2d-1]$, two parties perform as follows:
   (a) Both parties send (extend, r) to $\mathcal{F}^{p,r}_{\mathsf{sVOLE}}$, which returns $\{(m_h, u_h)\}_{h\in[r]}$ to $\mathcal{P}$ and $\{k_h\}_{h\in[r]}$ to $\mathcal{V}$ such that $k_h = m_h + u_h \cdot \Delta \in \mathbb{F}_{p^r}$ and $u_h \in \mathbb{F}_p$ for $h \in [r]$.
   (b) $\mathcal{P}$ computes $M_i := \sum_{h\in[r]} m_h \cdot X^{h-1} \in \mathbb{F}_{p^r}$ and $U_i := \sum_{h\in[r]} u_h \cdot X^{h-1} \in \mathbb{F}_{p^r}$, and $\mathcal{V}$ computes $K_i := \sum_{h\in[r]} k_h \cdot X^{h-1} \in \mathbb{F}_{p^r}$, where $K_i = M_i + U_i \cdot \Delta$.
2. $\mathcal{P}$ defines $g_1(x) = M_1 + U_1 \cdot x$, and $\mathcal{V}$ sets $B_1 = K_1$. If $d > 1$, from $i = 1$ to $d-1$, two parties execute as follows:
   (a) $\mathcal{P}$ computes the following univariate polynomial:
   $$g_{i+1}(x) = g_i(x) \cdot (M_{i+1} + U_{i+1} \cdot x) + (M_{d+i} + U_{d+i} \cdot x).$$
   (b) $\mathcal{V}$ computes $B_{i+1} := B_i \cdot K_{i+1} + K_{d+i}$.

   Then, $\mathcal{P}$ computes the coefficients $\{A_i\}_{i\in[0,d]}$ locally such that $g_d(x) = \sum_{i\in[0,d]} A_i \cdot x^i$, and $\mathcal{V}$ defines $B := B_d$.
3. $\mathcal{P}$ outputs $\{A_i\}_{i\in[0,d]}$; $\mathcal{V}$ outputs B.

Figure 3: Protocol for extended subfield VOLE in the $\mathcal{F}^{p,r}_{\mathsf{sVOLE}}$-hybrid model.

Proof. We first consider the case of a malicious prover (i.e., soundness and knowledge extraction) and then consider the case of a malicious verifier (i.e., zero knowledge). In each case, we construct a *probabilistic polynomial time* (PPT) simulator $\mathcal{S}$ given access to $\mathcal{F}_{\mathsf{ZK}}$, and running the adversary $\mathcal{A}$ as a subroutine while emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ for $\mathcal{A}$. We always implicitly assume that $\mathcal{S}$ passes all communication between adversary $\mathcal{A}$ and environment $\mathcal{Z}$.

Malicious prover. By emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ for $\mathcal{A}$ by choosing uniform $\Delta \in \mathbb{F}_{p^r}$, and recording all the values $\{\mu_i\}_{i\in[n]}$ and $\{\nu_i\}_{i\in[N]}$ and their corresponding MAC tags, which are received by $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ from adversary $\mathcal{A}$. These values define the corresponding keys in the natural way. When emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, $\mathcal{S}$ also receives $(A_0^*, A_1^*) \in (\mathbb{F}_{p^r})^2$ from $\mathcal{A}$ and defines B^* accordingly.
2. When $\mathcal{A}$ sends $\{\delta_i\}_{i\in\mathcal{I}_{\mathsf{in}}}$ in step 4, $\mathcal{S}$ computes $w_i := \delta_i + \mu_i \in \mathbb{F}_p$ for $i \in \mathcal{I}_{\mathsf{in}}$.
3. $\mathcal{S}$ executes the rest of the protocol as an honest verifier, using Δ and the keys defined in the first step. If the honest verifier outputs false, then $\mathcal{S}$ sends $\boldsymbol{w} = \perp$ and $\mathcal{C}$ to $\mathcal{F}_{\mathsf{ZK}}$ and aborts. If the honest verifier outputs true, then $\mathcal{S}$ sends $\boldsymbol{w}$ and $\mathcal{C}$ to $\mathcal{F}_{\mathsf{ZK}}$ where $\boldsymbol{w} = (w_1, \ldots, w_n)$ is defined as above.

Clearly, the view of adversary $\mathcal{A}$ simulated by $\mathcal{S}$ has the identical distribution as its view in the real-world execution. Whenever the verifier in the real-world execution outputs false, the verifier in the ideal-world execution outputs false as well (since $\mathcal{S}$ sends $\perp$ to $\mathcal{F}_{\mathsf{ZK}}$ in this case). Thus, it only remains to bound the probability that the verifier in the real-world execution outputs true but the witness $\boldsymbol{w}$ sent by $\mathcal{S}$ to $\mathcal{F}_{\mathsf{ZK}}$

Protocol $\Pi_{\mathsf{ZK}}^{p,r}$

Inputs: The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ hold a circuit $\mathcal{C}$ over any field $\mathbb{F}_p$ with t multiplication gates. Prover $\mathcal{P}$ also holds a witness $\boldsymbol{w}$ such that $\mathcal{C}(\boldsymbol{w}) = 1$ and $|\boldsymbol{w}| = n$ (i.e., $|\mathcal{I}_{\mathsf{in}}| = n$).

Preprocessing phase: Both the circuit and witness are unknown.

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$, which returns a uniform $\Delta \in \mathbb{F}_{p^r}$ to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ send $(\mathsf{extend}, n+t)$ to $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$, which returns authenticated values $\{[\mu_i]\}_{i\in[n]}$ and $\{[\nu_i]\}_{i\in[t]}$ to the parties.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\mathsf{VOPE}, 1)$ to $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$, which returns uniform (A_0^*, A_1^*) to $\mathcal{P}$ and B^* to $\mathcal{V}$, such that $B^* = A_0^* + A_1^* \cdot \Delta$.

Online phase: Now the circuit and witness are known by the parties.

4. For $i \in \mathcal{I}_{\mathsf{in}}$, $\mathcal{P}$ sends $\delta_i := w_i - \mu_i \in \mathbb{F}_p$ to $\mathcal{V}$, and then both parties compute $[w_i] := [\mu_i] + \delta_i$.
5. For each gate $(\alpha, \beta, \gamma, T) \in \mathcal{C}$, in a topological order:
   - If $T = \mathsf{Add}$, then two parties locally compute $[w_\gamma] := [w_\alpha] + [w_\beta]$.
   - If $T = \mathsf{Mult}$ and this is the i-th multiplication gate, $\mathcal{P}$ sends $d_i := w_\alpha \cdot w_\beta - \nu_i \in \mathbb{F}_p$ to $\mathcal{V}$, and then both parties compute $[w_\gamma] := [\nu_i] + d_i$ (with $w_\gamma = w_\alpha \cdot w_\beta$ in the honest case).
6. For the i-th multiplication gate, two parties hold an authenticated triple $([w_\alpha], [w_\beta], [w_\gamma])$ (with $k_i = m_i + w_i \cdot \Delta$ for $i \in \{\alpha, \beta, \gamma\}$) from the previous step and execute the following:
   - $\mathcal{P}$ computes $A_{0,i} := m_\alpha \cdot m_\beta \in \mathbb{F}_{p^r}$ and $A_{1,i} := w_\alpha \cdot m_\beta + w_\beta \cdot m_\alpha - m_\gamma \in \mathbb{F}_{p^r}$.
   - $\mathcal{V}$ computes $B_i := k_\alpha \cdot k_\beta - k_\gamma \cdot \Delta \in \mathbb{F}_{p^r}$.
7. $\mathcal{P}$ and $\mathcal{V}$ perform the following check to verify that $B_i = A_{0,i} + A_{1,i} \cdot \Delta$ for all $i \in [t]$.
   (a) $\mathcal{V}$ samples $\chi \leftarrow \mathbb{F}_{p^r}$ and sends it to $\mathcal{P}$.
   (b) $\mathcal{P}$ computes $U := \sum_{i\in[t]} A_{0,i} \cdot \chi^i + A_0^*$ and $V := \sum_{i\in[t]} A_{1,i} \cdot \chi^i + A_1^*$, and sends (U, V) to $\mathcal{V}$.
   (c) Then $\mathcal{V}$ computes $W := \sum_{i\in[t]} B_i \cdot \chi^i + B^*$ and checks that $W = U + V \cdot \Delta$. If the check fails, $\mathcal{V}$ outputs false and aborts.
8. For the single output wire h in the circuit $\mathcal{C}$, both parties hold $[w_h]$ with $k_h = m_h + w_h \cdot \Delta$, and check that $w_h = 1$ as follows:
   - In parallel with the previous step, $\mathcal{P}$ sends m_h to $\mathcal{V}$.
   - $\mathcal{V}$ checks that $k_h = m_h + \Delta$. If the check fails, then $\mathcal{V}$ outputs false. Otherwise, $\mathcal{V}$ outputs true.

