Sign Modules in Secure Arithmetic Circuits

Ching-Hua Yu

Chinghua.yu@gmail.com

Abstract

In this paper, we study the complexity of secure multiparty computation using only the secure arithmetic black-box of a finite field, counting the cost by the number of secure multiplications. We observe that a specific type of quadratic patterns exists in all finite fields, and the existence of these patterns can be utilized to improve the efficiency of secure computation to a remarkable extent.

We define *sign modules* as partial functions that simulate integer signs in an effective range using a polynomial number of arithmetic operations on a finite field. Let ℓ denote the bit-length of a finite field size. We show the existence of $\lfloor \ell/5 \rfloor$-"effective" sign modules in any finite field that has a sufficiently large characteristic. When ℓ is decided first, we further show the existence of prime fields that contain an $\Omega(\ell \log \ell)$-"effective" sign module and we propose an efficient probabilistic algorithm that finds concrete instances of sign modules.

Let $\mathcal{Z}_p$ be any odd prime field. Then, based on the existence of effective sign modules and providing a binary-expressed random number in $\mathcal{Z}_p$, prepared in the offline phase, we show that the computation of bitwise less-than can be improved from the best known result of $O(\ell)$ to $O(\sqrt{\frac{\ell}{\log \ell}})$ (with $O(1)$ rounds) in the online phase using only the $\mathcal{Z}_p$-arithmetic black-box. Accompanied by several related improvements, secure computation involving integer comparisons and modulo reductions can be improved from the best known result $O(\ell)$ to $O(\sqrt{\frac{\ell}{\log \ell}})$ (with $O(1)$ rounds), and a (deterministic) zero test can be improved from $O(\ell)$ to $O(1)$ in the online phase. Additionally, a tight-bound complexity of bit-decomposition is also obtained.

Keywords: Secure Multi-party Computation, Arithmetic Black-Box, Unconditionally Secure Protocol, Bit-Decomposition, Integer Comparison, Modulo Reduction, Zero Test.

Contents

1 Introduction

Secure multiparty computation (MPC) allows a number of parties to jointly evaluate a function with private inputs. A standard setting assumes that the inputs and outputs of the evaluated function are all shared secret values that use a linear secret sharing scheme, e.g., Shamir's polynomial sharing scheme, and a protocol is secure if the parties learn nothing about the real values of the inputs and the outputs.

General solutions can be based on a secure evaluation of Boolean circuits with secret binary inputs. However, for computation that involves a large number of additions and multiplications, such binary expression is inefficient.[1] Another type of solution is to choose a prime p that is greater than the predefined upper-bound of the computation, to express the inputs as elements of $\mathcal{Z}_p$, and to use $\mathcal{Z}_p$-arithmetic to simulate the task. Specifically, for arithmetic applications such as financial computation, auctions, data mining, statistical learning [8, 24, 9, 25] and the distributed generation of cryptographic keys [14], the second type of solution is usually considered more practical with respect to efficiency.

Aside from additions and multiplications, several non-arithmetic functions, involving integer comparison, zero test and modulo reduction, also play important roles in many arithmetic applications. Because these functions are much more expensive than a multiplication, efficient solutions for these functions can have strong influence on the overall efficiency. From existing results, we can use bit-decomposition, which converts a secret number into a secret binary set [13, 34, 26], as a generic solution to compute non-arithmetic functions in a bit-oriented way, or we can use more efficient solutions for specific problems such as zero test, integer comparison [27] and modulo reduction [26, 20]. Nevertheless, the known solutions to these problems require $O(\ell)$ secure multiplications, where $\ell = \log_2 p$ is the bit-length of the inputs, in both the online phase and the offline phase.

We observe that although these solutions are built in various ways, they all involve a fundamental part - using or emulating the computation of a bitwise less-than. However, because the computation is processed on $\mathcal{Z}_p$-arithmetic, a natural question is whether it is necessary to purely simulate binary computation with a bitwise approach or whether it is possible to simulate some small (but not constant-size) Boolean circuits using $O(1)$ secure multiplications.

Motivated by this possibility, this study begins with two observations. 1) The quadratic character of a finite field $\mathcal{Z}_P$ can be efficiently evaluated using $O(1)$ secure multiplications. 2) The distribution of quadratic residues/non-residues shares some similarity with a random distribution such that short regular patterns can be found in an arbitrary field. More specifically, we focus on integer signs and comparison-related problems, and define *sign modules* of $\mathcal{Z}_P$ by some specific character patterns of $\mathcal{Z}_P$. We defer a formal and generalized definition to Section 3, but intuitively, a sign module of $\mathcal{Z}_P$ serves as a partial function to simulate an integer sign in a limited range using polynomial number of $\mathcal{Z}_P$ operations. Then several questions arise: 1) To what extent can we simulate integer signs using a sign module of $\mathcal{Z}_p$? 2) How can we find a concrete instance of sign module? 3) Because of the existence of sign modules, what new results can be obtained regarding the complexity of secure computation of integer sign-related problems?

[1] Note that computation on ciphertexts, e.g., secret shares, is different from that on plaintexts. Hence the problem here is different from the computation on local machines, where we usually use binary expressions and operations.

1.1 Our Results

Let F_N be any finite field of an odd characteristic p. We show that several non-arithmetic functions can be computed much more efficient than known complexity in the F_N-arithmetic black-box ($\mathcal{F}_N$-ABB) model with unconditional security. (See [15] or Section 2.2 for a definition) The results are ascribed to some good property of F_N. The study of this paper can then be divided into two parts: 1) the good property of F_N and some more discussion regarding prime fields and 2) new constructions of secure protocols for lower complexity using this property.

In the first part, we define the *sign function* Sign_G over a finite group G that has a subnormal group G^+ such that $|G/G^+| = 2$. We show that $\mathsf{Sign}_{\mathcal{F}_N^*}$ (which is the quadratic character of $\mathcal{F}_N^*$), an instance of such a function, can be computed securely and efficiently in the $\mathcal{F}_N$-ABB model. Then, the evaluation of $\mathsf{Sign}_{\mathcal{F}_N^*}$ over a subset of $\mathcal{F}_N^*$ forms a *d-effective sign module* of $\mathcal{F}_N$ if the subset has a polynomial (especially linear in practice) mapping relationship with an ordered set $\{-d, ..., d\}$. Such an evaluation can then serve as a simulation of an integer sign in a limited range $-d$ to d. We show, through a deterministic demonstration, that $\mathcal{F}_N$ always contains a d-effective sign module with $d = \min(\lfloor \log_2 N/5 \rfloor, (p-3)/2)$, which is the good property mentioned above. Next, we discuss a slightly different question regarding the sign modules of prime fields with the goal of finding more effective sign modules. We show that for all ℓ, there always exists a prime $p \in \{2^{\ell-1}, 2^\ell\}$ such that $\mathcal{Z}_p$ contains an $\Omega(\ell \log \ell)$-effective sign module. In addition, some numerical data are also examined for practically applicable corollaries such as the existence of a $(2\ell+1)$-effective sign module in this setting when $\ell \geq 24$. Finally, we provide a probabilistic polynomial-time algorithm to sample a prime p of bit-length ℓ that is accompanied by an $\Omega(\ell)$-effective sign module of $\mathcal{Z}_p$.

In the second part, using only the arithmetic black-box construction of $\mathcal{Z}_P$ circuits and counting the cost by the number of secure multiplication gates (MULTs), we break the known linear complexity of several problems in the online phase based on the existence of effective sign modules. Our solutions rely on several improvements of sub-protocols. Nevertheless, there are three elementary constructions behind these solutions. Let $\ell = \lceil \log_2 p \rceil$ be the input length. First, the existence of $\Omega(\log p)$-effective sign modules implies that small range ($\Omega(\log p)$) integer comparison can be evaluated using $O(1)$ MULTs. Second, symmetric Boolean functions such as $\bigvee_{i=0}^{\log_2 p-1}[b_i]_p$ and $\bigwedge_{i=0}^{\log_2 p-1}[b_i]_p$ can now be implemented by $O(1)$ MULTs. Third, some $O(\ell)$-bits functions can be expressed as $O(\ell/log\ell)$-digit functions and have an $\Omega(log\ell)$ improvement factor. In the following, we enumerate our main results. The detailed cost estimation can be found in Section 6 and a comparison of works is shown in Table 1. All of the protocols run in constant rounds and have unconditional security.

- Given a binary-expressed random secret that is uniformly sampled from $\mathcal{Z}_P$ and prepared in the offline phase, we show that bitwise less-than can be evaluated using $O(\sqrt{\ell/log\ell})$ MULTs. This result is an improvement to the known complexity of $O(\ell)$ [13].

- Given a binary-expressed random secret that is uniformly sampled from $\mathcal{Z}_P$ and prepared in the offline phase, we show that a (deterministic) zero test can be evaluated using $O(1)$ MULTs in the online phase, which is an improvement to the known complexity of $O(\ell)$ [27].

- A tight bound complexity, $O(\ell)$, of bit-decomposition is also obtained in this setting (using constant-rounds, black-box construction and with unconditional security).

- Using bitwise less-than, our protocols of integer comparison and modulo reduction can be evaluated using $O(\sqrt{\ell/log\ell})$ MULTs in the online phase, which is an improvement to the known result of $O(\ell)$ [26, 20]. In addition, using our bitwise less-than solution, the computation of the offline phase is improved and then dominated by generating a few set of ℓ independent random bits, which suffers from an $\Omega(\ell)$ lower bound. (Nevertheless, some potential improvements that arise from releasing the condition of independence with a non-black-box approach are discussed in Section 8).

2 Preliminaries

2.1 Notations

Let $\mathcal{F}_N$ denote a finite field. When a secret is stored using a secret sharing scheme over $\mathcal{F}_N$ (e.g., Shamir's polynomial sharing [33]), it is denoted as $[x]_{\mathcal{F}_N}$. Let $\mathcal{Z}_p$ denote a field of congruence classes, where p is a prime with length $\ell = \lceil \log_2 p \rceil$. When a secret value x is stored in a secret sharing form over Z_p, it is denoted as $[x]_p$. We use $[x]^m$ to denote an m-dimensional secret vector $([x_{m-1}]_p, ..., [x_0]_p)$, where $x_i \in \mathcal{Z}_p, \forall i$. $[x]^m_{D(d)}$ is similar to $[x]^m$, but each $[x_i]_p$ is constrained by $x_i \in \{0, ..., d-1\}$. Specifically, when we write $[x]^m_{D(d)}$, rather than $[x]^m$, x is meaningful, and its value is $\sum_{i=0}^{m-1} x_i d^i$. We assume that the m in $[x]^m_{D(d)}$ is at most $\lceil \log_d p \rceil$. In other words, only $x \in \{0, ..., p-1\}$ is concerned. For d=2, we write $[x]^m_B$ shortly for $[x]^m_{D(2)}$. When $m = \lceil \log_d p \rceil$, $[x]^m_{D(d)}$ is abbreviated to $[x]_{D(d)}$ because the vector is a full base-d digit decomposition of x, and similarly, $[x]_B$ is abbreviated from $[x]^\ell_B$. We use the same symbol of multiplication for the inner product, $[x]_B \cdot [y]_B$. Because the operands are vectors now, it is clear to distinguish the two cases. We abuse the notation of comparison operators for $\mathcal{Z}_p$ elements. For example, $x \leq p/2$ means $x \in \{0, ..., \lfloor p/2 \rfloor\}$. Moreover, because $p - a(\mod p) \equiv -a(\mod p)$, without confusion, we also write $-a$ for short.We write $[r]_{\mathcal{F}_N} \in_R G$ when r is a random value which is uniformly and independently sampled from G.

We denote the set of quadratic residues and the set of quadratic non-residues in $\mathcal{Z}_p$ by $\mathcal{QR}_p$ and $\mathcal{NR}_p$ respectively, and $\mathcal{QR}^*_p = \{x^2 | x \in \mathcal{Z}^*_p\}$. In addition, p_k denotes the k^{th} prime. $\pi(x)$ denotes the prime counting function, which counts the number of primes that are less than or equal to x. $\pi(x; a, b) = \#\{\text{prime } p \mid p \equiv b \mod a, 0 < p \leq x\}$. $p_k\sharp$ denotes the product of the first k primes, which is known as primorial.

2.2 Model of Secure Arithmetic Black-Box

In this paper, we study the computation of several non-arithmetic functions, involving comparison and modulo reduction, using only secure arithmetic circuits. The construction of our protocols is purely based on the (secure) *arithmetic black-box* (ABB)[2]. ABB can be seen as an abstract computer. That is, assuming the underlying primitives are secure, we describe protocols by using secure arithmetic of a predefined and fixed field (or ring). Additionally, we do not describe individual actions of a party or a set of parties, but see all of the parties as an entity.

An $\mathcal{F}_N$-ABB scheme provides three operations for a number of parties: 1) securely storing an element x in $\mathcal{F}_N$ using a secret sharing form (e.g., Shamir's polynomial sharing [33], which is

[2]The term ABB is from [15], and the concept is implicit in many studies in the MPC literature.

denoted as $[x]_{\mathcal{F}_N}$. 2) fairly revealing a secret value, denoted as $c \leftarrow [x]_{\mathcal{F}_N}$, and 3)securely performing the $\mathcal{F}_N$ operations $+, \times$ for secret values. Note that, while an $\mathcal{F}_N$-ABB scheme only provides secure $\mathcal{F}_N$ operations for shared secret values, parties can perform any (polynomial-time) computation of revealed plaintexts. Moreover, we require that the ABB scheme guarantees two things for a finite field $\mathcal{F}_N$. 1) The secret sharing form is additively homomorphic such that linear combinations like $\sum_i a_i[x_i]_{\mathcal{F}_N}$ are costless in communication; and 2) The implementation of the secure multiplication, i.e., $[z]_{\mathcal{F}_N} \leftarrow [x]_{\mathcal{F}_N} \cdot [y]_{\mathcal{F}_N}$, satisfies *universal composability* (UC). Protocols remain secure in the UC framework [10] even when their multiple instances are arbitrarily composed with each other. We note that most updated ABB schemes have these two properties. A concrete ABB scheme can be implemented in several ways and is designed for different types/levels of security for different efficiency, e.g., passive or active, cryptographic or information-theoretical [7, 12], deterministic or statistical [17], honest or dishonest majority [22, 16]. The security type/level of high-level protocols would then depend on which concrete ABB scheme is used.

Remark 2.1. There is some ambiguity regarding the term (secure) arithmetic black-box in the literature. In some papers (especially of two-party computation), this term means that a secure multiplication protocol is assumed to be secure and used in a black-box way, but the processes of each party can be designed individually. In other words, such a model assumes a multiplication oracle. On the other hand, in accordance with [4, 13, 34, 26, 27], our setting is a more black-box setting, where only the whole behavior of the parties is described in protocols.

We say that a protocol is *unconditionally secure* in the ABB model if the protocol is secure given that the underlying ABB scheme is secure. Additionally, the security of the protocol should completely follow that of the ABB scheme without any extra assumption (e.g., without any additional security parameter for statistical security or an additional hardness assumption).

Definition 2.2 (Unconditional Security)**.** Let $[x]_{\mathcal{F}_N}, [y]_{\mathcal{F}_N}$ be the inputs and let $v_1, ..., v_k$ be the revealed values (through the second ABB operation) of protocol $\mathcal{P}$. Let $\mathcal{T} \subseteq 2^{[n]}$ be the collection of sets that are *not* allowed to jointly access a secret through the underlying secret sharing scheme of ABB. (For example, in a threshold-t secret sharing scheme, $\mathcal{T} = \{S \in 2^{[n]} || S| < t\}$.) If, for all $S \in \mathcal{T}$, we have $View_S([x]_{\mathcal{F}_N}|v_1, ..., v_k) = View_S([x]_{\mathcal{F}_N})$ and $View_S([y]_{\mathcal{F}_N}|v_1, ..., v_k) = View_S([y]_{\mathcal{F}_N})$, then $\mathcal{P}$ is unconditionally secure.