Figure 4: Zero-knowledge protocol for circuit satisfiability over any field in the $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$-hybrid model.

satisfies $\mathcal{C}(\boldsymbol{w}) = 0$. In the following, we show that if $\mathcal{C}(\boldsymbol{w}) = 0$ then the probability that the honest verifier in the real-world execution outputs true is at most $(t+3)/p^r$.

By induction, we prove that all the values on the wires in the circuit are correct. It is trivial that the values associated with the input wires and the output wires of Add gates are computed correctly. Therefore, we focus on analyzing the correctness of the values related to the output wires of Mult gates. When we analyze the correctness of the output value with respect to the i-th multiplication gate, we always assume that the output values associated with the first $(i-1)$ multiplication gates are correct by induction. For the i-th multiplication gate, two parties hold an authenticated triple $([w_\alpha], [w_\beta], [w_\gamma])$ with $w_\gamma = w_\alpha \cdot w_\beta + e_i$, where $e_i \in \mathbb{F}_p$ is an error chosen by adversary $\mathcal{A}$ by sending an incorrect value d_i' in step 5 of protocol $\Pi_{\mathsf{ZK}}^{p,r}$.

Thus, we have $k_\gamma = m_\gamma + w_\gamma \cdot \Delta = m_\gamma + (w_\alpha \cdot w_\beta) \cdot \Delta + e_i \cdot \Delta$. Further, we have:

$$\begin{aligned} B_i &= k_\alpha \cdot k_\beta - k_\gamma \cdot \Delta \\ &= (m_\alpha + w_\alpha \cdot \Delta) \cdot (m_\beta + w_\beta \cdot \Delta) - (m_\gamma + w_\alpha \cdot w_\beta \cdot \Delta + e_i \cdot \Delta) \cdot \Delta \\ &= m_\alpha \cdot m_\beta + (w_\alpha \cdot m_\beta + w_\beta \cdot m_\alpha - m_\gamma) \cdot \Delta - e_i \cdot \Delta^2 \\ &= A_{0,i} + A_{1,i} \cdot \Delta - e_i \cdot \Delta^2. \end{aligned}$$

In step 7 of the ZK protocol, $\mathcal{A}$ sends $U' = U + E_u$ and $V' = V + E_v$ to the honest verifier, where $U, V \in \mathbb{F}_{p^r}$ are computed following the protocol description, and $E_u, E_v \in \mathbb{F}_{p^r}$ are the adversarially chosen errors. Furthermore, we have the following:

$$\begin{aligned} W &= \sum_{i\in[t]} B_i \cdot \chi^i + B^* \\ &= \sum_{i\in[t]} \left(A_{0,i} + A_{1,i} \cdot \Delta - e_i \cdot \Delta^2\right) \cdot \chi^i + A_0^* + A_1^* \cdot \Delta \\ &= U + V \cdot \Delta - \Big(\sum_{i\in[t]} e_i \cdot \chi^i\Big) \cdot \Delta^2 \\ &= \left(U' - E_u\right) + \left(V' - E_v\right) \cdot \Delta - \Big(\sum_{i\in[t]} e_i \cdot \chi^i\Big) \cdot \Delta^2. \end{aligned}$$

If the check passes in step 7, then we have that $W = U' + V' \cdot \Delta$. Therefore, we obtain that

$$E_u + E_v \cdot \Delta + \Big(\sum_{i\in[t]} e_i \cdot \chi^i\Big) \cdot \Delta^2 = 0.$$

If $\sum_{i\in[t]} e_i \cdot \chi^i \neq 0$, then the above equation holds with probability at most $2/p^r$, as $\Delta \in \mathbb{F}_{p^r}$ is uniformly random and kept secret from the adversary's view. Below, we consider that $\sum_{i\in[t]} e_i \cdot \chi^i = 0$. If there exists some $i \in [t]$ such that $e_i \neq 0$, the probability that $\sum_{i\in[t]} e_i \cdot \chi^i = 0$ is at most t/p^r, as χ is sampled uniformly at random after e_i for all $i \in [t]$ have been determined. Overall, all the values on the wires in the circuit are correct, except with probability at most $(t+2)/p^r$.

Now, we assume that all the values on the wires in the circuit are correct. If $\mathcal{C}(\boldsymbol{w}) = 0$ but the honest verifier outputs true in step 8, then adversary $\mathcal{A}$ must send $m_h + \Delta$ to the honest verifier where m_h is a MAC tag on output wire h known by $\mathcal{A}$. In other words, $\mathcal{A}$ learns Δ, which occurs with probability at most $1/p^r$.

In conclusion, any unbounded environment $\mathcal{Z}$ cannot distinguish between the real-world execution and ideal-world execution, except with probability $(t+3)/p^r$.

Malicious verifier. If $\mathcal{S}$ receives false from $\mathcal{F}_{\mathsf{ZK}}$, then it simply aborts. Otherwise, $\mathcal{S}$ interacts with adversary $\mathcal{A}$ as follows:

1. In the preprocessing phase, $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\text{ext-sVOLE}}$ by recording the global key Δ and the keys for all the authenticated values, that are sent to this functionality by $\mathcal{A}$. $\mathcal{S}$ also receives $B^* \in \mathbb{F}_{p^r}$ from $\mathcal{A}$ when emulating $\mathcal{F}^{p,r}_{\text{ext-sVOLE}}$.

2. $\mathcal{S}$ executes steps 4–5 of protocol $\Pi^{p,r}_{\mathsf{ZK}}$ by sending uniformly random δ_i for each $i \in \mathcal{I}_{\mathsf{in}}$ and d_i for the i-th multiplication gate to adversary $\mathcal{A}$.

3. $\mathcal{S}$ executes steps 6–7 of the protocol as an honest prover, except that sampling $V \leftarrow \mathbb{F}_{p^r}$ and computing $U := W - V \cdot \Delta$ where W is computed using Δ, B^* and the keys received from $\mathcal{A}$ following the protocol specification.

4. In step 8 of the protocol, $\mathcal{S}$ computes k_h (based on the keys sent to $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$ by $\mathcal{A}$) and then sets $m_h := k_h + \Delta$, where h is the single output wire. Then, $\mathcal{S}$ sends m_h to $\mathcal{A}$.