In the following two lemmas, we reason about the security of the protocols in the ABB model. Because we assume that the underlying primitives of ABB are secure, the secret values can only be leaked through the reveal operation of ABB. Hence when a protocol is constructed in a pure ABB framework, we can only check whether the revealed values contain any information to ensure the security. Lemma 2.3 is straightforward, and Lemma 2.4 simply follows Lemma 2.3 and the definition of ABB and unconditional security. Both lemmas are self-explanatory.

Lemma 2.3. *Let $\mathcal{T} \subseteq 2^{[n]}$ be the collection of sets that are* not *allowed to jointly access a secret through the underlying secret sharing scheme of ABB.*

1) *Let r be a secret random value which is uniformly and independently sampled from $\mathcal{F}_N$. Then, for all $S \in \mathcal{T}$ and for all $x \in \mathcal{F}_N$, we have $View_S([x]_{F_N}|x + r) \equiv_s View_S([x]_{F_N})$.*

2) *Let r be a secret random value which is uniformly and independently sampled from $\mathcal{F}_N^*$. Then, for all $S \in \mathcal{T}$ and for all $x \in \mathcal{F}_N^*$, we have $View_S([x]_{F_N}|x \cdot r) \equiv_s View_S([x]_{F_N})$.*

Lemma 2.4. *If all of the revealed values of an ABB-based protocol only involve the following two cases, then the protocol is unconditionally secure.*

1) $[r]_{\mathcal{F}_N} \in_R \mathcal{F}_N$; $z \leftarrow [x]_{\mathcal{F}_N} + [r]_{\mathcal{F}_N}$.

2) $[r]_{\mathcal{F}_N} \in_R \mathcal{F}_N^*$; $z \leftarrow [x]_{\mathcal{F}_N} \cdot [r]_{\mathcal{F}_N}$.

All of the protocols in this paper are implemented by constant-depth circuits with unconditional security in the ABB model, and the complexity is measured according to how many secure multiplications are used. We abbreviate the secure multiplication to MULT as a unit. Regarding the other two basic operations, because the computation/communication cost of both $c \leftarrow [x]_{\mathcal{F}_N}$ and $c \leftarrow [x]_{\mathcal{F}_N} \cdot [y]_{\mathcal{F}_N}$ are not more than $[z]_{\mathcal{F}_N} \leftarrow [x]_{\mathcal{F}_N} \cdot [y]_{\mathcal{F}_N}$, we simply count their costs by one MULT.

2.3 Secret Random Elements

Several of our protocols require computation involving secret random elements. Because they are unrelated to the inputs, the parties can prepare them in the offline phase. Here we introduce three basic types of random element generation. We describe them using $\mathcal{Z}_p$-arithmetic; similar methods exist for general $\mathcal{F}_N$-arithmetic.

Random Elements. The method for generating a secret sharing of a uniformly random, unknown element in $\mathcal{Z}_p$ depends on the security type of the concrete ABB scheme. However, a standard framework is the following.[3] Each party P_i (uniformly and independently) samples a random value r_i from $\mathcal{Z}_p$ and sends its shares to all of the parties according to the secret sharing scheme. Then, the parties add up the secret shares, $[r]_p = \sum_{i=1}^{n} [r_i]_p$. In the active setting, if the j-th party cheats on the value of r_j, the randomness of r can still be achieved by the other parties. When at least one honest party contributes to the sum, r will be random and private. The cost is no more than one MULT. We denote this operation as $[r]_p \leftarrow \mathcal{Z}_p$.

Random Bits. Here a random bit is a secret $[b]_p$ with constraint $b \in \{0, 1\}$. To generate a random bit, the parties first perform $[r]_p \leftarrow \mathcal{Z}_p$, $r^2 \leftarrow [r]_p \cdot [r]_p$ and check whether $r^2 = 0$, to abort or not. If $r^2 = 0$, then the parties the parties output $[b]_p \leftarrow ([r]_p / \sqrt{r^2} + 1)/2$. The probability that $r^2 = 0$ is only $1/p$, which is negligible. The whole procedure costs approximately 2 MULTs and 2 rounds. [13]

Random Solved Bits. A set of random solved bits (SolvedRan) $[r]_B = ([r_{\ell-1}]_p, ..., [r_0]_p)$ can be generated first by generating ℓ random bits, and then the parties check whether $[r]_B < p$ to abort or not. Following the assumption in [13, 27, 34, 26], we run four candidates in parallel to obtain a low aborting probability. In the literature, the test $[r]_B < p$ dominates the cost of this functionality, but this procedure can be improved to be sublinear in ℓ by using the solution in Section 4. However, it still costs $O(\ell)$ MULTs to generate ℓ random bits. We discuss this constraint further in Section 8.

3 Existence of d-Effective Sign Modules of Finite Fields

In this section, we show that any finite field of an odd characteristic has a good property such that we can construct efficient secure protocols using this property (shown later, in Section 6). We abuse

[3] Because this procedure is non-black-box, this function can be assumed to be one of the basic operations in the ABB model.

the term "sign" in a group $(G, \diamond)$ that has a subnormal group G^+, which has exactly 2 cosets G^+ and G^-. In other words, the order of the quotient group G/G^+ is 2. For all $x_1, r_1 \in G^+, x_2, r_2 \in G^-$, we have the following properties: $x_1 \diamond r_1, x_2 \diamond r_2 \in G^+$, and $x_1 \diamond r_2, x_2 \diamond r_1 \in G^-$. Denote the identity of G^+ by g^+, and let g^- be the element in G^- such that $g^- \diamond g^- = g^+$. Accordingly, the "sign" of x in G is defined as $\mathsf{Sign}_G(x) = g^+$ if $x \in G^+$; else, g^-.

While we define sign functions over a finite group G, integer signs are usually more of a concern in most applications. Therefore, in the next step, we attempt to map integers in a limited range to a subset of G and simulate the integer sign by Sign_G. We will first focus on the computation over finite fields. Let p be an odd prime, and let $\mathcal{F}_N$ be a finite field, where $N = p^n$. In addition, let id be the multiplicative identity of $\mathcal{F}_N$. Then, G should be a subgroup of $\mathcal{F}_N$. Assume that integers are encoded in a naturally way, using $\{\mathsf{id}, ..., (p-1)/2 \cdot \mathsf{id}\}$ for positive numbers and $\{(p-1)\mathsf{id}, ..., (p+1)/2 \cdot \mathsf{id}\}$ for negative numbers. We hope that the mapping from these elements to G is efficient and that the effective simulation range of Sign_G is large.

Definition 3.1 (d-effective sign module)**.** Let $\mathcal{F}_N$ be a finite field of an odd characteristic, and let id be the multiplicative identity of $\mathcal{F}_N$; let G and G^+ be two multiplicative subgroups of $\mathcal{F}_N^*$ with $G^+ \subset G$ and $|G : G^+| = 2$, and let the other coset of G^+ be G^-. In addition, let $f(x) : \mathcal{F}_N \to G$ be a polynomial mapping function such that for all x in $\{0, \mathsf{id}, ..., d \cdot \mathsf{id}\}$, $f(x)$ is in G^+, and for all x in $\{(N-1)\mathsf{id}, ..., (N-d)\mathsf{id}\}$, $f(x)$ is in G^-. Then (f, Sign_G) forms a d-effective sign module over $\mathcal{F}_N$.

Note that there may be many choices for the mapping function $f(x)$. However, when the input x is a secret in secure computation, it is inefficient to implement the mapping by an arbitrary table. Instead, we expect a very efficient mapping, such as a linear mapping or a low-degree polynomial. Also note that $\mathcal{F}_N$ is restricted to have an odd characteristic; otherwise, $\mathcal{F}_N^*$ does not contain any subgroups G, G^+ such that $|G : G^+| = 2$ because $|\mathcal{F}_N^*|$ is odd.

By default, we use the largest multiplicative subgroup, $\mathcal{F}_N^*$, to compute the sign. Note that F_N^* is a valid group for constructing sign modules because the nonzero quadratic residues in $\mathcal{F}_N$ form a subgroup of $\mathcal{F}_N^*$, and the subgroup has two cosets. The other coset is the set of quadratic non-residues in $\mathcal{F}_N$. $\mathsf{Sign}_{\mathcal{F}_N^*}(x)$ can be computed from $x^{(N-1)/2}$ with $O(\log^3 N)$ bit operations, or we can use a generalized version of the Jacobi symbol algorithm for any finite field, to obtain better efficiency ($O(\log^2 N)$ bit operations) [3].

Then, in Theorem 3.3, we show that there is always an $\Omega(\log N)$-effective sign module of $\mathcal{F}_N$ when $p > \log_2 N$. To this aim, we first derive the number of a specific subset of d-effective sign modules of $\mathcal{F}_N$, as described in Lemma 3.2. Although a similar result of Lemma 3.2 for modular fields can be found in [28], their proofs are through probabilistic demonstration, which only implies a statistically asymptotic guarantee. Therefore, for completeness, we show a deterministic proof for general finite fields along the lines of [2, 19] in Section 3.1.

Lemma 3.2. *Let $\mathcal{F}_N$ be a finite field, where $N = p^n$, where p is an odd prime. Let a be a nonzero element in $\mathcal{F}_N$, and let d be an integer, $0 \le d < (p-1)/2$. Let $S = \{x \in \mathcal{F}_N | x, x+a, ..., x+d \cdot a$ are quadratic residues, and $x-a, ..., x-d \cdot a$ are quadratic non-residues $\}$. Then, $|S|$ is in the range $\left(\frac{N}{2^{2d+1}} - d - \frac{1}{2}\right) \pm \sqrt{N}\left(d - \frac{1}{2}\right)$*

Theorem 3.3. *For all $N = p^n$, where p is an odd prime, $\mathcal{F}_N$ contains a d-effective sign module with $d = \min(\lfloor \log_2 N/5 \rfloor, (p-3)/2)$.*

Proof. Following Lemma 3.2 and its notation, we have that $|S| > 0$ if d satisfies the following:

$$\frac{N}{2^{2d+1}} > \sqrt{N}\left(d - \frac{1}{2}\right) + \left(d + \frac{1}{2}\right).$$

Because $N \geq 1$, it is adequate to request the following:

$$d < \frac{1}{4}\log_2 N - 1 - \frac{1}{2}\log_2 d.$$

On the other hand, we require $d < (p-1)/2$ so that $x - d \cdot a, ..., x + d \cdot a$ are distinct. Hence, we conclude that there is a d-effective sign module of $\mathcal{F}_N$ with $d = \min(\lfloor (\log_2 N)/5 \rfloor, (p-3)/2)$.

□

3.1 Proof of Lemma 3.2

Let $\chi : \mathcal{F}_N \to \{-1, 0, 1\}$ be the quadratic character of $\mathcal{F}_N$, i.e., $\chi(x) = 1$ if x is a quadratic residue; $\chi(x) = 0$ if $x = 0$; $\chi(x) = -1$ if x is a quadratic non-residue.

Theorem 3.4 (Weil bound)**.** *Let $f(x) \in \mathcal{F}_N[x]$ be a polynomial of positive degree which is not of the form $h^2(x)$ for any $h(x) \in \mathcal{F}_N[x]$, and let t be the number of distinct roots of $f(x)$ in the algebraic closure of $\mathcal{F}_N$. Then*

$$\left| \sum_{x \in \mathcal{F}_N} \chi(f(x)) \right| \leq (t-1)\sqrt{N}.$$

(See e.g., p.43, Theorem 2C of [32].)

Let $H(x) = \prod_{i=1}^{d}(1 - \chi(x - i \cdot a)) \cdot \prod_{i=0}^{d}(1 + \chi(x + i \cdot a))$,
$S_1 = \{x \in \mathcal{F}_N | \chi(x-a) = ... = \chi(x - d \cdot a) = -1, \chi(x) = \chi(x+a) = ... = \chi(x + d \cdot a) = 1\}$,
$S_2 = \{x \in \mathcal{F}_N | \chi(x-a) = 1$ or... or $\chi(x - d \cdot a) = 1$ or $\chi(x) = -1$ or..., or $\chi(x + d \cdot a) = -1\}$,
and $S_3 = \mathcal{F}_N - S_1 - S_2$.

Note that the condition of S_3 is similar to S_1 except one of $\chi(x - d \cdot a), ..., \chi(x + d \cdot a)$ is zero. Since $\mathcal{F}_N$ only contains one zero, we have $|S_3| = 2d + 1$. Let $|S_1| = s$. Then, due to $\sum_{x \in S_2} H(x) = 0$,

$$\sum_{x \in \mathcal{F}_N} H(x) = 2^{2d+1}s + 2^{2d}(2d+1).$$

On the other hand, $H(x)$ can be expanded as

$$H(x) = 1 + \sum_{m=1}^{d}\sum_{n=1}^{d+1}(-1)^m \sum_{\substack{1 \leq i_1 < ... < i_m \leq d \\ 0 \leq j_1 < ... < j_n \leq d}} \chi((x - i_1 \cdot a)...(x - i_m \cdot a)(x + j_1 \cdot a)...(x + j_n \cdot a)).$$

Let $f(x) = (x - d \cdot a) \cdot ... \cdot (x + d \cdot a)$. Since $2d + 1 < p$ and $f(x)$ has an odd number of roots, $f(x)$ can not be written as $h(x)^2$ for some $h(x) \in \mathcal{F}_N[x]$. Therefore, applying Theorem 3.4 we have

$$\begin{aligned}
\left|2^{2d+1}s + 2^{2d}(2d+1) - N\right| &\leq \sum_{m=1}^{d}\sum_{n=1}^{d+1}\sum_{\substack{1\leq i_1<\ldots<i_m\leq d\\ 0\leq j_1<\ldots<j_n\leq d}}\left|\sum_{x\in\mathcal{F}_N}\chi((x-i_1\cdot a)\ldots(x-i_m\cdot a)(x+j_1\cdot a)\ldots(x+j_n\cdot a))\right| \\
&\leq \sum_{m=1}^{d}\sum_{n=1}^{d+1} C_m^d C_n^{d+1}(m+n-1)\sqrt{N} \\
&< 2^{2d}(2d-1)\sqrt{N}.
\end{aligned}$$

Hence

$$\frac{N}{2^{2d+1}} - d - \frac{1}{2} - \sqrt{N}(d-\frac{1}{2}) < s < \frac{N}{2^{2d+1}} - d - \frac{1}{2} + \sqrt{N}(d-\frac{1}{2}).$$

□

4 Concrete Sign Modules of Prime Fields

In this section, we describe more results regarding sign modules of prime fields expressed in the form of $\mathcal{Z}_p$, a ring of congruence classes of a prime characteristic.

First, we observe that in many applications, an adequate field size is required instead of a specific p. Hence, we ask the following question: when ℓ is given, is there a prime $p \in [2^{\ell-1}, 2^\ell]$ such that $\mathcal{Z}_p$ has a sign module whose effectiveness is better than the bound described in Theorem 3.3? We will show the existence of $\Omega(\log p \log\log p)$-effective sign modules of $\mathcal{Z}_p$ in this setting. Sign modules with such effectiveness can provide simpler and more efficient construction of secure protocols. Specifically, a d-effective sign module with $d > 2\log_2 p$ supports secure protocols such as $\mathsf{OR}^*([x]_B)$ and $\mathsf{AND}^*([x]_B)$, to compute using 1 MULT in the online phase (shown later, in Section 6).

Second, although by Theorem 3.3 we know that there is always an $\Omega(\log p)$-effective sign module of $\mathcal{Z}_p$, we have no idea how to find one of such sign modules. A trivial but inefficient method is to sample a prime $p \in [2^{\ell-1}, 2^\ell]$, involving a primality test with a success probability of $\Omega(1/\log p)$ (due to the prime density) first and then to sample an element in $\mathcal{Z}_p$. However, by Lemma 3.1, the success probability through random sampling is only $2^{-\theta(\log p)}$, and the computation in each sampling mainly involves quadratic character tests (each of which costs $O(\log^2 p)$ bit operations using [3]). When ℓ is given first, we show a randomized method to sample a prime p of length ℓ accompanied by an $\Omega(\log p)$-effective sign module of $\mathcal{Z}_p$ with a success probability of $\Omega(1/\log p)$. The computation of each sampling mainly involves a primality test, which can be done in $\tilde{O}(\log^6 p)$ [4] by [23]. The intuition is that, because the distribution of the quadratic residue/non-residue is quite uniform in $\mathcal{Z}_p$, the density of a certain pattern is quite small. On the other hand, the density of primes in arithmetic progressions is much larger. Hence, it would be more efficient to express the pattern by certain arithmetic progressions first and then to sample a prime among them.

4.1 $\pm d$-CQRN Scheme

Theorem 3.3 states the existence of an $\Omega(\ell)$-effective sign module in $\mathcal{Z}_p$ for each prime $p \in [2^{\ell-1}, 2^\ell]$. By intuition, if we choose the most effective sign module among the sign modules of those prime

[4] The notation $\tilde{O}(x)$ signifies a bound $c_1 x(\log x)^{c_2}$ for suitable positive constants c_1 and c_2.

fields, it is possible to promote the effectiveness. To show that this possibility is true, we will prove the existence of an $\Omega(\ell \log \ell)$-effective sign module when ℓ is given under the *generalized Riemann hypothesis* (GRH).

Theorem 4.1. *Assume GRH. Then, there exists a prime $p \in [2^{\ell-1}, 2^{\ell}]$ such that $\mathcal{Z}_p$ has an $\Omega(\ell \log \ell)$-effective sign module.*

Instead of purely proving the existence, we provide concrete construction. Specifically, we show that a simple pattern of quadratic character is adequate to obtain the result. (In Section 4.2, we reason that this simple pattern is already a statistically optimal choice.)

Definition 4.2. We say that a prime p is qualified for *$\pm d$-Consecutive Quadratic Residues and Non-residues ($\pm d$-CQRN)* if and only if $1, ..., d \in \mathcal{Q}R_p$ and $-1, ..., -d \in \mathcal{N}R_p$.
Note that, if p is qualified for $\pm p_k$-CQRN, then p is also qualified for $\pm d$-CQRN for d up to $p_{k+1}-1$.

Note that, when p is qualified for $\pm d$-CQRN, $(x, \mathsf{Sign}_{\mathcal{Z}_p})$ forms a d-effective sign module of $\mathcal{Z}_p$. We will prove Theorem 4.1 by showing that there is a prime $p \in \{2^{\ell-1}, 2^{\ell}\}$ that is qualified for $\pm d$-CQRN with $d = \Omega(\ell \log \ell)$. This proof is accomplished by elucidating the relationship between the range of primes and the number of primes that are qualified for $\pm d$-CQRN. Specifically, we would like to know whether the number of primes with length ℓ that are qualified for $\pm d$-CQRN, for some $d = \Omega(\ell \log \ell)$, is greater than 0.

Definition 4.3. Let $N(x, d)$ denote the number of the primes that are smaller than x and are qualified for $\pm d$-CQRN.

First, in the following lemma, we show that the conditions of $\pm d$-CQRN can be expressed as the union of $\prod_{i=2}^{k} \frac{p_i-1}{2}$ sets of linear equations, where $k = \pi(d)$.

Lemma 4.4. *Let $k = \pi(d)$ and let p be a prime, $p > 3$. Then, p is qualified for $\pm d$-CQRN if and only if*

$$\begin{cases} p \equiv -1 \mod 8, \\ p \equiv q_i \mod p_i, \forall i = 2, ..., k, \end{cases} \tag{1}$$

$$\text{for some } \begin{cases} q_i \in \mathcal{Q}R_{p_i}, & \text{if } p_i \equiv 1 \mod 4; \\ q_i \in \mathcal{N}R_{p_i}, & \text{if } p_i \equiv 3 \mod 4. \end{cases} \tag{2}$$

The proof is provided in Section 5.

Then, by Lemma 4.4, to find a prime p that is qualified for $\pm d$-CQRN is equivalent to finding a prime solution of any of these equation sets. Because each equation corresponds to an arithmetic progression, by Dirichlet's theorem, we know that there are infinitely many prime solutions. Furthermore, if the distribution of primes were perfect independent random, then for all x and $i \geq 2$, $E\left[\sum_{q_i \in \mathcal{Q}R_{p_i}} \pi(x; p_i, q_i)\right] = E\left[\sum_{q_i \in \mathcal{N}R_{p_i}} \pi(x; p_i, q_i)\right] = \pi(x)/2, E\left[\pi(x; 8, 7)\right] = \pi(x)/4$, and thus, $E\left[N(x, d)\right] = \pi(x)/2^{k+1}$. We know that the real distribution is quite close to the behavior of random, and that these expected results can be obtained asymptotically when x grows to infinity by, e.g., Dirichlet's theorem. However, because the distribution of primes and the pattern of quadratic residues/non-residues is not truly random, the proof should be taken more carefully when x is not very large and when deterministic results instead of probabilistically asymptotic results are considered.

Lemma 4.5. *Assume GRH. Then we have*

$$\left| N(x,d) - \left(\frac{1}{2}\right)^{k+1} \pi(x) \right| < O\left(\left(\frac{3}{2}\right)^{k-1} \frac{\sqrt{x}}{\log x} \right), \text{ where } k = \pi(d).$$

The proof is provided in Section 5.2.

Using Lemma 4.5, we prove Theorem 4.1 by showing that there exists $d \in \Omega(\ell \log \ell)$ such that $N(2^\ell, d) - N(2^{\ell-1}, d) > 0$ when ℓ is sufficiently large.

Proof of Theorem 4.1.
Let $k = \pi(d)$ (i.e., $p_k \leq d < p_{k+1}$). By Lemma 4.5, there are constants A and B such that for all $d > A$,

$$\left(\frac{1}{2}\right)^{k+1} \frac{2^\ell}{\ell \ln 2} - B\left(\frac{3}{2}\right)^{k-1} \frac{2^{\ell/2}}{\ell} < N(2^\ell, d) < \left(\frac{1}{2}\right)^{k+1} \frac{2^\ell}{\ell \ln 2} + B\left(\frac{3}{2}\right)^{k-1} \frac{2^{\ell/2}}{\ell}.$$

Then,

$$N(2^\ell, d) - N(2^{\ell-1}, d) > \left(\frac{1}{2}\right)^{k+1} \frac{2^\ell}{\ell \ln 2} \left(1 - \frac{1}{2(\ell-1)}\right) - 2B\left(\frac{3}{2}\right)^{k-1} \frac{2^{\ell/2}}{\ell}.$$

To obtain the maximum k that satisfies $N(2^\ell, d) - N(2^{\ell-1}, d) > 0$ from the above condition, we set

$$k = \left\lceil \frac{1}{\log_2 3} \left(\frac{\ell}{2} - 1 - \log_2 B - \log \ln 2 - \log\left(1 - \frac{1}{2(\ell-1)}\right)\right)\right\rceil.$$

Hence, $k = \Omega(\ell)$. Besides, $k = \pi(d) > \frac{d}{\ln d}, \forall d \geq 17$. [30]

This implies that there exists constants C and D such that for all $\ell > C, N(2^\ell, D\ell \log \ell) - N(2^{\ell-1}, D\ell \log \ell) > 0$ (for the reason $\pi(\ell \log \ell) = O(\ell)$). Hence we conclude the existence of $\Omega(\ell \log \ell)$-effective sign modules. □

Corollary 4.6. *Assume GRH. Then there is a constant C such that for all $\ell \geq C$, there always exists a prime $p \in \mathcal{Z}_p$ which contains a $(2\ell+1)$-effective sign module (using $\pm(2\ell+1)$-CQRN scheme).*

Corollary 4.6 is simply implied by the stronger result described in Theorem 4.1. From the proof of Theorem 4.1, we can expect that $N(2^\ell, 2\ell+1) - N(2^{\ell-1}, 2\ell+1) >> 1$ when ℓ is sufficiently large. Furthermore, using numerical data as an auxiliary verification (See Appendix A and Table 3), we show $C = 24$ is only a small number.

4.2 $\pm d$-CQRN is Statistically Optimal

In Theorem 4.1 and Corollary 4.6, we have shown that the more effective sign modules can be obtained in the setting where we fix ℓ first and select a prime p of $\mathcal{Z}_p$ between $2^{\ell-1}$ and 2^ℓ. There follows a natural question of whether $\pm d$-CQRN is the best choice for the mapping of sign modules in $\mathcal{Z}_p$. In the following lemma, we show that this choice is the best at least when we consider only polynomial mapping functions. Moreover, for computation based on secure arithmetic circuits, because only polynomial functions (with a constant degree) can be evaluated efficiently, this result implies that the simple $\pm d$-CQRN is already shown to be the best choice.

Lemma 4.7. *Consider the set of polynomial mapping functions that maps* $\{0, ..., d\}$ *to a subset in* $\mathcal{Q}R_p$ *and* $\{-1, ..., -d\}$ *to a subset in* $\mathcal{N}R_p$*. The simple mapping of* $\pm d$*-CQRN* ($f(x) = x$) *is (statistically) optimal.*

Proof. First, instead of mapping x to x, consider an addition operation, i.e., a shift mapping from x to $x + a$ for a nonzero constant a. This mapping scheme would break the simple requirement of $-1 \in \mathcal{N}R_p$ as Requirement 3.2 and would require more conditions for $\{-1, ..., -d\} \in \mathcal{N}R_p$. On the other hand, the same number of conditions for $\{1, ..., d\} \in \mathcal{Q}R_p$ remains. Therefore, similar to the derivation of Equation 1 in Lemma 4.4 for $\pm d$-CQRN, the conditions of this mapping scheme can also be derived but with considerable more constraint equations. Hence, follow a similar derivation as in Lemma 3.2, the number of primes $N(x, k)$ in this scheme would be (statistically) considerably less than than the number in $\pm d$-CQRN.

Second, consider the multiplication operation of mapping x to ax for a constant a. This mapping scheme would only add an additional condition, $a \in \mathcal{Q}R_p$; thus, it could not be better than the simple mapping scheme.

Then, the above inferences imply that any arithmetic combination (i.e., a polynomial mapping function) cannot gain advantages, and the simple mapping is already the optimum mapping among these functions.

□

4.3 Finding Concrete Sign Modules

Because we prove the existence of effective sign modules by constructing concrete instances, a related method for finding an effective sign module can also be deduced. First, note that in Equation 1 and 2, for all $i = 2$ to k, p_i has $\frac{p_i-1}{2}$ non-zero quadratic residues and $\frac{p_i-1}{2}$ quadratic non-residues. Accordingly, there are $\prod_{i=2}^{k} \frac{p_i-1}{2}$ sets of k-simultaneous equations. Using the Chinese Remainder Theorem, each set of the k-simultaneous equations can also be expressed as follows:

$$p = -m_1 \frac{Q}{8} + \sum_{i=2}^{k} m_i \cdot \frac{Q}{p_i} \cdot q_i + c \cdot Q, c \in \mathbb{Z}, p > 0, \tag{3}$$

where $Q = 8 \cdot \prod_{i=2}^{k} p_i = 4 \cdot p_k\sharp$, and $m_1, m_2, ..., m_k$ satisfy

$$m_1 \frac{Q}{8} + \sum_{i=2}^{k} m_i \cdot \frac{Q}{p_i} = 1. \tag{4}$$

Note that, because $\frac{Q}{8}, \frac{Q}{p_1}..., \frac{Q}{p_k}$ are coprime to each other, the value of $m_1, ..., m_k$ can be solved by the Euclidean Algorithm efficiently.

Hence, by Lemma 4.4, one way to find a qualified prime $p < Q$ is by randomly sampling these $\prod_{i=2}^{k} \frac{p_i-1}{2}$ sets' $q_i's$, substituting them into Equation 3 and testing whether p is a prime. To perform this task, we are required to find all of the q_i's that satisfy Equation 2 and to compute the m_i's in Equation 4 first; however, these steps can be performed in polynomial time and need to only be

computed once. Then, because Equation 3.3 can be computed efficiently, the main computation involves a primality test, which can be accomplished in $\tilde{O}(\log^6 p)$ by [23]. Furthermore, instead of $p < Q$, we can also ask $2^{\ell-1} < p < 2^{\ell}$ for a given ℓ. An algorithm for this setting is shown in Figure 1. Note that $[(a-1)Q, aQ] \cap [2^{\ell-1}, 2^{\ell}] \geq Q/2$

Input: $\ell > 2$

Output: a prime $p \in [2^{\ell-1}, 2^{\ell+1}]$, qualified for $\pm d$-CQRN.

1. Set k such that $4p_{k-1}\sharp < 2^{\ell} < 4p_k\sharp$.
2. Set $Q = 4p_{k-1}\sharp$. Set $a = argmax_t|[(t-1)Q, tQ] \cap [2^{\ell-1}, 2^{\ell}]|$.
3. Randomly sample q_i's according to Equation 2, and substitute q_i's into Equation 3 to compute the only p such that $(a-1)Q < p < aQ$.
4. Do primality test. If p is a prime in $[2^{\ell-1}, 2^{\ell}]$, then output p; else, go to Step 3.

Figure 1: An algorithm for sampling a qualified prime

Lemma 4.8. *The success probability in each sampling of the algorithm in Figure 1 (Step 3) is* $\Omega(1/\log Q)$*, and any output* p *of the algorithm is qualified for* $\pm d$*-CQRN with* $d = \Omega(\log p)$*.*

Proof. By Lemma 4.2, the number of primes that are between $(a-1)Q$ and aQ and are qualified for $\pm p_{k-1}$- CQRN is $\theta\left(2^{-k}Q/\log Q\right)$. By Lemma 4.4, all of these qualified primes are included in the $\prod_{i=2}^{k-1} \frac{p_i-1}{2} = O(2^{-k}Q)$ sets of solutions in Equation 2. Hence, if we sample these solutions randomly (by sampling $q_i's$ satisfying Equation 2), then the success probability is $\Omega(1/\log Q)$. Additionally, $a < p_k$ and $\log_2 p - \log_2 a < \log_2 Q = p_{k-1}(1 + o(1))/\ln 2$ (from the Chebyshev function) implies $p_{k-1} = \Omega(\log p)$. Hence, we have that any output p is qualified by $\pm d$-CQRN with $d = \Omega(\log p)$. □

5 Supplementary Proofs for Section 4.1

To be self-contained with respect to references in our proof, we list the basic lemmas of quadratic residues, which are Lemma 5.1, Lemma 5.2 and Theorem 5.3; these lemmas can be found in books on number theory. Let a be an integer, and let p be an odd prime. Denote the Legendre symbol by $(\frac{a}{p})$ (which is 0 if $a|p$, 1 if $a \in \mathcal{Q}R_p^*$, and -1 if $a \in \mathcal{N}R_p$).