Since $\{\mu_i\}, \{\nu_i\}$ and A_1^* are uniformly random and perfectly hidden against the view of adversary $\mathcal{A}$, we easily obtain that the view of $\mathcal{A}$ simulated by $\mathcal{S}$ is distributed identically to its view in the real protocol execution. This completes the proof. □

In the protocol $\Pi_{\mathsf{ZK}}^{p,r}$ shown in Figure 4, if we set $p = 2^{61} - 1$ and $r = 1$, then the computation of χ^i for $i \in [t]$ is expensive (especially for large t). We can replace χ^i for $i \in [t]$ with independent uniform coefficient χ_i for $i \in [t]$ to obtain better computation efficiency. In this case, the verifier can send a random seed in $\{0,1\}^\kappa$ to the prover, and then both parties compute $\chi_1, \ldots, \chi_t$ using the seed and a random oracle. Now, the soundness error is bounded by $q/2^\kappa + 4/p^r$, where q is an upper bound of the number of random oracle queries made by the adversary. When using the random oracle, the security is guaranteed in the computational sense.

Non-interactive online phase. In the online phase of our protocol $\Pi_{\mathsf{ZK}}^{p,r}$, the verifier only sends a random coefficient χ to the prover. Thus, the communication cost is one field element per multiplication gate even without random oracle. But the online phase needs communication of three rounds.

We can use the Fiat-Shamir heuristic to make the online phase *non-interactive* at the cost of that the information-theoretic security is degraded to the computation security. Specifically, both parties can compute $\chi \in \mathbb{F}_{p^r}$ as $\mathsf{H}(d_1, \ldots, d_t)$, where $\mathsf{H} : \{0,1\}^* \to \mathbb{F}_{p^r}$ is a cryptographic hash function modeled as a random oracle and $p^r \geq 2^\kappa$. In this case, the soundness error for the batch check of multiplication gates together with the correctness of the single output is now bounded by $(q_{\mathsf{H}} + t + 3)/p^r \leq (q_{\mathsf{H}} + t + 3)/2^\kappa$, where q_{H} is an upper bound of the number of H queries made by the adversary. When we set $p = 2$ and $r = 128$, we can obtain a non-interactive online phase with a blazing-fast computation given hardware-instruction support.

5 Zero-Knowledge Proof for Polynomial Sets over Any Field

Recall that we have explained the intuition of our ZK protocol for polynomial sets in Section 3.2. Thus, we directly show the detailed protocol in Figure 5, and provide the formal security proof. We also present the practical applications of the ZK protocol, including how to optimize the zero-knowledge proofs for proving matrix multiplication, proving knowledge of a solution to an SIS problem, proving integer multiplication over a ring and proving the circuits with some level of weak uniformity.

In the ZK protocol shown in Figure 5, the prover can compute the coefficients $\{A_{i,h}\}_{h \in [0,d-1]}$ of polynomial $g_i(x)$ for $i \in [t]$ in the following *generic* way.

- $\mathcal{P}$ computes $y_{i,j} := g_i(\alpha_j)$ for $j \in [d+1]$, where $\alpha_1, \ldots, \alpha_{d+1}$ are any $d+1$ different fixed points over extension field $\mathbb{F}_{p^r}$.
- Then $\mathcal{P}$ can compute $g_i(x) = \sum_{j \in [d+1]} y_{i,j} \cdot \delta_j(x)$, where $\delta_j(x) = \prod_{k \neq j} \frac{x - \alpha_k}{\alpha_j - \alpha_k}$ is a fixed d-degree polynomial that can be precomputed in the preprocessing phase.

In a lot of practical applications, the polynomials $\{g_i(x)\}$ are usually simple, and thus the coefficients can be computed efficiently without the need of using the above generic approach.

5.1 Proof of Security

When both parties are honest, it is not hard to see that the verifier will always output `true` with probability 1. Specifically, from $k_i = m_i + w_i \cdot \Delta$ for $i \in [n]$ and the description in Section 3.2, we have that

Protocol $\Pi^{p,r}_{\mathsf{polyZK}}$

Inputs: The prover $\mathcal{P}$ and verifier $\mathcal{V}$ hold t number of d-degree polynomials $f_1, \ldots, f_t$ all over n variables. Each polynomial f_i is represented as $f_i = \sum_{h\in[0,d]} f_{i,h}$ where all terms in $f_{i,h}$ have degree h. $\mathcal{P}$ also holds a witness $\boldsymbol{w} \in \mathbb{F}_p^n$, such that $f_i(\boldsymbol{w}) = 0$ for all $i \in [t]$.

Preprocessing phase:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, which sends $\Delta \in \mathbb{F}_{p^r}$ to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ send (extend, n) to $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, which returns authenticated values $\{[s_i]\}_{i\in[n]}$ to the parties.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\mathsf{VOPE}, d-1)$ to $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, which returns $\{A^*_h\}_{h\in[0,d-1]}$ to $\mathcal{P}$, B^* to $\mathcal{V}$, such that $\sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h = B^*$.

Online phase:

4. For $i \in [n]$, $\mathcal{P}$ sends $\delta_i := w_i - s_i \in \mathbb{F}_p$ to $\mathcal{V}$, and both parties compute $[w_i] := [s_i] + \delta_i$. Note that $k_i = m_i + w_i \cdot \Delta$ for $i \in [n]$.
5. From $i = 1$ to t, for the i-th polynomial f_i, two parties perform the following:
   - $\mathcal{V}$ computes $B_i := \sum_{h\in[0,d]} f_{i,h}(k_1, \ldots, k_n) \cdot \Delta^{d-h}$.
   - $\mathcal{P}$ defines a univariate d-degree polynomial over field $\mathbb{F}_{p^r}$ as $g_i(x) = \sum_{h\in[0,d]} f_{i,h}(m_1 + w_1 \cdot x, \ldots, m_n + w_n \cdot x) \cdot x^{d-h}$, and computes the coefficients $\{A_{i,h}\}_{h\in[0,d]}$ such that $g_i(x) = \sum_{h\in[0,d]} A_{i,h} \cdot x^h$. Since $A_{i,d} = f_i(w_1, \ldots, w_n) = 0$, $g_i(x)$ can be written as $\sum_{h\in[0,d-1]} A_{i,h} \cdot x^h$.
6. Two parties perform the following to check that $\sum_{h\in[0,d-1]} A_{i,h} \cdot \Delta^h = B_i$ for all $i \in [t]$:
   (a) $\mathcal{V}$ samples $\chi \leftarrow \mathbb{F}_{p^r}$ and sends it to $\mathcal{P}$.
   (b) For all $h \in [0, d-1]$, $\mathcal{P}$ computes $U_h := \sum_{i\in[t]} A_{i,h} \cdot \chi^i + A^*_h$ and sends it to $\mathcal{V}$.
   (c) $\mathcal{V}$ computes $W := \sum_{i\in[t]} B_i \cdot \chi^i + B^*$ and checks that $W = \sum_{h\in[0,d-1]} U_h \cdot \Delta^h$. If the check fails, $\mathcal{V}$ outputs false; otherwise it outputs true.

Figure 5: Zero-knowledge for polynomial satisfiability over any field in the $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$-hybrid model.

$B_i = \sum_{h\in[0,d-1]} A_{i,h} \cdot \Delta^h$ for all $i \in [t]$. Together with $\sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h = B^*$, we obtain that the following holds:

$$\begin{aligned} W &= \sum_{i\in[t]} B_i \cdot \chi^i + B^* \\ &= \sum_{i\in[t]} \Big(\sum_{h\in[0,d-1]} A_{i,h} \cdot \Delta^h \Big) \cdot \chi^i + \sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h \\ &= \sum_{h\in[0,d-1]} \Big(\sum_{i\in[t]} A_{i,h} \cdot \chi^i + A^*_h \Big) \cdot \Delta^h = \sum_{h\in[0,d-1]} U_h \cdot \Delta^h. \end{aligned}$$

Thus, our protocol shown in Figure 5 has the perfect completeness.