Lemma 5.1. *Euler's criterion determines whether* a *is a quadratic residue modulo* p*. Let* p *be an odd prime, and let* a *be an integer that is coprime to* p*. Euler's criterion can be expressed by using the Legendre symbol, as* $(\frac{a}{p}) \equiv a^{(p-1)/2} \mod p$*.*

Lemma 5.2. *The following properties can be derived from Euler's criterion :* ① $(\frac{a}{p}) = (\frac{p+a}{p})$*.* ② $(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p})$*.* ③ $(\frac{a^2}{p}) = 1$ *if* $a \nmid p$*.* ④ $(\frac{-1}{p}) = (-1)^{(p-1)/2}$*. In addition, the solution to* $(\frac{2}{p})$ *can be derived from Gauss's Lemma:* ⑤ $(\frac{2}{p}) \equiv \begin{cases} 1, & p \equiv \pm 1 \mod 8; \\ -1, & p \equiv \pm 3 \mod 8. \end{cases}$

Theorem 5.3 (The law of quadratic reciprocity)**.** *Let p and q be two distinct odd primes. Gauss's statement of the law of quadratic reciprocity can be formulated using the Legendre symbol:* $(\frac{q}{p}) = \begin{cases} (\frac{p}{q}), & \text{if } p \equiv 1 \mod 4 \text{ or } q \equiv 1 \mod 4; \\ (\frac{-p}{q}), & \text{if } p \equiv q \equiv 3 \mod 4. \end{cases}$

5.1 Proof of Lemma 4.4

By Lemma 5.2 ②, the following two requirements are equivalent to the conditions of $\pm d$-CQRN for a prime p in Definition 4.2.

$$\begin{cases} 1, ..., d \in \mathcal{Q}R_p & \textbf{(Requirement 1)} \\ -1 \in \mathcal{N}R_p & \textbf{(Requirement 2)} \end{cases}$$

First, note that the condition of p for Requirement 2 is implied by Lemma 5.2 ④. In other words, $p \equiv 3 \mod 4$. Second, Requirement 1 can be reduced to $p_1, ..., p_k \in \mathcal{Q}R_p$. Then, according to the law of quadratic reciprocity (Theorem 5.3), the conditions of p are derived as follows:

1. $\forall p_i \equiv 1 \mod 4$, $p_i \in \mathcal{Q}R_p$ if and only if $p \in \mathcal{Q}R_{p_i}$, which implies that $p \equiv q_i \mod p_i$ for some $q_i \in \mathcal{Q}R_{p_i}$ (by Lemma 5.2 ①).
2. $\forall p_i \equiv 3 \mod 4$, $-p_i \in \mathcal{Q}R_p$ if and only if $p \in \mathcal{Q}R_{p_i}$. This result implies that $p_i \in \mathcal{Q}R_p$ if and only if (1) $-p_i \in \mathcal{Q}R_p$ and $-1 \in \mathcal{Q}R_p$ or (2) $-p_i \in \mathcal{N}R_p$ and $-1 \in \mathcal{N}R_p$. Condition (1) contradicts Requirement 2; hence, we retain condition (2), which implies that $p \in \mathcal{N}R_{p_i}$. This step leads to $p \equiv q_i \mod p_i$ for some $q_i \in \mathcal{N}R_{p_i}$ (by Lemma 5.2 ①).
3. The condition of the first prime $p_1 = 2$ is provided by Lemma 5.2 ⑤. However, $p \equiv 1 \mod 8$ contradicts Requirement 2; thus, only the condition $p \equiv -1 \mod 8$ is left.

The above conditions of p for Requirement 1 and 2 can then be summarized as Equation 1 and 2 in Lemma 4.4. □

5.2 Proof of Lemma 4.5

Here we prove a more generalized theorem, of which Lemma 4.5 is an immediate result. Note that in the following, p denotes a prime.

Theorem 5.4. *Assume GRH and let $\chi^{a_1}, ..., \chi^{a_k}$ be any nonprinciple characters with $2 \nmid a_i, \forall i$ and $(a_i, a_j) = 1, \forall i \neq j$. Then,*

$$\sum_{\substack{p \leq x \\ p \mod 8 = 7 \\ p \nmid a_1 \cdots a_k}} \prod_{i=1}^{k} \frac{1 \pm \chi^{a_i}(p)}{2} = 2^{-(k+2)}\pi(x) + O\left(\left(\frac{3}{2}\right)^{k-1} \frac{\sqrt{x}}{\log x}\right)$$

To prove the theorem, we require several preparing lemmas. First, we introduce a known inequality of nonprinciple characters summing over primes.

Lemma 5.5 (Theorem 1 of [6] or (1.3) of [5] or (2.12) of [31])**.** *Assume GRH. For all nonprinciple character χ^a modulo a, we have*

$$\left|\sum_{p\le x, p\nmid a}\frac{1\pm\chi^a(p)}{2}-\frac{\pi(x)}{2}\right|\le\alpha=O\left(\frac{\sqrt{x}}{\log x}\right).$$

Lemma 5.6. *Assume GRH and let χ^a and χ^b be two quadratic characters modulo a and b respectively with $(a,b)=1$. Then we have*

$$\left|\sum_{p\le x, p\nmid ab}\left(\frac{1\pm\chi^a(p)}{2}\right)\left(\frac{1\pm\chi^b(p)}{2}\right)-\frac{\pi(x)}{4}\right|\le\beta=O\left(\frac{\sqrt{x}}{\log x}\right).$$

Proof. Let

$$f_{\pm,\pm}=\sum_{p\le x, p\nmid ab}\left(\frac{1\pm\chi^a(p)}{2}\right)\left(\frac{1\pm\chi^b(p)}{2}\right)-\frac{\pi(x)}{4}.$$

By Lemma 5.5, each of $|f_{+,+}+f_{+,-}|, |f_{-,+}+f_{-,-}|, |f_{+,+}+f_{-,+}|$ and $|f_{+,-}+f_{-,-}|$ is bounded by $\alpha=O\left(\frac{\sqrt{x}}{\log x}\right)$. Besides, from the multiplicity property of Dirichlet character,

$$f_{+,+}+f_{-,-}=\sum_{p\le x, p\nmid ab}\frac{1\pm\chi^a(p)\chi^b(p)}{2}=\sum_{p\le x, p\nmid ab}\frac{1\pm\chi^{ab}(p)}{2},$$

where $\chi^{ab}(p)$ is a nonprinciple character modulo ab. Hence we also have $|f_{+,+}+f_{-,-}|\le\alpha$. W.l.o.g. assume $f_{+,+}=\alpha+\gamma$ and $\gamma\ge 0$. Then we have $f_{+,-}\le-\gamma$ and $f_{-,-}\le-\gamma$. However, $f_{+,-}+f_{-,-}\ge-\alpha$. This leads to $\gamma\le\alpha/2$ (as the following table). Similarly, if $f_{+,+}=-\alpha+\gamma$ and $\gamma\le 0$, we also have $\gamma\ge-\alpha/2$.

$f_{\text{row,line}}$	+	-
+	$=\alpha+\gamma$	$\le-\gamma$
−	$\le-\gamma$	$\le-\gamma$

Therefore

$$\left|\sum_{p\le x, p\nmid ab}\left(\frac{1\pm\chi^a(p)}{2}\right)\left(\frac{1\pm\chi^b(p)}{2}\right)-\frac{\pi(x)}{4}\right|\le 3\alpha/2=O\left(\frac{\sqrt{x}}{\log x}\right).$$

□

Lemma 5.7. *Assume GRH and let χ^a be a quadratic characters modulo a. Then we have*

$$\left|\sum_{\substack{p\le x, p\nmid a\\ p \mod 8=7}}\frac{1\pm\chi^a(p)}{2}-\frac{\pi(x)}{8}\right|\le\gamma=O\left(\frac{\sqrt{x}}{\log x}\right).$$

Proof. Let χ_1, χ_2 and χ_3 be the three nonprinciple characters modulo 8:

$$\begin{array}{lll} \chi_1(1) = \chi_1(3) = 1 & \& & \chi_1(5) = \chi_1(7) = -1, \\ \chi_2(1) = \chi_2(5) = 1 & \& & \chi_2(3) = \chi_2(7) = -1, \\ \chi_3(1) = \chi_3(7) = 1 & \& & \chi_3(3) = \chi_3(5) = -1. \end{array}$$

For all $i = 1$ to 3, let

$$f_{2i+1,\pm}(x) = \sum_{\substack{p \le x, p \nmid a \\ x \mod 8 = 2i+1}} \left(\frac{1 \pm \chi^a(x)}{2} \right) - \frac{\pi(x)}{8}.$$

Note that by Lemma 5.6,

$$|f_{3,\pm}(x) + f_{7,\pm}(x)| = \left| \sum_{p \le x, p \nmid a} \left(\frac{1 - \chi_2(p)}{2} \right) \left(\frac{1 \pm \chi^a(p)}{2} \right) - \frac{\pi(x)}{4} \right| \le \beta = O\left(\frac{\sqrt{x}}{\log x} \right).$$

Similarly, $|f_{1,\pm}(x) + f_{7,\pm}(x)|$ and $|f_{1,\pm}(x) + f_{3,\pm}(x)|$ are also bounded by β.

$f_{1,+}$	$f_{3,+}$
$f_{5,+}$	$f_{7,+}$

$f_{1,-}$	$f_{3,-}$
$f_{5,-}$	$f_{7,-}$

Assume $f_{7,+} = \beta + \gamma, \gamma \ge 0$, which implies $f_{1,+} \le -\gamma$ and $f_{3,+} \le -\gamma$. Hence, $-\beta \le f_{1,+} + f_{3,+} \le -2\gamma$, which implies $\gamma \le \beta/2$.

Similarly, assume $f_{7,+} = -\beta + \gamma, \gamma \le 0$, we have $\gamma \ge -\beta/2$. These combines to $|f_{7,+}| \le \frac{3}{2}\beta$.

Similarly, we also have $|f_{7,-}| \le \frac{3}{2}\beta$ and complete to proof. □

Proof of Theorem 5.4.

1. Let a_i be one of $a_1, ..., a_k$ and let $f^{(1)}_{\pm}(x) = \sum_{\substack{p \le x, p \nmid a_i \\ x \mod 8 = 7}} \left(\frac{1 \pm \chi^{a_i}(x)}{2} \right) - \frac{\pi(x)}{8}$. By Lemma 5.7, we have $|f^{(1)}_{\pm}(x)| \le \alpha_1 = O\left(\frac{\sqrt{x}}{\log x} \right)$.

2. Let a_i, a_j be two of $a_1, ..., a_k$ and let $f^{(2)}_{\pm,\pm}(x) = \sum_{\substack{p \le x, p \nmid a_i a_j \\ x \mod 8 = 7}} \left(\frac{1 \pm \chi^{a_i}(x)}{2} \right) \left(\frac{1 \pm \chi^{a_j}(x)}{2} \right) - \frac{\pi(x)}{16}$. From Step 1, we have both of $|f^{(2)}_{+,+}(x) + f^{(2)}_{+,-}(x)|$ and $|f^{(2)}_{+,-}(x) + f^{(2)}_{-,-}(x)|$ are bounded by α_1. Besides, from the multiplicity property of Dirichlet's character, we also have $|f_{+,+} + f_{-,-}| = \sum_{\substack{p \le x, p \nmid a \\ x \mod 8 = 7}} \left(\frac{1 \pm \chi^{a_i}(x)\chi^{a_j}(x)}{2} \right) - \frac{\pi(x)}{8} \le \alpha_1$.

   Then, w.l.o.g. assume $f^{(2)}_{+,+}(x) = \alpha_1 + \gamma^{(1)}, \gamma^{(1)} \ge 0$. We have $f^{(2)}_{+,-}(x) \le \gamma^{(1)}$ and $f^{(2)}_{-,-}(x) \le \gamma^{(1)}$. These lead to $-\alpha_1 \le f^{(2)}_{+,+} + f^{(2)}_{-,-} \le -2\gamma$, which implies $\gamma \le \alpha_1/2$. From similar inferences, we have $|f^{(2)}_{\pm,\pm}| \le \alpha_{(2)} \le \frac{3}{2}\alpha_{(1)}$.

3. By induction, we have

$$\left| \sum_{\substack{p \le x \\ p \equiv 7 \bmod 8 \\ p \nmid a_1 \cdots a_k}} \prod_{i=1}^{k} \frac{1 \pm \chi^{a_i}(p)}{2} - 2^{-(k+2)}\pi(x) \right| \le \alpha_k \le \left(\frac{3}{2}\right)^{k-1} \alpha_1 = O\left(\left(\frac{3}{2}\right)^{k-1} \frac{\sqrt{x}}{\log x}\right)$$

□

6 Secure Protocols

To set up a secure computation, first, the parties decide the input size ℓ, which usually depends on the domain of the plain text, on how many elements and functions are involved, on the range of the computation, and on the accuracy that is designed. According to Corollary 4.6, there always exist primes with bit-length ℓ that are qualified for $\pm(2\ell+1)$-CQRN when ℓ is sufficiently large. Hence after ℓ is decided, the parties agree on a public qualified prime p and perform computation with the $\mathcal{Z}_p$-ABB. Under this setting, we construct protocols and estimate costs based on the use of such $(2\ell+1)$-effective sign modules, for example. Nevertheless, because by Theorem 3.3, there exists $\Omega(\ell)$-effective sign modules for any finite field with an odd characteristic, similar (almost the same) protocols with the same complexity can be obtained when secure arithmetic of an arbitrary finite field of a sufficiently large characteristic is used. The small difference of the implementation is described in the content.

First in Section 6.1, we implement a $\pm\ell$-Sign protocol ($[z]_p \leftarrow \mathsf{Sign}_{\pm\ell}([x]_p)$), which tests the "sign" of $x \in \{p-\ell, ..., p-1\} \cup \{0, ..., \ell\}$. The $\pm\ell$-Sign implies protocols of the ℓ-CMP family (e.g., $[z]_p \leftarrow \mathsf{CMP}_\ell([x]_p \ge [y]_p)$), which can compare values within a small range. The ℓ-CMP family then servers as a building block. In Section 6.2, we implement several Boolean functionalities involving Threshold-k, AND^*, OR^*, Bits Equality Test, Prefix OR/AND, and the Masking Vector. These entities are useful for constructing higher-level protocols that are not limited here. In Section 6.3, we construct elementary bitwise and digit-wise less-than solutions, which have sublinear complexity but are not optimal. In Section 6.4, we present the first tight bound solution to bit decomposition in the ABB model and a digit decomposition protocol. Then, in Section 6.5, combining all of these techniques, we propose our main result for secure protocols - an efficient solution of bitwise less-thans. In Section 6.6, we summarize the use of CMP_ℓ and bitwise less-than in high-level applications that involve zero-test, integer comparison and modulo reduction. The protocol hierarchy is shown in Figure 2.

Our protocols split into online and offline phases. In the offline phase, the parties prepare (conditioned) random elements that are independent of the inputs. A set of random elements may have a condition on each other. However, different sets of random elements are required to be i.i.d. in sampling so that they can be used in the UC-based construction [10]. Because our protocols are described in a hierarchical structure, for higher-level protocols, we describe the offline phase only when additional random elements are required. Regarding the cost estimation, although the two phases are estimated separately, the cost of the offline phase does not affect the complexity of the whole procedure for the lower level protocols (Section 6.1 - 6.3). For those protocols, the two phase model provides only a minor optimization. However, it makes a difference for the higher level

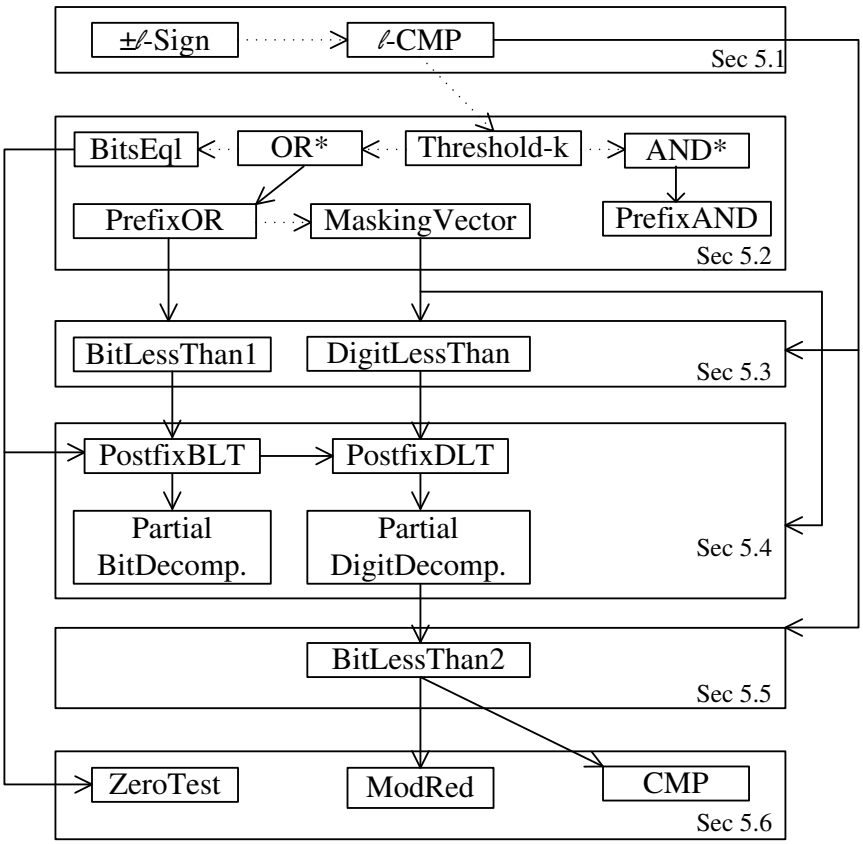

Figure 2: Protocol hierarchy. The dotted lines denote trivial reductions.

protocols (Section 6.4 - 6.6), where we require one or several sets of random-solved-bits ($\mathsf{SolvedRan}$) (See, e.g., [13] or Section 2.3 for a definition) that are prepared in the offline phase. In addition, we discuss an improvement of the complexity of random-solved-bits generation in the ABB model in Section 6.3 and some non-black-box manners for advanced improvement in Section 8.

Security. Our protocols are constructed strictly in the $\mathcal{Z}_p$-ABB model; and thus, the unconditional security of each protocol directly follows Lemma 2.4. Because this construct is easy to check, we will spare the routine description.

6.1 Small Range Comparison

$\pm\ell$-Sign: $[z]_p \leftarrow \mathsf{Sign}_{\pm\ell}([x]_p), \forall x \in \{-\ell, ..., \ell\}$

offline: generate random pairs $\{[r]_p, [s]_p\}$, where $r \in \mathcal{Z}_p^*, s = \mathsf{Sign}_{\mathcal{Z}_p}(r)$.

1. $[a]_p \leftarrow \mathcal{Z}_p, [b]_p \leftarrow \mathcal{Z}_p$
2. $[c]_p \leftarrow [a]_p \cdot [a]_p, [d]_p \leftarrow [a]_p \cdot [b]_p$
3. $[e]_p \leftarrow [c]_p \cdot [d]_p, f \leftarrow [d]_p \cdot [d]_p$, if $f = 0$, abort.
4. $[r]_p \leftarrow (\sqrt{f})^{-1}[e]_p, [s]_p \leftarrow (\sqrt{f})^{-1}[d]_p$

Online:

1. $x' \leftarrow [2x+1]_p \cdot [r]_p$
2. $[z]_p \leftarrow \mathsf{Sign}_{\mathcal{Z}_p}(x') \cdot [s]_p$

ℓ-CMP family: $\forall x, y \in \{0, ..., \ell\}$, compare x and y.

$\mathsf{CMP}_\ell([x]_p \leq [y]_p) = \mathsf{CMP}_\ell([y]_p \geq [x]_p) \leftarrow 2^{-1}(\mathsf{Sign}_{\pm\ell}([y-x]_p) + 1)$

$\mathsf{CMP}_\ell([x]_p < [y]_p) = \mathsf{CMP}_\ell([y]_p > [x]_p) \leftarrow 1 - \mathsf{CMP}_\ell([y] \leq [x])$

$\mathsf{CMP}_\ell([x]_p = [y]_p) \leftarrow \mathsf{CMP}_\ell([x]_p \leq [y]_p) \cdot \mathsf{CMP}_\ell([x]_p \geq [y]_p)$

Figure 3: Implementation of $\pm\ell$-Sign and ℓ-CMP

Small range sign. The definition of the sign function here is slightly different from that of Sign_G, which is defined on a multiplicative group in Section 3. We involve the additive 0 in $\mathcal{Z}_p$. In other words, the evaluation is effective when the input is in $\{-\ell, ..., \ell\}$. Protocol $\pm\ell$-Sign: $[z]_p \leftarrow \ell([x]_p)$ implements this partial function over $\mathcal{Z}_p$. It outputs $[1]_p$ when x is in $\{0, ..., \ell\}$ and $[-1]_p$ when x is in $\{-1, ..., -d\}$.

The concrete scheme is shown in Figure 3. In the offline phase, a random secret $[r]_p$ and its "sign" $[s]_p$ are prepared. If the protocol does not abort, then r is uniformly sampled from $\mathcal{Z}_p^*$. The only revealed value is $f = a^2b^2$, which is independent of $s = \mathsf{Sign}_{Z_p}(ab)$ and a^2, and thus, independent of $r = a^2s$. If $f = 0$, then the protocol aborts. This event occurs with a probability that is less than $2/p$. In the online phase, because x is allowed to be 0, we weed out this case by shifting x to $2x + 1$. Then, $2x + 1$ has the same "sign" as x when $x \neq 0$ and is "positive" when $x = 0$. Then a random secret $[r]_p$ is used to protect $[2x+1]_p$. Because r is a uniformly random value

in $\mathcal{Z}_p^*$, no information of x is leaked. Finally, the "sign" of x can be inferred through the "sign" of r and x'. The price of shifting x to $2x+1$ is that the effective range of the input is approximately half of the range when $[x]_p \neq 0$ is ensured. Because p is qualified for $\pm(2\ell+1)$-CQRN, the output value is valid.

Complexity. Note that, because p is a prime, both $(\sqrt{d})^{-1}$ and $\mathsf{Sign}_{\mathcal{Z}_p}(x')$ can be computed efficiently in polynomial time. The offline phase takes approximately 6 MULTs and 3 rounds, while the online phase takes only 1 MULT and 1 round.

Small range comparison. The output of $\pm\ell$-Sign in $\{[1]_p, [-1]_p\}$ is easy to convert to $\{[1]_p, [0]_p\}$. Hence, $\pm\ell$-Sign implies the ℓ-CMP family, which is designed for small range comparison. Because for all $x, y \in \{0, ...\ell\}$, we have $x - y \in \{-\ell, ..., \ell\}$, $x - y \in \{-\ell, ..., \ell\}$, the test is valid. The cost of the ℓ-CMP family is the same as $\pm\ell$-Sign. The ℓ-CMP family then serves as a building block that is similar to the MULT protocol in this paper.

While we construct the protocol using $\pm(2\ell+1)$-CQRN of $\mathcal{Z}_p$, it is easy to generalize the result by using any $\Omega(\log p)$-effective sign module of any prime field. We summarize this generalized result in the following theorem.

Theorem 6.1. *For all odd prime p, there exists a number $\phi = \Omega(\log p)$ such that CMP_ϕ can be computed in the $\mathcal{Z}_p$-ABB model using $O(1)$ MULTs and $O(1)$ rounds.*

Proof. By Lemma 3.2 and Theorem 3.3, $\mathcal{Z}_p$ contains a d-effective sign module $(\beta + x\alpha, \mathsf{Sign}_{\mathcal{Z}_p^*})$ with $d = \Omega(\log p)$. This sign module suffices to construct a CMP_ϕ protocol with $\phi \geq (d-1)/2$, using a similar procedure in Figure 3, and we need to only modify Step 4 in the offline phase and Step 1 in the online phase of $\pm\ell$-Sign. First, note that -1 is not necessarily in $\mathcal{N}R_p$, and $a^2 s$ can always be in $\mathcal{Q}R_p^*$, regardless of whether $s = 1$ or -1. Let γ be any element in $\mathcal{N}R_p$. Hence in the offline phase, we replace $[r]_p \leftarrow (\sqrt{f})^{-1}[e]_p$ with $[r]_p \leftarrow (1-\gamma)(2\sqrt{f})^{-1}[e]_p + (1+\gamma)2^{-1}[c]_p$ such that $r = a^2$ when $s = 1$ and $r = \gamma a^2$ when $s = -1$. Second in the online phase, if $\beta, \beta+\alpha, ..., \beta+d\alpha$ are all non-zero, then we replace $x' \leftarrow [2x+1]_p \cdot [r]_p$ with $x' \leftarrow [\beta + x\alpha]_p \cdot [r]_p$; otherwise, if one of $\beta + (2k-1)\alpha$, for $k = 1, ...(d+1)/2$, is zero, we replace it with $x' \leftarrow [\beta + 2x\alpha]_p \cdot [r]_p$; otherwise, we replace it with $x' \leftarrow [\beta + (2x+1)\alpha]_p \cdot [r]_p$. Therefore, with these simple replacements, the whole complexity of computing CMP_ϕ in the $\mathcal{Z}_p$-ABB model remains $O(1)$ MULTs and $O(1)$ rounds. □

In addition, because Theorem 3.3 provides a guarantee for general finite fields, a similar result can be obtained when we use an $\mathcal{F}_N$-ABB scheme, even though it is more natural to use a modular field/ring in arithmetic computation. We describe this extension in the following corollary, whose proof is similar to that of Theorem 6.1 and omitted. For the same reason, the remaining theorems for $\mathcal{Z}_p$ in this section can be extended for the case of $\mathcal{F}_N$. We will omit these similar statements.

Corollary 6.2. *Let p be an odd prime, and let $N = p^n$, where $n = O(p/logp)$, i.e., $p = \Omega(\log N)$. Then, for any $\phi = \Omega(\log N)$, CMP_ϕ can be computed in the $\mathcal{F}_N$-ABB model using $O(1)$ MULTs and $O(1)$ rounds.*

Threshold-k: Test whether the number of 1's in $[x]_B^m$ is greater than or equal to k, where $x \in \mathcal{Z}_p$. (We only concern a meaningful value of $x \in \{0, ..., p-1\}$.)

$\mathsf{Threshold}([x]_B^m, k) \leftarrow \mathsf{CMP}_\ell(\sum_{i=0}^{m-1}[x_i]_p \geq k)$

$\mathsf{OR}^*([x]_B^m) = \mathsf{Threshold}([x]_B^m, 1)$

$\mathsf{AND}^*([x]_B^m) = \mathsf{Threshold}([x]_B^m, m)$

Bits Equality Test: Test whether a secret $[x]_B^m$ is equal to a public value a, where $x, a \in \mathcal{Z}_p$. When $m < \ell$, set $x_i = 0, \forall i = \ell - 1, ..., m$.

$\mathsf{BitsEql}([x]_B^m, a) \leftarrow \mathsf{CMP}_\ell(\sum_{i=0}^{\ell-1}[x_i]_p \oplus a_i \leq 0)$, where $a = \sum_{i=0}^{m-1} 2^i a_i$.

Prefix-OR: $[z]^m \leftarrow \mathsf{PrefixOR}([x]_B^m)$, $x \in \mathcal{Z}_p$

$\forall i = m-1, ..., 0$, in parallel: $[z_i]_p \leftarrow \mathsf{CMP}_\ell(\sum_{j=i}^{m-1}[x_i]_p \geq 1)$ in parallel.

Prefix-AND: $[z]^m \leftarrow \mathsf{PrefixAND}([x]_B^m)$, $x \in \mathcal{Z}_p$

$\forall i = m-1, ..., 0$, in parallel: $[z_i]_p \leftarrow \mathsf{CMP}_\ell(\sum_{j=i}^{m-1}[x_i]_p \geq m)$ in parallel.

Masking Vector: $[z]_B^m \leftarrow \mathsf{MV}([x]_B^m), \forall x \in Z_p$. Return a Boolean vector indicating the location of the first nonzero bit.

1. $[u]_B^m \leftarrow \mathsf{PrefixOR}([x]_B^m)$
2. $\forall i = 0..., m-2, [z_i]_p \leftarrow [u_i]_p - [u_{i+1}]_p$,
3. $[z_{m-1}]_p \leftarrow [u_{m-1}]_p$

Figure 4: Implementation of several Boolean functionalities

6.2 Useful Boolean Functionalities

Threshold-k tests whether the number of 1's of a bit-decomposed secret (a Boolean vector) $[x]_B^m = \{[x_{m-1}]_p, ..., [x_0]_p]\}, x_i \in \{0,1\}, x = \sum_{i=0}^{m-1} \in \{0, ..., p-1\}, m \leq \ell$, is greater than or equal to k. Because for all $x \in \{0, ..., p-1\}$, this number is small ($\sum_{i=0}^{m-1} x_i \leq m \leq \ell$), we can use one ℓ-CMP to handle it. Threshold-k is the first apparent use of the Sign module. In most real applications, we do not aim to construct general Boolean functions by secure arithmetic circuits. Instead, the main purpose of arithmetic circuits is for arithmetic computation, and only Boolean functions fulfilling the arithmetic purposes are of interested. Threshold-k plays a key rule in many of these functions. For example, OR^* computes $[x_m - 1]_p \vee, ..., \vee [x_0]_p$ and AND^* computes $[x_m - 1]_p \wedge, ..., \wedge [x_0]_p$. OR^* is equivalent to Threshold-1 while AND^* is equivalent to Threshold-m. In addition, *Bits-Equality-Test* tests whether a bit-decomposed secret $[x]_B$ is equal to a public value a. Because a is public, it can be decomposed into $a_0, ..., a_{\ell-1}$, where $a = \sum_{i=0}^{\ell-1} a_i 2^i$, and the exclusive-or of $[x_i]_p$ and a_i ($[x_i] \oplus a_i = [x_i]_p + a_i - 2a_i[x_i]_p$) can be computed locally without communication. When the input only has m bits (i.e., $[x]_B^m$) with $m < \ell$, we can first extend it to $[x]_B$ by adding value-0 elements.

Complexity. Because there are only trivial reductions (communication free with polynomial-time computation), the cost of these protocols (Threshold-k, OR^*, AND^* and $\mathsf{BitsEql}$) is equal to that of CMP_ℓ.