Theorem 3. *Protocol $\Pi^{p,r}_{\mathsf{polyZK}}$ UC-realizes functionality $\mathcal{F}_{\mathsf{ZK}}$ that proves polynomial satisfiability in the $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$-hybrid model with soundness error $(d+t)/p^r$ and the information-theoretic security.*

Proof. We first consider the case of a malicious prover (i.e., soundness and knowledge extraction) and then consider the case of a malicious verifier (i.e., zero knowledge). In each case, we construct a PPT simulator

$\mathcal{S}$, which given access to $\mathcal{F}_{\mathsf{ZK}}$, runs the adversary $\mathcal{A}$ as a subroutine while emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ for $\mathcal{A}$. We always implicitly assume that $\mathcal{S}$ passes all communication between adversary $\mathcal{A}$ and environment $\mathcal{Z}$.

Malicious prover. $\mathcal{S}$ emulates functionality $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ and interacts with adversary $\mathcal{A}$ as follows:

1. $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ for $\mathcal{A}$ by choosing uniform $\Delta \in \mathbb{F}_{p^r}$, and recording all the values $\{s_i\}_{i\in[n]}$ and their corresponding MAC tags that are received by $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ from adversary $\mathcal{A}$. These values define the corresponding keys in the natural way. When emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$, $\mathcal{S}$ also receives $\{A^*_h\}_{h\in[0,d-1]}$ from $\mathcal{A}$ and defines $B^* = \sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h$.
2. When $\mathcal{A}$ sends $\{\delta_i\}_{i\in[n]}$ in step 4, $\mathcal{S}$ extracts the witness as $w_i := \delta_i + s_i$ for $i \in [n]$.
3. $\mathcal{S}$ executes the remaining part of protocol $\Pi^{p,r}_{\mathsf{polyZK}}$ as an honest verifier, using Δ and the keys defined in the first step. If the honest verifier outputs false, then $\mathcal{S}$ sends $\boldsymbol{w} = \perp$ and $\mathcal{C}$ to $\mathcal{F}_{\mathsf{ZK}}$ and aborts. If the honest verifier outputs true, $\mathcal{S}$ sends $\boldsymbol{w}$ and $\mathcal{C}$ to $\mathcal{F}_{\mathsf{ZK}}$ where $\boldsymbol{w} = (w_1, \ldots, w_n)$ is extracted by $\mathcal{S}$ as above.

It is easy to see that the view of the adversary simulated by $\mathcal{S}$ has the identical distribution as its view in the real-world execution. Whenever the honest verifier in the real-world execution outputs false, the honest verifier in the ideal-world execution outputs false as well (since $\mathcal{S}$ sends $\perp$ to $\mathcal{F}_{\mathsf{ZK}}$ in this case). Therefore, we only need to bound the probability that the verifier in the real-world execution outputs true but the witness $\boldsymbol{w}$ sent by $\mathcal{S}$ to $\mathcal{F}_{\mathsf{ZK}}$ satisfies that $f_i(\boldsymbol{w}) \neq 0$ for some $i \in [t]$. Below, we show that this happens with probability at most $(d+t)/p^r$.

Let $f_i(\boldsymbol{w}) = f_i(w_1, \ldots, w_n) = y_i$ with some $y_i \in \mathbb{F}_p$ for each $i \in [t]$, where $\boldsymbol{w} = (w_1, \ldots, w_n)$ is a vector extracted by $\mathcal{S}$. According to the definition of B_i for $i \in [t]$, we have the following:

$$\begin{aligned} B_i &= \sum_{h\in[0,d]} f_{i,h}(k_1, \ldots, k_n) \cdot \Delta^{d-h} \\ &= \sum_{h\in[0,d]} f_{i,h}(m_1 + w_1 \cdot \Delta, \ldots, m_n + w_n \cdot \Delta) \cdot \Delta^{d-h} \\ &= \sum_{h\in[0,d-1]} A_{i,h} \cdot \Delta^h + y_i \cdot \Delta^d. \end{aligned}$$

In step 6, $\mathcal{S}$ receives $U'_h = U_h + E_h$ for $h \in [0, d-1]$ from adversary $\mathcal{A}$, where U_h is computed with $\boldsymbol{w}$ and the corresponding MACs following the protocol specification, and $E_h \in \mathbb{F}_{p^r}$ is an adversarially chosen error. Together with $B^* = \sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h$, we obtain that the following equation holds:

$$\begin{aligned} W &= \sum_{i\in[t]} B_i \cdot \chi^i + B^* \\ &= \sum_{i\in[t]} \Big(\sum_{h\in[0,d-1]} A_{i,h} \cdot \Delta^h + y_i \cdot \Delta^d \Big) \cdot \chi^i + \sum_{h\in[0,d-1]} A^*_h \cdot \Delta^h \\ &= \Big(\sum_{i\in[t]} y_i \cdot \chi^i \Big) \cdot \Delta^d + \sum_{h\in[0,d-1]} \Big(\sum_{i\in[t]} A_{i,h} \cdot \chi^i + A^*_h \Big) \cdot \Delta^h \\ &= \Big(\sum_{i\in[t]} y_i \cdot \chi^i \Big) \cdot \Delta^d + \sum_{h\in[0,d-1]} U'_h \cdot \Delta^h - \sum_{h\in[0,d-1]} E_h \cdot \Delta^h. \end{aligned}$$

If the honest verifier outputs true, then we have $W = \sum_{h\in[0,d-1]} U'_h \cdot \Delta^h$. Therefore, we have the following:

$$\Big(\sum_{i\in[t]} y_i \cdot \chi^i \Big) \cdot \Delta^d - E_{d-1} \cdot \Delta^{d-1} - \cdots - E_1 \cdot \Delta - E_0 = 0.$$

If $\sum_{i\in[t]} y_i \cdot \chi^i \neq 0$, the probability that the above equation holds is at most d/p^r, as $\Delta \in \mathbb{F}_{p^r}$ is uniformly random and kept secret from the adversary's view. In the following, we assume that $\sum_{i\in[t]} y_i \cdot \chi^i = 0$. If there exists some $i \in [t]$ such that $y_i \neq 0$, then that probability that $\sum_{i\in[t]} y_i \cdot \chi^i = 0$ is at most t/p^r, since $\chi \in \mathbb{F}_{p^r}$ is sampled uniformly at random after y_i for all $i \in [t]$ have been defined. Overall, the probability that the honest verifier outputs true but $f_i(\boldsymbol{w}) \neq 0$ for some $i \in [t]$ is bounded by $(d+t)/p^r$. In conclusion, any unbounded environment $\mathcal{Z}$ cannot distinguish between the real-world execution and ideal-world execution, except with probability $(d+t)/p^r$.

Malicious verifier. If $\mathcal{S}$ receives false from $\mathcal{F}_{\mathsf{ZK}}$, then it simply aborts. Otherwise, $\mathcal{S}$ interacts with adversary $\mathcal{A}$ as follows:

1. In the preprocessing phase, $\mathcal{S}$ emulates $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$ by recording the global key Δ and the keys for all the authenticated values, which are received from adversary $\mathcal{A}$. Additionally, $\mathcal{S}$ also receives $B^* \in \mathbb{F}_{p^r}$ from $\mathcal{A}$ by emulating $\mathcal{F}^{p,r}_{\mathsf{ext\text{-}sVOLE}}$.
2. $\mathcal{S}$ executes the step 4 of protocol $\Pi^{p,r}_{\mathsf{polyZK}}$ by sending uniform $\delta_i \in \mathbb{F}_p$ for $i \in [n]$ to adversary $\mathcal{A}$.
3. For steps 5–6 of the ZK protocol, $\mathcal{S}$ computes W by using Δ, the keys and B^* received from $\mathcal{A}$ following the protocol description, and then samples $U_1, \ldots, U_{d-1} \leftarrow \mathbb{F}_{p^r}$ and computing $U_0 := W - \sum_{h\in[d-1]} U_h \cdot \Delta^h$. Then, $\mathcal{S}$ sends $U_0, \ldots, U_{d-1}$ to $\mathcal{A}$.

Note that $\{s_i\}$ and $\{A_h^*\}_{h\in[d-1]}$ are sampled uniformly at random and kept secret from the view of adversary $\mathcal{A}$. Therefore, we easily obtain that the view of $\mathcal{A}$ simulated by $\mathcal{S}$ is distributed identically to its view in the real-world execution, which completes the proof. $\square$

For the polynomial-based ZK protocol, we can also use the Fiat-Shamir heuristic to make the online phase *non-interactive* at the cost of that the security is guaranteed in the computational sense.

5.2 Optimizing Practical Applications

In the following applications, for the sake of simplicity, we always assume that $p^r \approx 2^\kappa$ as the Fiat-Shamir heuristic is assumed to be implicitly used in the applications. For the interactive case, we can also extend the applications to smaller extension fields, as long as the soundness error is assured negligible in ρ. In this section, the communication cost is computed in the sVOLE-hybrid model.[2]

Optimizing matrix multiplication. The prover wants to prove that $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$, where $\mathbf{A}, \mathbf{B} \in \mathbb{F}_p^{n\times n}$ are two secret matrices and $\mathbf{C} \in \mathbb{F}_p^{n\times n}$ is a public matrix known by the verifier. Using the circuit-based ZK protocol shown in Figure 4, this will need communication of $(2n^2 + n^3)\log p + 2\kappa$ bits.

Using the polynomial-based ZK protocol described in Figure 5, we can directly obtain a ZK protocol for inner product of two n-length vectors with communication of $2n \log p + 2\kappa$ bits, by defining a polynomial $f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i\in[n]} x_i \cdot y_i$ for two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_p^n$. The communication complexity remains unchanged, even if the inner product of t vector pairs needs to be proved. This immediately gives us a ZK protocol for proving matrix multiplication with communication of $2n^2 \log p + 2\kappa$ bits, since a matrix multiplication can be written as the inner-product of n^2 vector pairs, where the communication of $2n^2 \log p$ bits is used to commit the entries in matrices $\mathbf{A}$ and $\mathbf{B}$ using sVOLE.

Proving solutions to lattice problems. Here, we assume the prover has a binary vector $\boldsymbol{s} \in \{0,1\}^m$ and intends to prove that $\mathbf{A} \cdot \boldsymbol{s} = \boldsymbol{t}$, with public matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and vector $\boldsymbol{t} \in \mathbb{Z}_q^n$. Here we assume that q is a prime. The SIS problem has been considered in prior work such as [BN20, WYKW20]. Our ZK

[2] We note that the communication for generating sVOLE correlations is *sublinear* to the number of resulting sVOLE correlations, using the recent LPN-based protocols [SGRR19, BCG+19a, YWL+20, WYKW20].

Threads	Boolean Circuits					Arithmetic Circuits				
	10 Mbps	20 Mbps	30 Mbps	50 Mbps	Local-host	100 Mbps	500 Mbps	1 Gbps	2 Gbps	Local-host
1	4.4 M	6.2 M	7.0 M	7.5 M	7.6 M	1.2 M	3.4 M	4.2 M	4.8 M	4.8 M
2	5.3 M	8.1 M	9.9 M	11.8 M	11.8 M	1.3 M	4.4 M	6.1 M	7.0 M	7.1 M
3	5.7 M	9.1 M	11.4 M	13.9 M	14.3 M	1.4 M	4.9 M	7.2 M	8.4 M	8.4 M
4	5.8 M	9.9 M	12.2 M	14.9 M	15.8 M	1.4 M	5.0 M	7.5 M	8.9 M	8.9 M

Table 2: **Benchmark the performance of our circuit-based ZK protocol.** The benchmark results are the number of AND/MULT gates per second that can be proven using our protocol, where "M" means "million". Benchmark was obtained with different network settings and number of threads.

protocol for polynomial sets can be used to prove the statements more efficiently. To commit to all input bits $s_1, \dots, s_m$, the ZK protocol takes $m \log q$ bits. Then we need to show that 1) the prover indeed commits bits and that 2) the above linear equation holds. All of the above can be modeled as a set of $m + n$ polynomials with degree at most 2. In particular, we need to show that $s_i \cdot (s_i - 1) = s_i^2 - s_i = 0$ for all $i \in [m]$ and that $\sum_{j \in [m]} a_{i,j} \cdot s_j - t_i = 0$ for $i \in [n]$ where $a_{i,j}$ is the entry in the i-th row and j-th column of matrix $\mathbf{A}$. Since the polynomial degree is at most 2, the communication cost would be 2 elements over the extension field, each of size roughly κ bits. Therefore, the *total* communication cost is $m \log q + 2\kappa$ bits.

If the secret vector $\boldsymbol{s}$ is in $[-B, B]^m$ (with a small integer B) instead of a binary vector, which has also been addressed by prior work [BLS19, BCOS20, ENS20, WYKW20], we would need a degree-$(2B + 1)$ polynomial to prove that $f(s_i) = 0$ for all $i \in [m]$ where $f(x) = \Pi_{j \in [-B,B]}(x - j)$, with the total communication cost of $m \log q + (2B + 1)\kappa$ bits.

In Section 6.2, we evaluate the concrete performance to demonstrate that our polynomial-based ZK protocol significantly outperforms prior work for proving knowledge of solutions to SIS problems.

Optimizing integer operations over a ring. Arithmetic operations over a field may often be sufficient for some applications. However, for applications where matching cleartext computation is crucial, the statement to be proven may require native computation over a ring $\mathbb{Z}_{2^n}$ such as $\mathbb{Z}_{2^{32}}$. In this case, one may naturally think about ring operations. Here we explore an alternative approach.

Our idea is to view integer multiplication over $\mathbb{Z}_{2^n}$ as a set of n polynomials that take $2n$ variables as input. In this case, the maximum degree for these polynomials is $8n^2$, since the Boolean circuit for integer multiplication has a depth $2 \log n + 3$ [BHWK16]. The communication cost for proving a set of integer multiplications would become linear to n, when the number of integer multiplications to be proven is large. In particular, if there are t integer multiplications to be proven with $t \approx 8n\kappa$, the amortized communication cost for each multiplication will be $3n + \frac{8n^2\kappa}{t} \approx 4n$ bits.

Optimizing for circuits with weak uniformity. Inspired by the above concrete examples, we summarize a blueprint to optimize circuits with some level of weak uniformity (i.e., the polynomial representations of sub-circuits are all bounded by some degree d).

Assume that the circuit to be proven is C, which contains t multiplication gates. We let $C_1, \dots, C_k$ be k non-overlapping sub-circuits of C, such that for sub-circuit C_i, it has t_i multiplication gates without counting the multiplication gates that include the output wires of C_i. Each sub-circuit can be represented as a set of polynomials with the degree at most d. In a nutshell, our ZK protocol can be constructed as follows:

1. Use sVOLE to commit to all the wire values in $C \setminus \{C_1, \dots, C_k\}$, including the input wires go into the sub-circuits $C_1, \dots, C_k$ and the output wires go out of these sub-circuits.