In addition, *Prefix-OR* and *Prefix-AND* run m instances of OR^* and AND^* respectively, in parallel, and *Masking Vector* (MV) can be reduced to Prefix-OR. Hence, each of these protocols requires m invocations of CMP_ℓ.[5]

Similar to the generalization of Theorem 6.1, we can extend these results for general $\mathcal{Z}_p$. By Theorem 6.1, CMP_ϕ, where $\phi = \Omega(\log p)$, can be computed using $O(1)$ MULTs and $O(1)$ rounds in the $\mathcal{Z}_p$-ABB model. This implies that $\mathsf{Threshold}$ can be composed by a constant number (at most $\lceil \ell/\phi \rceil + 1$) of CMP_ϕ. Therefore, the complexity remains the same. We describe this extension in the following theorem.

Theorem 6.3. *For all odd prime p and for all $x \in \mathcal{Z}_p$, given $[x]_B$ as the input,* $\mathsf{Threshold}, \mathsf{OR}^*, \mathsf{AND}^*$ *and* $\mathsf{BitsEql}$ *can be computed using $O(1)$* MULT*s and $O(1)$ rounds, and* $\mathsf{PrefixOR}, \mathsf{PrefixAND}$ *and* MV *can be computed using $O(m)$* MULT*s and $O(1)$ rounds in the $\mathcal{Z}_p$-ABB model.*

6.3 Elementary Bit/Digit-wise Less-Than

Here we implement our first version of bit/digit-wise less-than called $\mathsf{BLT1}/\mathsf{DLT}$, which will be used in constructing an advanced solution. Although $\mathsf{BLT1}$ and DLT are elementary, they already have sublinear complexity. The two protocols follow a logic similar to that of [13] and are inspired by the fact that CMP_ℓ compares $\lfloor \log_2 \ell \rfloor$-bits values using only 1 MULT online (plus 6 MULTs offline). Given two bit/digit-decomposed values as the inputs, the solution intuition is to find the first distinct bit/digit pair from the most significant bit/digit, and then to return the comparison of these two bits/digits. The concrete schemes are shown in Figure 5.

Digit-wise Less-Than compares two digit-decomposed secrets ($[z]_p \leftarrow \mathsf{DLT}([x]_{D(d)}^m < [y]_{D(d)}^m), d \leq \ell$). Note that for all i, $0 \leq x_i, y_i < d \leq \ell$, and thus x_i and y_i can be compared using one CMP_ℓ. Our

[5]Since the whole (offline and online) procedure of ℓ-CMP costs 7 s, the Prefix OR/AND protocol can be improved by computing the significant bits ($i = \ell - 1, ..., \ell - 6$) directly by the MULT protocol. Nevertheless this result improvement is minor when ℓ is large, so we omit the detail.

Digit-wise Less-Than: $[z]_p \leftarrow \mathsf{DLT}([x]^m_{D(d)} < [y]^m_{D(d)}), d \leq \ell$

1. $\forall i = 0, ..., m-1,$ in parallel: $[e_i]_p \leftarrow \mathsf{CMP}_\ell([x_i]_p > [y_i]_p)$, $[f_i]_p \leftarrow \mathsf{CMP}_\ell([x_i]_p < [y_i]_p)$
2. $\forall i = 0, ..., m-1,$ in parallel: $[u_i]^m_p \leftarrow [e_i]_p + [f_i]_p - 2[e_i]_p \cdot [f_i]_p$
3. $[v]^m_B \leftarrow \mathsf{PrefixOR}([u]^m_B)$
4. $[z]_p \leftarrow 1 - [v]^m_B \cdot [e]^m_B$

Bitwise Less-Than (ver. 1):

When the inputs involve two secrets: $[z]_p \leftarrow \mathsf{BLT1}([x]^m_B < [y]^m_B)$

1. Set $a = \lfloor \log_2 \ell \rfloor$, $d = 2^a$ and $\omega = \lceil m/a \rceil$
2. $\forall i = 0, ..., \omega - 1, [x'_i] \leftarrow \sum_{j=0}^{a-1} [x_{i \cdot d+j}]_p,\ [y'_i] \leftarrow \sum_{j=0}^{a-1} [y_{i \cdot d+j}]_p$
3. $[z]_p \leftarrow \mathsf{DLT}([x']^\omega_{D(d)} < [y]^\omega_{D(d)})$

When the inputs involve one secret: $[z]_p \leftarrow \mathsf{BLT1}([x]^m_B < a), a = \sum_{i=0}^{m-1} a_i 2^i$

1. Set $a = \lfloor \log_2 \ell \rfloor$, $d = 2^a$ and $\omega = \lceil m/a \rceil$
2. $\forall i = 0, ..., m-1, [e_i]_p \leftarrow [x_i]_p \oplus a_i$
3. $\forall i = 0, ..., \omega - 1,$ in parallel: $[f_i]_p \leftarrow \mathsf{CMP}_\ell(\sum_{j=0}^{a-1} [e_{i \cdot d+j}]_p \geq 1)$
4. $[g]^\omega_B \leftarrow MV([f]^\omega_B)$
5. $[u]_p \leftarrow [g]^\omega_B \cdot [x]^\omega_B$ (Note that $[x]^m_B$ implies $[x]^\omega_B$)
6. $[v]_p \leftarrow [g]^\omega_B \cdot [a]^\omega_B$ (Note that a implies $[a]^\omega_B$) (communication-free)
7. $[z]_p \leftarrow \mathsf{CMP}_\ell([u]_p < [v]_p)$

For $[z]_p \leftarrow \mathsf{BLT1}([x]^m_B > a)$, replace Step 7 by $[z]_p \leftarrow \mathsf{CMP}_\ell([u]_p > [v]_p)$

Figure 5: Implementation of DLT and BLT1

DLT is designed as an efficient component for higher-level protocols. Hence, the value of d would be set to ℓ or close to ℓ so that the complexity can be improved by an $\Omega(\log \ell)$ factor. Although DLT is not designed for a general digit purpose, an extension for a general digit purpose is not difficult to obtain. Nevertheless, the improvement factor is still bounded by $\Omega(\log \ell)$ because p is only assumed to be qualified for $\pm(2\ell+1)$-CQRN.

Bitwise less-than compares two bit-decomposed values. Here, we proposed an elementary scheme called BLT1. In BLT1 when the inputs are two secrets, it is reduced only to DLT by rearranging the inputs to base-$2^{\lfloor \log_2 \ell \rfloor}$ digit-decomposed values. Moreover, when the inputs are one secret and one public value, we improve it by direct implementation.

Complexity. DLT involves $3m+1$ CMP$'_\ell s$ and m MULTs. The total cost is 3 rounds and $18m+6$ MULTs in the offline phase and 4 rounds and $4m+1$ MULTs in the online phase. In the case of two secret inputs, m-bits BLT1 is reduced to ω-bits DLT, where $\omega = \lceil m/\lfloor \log_2 \ell \rfloor \rceil$. In the case of one secret and one public input, BLT1 involves $2\omega+1$ CMP$'_\ell s$ and ω MULTs. The total cost is 3 rounds and $12\omega+6$ MULTs in the offline phase and 4 rounds and $3\omega+1$ MULTs in the online phase.

By Theorems 6.1 and 6.3, we also have the following extension.

Theorem 6.4. *For all $\mathcal{Z}_p$, $[z]_p \leftarrow \mathsf{BLT1}([x]_B^m < [y]_B^m)$ can be implemented by $\mathcal{Z}_p$-ABB using $O(m/\log \ell)$ MULTs and $O(1)$ rounds.*

Remark 6.5. [Using BLT1 to improve SolvedRan] The definition and standard implementation of SolvedRan can be found in [13, 27] or in Section 2.3. In the process of SolvedRan, after generating $\ell = \lceil \log_2 p \rceil$ random secret bits $\{[r_{\ell-1}]_p, ..., [r_0]_p\}$, we must test whether $\sum_{i=0}^{\ell-1} 2^i r_i < N$, to ensure that r is uniformly sampled from $\mathcal{Z}_p$. Using BLT1, the test costs $O(\ell/\log \ell)$ MULTs. Thus, the original dominant term in SolvedRan is improved and becomes minor. However, generating ℓ independent random secret bits still costs $O(\ell)$ MULTs which becomes the dominant computation.

6.4 Partial Bit/Digit-Decomposition

Bit-decomposition decomposes a secret $[x]_p$ to a set of binary based sharing $[x]_B$. This approach is a standard method that addresses the both worlds - the Boolean functions of $[x]_B$ and the arithmetic functions of $[x]_p$. Because there are ℓ outputs of nontrivial functions that involve ℓ inputs, the problem has a lower bound $\Omega(\ell)$. Here, we propose the first tight bound solution for the ABB model. Because our solution is based on the CMP$_\ell$ protocol, which only costs a few MULTs, it is efficient in practice. Moreover, when the input is conditioned on $x < 2^m$, instead of decomposing the whole ℓ bits blindly, we would expect to perform partial decomposition for higher efficiency.

Our solution is based on the method of the postfix comparison in [34] with variations such that a partial bit-decomposition ($[x]_B^m \leftarrow \mathsf{BitDec}([x]_p, m), x < 2^m$) can be efficiently evaluated when our sign-module based approaches are applied. First, a protocol postfix bit-less-than is defined as $[z]_B^m \leftarrow \mathsf{PostfixBLT}([x]_B^m > [y]_B^m)$, where $[z_i]_p = ([(x \mod 2^{i+1}) > (y \mod 2^{i+1})]_p)$. Following the notation in [34] for clear expression, we define a protocol named full bitwise less-than: $([z^\top]_p, [z^\perp]_p) \leftarrow \mathsf{FullBLT}([x]_B^m, [y]_B^m)$, $z^\top, z^\perp \in \{0,1\}$. $z^\top = 1$ if and only if $x > y$; $z^\perp = 1$ if and only if $x < y$. The outputs of FullBLT involve the complete information of the comparison. In addition, the comparison job can be divided and concatenated.

Bits Comparison: $([z^{\top}]_p, [z^{\perp}]_p) \leftarrow \mathsf{FullBLT}([x]_B^m, [y]_B^m)$

1. $[z^{\top}]_p \leftarrow \mathsf{BLT1}([x]_B^m > [y]_B^m)$
2. $[z^{\top}]_p \leftarrow \mathsf{BLT1}([x]_B^m < [y]_B^m)$

Postfix Bit-Less-Than: $[z]_B^m \leftarrow \mathsf{PostfixBLT}([x]_B^m > [y]_B^m)$,

1. Divide and expand $([x]_B^m, [y]_B^m)$ into a complete binary tree.
2. $\mathsf{Tree} \leftarrow$ Compute the $\mathsf{FullBLT}$ of each node in parallel (involving $2m-1$ pairs of the $\mathsf{FullBLT}$ results).
3. $\forall i = 0, ...m-1$, in parallel: $[z_i]_p = ([(x \mod 2^{i+1}) > (y \mod 2^{i+1})]_p) \leftarrow \mathsf{CMP}_{\ell}([u_i]_p > [v_i]_p)$, where $([u_i]_p, [v_i]_p)$ is obtained by concatenating at most 1 pair on each level of Tree. (Note $x \mod 2^i = \sum_{j=0}^{i-1} x_j 2^j$.)

Partial Bit-Decomposition: $[x]_B^m \leftarrow \mathsf{BitDec}([x]_p, m), x < 2^m$

offline: Prepare a set of random solved bits $\{[r]_B, [r]_p\}$.

online:

1. $c \leftarrow [x]_p + [r]_p$
2. If $c \geq 2^m - 1$, set $f = 1$; else, $[f]_p \leftarrow \mathsf{BLT1}([r]_B^m \leq c)$
3. $\forall i = 0, ..., m-1, [y_i]_p \leftarrow [f]_p \cdot [c_i]_p + (1 - [f]_p) \cdot [(c+p)_i]_p$
4. $[u]_B^m \leftarrow \mathsf{PostfixBLT}([r]_B^m > [y]_B^m)$
5. $[x \mod 2^m]_p \leftarrow [x]_p$
   $\forall i = m-1, ..., 1, [x \mod 2^i]_p \leftarrow [y \mod 2^i]_p + 2^i[u_i]_p - [r \mod 2^i]_p$
6. $\forall i = m-1, ..., 1, [x_i]_p \leftarrow 2^{-i}([x \mod 2^{i+1}]_p - [x \mod 2^i]_p)$
   $[x_0]_p \leftarrow [x \mod 2]_p$

Figure 6: Implementation of Partial Bit-Decomposition

Definition 6.6 (Concatenation Operator)**.** Let $||$ denote the binary concatenation operator for two pairs of bit-decomposition values, as follows: $([x]_B^a, [y]_B^a)||([x']_B^b, [y']_B^b) = ([u]_B^{a+b}, [v]_B^{a+b})$, where $[u]_B^{a+b} = \{[x_{a-1}, ..., x_0, x'_{b-1}, ..., x'_0]\}$, $[v]_B^{a+b} = \{[y_{a-1}, ..., y_0, y'_{b-1}, ..., y'_0]\}$.

Divide $[x]_B^m$ into two parts by setting $[x^\top]_B^{m-a} = \{[x_{m-1}]_p, ..., [x_a]_p\}$ and $[x^\perp]_B^a = \{[x_{a-1}]_p, ..., [x_0]_p\}$ for any $0 < a < m$, and similarly also divide $[y]_B^m$ into $[y^\top]_B^{m-a}$ and $[y^\perp]_B^a$. We have

$$\mathsf{FullBLT}([x]_B^m, [y]_B^m) = \mathsf{FullBLT}((\mathsf{FullBLT}([x^\top]_B^{m-a}, [y^\top]_B^{m-a}))||(\mathsf{FullBLT}([x^\perp]_B^a, [y^\perp]_B^a))).$$

Lemma 6.7. *Assume that p is qualified for $\pm(2\ell{+}1)$-CQRN. In the $\mathcal{Z}_p$-ABB model,* $\mathsf{PostfixBLT}([x]_B^m > [y]_B^m)$ *can be computed using $O(1)$ rounds and $O(m)$* MULT*s (approximately 3 rounds and $33m$* MULT*s in the offline phase and 5 rounds and $7m$* MULT*s in the online phase).*

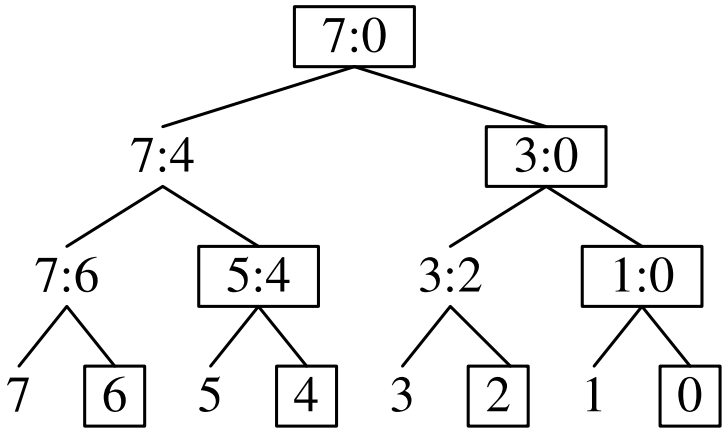

Figure 7: A demonstration of the comparison tree in PostfixBLT. The number denotes the bit index. Note that only the right children and the root (circled by boxes) are involved.