   This step takes $t - \sum_{i=1}^{k} t_i$ elements over $\mathbb{F}_p$ for communication.

2. Prove that all multiplication gates in $C \setminus \{C_1, \dots, C_k\}$ are computed correctly using our ZK protocol for circuit satisfiability shown in Figure 4.

This step takes 2κ bits of communication.

3. For each sub-circuit C_i, represent it as a set of polynomials, one for each output of C_i. Prove that all the polynomials with respect to all sub-circuits are computed correctly using our ZK protocol for polynomial satisfiability shown in Figure 5.

   This step takes $d\kappa$ bits of communication, because all input and output wire values have already been committed with sVOLE.

In summary, the communication of the above protocol is essentially $(t - \sum_{i \in [k]} t_i) \log p + (d + 2)\kappa$ bits. Now the task is really about how to "dig" as many "holes" as possible from C, while keeping all holes relatively simple. In practice, this is fairly common, as the real-life computations are written in succinct libraries, which means the same subroutine is often called for many times. We leave it as a future work to fully explore its potential and build an automated optimizer to maximize the practical efficiency.

6 Implementation and Benchmarking

We implemented our ZK protocols and report their performance. Unless otherwise specified, our evaluation results are reported over two Amazon EC2 machines of type `m5.2xlarge` with throttled network bandwidth (with latency about 0.1 ms) and one thread. Each machine has 8 virtual CPUs, which means 4 CPU cores. We instantiate the COT protocol (i.e., sVOLE with $p = 2$ and $r = \kappa$) and the VOLE protocol over a 61-bit field by using the recent protocols [YWL+20, WYKW20], and use SHA-256 as the cryptographic hash function modeled as a random oracle. We take advantage of hardware AES-NI and binary-field multiplication when applicable. All our implementations achieve computational security parameter $\kappa = 128$ and statistical security parameter $\rho \approx 100$ for Boolean circuits, and $\kappa = 128$ and $\rho \geq 40$ for arithmetic circuits over a 61-bit field where Mersenne prime $p = 2^{61} - 1$ is used as in prior work.

6.1 Benchmarking Our Circuit-based ZK Proof

We benchmarked the performance of our ZK protocol by proving circuits with 3×10^8 AND/MULT gates. Similar to prior work [WYKW20, BMRS20], we observe that the performance does not depend on the shape of the circuit and is linear to the circuit size; and thus we focus on the speed in terms of "million gates per second". In Table 2, we benchmarked the performance of our circuit-based ZK protocol under different network settings and number of threads. The performance of our protocol ranges from 4.4 million to 15.8 million AND gates per second (or from 1.2 million to 8.9 million multiplication gates per second), depending on the network setting and number of threads. When we increase the number of threads and/or the network bandwidth, we could see an increase in the performance. The computation becomes the efficiency bottleneck of our protocol for Boolean circuits (resp., arithmetic circuits) when the network bandwidth is increased to 50 Mbps (resp., 2 Gbps), and thus the performance is not improved much beyond that.

Comparison with prior work. We compared the performance of our ZK protocol QuickSilver and prior related work in Table 1. Since Mac′n′Cheese [BMRS20] only reported the performance of their protocol with one thread and local-host, we compare the performance of all protocols using this setup. In the Boolean setting, we observe $6\times$ improvement in computation and $7\times$ improvement in communication compared to the state-of-the-art protocol Wolverine [WYKW20]. For arithmetic circuits, our protocol improves by at least $7\times$ in computation and $3\times$–$4\times$ in communication compared to Wolverine and Mac′n′Cheese. Note that Wolverine studied the performance of their ZK protocol when used for DECO [ZMM+20] and Blind CA [WAP+19], as well as other applications like Merkle trees, and proving bugs in a set of code snippets [HK20]. Our performance improvement described as above directly translates to the improvements for all of these applications.

Instance Information			Boolean Circuits		Arithmetic Circuits	
Type	Price cents/hour	CPU	Speed gates/sec	Cost gates/cent	Speed gates/sec	Cost gates/cent
`c6g.medium`	1.9	ARM	5.3 M	10.0 B	2.2 M	4.1 B
`c5.large`	4.7	Intel	5.9 M	4.5 B	2.9 M	2.2 B
`c5a.large`	4.2	AMD	7.3 M	6.3 B	3.0 M	2.6 B

Table 3: **Performance of stress-testing our ZK protocol on different Amazon EC2 instances.** All instances have 2 vCPUs and 1 GB memory.

	Binary field $\mathbb{F}_2$			Large field $\mathbb{F}_{2^{61}-1}$		
Length of vectors	10^6	10^7	10^8	10^6	10^7	10^8
Process witness (s)	0.24	2.4	23.8	0.39	3.9	39.2
Prove inner product (ms)	36.6	69.3	423.5	42.8	100.3	703.8

Table 4: **Performance of our ZK protocol for inner product.** We report separately the cost to process the witness and the cost to prove the inner product after the witness was processed.

Protocol	Execution Time	Communication
Virgo [ZXZS20]	357 s	221 KB
Wolverine [WYKW20]	1627 s	34 GB
Mac′n′Cheese [BMRS20]	2684 s	25.8 GB
QuickSilver (Circuit)	316 s	8.6 GB
QuickSilver (Polynomial)	10 s	25.2 MB

Table 5: **Performance of proving matrix multiplication using various protocols.** Every protocol is used to prove knowledge of two 1024×1024 matrices over a 61-bit field, whose product is a public matrix. The execution time for Wolverine and Mac′n′Cheese is based on local-host, while our protocols and Virgo are based on a 500 Mbps network. While Virgo uses about 148 GB memory, other protocols use the memory of about 1 GB.

Stress-testing of our ZK protocol. We stress-test our circuit-based ZK protocol on the cheapest instance of Amazon EC2 that only costs 2 to 5 cents per hour, and summarize the experimental results in Table 3.[3] For all protocol executions, we use only a single thread. The Boolean circuits (resp., arithmetic circuits) are tested under the network bandwidth of 20 Mbps (resp., 500 Mbps). Although the computational power and memory are limited, our protocol still achieves high throughput. The speed for computing Boolean circuits ranges from 5.3 million to 7.3 million gates per second and the speed for arithmetic circuits ranges from 2.2 million to 3 million gates per second. Taking the low cost into consideration, our ZK protocol is very *affordable*. The cost to prove Boolean circuits is about 10 billion gates per cent using the ARM CPU; the cost of proving arithmetic circuits is roughly 2.2–4.1 billion gates per cent.

[3]Price is based on AWS defined-duration spot instances. There are cheaper t3.medium, t3a.medium burstable instances, but the cost is higher than the instances in Table 3 unless the average CPU usage is kept below 20%.

Protocol	ENS [ENS20]	Wolverine [WYKW20]	QuickSilver
Communication	53 KB	32.8 KB	4.1 KB
Execution time	–	220 ms	2 ms

Table 6: **Performance comparison of our ZK protocol** QuickSilver **vs. prior work for proving knowledge of an SIS solution.** The solution is assumed to be a ternary-vector and $n = 2048, m = 1024, \log q = 32$.

6.2 Benchmarking Our Polynomial-based ZK Proof

While our ZK protocol for polynomial satisfiability is generic, here we focus on some useful applications that use low-degree polynomials to demonstrate how powerful it can be. We leave exploration of compiler-based optimization and more complicated examples as the future work. In all of the experiments below, we use the network bandwidth of 20 Mbps for a binary field and 500 Mbps for a 61-bit field, and always use a single thread.

Inner product. In this benchmark, the witness consists of two vectors of n field elements (namely $\boldsymbol{x}$ and $\boldsymbol{y}$), and the prover wants to prove that the inner product of two vectors $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i \in [n]} x_i \cdot y_i$ equals to some public value. We report the cost of processing the witness and the cost of proving the inner product separately in Table 4. We found that processing the witness could be free in a larger computation. This is because when using inner product as a sub-circuit in a larger circuit, the input witness of this subcircuit is the output of some prior computation and thus need not be processed again. In this case, the cost of the ZK proof for inner product is simply the second line. We can see that even for proving inner product of two vectors of length 10^8, the cost is very small.

Matrix multiplication. We report the performance of our ZK protocol for proving matrix multiplication, and compare it with prior work in Table 5. We observe that our polynomial-based ZK protocol is 31$\times$ faster than our own circuit-based protocol, which is already faster than prior protocols. It also uses 340$\times$ less communication than our circuit-based protocol. Our proof size is still significantly larger than Virgo, but our ZK protocols (QuickSilver) benefit in other aspects including execution time and memory usage. We attempted to test on a recent zk-SNARK implementation Spartan [Set20], but the program used more than 600 GB memory before killed.

Proving knowledge of solutions to lattice-based problems. Here we focus on proving knowledge of a solution to a *short integer solution* (SIS) problem. We assume that the prover knows a vector $\boldsymbol{s} \in [-B, B]^m$, such that $\mathbf{A} \cdot \boldsymbol{s} = \boldsymbol{t}$, where both parties know the public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vector $\boldsymbol{t} \in \mathbb{Z}_q^n$ (here we assume that q is a prime). Checking the matrix multiplication is easy since the matrix $\mathbf{A}$ is public and thus the main work is to check that all coordinates in $\boldsymbol{s}$ are bounded. This can be done by proving that $\Pi_{j \in [-B,B]}(s_i - j) = 0$ for all $i \in [m]$. In typical SIS problems, e.g., the one studied in the recent work [ENS20], B is set to 1, resulting in a degree-3 polynomial. The checking procedure of our ZK protocol is essentially *free* compared to the cost of obtaining the committed input to the polynomial. We show the performance comparison in Table 6, where the execution time for [ENS20] is not available from their paper. Due to our improved protocol for low-degree polynomials, our protocol is significantly faster and uses less communication than prior work.

Acknowledgements

The authors would like to thank Steve Lu for the helpful discussion on the cost of Amazon EC2. This material is based upon work supported in part by DARPA under Contract No. HR001120C0087. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Work of Pratik Sarkar is supported by the DARPA SIEVE project and NSF awards 1931714, 1414119. Work of Kang Yang is supported in part by the National Key Research and Development Program of China (Grant Nos. 2018YFB0804105, 2020YFA0309703), and in part by the National Natural Science Foundation of China (Grant Nos. 61932019, 61802021).

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BCG$^+$19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 11–15, 2019.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[BCOS20] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum SNARKs for RSIS and RLWE and their applications to privacy. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 247–267, Paris, France, April 15–17 2020. Springer, Heidelberg, Germany.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Recursive zk-snarks from any additive polynomial commitment scheme. Cryptology ePrint Archive, Report 2020/1536, 2020. https://eprint.iacr.org/2020/1536.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[BFH$^+$20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 2025–2038, Virtual Event, USA, November 9–13, 2020. ACM Press.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[BHWK16] Niklas Büscher, Andreas Holzer, Alina Weber, and Stefan Katzenbeisser. Compiling low depth circuits for practical secure computation. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 80–98, Heraklion, Greece, September 26–30, 2016. Springer, Heidelberg, Germany.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BMR16] Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for Boolean and arithmetic circuits. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 565–577, Vienna, Austria, October 24–28, 2016. ACM Press.

[BMRS20] Carsten Baum, Alex J. Malozemoff, Marc Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for arithmetic circuits with nested disjunctions. Cryptology ePrint Archive, Report 2020/1410, 2020. https://eprint.iacr.org/2020/1410.

[BN20] Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CDG$^+$17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[DIO20] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. Cryptology ePrint Archive, Report 2020/1446, 2020. `https://eprint.iacr.org/2020/1446`.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT 2020, Part II*, LNCS, pages 259–288. Springer, Heidelberg, Germany, December 2020.

[FNO15] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 191–219, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083, Austin, TX, USA, August 10–12, 2016. USENIX Association.

[HK20] David Heath and Vladimir Kolesnikov. Stacked garbling for disjunctive zero-knowledge proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 569–598, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 598–628, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 955–966, Berlin, Germany, November 4–8, 2013. ACM Press.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[KMR14] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[KP17] Yashvanth Kondi and Arpita Patra. Privacy-free garbled circuits for formulas: Size zero and information-theoretic. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 188–222, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

[MPas15] Tal Malkin, Valerio Pastro, and abhi shelat. The whole is greater than the sum of its parts: Linear garbling and applications. https://simons.berkeley.edu/talks/tal-malkin-2015-06-10, 2015.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139, 1999.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[SGRR19] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, November 11–15, 2019.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020. https://eprint.iacr.org/2020/1275.

[WAP+19] Liang Wang, Gilad Asharov, Rafael Pass, Thomas Ristenpart, and abhi shelat. Blind certificate authorities. In *2019 IEEE Symposium on Security and Privacy*, pages 1015–1032, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[WYKW20] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. Cryptology ePrint Archive, Report 2020/925, 2020. https://eprint.iacr.org/2020/925.

[YWL+20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 1607–1626, Virtual Event, USA, November 9–13, 2020. ACM Press.

[ZMM+20] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating web data using decentralized oracles for TLS. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 1919–1938, Virtual Event, USA, November 9–13, 2020. ACM Press.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press.

A More Details on Preliminaries

Universal composability. We use the universal composability (UC) framework [Can01] to prove security in the presence of a malicious, static adversary. We say that a protocol Π *UC-realizes* an ideal functionality $\mathcal{F}$ if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for any PPT environment $\mathcal{Z}$ with arbitrary auxiliary input z, the output distribution of $\mathcal{Z}$ in the *real-world* execution where the parties interact with $\mathcal{A}$ and execute Π is indistinguishable from the output distribution of $\mathcal{Z}$ in the *ideal-world* execution where the parties interact with $\mathcal{S}$ and $\mathcal{F}$.

We prove the security of our protocols in the $\mathcal{G}$-hybrid model, where the parties execute a protocol with real messages and also have access to an ideal functionality $\mathcal{G}$. We say that protocol Π UC-realizes functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid model with statistical error $1/2^\rho$ and the information-theoretic security (resp., the computational security), if the distinguishing probability of environment $\mathcal{Z}$ between the real-world execution and ideal-world execution is bounded by $1/2^\rho$ (resp., $1/2^\rho + \mathsf{negl}(\kappa)$).

Global-key query for sVOLE. In the malicious setting, if the sVOLE protocol [WYKW20] over any field is used to securely realize $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$ shown in Figure 6, then we can guarantee the UC security when the following single *global-key query* is added into $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$.

Functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$

Initialize: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$, sample $\Delta \leftarrow \mathbb{F}_{p^r}$ if $\mathcal{V}$ is honest, and receive $\Delta \in \mathbb{F}_{p^r}$ from the adversary otherwise. Store Δ and send it to $\mathcal{V}$, and ignore all subsequent (init) commands.