Proof. First, divide and expand $\mathsf{FullBLT}([x]_B^m > [y]_B^m)$ into a complete binary tree Tree. The root (on level $\lfloor log_2 m \rfloor$) is simply the full comparison $\mathsf{FullBLT}([x]_B^m > [y]_B^m)$ while the leaves are $\mathsf{FullBLT}([x_{m-1}]_p, [y_{m-1}]_p)$,...,$\mathsf{FullBLT}([x_0]_p, [y_0]_p)$. (See Figure 7 for example)

Then, assume that $\mathsf{FullBLT}([x]_B^\nu, [y]_B^\nu)$ can be computed using R rounds and $C(\nu) = a\nu + b$ MULTs. Each node on level i requires at most $C(2^i)$ MULTs. Specifically, we skip the computation of leaves because their inputs are already one-bit pairs. The total cost of computing all of the nodes except for the leaves in parallel is the following:

$\sum_{i=1}^{\lfloor \log_2 m \rfloor} C(2^i) \cdot 2^{\lfloor \log_2 m \rfloor - i} = a \sum_{i=1}^{\lfloor \log_2 m \rfloor} 2^i \cdot \frac{m}{2^i} + b \sum_{i=1}^{\lfloor \log_2 m \rfloor} \frac{m}{2^i} = m(a \log_2 m + 2b) - b.$

$\mathsf{FullBLT}([x]_B^\nu, [y]_B^\nu)$ can be implemented by two invocations of BLT1. Therefore, it costs 3 rounds and $(36\nu/\lfloor \log_2 \ell \rfloor + 12)$ MULTs in the offline phase and 4 rounds and $8\nu/\lfloor \log_2 \ell \rfloor + 2$ MULTs in the online phase. The total costs is at most 3 rounds and $60m$ MULTs in the offline phase and 4 rounds and $12m$ MULTs in the online phase.

Then, note that comparing $[x \mod 2^i]_p$ and $[y \mod 2^i]_p$ is equal to comparing the concatenation of at most $\lfloor \log_2 m \rfloor$ nodes of $Tree$ (at most one on each level), and that the value of $\lfloor \log_2 m \rfloor$ bits is less than or equal to $m \leq \ell$. Hence, for all $i = 0$ to $m-1, [z_i]_p \leftarrow [(x \mod 2^i) > (y$

mod $2^i)]_p$ can be evaluated by one CMP_ℓ. These steps cost $6m$ MULTs in the offline phase and m MULTs and one additional round in the online phase.

Furthermore, because only the right children and the root in Tree are used, we can save the computation of the left children. (See Figure 7) To sum up, the total cost of $\mathsf{PostfixBLT}([x]^m_B > [y]^m_B)$ is at most 3 rounds and $33m$ MULTs in the offline phase and 5 rounds and $7m$ in the online phase. □

Digits Comparison: $([z^\top]_p, [z^\perp]_p) \leftarrow \mathsf{FullCMP}_\ell([x]^m_{D(d)}, [y]^m_{D(d)})$

1. $[z^\top]_p \leftarrow \mathsf{DLT}([x]^m_{D(d)} > [y]^m_{D(d)})$
2. $[z^\top]_p \leftarrow \mathsf{DLT}([x]^m_{D(d)} < [y]^m_{D(d)})$

Postfix Digit-Less Than: $[z]^m_B \leftarrow \mathsf{PostfixDLT}([x]^m_{D(d)} > [y]^m_{D(d)})$,

1. $\forall i = 0, ..., m-1, ([x'_i]_p, [y'_i]_p) \leftarrow \mathsf{FullCMP}_\ell([x_i]_p, [y_i]_p)$.
2. $[z]^m_B \leftarrow \mathsf{PostfixBLT}([x']^m_B, [y']^m_B)$.

Partial Digit-Decomposition: $[x]^m_{D(d)} \leftarrow \mathsf{DigitDec}([x]_p, d, m), x < d^m < p$

offline: Prepare a set of random solved digits $\{[r]_{D(d)}, [r]_p\}$.

online:

1. $c \leftarrow [x]_p + [r]_p$
2. If $c \geq d^m - 1$, set $f = 1$; else, $[f]_p \leftarrow \mathsf{DLT}([r]^m_{D(d)} \leq c)$
3. $\forall i = 0, ..., m-1, [y_i]_p \leftarrow [f]_p \cdot [c_i]_p + (1 - [f]_p) \cdot [(c+p)_i]_p$
4. $[u]^m_B \leftarrow \mathsf{PostfixDLT}([r]^m_{D(d)} > [y]^m_{D(d)})$
5. $[x \mod d^m]_p \leftarrow [x]_p$
   $\forall i = m-1, ..., 1, [x \mod d^i]_p \leftarrow [y \mod d^i]_p + d^i[u_i]_p - [r \mod d^i]_p$
6. $\forall i = m-1, ..., 1, [x_i]_p \leftarrow d^{-i}([x \mod d^{i+1}]_p - [x \mod d^i]_p)$
   $[x_0]_p \leftarrow [x \mod d]_p$

Figure 8: Implementation of Partial Digit-Decomposition

The concrete schemes of $\mathsf{FullBLT}, \mathsf{PostfixBLT}$, and BitDec are shown in Figure 6. In BitDec, the design of Steps 2 and 3 are from the observation that, for $x < 2^m, r < p$, if $(c \geq 2^m - 1 \geq x) \vee (c \geq r \mod 2^m)$ is true, no wrap-around modulo p occurs in $x + r \mod p$.

Complexity. BitDec requires an additional set of random solved bits that are prepared in the offline phase. The total cost of BitDec is 1 $\mathsf{PostfixBLT}([r]^m_B > [y]^m_B) + 1$ $\mathsf{BLT1}([r]^m_B \leq c) + 1\mathsf{SolvedRan}$ (in different rounds) + m MULTs (in parallel), where $\mathsf{BLT1}$ only has sublinear complexity. Hence, the cost is less than 3 rounds and 34m MULTs plus 1 $\mathsf{SolvedRan}$ in the offline phase and 11 rounds and 9m MULTs in the online phase.

By Theorems 6.1 and 6.4, we also have the following extension.

Theorem 6.8. *For all $\mathcal{Z}_p$ and for all x in $\mathcal{Z}_p$, there exists a $\mathcal{Z}_p$-ABB protocol that computes the bit-decomposition of x, $[x]_B \leftarrow \mathsf{BitDec}([x]_p)$, using $O(\log p)$ MULTs and $O(1)$ rounds.*

Digit-Decomposition decomposes a secret $[x]_p$ to a set of base-d sharing $[x]_{D(d)}$. For some applications, using a digit-decomposition is more efficient than using a bit-decomposition because fewer elements are involved (e.g., our higher-level protocols are covered later). By a similar construction of BitDec, we can also construct a partial digit-wise decomposition $[x]^m_{D(d)} \leftarrow \mathsf{DigitDec}([x]_p, d, m), x < d^m$. Here, we are only concerned with $d \leq \ell$ because $\mathsf{DigitDec}$ is designed as an efficient component for higher construction and because p is only assumed to be qualified for $\pm(2\ell+1)$-CQRN, although it is not difficult to construct a general digit version. The difference is that we replace $\mathsf{BLT1}$ by DLT and $\mathsf{PostfixBLT}$ by a $\mathsf{PostfixDLT}$, which is defined as $[z]^m_B \leftarrow \mathsf{PostfixDLT}([x]^m_{D(d)}, [y]^m_{D(d)})$, where $[z_i]_p = ([(x \mod d^{i+1}) > (y \mod d^{i+1})]_p)$. $\mathsf{PostfixDLT}$ is the main job in $\mathsf{DigitDec}$. To perform this step, we define $\mathsf{FullCMP}_\ell : ([z^\top]_p, [z^\perp]_p) \leftarrow \mathsf{FullCMP}_\ell([x]_p, [y]_p)$ by an analog definition of $\mathsf{FullBLT}$ (Definition 6.6). Similarly, $\mathsf{FullCMP}_\ell$ is implemented by two CMP_ℓs. Then, $\mathsf{PostfixDLT}$ can be reduced to $\mathsf{PostfixBLT}$ by computing $([x'_i]_p, [y'_i]_p) \leftarrow \mathsf{FullCMP}_\ell([x_i]_p, [y_i]_p)$ for $i = 0$ to $m-1$ first and then $[z]^m_B \leftarrow \mathsf{PostfixBLT}([x']^m_B, [y']^m_B)$. (Recall that we use notation $x = \sum_{i=0}^{m-1} x_i 2^i$.)

Similar to BitDec, $\mathsf{DigitDec}$ requires a set of random solved digits $[r]_{D(d)}, [r]_p$ which are prepared in the offline phase. A set of random solved digits can be generated from processes similar to $\mathsf{SolvedRan}$. Specifically, when $d = 2^\rho$ for any positive integer ρ, $[r]_{D(d)}$ can be obtained by rearranging $[r]_B$.

Complexity. Hence, if we set $d = 2^{\lfloor \log_2 \ell \rfloor}$, then plus m invocations of $\mathsf{FullCMP}_\ell$, the total cost of $\mathsf{DigitDec}$ is less than 3 rounds and 46m MULTs plus 1 $\mathsf{SolvedRan}$ in the offline phase and 13 rounds and 11m MULTs in the online phase. The complete procedure of $\mathsf{FullCMP}_\ell$, $\mathsf{PostfixDLT}$ and $\mathsf{DigitDec}$ are shown in Figure 8.

Similar to the summarization in Theorem 6.8, we also have the following lemma.

Lemma 6.9. *For all $\mathcal{Z}_p$ and for all $x < d^m$, where $d = O(\log p)$, if a d-digit-wise random $[r]_{D(d)}$, where $r \in_R \mathcal{Z}_p$, is prepared in the offline phase, then $[x]^m_{D(d)} \leftarrow \mathsf{DigitDec}([x]_p)$ can be implemented by $\mathcal{Z}_p$-ABB using $O(m)$ MULTs and $O(1)$ rounds in the online phase.*

6.5 Bitwise Less-Than

Combining all of the techniques above, we propose our main result regarding the bitwise less-than. This approach follows the logic of finding the first distinct bit between x and a. First we rearrange the bits of $[x_i] \oplus a_i, \forall i = 0, ..., m-1$ into $\mu \times (\lambda\rho)$ blocks, where $\rho = \lfloor \log_2 \ell \rfloor$, $\lambda = \left\lfloor \kappa \cdot \sqrt{m/\rho} \right\rfloor$ and $\mu = \lceil m/(\lambda\rho) \rceil$, where κ is an optimization parameter. The idea is to locate the first non-zero row (which can be accomplished efficiently by OR^* and MV) and then to locate the first non-zero element of that row (which requires $\mathsf{DigitDec}$, CMP_ℓ and MV) of the blocks. Certainly, the whole computation should be conducted privately so that the location information is stored in secret vectors ($[u']^\mu_B$ and $[g]^\lambda_B$). Finally, using an inner product, we derive the result of $([x]^m_B < a)$. The concrete protocol is shown in Figure 9.

Bitwise Less-Than (ver. 2): $[z]_p \leftarrow \mathsf{BLT2}([x]_B^m < a)$:

1. $\forall i = 0, ..., m-1, [y_i]_p \leftarrow [x_i]_p \oplus a_i$.
2. Let $\rho = \lfloor \log_2 \ell \rfloor$, $d = 2^\rho$. Set $\lambda = \left\lfloor \kappa \cdot \sqrt{m/\rho} \right\rfloor, \mu = \lceil m/(\lambda\rho) \rceil$, where κ is an optimization parameter.
   Rearrange $[y]_B$ into $\mu \times (\lambda\rho)$ blocks, while set the first/last $(m - \lambda\mu\rho)$ empty blocks to be 0. Rearrange a similarly. (Now $a = \sum_{i=0}^{\mu} \sum_{j=0}^{\lambda} a_{i,j}$)
3. $\forall i = 0, ..., \mu-1, [u_i]_p \leftarrow \bigvee_{i=0}^{\lambda\rho-1} [y_{i,j}]_p$ (using the OR^* protocol)
4. $[u']_B^\mu \leftarrow \mathsf{MV}([u]_B^\mu)$
5. $\forall j = 0, ..., \lambda-1, [v_j]_p \leftarrow \sum_{i=0}^{\mu-1} \sum_{k=0}^{\rho-1} [u'_i]_p \cdot 2^k a_{i,j\rho+k}$
6. $\forall i = 0, ..., \mu-1, [w'_i]_p \leftarrow \sum_{j=0}^{\lambda\rho-1} 2^j [x_{i,j}]_p$
7. $[w]_p \leftarrow [w']^\mu \cdot [u']_B^\mu$
8. $[w]_{D(d)}^\lambda \leftarrow \mathsf{DigitDec}([w]_p, d, \lambda)$
9. $\forall i = 0, ..., \lambda-1, [e_i]_p \leftarrow \mathsf{CMP}_\ell([w_i]_p < [v_i]_p$
10. $\forall i = 0, ..., \lambda-1, [f_i]_p \leftarrow \mathsf{CMP}_\ell([w_i]_p > [v_i]_p$.
11. $[g]_B^\lambda \leftarrow \mathsf{MV}([e+f]_B^\lambda)$
12. $[z]_p \leftarrow [g]_B^\lambda \cdot [e]_B^\lambda$

For $[z]_p \leftarrow \mathsf{BLT2}([x]_B^m > a)$, replace Step 12 by $[z]_p \leftarrow [g]_B^\lambda \cdot [f]_B^\lambda$.

Figure 9: Implementation of $\mathsf{BLT2}$

Complexity. BLT2 involves $\mu + \lambda$ MULTs, $\mu + 2\lambda$ CMP_ℓ , 1 $\mathsf{MV}([u]_B^\mu)$, 1 $\mathsf{MV}([e+f]_B^\lambda)$, and 1 $\mathsf{DigitDec}([w]_p, d, \lambda)$. For the optimal adjustment, by setting κ to $\sqrt{1/5}$, the total cost is less than 3 rounds and $58\sqrt{m/\log_2 \ell}$ plus one SolvedRan in the offline phase and 15 rounds and $14\sqrt{m/\log_2 \ell}$ in the online phase.

By Theorems 6.1 and 6.3 and Lemma 6.9, we have also the following extension.

Theorem 6.10. *For all $\mathcal{Z}_p$, given a binary-expressed random $[r]_B$, where $r \in_R \mathcal{Z}_p$, prepared in the offline phase, $[z]_p \leftarrow \mathsf{BLT2}([x]_B^m < a)$ can be implemented by $\mathcal{Z}_p$-ABB using $O(\sqrt{m/\log\log p})$ MULTs and $O(1)$ rounds in the online phase.*

6.6 Applications

Our bitwise less-than solution implies new complexity bounds for several high-level applications in the online phase. For example, with a set of random solved bits $\{[r]_B, [r]_p\}$ that are prepared in the offline phase, a Zero Test (testing whether $[x]_p$ is zero) can be performed by one BitsEql, or equivalently one CMP_ℓ, in the online phase. Several known reductions of applications that were related to bitwise less-than involving the least significant bit [27], integer comparison [27] and modulo reduction [26] are summarized in Figure 10. When a few sets of $\{[r]_B, [r]_p\}$, depending on how many invocations of bitwise less-than are used in the protocol, have been prepared in the offline phase, the remaining computation of these applications can be performed using $O(1)$ rounds and $O(\sqrt{\ell/log\ell})$ MULTs.

7 Related Works and Comparison

This paper is the first study of the "sign" module and its effect on secure arithmetic circuits. However, several results of bit-decomposition, comparison, modulo reduction and equality test based on secure arithmetic circuits have been shown in the previous works and serve as good guides to the problem in this paper. In the following description, we focus on the complexity in the constant-depth arithmetic circuits and leave the detailed cost estimation for Section 7.1.