Extend: This procedure can be run multiple times. Upon receiving (extend, ℓ) from $\mathcal{P}$ and $\mathcal{V}$, execute the following:

1. If $\mathcal{V}$ is honest, sample $\boldsymbol{k} \leftarrow \mathbb{F}_{p^r}^{\ell}$. Otherwise, receive $\boldsymbol{k} \in \mathbb{F}_{p^r}^{\ell}$ from the adversary.
2. If $\mathcal{P}$ is honest, sample $\boldsymbol{x} \leftarrow \mathbb{F}_p^{\ell}$ and compute $\boldsymbol{m} := \boldsymbol{k} - \Delta \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^{\ell}$. Otherwise, receive $\boldsymbol{x} \in \mathbb{F}_p^{\ell}$ and $\boldsymbol{m} \in \mathbb{F}_{p^r}^{\ell}$ from the adversary, and then recompute $\boldsymbol{k} := \boldsymbol{m} + \Delta \cdot \boldsymbol{x} \in \mathbb{F}_{p^r}^{\ell}$.
3. Send $(\boldsymbol{x}, \boldsymbol{m})$ to $\mathcal{P}$ and $\boldsymbol{k}$ to $\mathcal{V}$.

Figure 6: Functionality for subfield VOLE.

- If $\mathcal{P}$ is corrupted, receive $(\mathsf{guess}, \Delta')$ from the adversary. If $\Delta' = \Delta$, then send $\mathsf{success}$ to $\mathcal{P}$ and ignore any subsequent global-key query. Otherwise, send abort to both parties and abort.

This global-key query allows the adversary to guess Δ only once, which only increases the successful probability of the adversary to guess Δ by at most $1/p^r$. If the more efficient sVOLE protocol [YWL+20] for only the case of $p = 2$ and $r = \kappa$ is used, then we need to strengthen the global-key query from the single query to any polynomial number of queries, where this functionality will not abort for an incorrect guess. This type of global-key queries has been used in MPC protocols such as [NNOB12, HSS17]. In this case, the successful probability of the adversary to guess Δ is increased to at most $q/2^{\kappa}$, where q is the number of global-key queries.

For the sake of simplicity, we omit the global-key query, even if we use the protocols [WYKW20, YWL+20] to UC-realize $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$.

B Proof of Theorem 1

Proof. We easily construct a PPT simulator $\mathcal{S}$ given access to $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$, and running the (unbounded) adversary $\mathcal{A}$ as a subroutine while emulating $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$ for $\mathcal{A}$. In particular, there is no communication between $\mathcal{P}$ and $\mathcal{V}$. Thus, $\mathcal{S}$ can emulate $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$ and record all the values sent by $\mathcal{A}$ to $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$, and then compute the output value for the corrupted party following the protocol specification, and send it to functionality $\mathcal{F}_{\mathsf{ext\text{-}sVOLE}}^{p,r}$. It is trivial to see that the simulation is perfect.

Below, we show that the output of the honest party is perfectly indistinguishable between the real-world execution and ideal-world execution. We first prove if both parties compute their output *locally* following the protocol specification, then their outputs satisfy the correct VOPE correlation. Specifically, from $k_h = m_h + u_h \cdot \Delta$ for $h \in [r]$, we easily obtain that for each $i \in [d]$,

$$\begin{aligned} K_i &= \sum_{h \in [r]} k_h \cdot X^{h-1} = \sum_{h \in [r]} (m_h + u_h \cdot \Delta) \cdot X^{h-1} \\ &= \sum_{h \in [r]} m_h \cdot X^{h-1} + \Big(\sum_{h \in [r]} u_h \cdot X^{h-1} \Big) \cdot \Delta \\ &= M_i + U_i \cdot \Delta. \end{aligned}$$

It is easy to see that $B_1 = K_1 = M_1 + U_1 \cdot \Delta = g_1(\Delta)$. Below, we prove by induction. In the i-th iteration with $i \in [d-1]$, assuming that $B_i = g_i(\Delta)$, we have the following holds:

$$\begin{aligned} B_{i+1} &= B_i \cdot K_{i+1} + K_{d+i} \\ &= g_i(\Delta) \cdot (M_{i+1} + U_{i+1} \cdot \Delta) + (M_{d+i} + U_{d+i} \cdot \Delta) \\ &= g_{i+1}(\Delta). \end{aligned}$$

Therefore, we obtain that $B = B_d = g_d(\Delta) = \sum_{i \in [0,d]} A_i \cdot \Delta^i$.

If $\mathcal{V}$ is honest, then its output is always defined by the output of malicious party $\mathcal{P}$ and Δ (i.e., $B = \sum_{i \in [0,d]} A_i \cdot \Delta^i$) in both worlds. In the following, we consider the case that $\mathcal{P}$ is honest but $\mathcal{V}$ is malicious. The output values $A_0, \ldots, A_d$ for $\mathcal{P}$ are uniformly random such that $B = \sum_{i \in [0,d]} A_i \cdot \Delta^i$ in the ideal-world execution, where B is the output of malicious party $\mathcal{V}$. In the real protocol execution, A_i for each $i \in [0,d]$ is computed as the coefficient of item x^i for polynomial $g_d(x)$. According to the definition of $\mathcal{F}_{\mathsf{sVOLE}}^{p,r}$, u_h for $h \in [r]$ is uniform in $\mathbb{F}_p$. Therefore, for $i \in [2d-1]$, we have that $U_i := \sum_{h \in [r]} u_h \cdot X^{h-1}$ is uniformly random in $\mathbb{F}_{p^r}$. For $i \in [d]$, we prove by induction that each coefficient of $g_i(x)$ except for constant term is uniformly distributed in $\mathbb{F}_{p^r}$, except with probability at most $1/p^r$. This holds for $g_1(x) = M_1 + U_1 \cdot x$ with probability 1. In the i-th iteration with $i \in [d-1]$, we have that the coefficients $A_{i,1}, \ldots, A_{i,i}$ of degree-i polynomial $g_i(x)$ are uniform by the induction assumption. From the definition of $g_{i+1}(x)$, we obtain the following holds:

$$\begin{aligned} g_{i+1}(x) &= g_i(x) \cdot (M_{i+1} + U_{i+1} \cdot x) + (M_{d+i} + U_{d+i} \cdot x) \\ &= \left(\sum_{h=0}^{i} A_{i,h} \cdot x^h \right) \cdot (M_{i+1} + U_{i+1} \cdot x) + (M_{d+i} + U_{d+i} \cdot x) \\ &= (A_{i,0} \cdot M_{i+1} + M_{d+i}) + (A_{i,1} \cdot M_{i+1} + A_{i,0} \cdot U_{i+1} + U_{d+i}) \cdot x \\ &\quad \sum_{h=2}^{i} (A_{i,h} \cdot M_{i+1} + A_{i,h-1} \cdot U_{i+1}) \cdot x^h + A_{i,i} \cdot U_{i+1} \cdot x^{i+1}. \end{aligned}$$

From the uniformity of U_{d+i}, we directly obtain that the 1-degree term of $g_{i+1}(x)$ is uniform. If $U_{i+1} \neq 0$ except with probability $1/p^r$, then the h-degree term of $g_{i+1}(x)$ for $h \in [2, i+1]$ is uniform from the uniformity of $A_{i,h-1}$. Overall, except with probability $1/p^r$, each coefficient of $g_{i+1}(x)$ except for constant term is uniformly random. Therefore, the coefficients $A_1, \ldots, A_d$ of polynomial $g_d(x)$ are uniform over $\mathbb{F}_{p^r}$, except with probability at most $(d-1)/p^r$. Together with $B = g_d(\Delta)$, we have that $A_0 = B - \sum_{i \in [d]} A_i \cdot \Delta^i$, which completes the proof. □