Bit-decomposition In [13], Damgård et al. introduce the first constant-rounds bit-decomposition protocol with perfect security. Their method is based on the computation of the prefix-carries function. Expressing the carry computation as a semigroup product of the elements *carry, propagate, and kill*, they follow a (Boolean) construction in [11] and obtain a $O(\ell \log \ell)$ complexity. Latter in [34], Using a tree construction and postfix-comparison, Toft improves the complexity to $O(\ell \log^* \ell)$, and in [29], Reistad and Toft further improve it to $O(\ell)$ but only with statistical security against passive adversaries.

Symmetric Boolean functions are Boolean functions with values that depend only on the number of ones in the input. Expressing them as a polynomial function bit-summation (i.e., $f(x_0, ..., x_{\ell-1}) = \phi(1 + \sum_{i=0}^{\ell-1})$), Damgård et al. [13] use the prefix-multiplication (a variation of unbounded fan-in multiplication in [4]) to compute it. They obtain a $O(\ell)$ complexity for symmetric functions involving frequently used AND* and OR*.

Integer Comparison Here we only enumerate representative results that use constant-depth arithmetic circuits and are unconditionally secure. The kernel circuits of secure comparison can be expressed as *bitwise less-than*. In [13], although their bitwise less-than protocol has complexity $O(\ell)$, their secure comparison protocol requires $O(\ell \log \ell)$ because of the complexity of their

Zero Test: $[z]_p \leftarrow ([x]_p = 0)$

offline: Prepare a pair of random solved bits $\{[r]_B, [r]_p\}$.

online:

1. $c \leftarrow [x]_p + [r]_p$.
2. $[z]_p \leftarrow BitsEql([r]_B, r)$.

Least Significant Bit(LSB) [27]: $[z]_p \leftarrow ([x]_p \leq p/2)$

offline: Prepare a pair of random solved bits $\{[r]_B, [r]_p\}$.

online:

1. $c \leftarrow [2x]_p + [r]_p$.
2. $[(2x_0] \leftarrow ([r]_B \leq c) \cdot (c_0 \oplus [r_0]_p) + (1 - ([r]_B \leq c)) \cdot (1 - c_0 \oplus [r_0]_p)$.
3. $[z]_p \leftarrow 1 - [(2x)_0]_p$.

Integer Comparison [27]: $[z]_p \leftarrow ([x]_p \leq [y]_p)$

1. $[u]_p \leftarrow ([x]_p \leq p/2)$; $[v]_p \leftarrow ([y]_p \leq p/2)$; $[w]_p \leftarrow ([x - y]_p \leq p/2)$;.
2. $[z]_p \leftarrow [u]_p(1 - [v]_p) + (1 - [u]_p)(1 - [v]_p)(1 - [w]_p) + [u]_p[v]_p(1 - [w]_p)$

Modulo Reduction [26]: $[z]_p \leftarrow ([x \mod q]_p = 0), \forall q \in \{2, ..., p-1\}$

offline: Prepare a set $\{[r]_{D(q),B}, [r]_p\}$, where $[r]_{D(q),B} = \{[r_{m-1}]_B, ..., [r_0]_B\}$, where $\rho = \lceil \log_2 q \rceil$ and $m = \ell/\rho$, and $r_i \in \{0, ..., q-1\}$.

online:

1. $c \leftarrow [x]_p + [r]_p$.
2. $c_0 \leftarrow c \mod q$; $c'_0 \leftarrow c + p \mod q$.
3. $[u]_p \leftarrow ([r]_B \leq c)$; $[v]_p \leftarrow ([r_0]_B \leq c_0)$; $[w]_p \leftarrow ([r_0]_B \leq c'_0)$.
4. $[z]_p \leftarrow [u]_p([v]_p(c - [r_0]_p) + (1 - [v]_p)(q + c_0 - [r_0]_p)) + (1 - [u]_p)([w]_p(c - [r_0]_p) + (1 - [w]_p)(q + c_0 - [r_0]_p))$.

Figure 10: BLT-realted Applications

bit-decomposition. Later in [27], NIshide and Ohta propose an approach without using a bit-decomposition protocol and obtain a $O(\ell)$ complexity.

Modulo reduction One way to compute modulo reduction is through bit-decomposition and linear re-composition of the shares as described in [13]. However, in [13], this approach requires $O(\ell \log \ell)$ because of bit-decomposition. Recently, in [26], Ning and Xu use an idea that is similar to [27] (the random masking trick), to avoid bit-decomposition and to obtain a $O(\ell)$ complexity.

7.1 Cost Estimation and Comparison

In Table 1, we estimate the cost of our zero test, integer comparison and modulo reduction protocols and compare them with the best known solutions in the same setting (i.e., with unconditional security in the $\mathcal{Z}_p$-ABB model). For simplicity, we omit the constant terms that are smaller than the coefficient of the dominant terms in the complexity.

Bit Decomposition. The analysis can be found in Section 6.4. Adding $8\ell + 60\ell/\log_2 \ell$ of one SolvedRan in the offline phase, we complete the cost estimation of our BitDec.

AND*/OR*. As described in Section 6.2, AND* and OR* (as well as Threshold and BitsEql) can be trivially reduced to CMP_ℓ. Hence their cost is the same as the cost of one CMP_ℓ.

(Deterministic) Zero Test. The offline phase mainly involves one invocation of SolvedRan. The trimmed estimation of this cost, which is described in [27], is 7 rounds and 76ℓ MULTs. Using the improvement by Remark 6.5 and following the same amortized analysis (first used in [13]), which generates four candidates of $[r]_B$ in parallel to ensure a low aborting (when $r \geq p$) rate, the cost of SolvedRan can be reduced to $8\ell + 60\ell/\log_2 \ell$. Leaving out SolvedRan, the zero test protocol in [27] mainly involves one AND*. However, without using the sign module, their solution (from [13]) requires 3 rounds and 5ℓ MULTs.

Integer Comparison. In the offline phase, our solution involves mainly 6 invocations of SolvedRan, while the solution of [27] mainly involves 3 invocations of SolvedRan. However, because of the improvement of SolvedRan, we still have a minor improvement on the coefficient of the dominant term. The cost of the online phase is ascribed mainly to 3 invocations of bitwise less-thans.

Problem	Scheme	Offline		Online	
		Rounds	MULTs	Rounds	MULTs
Bit Decomposition	[34]	7	$52\ell + 24\sqrt{\ell}$	26	$57\ell \log^* \ell + 19\ell + 14\sqrt{\ell}\log^* \ell + 8\sqrt{\ell}$
	Proposed	9	$42\ell + 60\ell/\log_2 \ell$	11	9ℓ
AND*, OR*	[13]	2	3ℓ	2	2ℓ
	Proposed	3	6	1	1
(Deterministic) Zero Test	[27]	7	76ℓ	3	5ℓ
	Proposed	9	$8\ell + 60\ell/\log_2 \ell$	1	1
Integer Comparison	[27]	7	228ℓ	8	51ℓ
	Proposed	9	$48\ell + 360\ell/\log_2 \ell + 174\sqrt{\ell/\log_2 \ell}$	18	$42\sqrt{\ell/\log_2 \ell}$
Modulo reduction	[26]	7	312ℓ	8	42ℓ
	Proposed	9	$56\ell + 420\ell/\log_2 \ell + 174\sqrt{\ell/\log_2 \ell}$	17	$42\sqrt{\ell/\log_2 \ell}$

Table 1: Cost Estimation of Several Unconditional Secure Multiparty Protocols in the $\mathcal{Z}_p$-ABB Model. $\ell = \log_2 p$.

Modulo Reduction. The offline phase of modulo reduction is slightly more complex. It requires a pair of $\{[r]_{D(q),B}, [r]_p\}$. This approach must first generate $[r_i]_B^\rho$ for $i = 0, ..., m-1$ first. Using our improved SolvedRan, this step requires approximately 9 rounds and $32\ell + 240\ell/\log_2 \ell$ MULTs. Adding 3 invocations of SolvedRan that are used in BLT2, these components sum to $56\ell + 420\ell \log_2 \ell$. The online phase is ascribed mainly to 3 invocations of bitwise lese-than.

8 Discussion and Open Issue

Finally we list four topics that are related to this paper and discuss a few open questions for each of them.

8.1 Sign Modules of Modular Rings

In Sections 3 and 4, we show the existence of $\Omega(\log N)$-effective sign modules in any finite field $\mathcal{F}_N$ with $N = p^a$ and $p = O(\log N)$. We show more results regarding sign modules of prime fields; given an integer ℓ, we propose an efficient randomized algorithm to find an $\Omega(\ell)$-effective sign module and to show the existence of an $\Omega(\ell \log \ell)$-sign module of $\mathcal{Z}_p$ with $\ell = \lceil \log_2 p \rceil$.

We can also discuss the case of finite rings. Although Definition 3.1 is described using finite fields, the same definition works for finite rings. Moreover, because modular rings (rings of congruence classes) are more natural for simulating integer arithmetic as well as integer comparison and bit-decomposition, explicit results of a modular ring $\mathcal{Z}_N$ may be more interesting for these applications. Using a similar idea for the proofs in Section 3 for $\mathcal{F}_N$ or Section 4 for $\mathcal{Z}_p$, it is possible to prove the existence of an effective sign module of $\mathcal{Z}_N$. However, this extension is not trivial and requires several technical solutions. Next, we suggest some open issues regarding sign modules of modular rings. How can we find a concrete sign module? How can we use a sign module of $\mathcal{Z}_N$ in applications? Note that, because we need to consider $\mathcal{Z}_N^*$ carefully, additional constraints for usage are accompanied. Additionally, note that several operations in $\mathcal{Z}_N$ need some factorization information with respect to N. For example, the known efficient methods for computing the quadratic character require $\lambda(N)$ (Carmichael function), and the known efficient methods for computing $\sqrt{x}$ and sampling of a random value in $\mathcal{Z}_N^*$ require the factorization of N.

8.2 Random Elements

In this paper, the computation is divided into the offline phase and the online phase. Thus the problem is also divided into two parts. The main work in the offline phase is to generate a set of secret random bits $\{[b_0]_p, ..., [b_{\ell-1}]_p\}$, where $\ell = \log_2 p$ or a set of random digits $\{[d_0]_p, ..., [d_{m-1}]_p\}$, where $m = \lceil \ell/\log_2 d \rceil$ for base-d digits.

The random elements can be seen as valuable resources and some corresponding commodity models can be discussed. However, note that if we require each bit to be independent, the computation complexity is lower-bounded by $\Omega(\ell)$ MULTs for generating ℓ secret random bits using an arithmetic black-box. However, some improvements are possible in non-black-box settings. First, because each secret random bit is generated independently and in parallel, the generation of ℓ secret random bits could be divided into subsets of parties. Although this improvement may work in some specific settings, the implementation should be considered with concrete security models. Second, although computing ℓ independent random bits is lower-bounded by $\Omega\{\ell\}$, this does not imply that

the lower bound of computing m independent random digits is also $\Omega(\ell)$. An open question is whether we can generate m secret random digits using $O(m)$ MULTs in the ABB setting (or using $O(m \log p)$ communication bits in non-black-box settings). One candidate solution is to involve integer-based sharing and compromise with statistical security. We refer readers to Algesheimer et al.'s modulo reduction protocol described in [1] for this direction. However, the protocols in [1] are only passively secure, and it remains unknown whether it can be promoted to be actively secure without losing too much efficiency. Finally, using non-independent pseudo-random bits with weaker security is also a compromise for higher efficiency.

8.3 Extended Applications

First, the techniques that are developed in this paper can be used to build higher-level protocols of, e.g., integer division, the greatest common divisor and the k-th ranked element. The existence of effective sign modules of $\mathcal{Z}_p$ implies that the online phase of these protocols can be improved from known results.

Second, in Section 4, we use a specific pattern of quadratic residues and non-residues to construct instances of sign modules of $\mathcal{Z}_p$. Because of the existence of specific types of patterns in finite fields, it is possible to define more modules (other than sign modules) using patterns of quadratic residues/no-residues, or patterns of other characters of finite fields for various purposes.

8.4 Fully Homomorphic Encryption-Based Computation

Fully Homomorphic Encryption (FHE) supports computation of arbitrary circuits on ciphertexts homomorphically, which implies a non-interactive model in secure multiparty computation. However, there are still several challenges to making FHE practical for secure arithmetic computation. The first challenge concerns the practical efficiency which is affected drastically by security parameters. One may like to refer to a recent demonstration [21], a real implementation of a variant of Gentry's scheme [18]. Nevertheless, this challenge could be conquered gradually by successive FHE schemes in the future. The second challenge is that a concrete scheme of an arithmetic version of FHE (AFHE) (e.g., $\mathcal{F}_N$-arithmetic or $\mathcal{Z}_N$-arithmetic) is still an open problem. We can emulate only the computation with Boolean circuits using current schemes. Even if these challenges are solved, because the advantages of arithmetic circuits v.s. those of Boolean circuits are independent of whether we use an FHE scheme, the problem of efficient non-arithmetic computation using secure arithmetic circuits remains. Specifically, from the results of this paper, we provide some prepared notes for cases where an AFHE scheme is available. Although we leave strict definitions and proofs here, it is reasonable to conjecture that, in the non-interactive model of AFHE, computation involving bit-decomposition, modulo reduction, comparison and zero tests will be inefficient and will suffer from a high lower-bound. Therefore, an interesting question becomes the following: how much could the situation be improved if we allow one or a small number of interactions in AFHE-based computation? One potential scheme is to compare shared secrets with ciphertexts of AFHE and to compare the reveal operation in our protocols with the decryption of intermediary ciphertexts (which requires interaction). Then, from a similar idea of the protocols in this paper and given some random ciphertexts prepared in the offline phase, the computation cost should be improved remarkably in the online phase.

References

[1] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *22nd CRYPTO*, pages 417 – 432, 2002.

[2] L. Babai, A. Gál, J. Kollár, L. Rónyai, T. Szabó, and A. Wigderson. Extremal bipartite graphs and superpolynomial lower bounds for monotone span programs. In *Proc. 28th STOC*, pages 603–611, 1996.

[3] E. BACH. A Note on Square Roots in Finite Fields. *IEEE Transactions on Information Theory*, 36(6):1494 – 11498, Nov. 1990.

[4] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *8th ACM Symposium on Principles of Distributed Computing*, pages 201–209, 1989.

[5] C. Bays, K. Ford, R. H. Hudson, and M. Rubinstein. Zeros of Dirichlet L-functions near the real axis and Chebyshev's bias. *Journal of Number Theory*, 87(1):54–76, 2001.

[6] C. Bays and R. H. Hudson. Zeroes of Dirichlet L-Functions and irregularities in the distribution of primes. *Mathematics of Computation*, 69(230):861–866, 1999.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th STOC*, pages 1–10, 1988.

[8] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Multi-party computation goes live. In *Cryptology ePrint Archive, Report*, 2008.

[9] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *Proc. 14th CCS*, pages 486–497, 2007.

[10] R. Canetti. Universally Composable Security: A new paradigm for cryptographic protocols. In *Proc. 42nd FOCS*, pages 136–145, 2001.

[11] A. K. Chandra, S. Fortune, and R. Lipton. Unbounded fan-in circuits and associative functions. In *STOC*, pages 52 – 60, 1983.

[12] D. Chaum, C. Crepéau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th STOC*, pages 11–19, 1988.

[13] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Proc. 3rd TCC*, pages 285–304, 2006.

[14] I. Damgård and G. L. Mikkelsen. Efficient robust and constant-round distributed RSA key generation. In *Proc. 7th TCC*, pages 183–200, 2010.

[15] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, pages 247–264, 2003.

[16] I. Damgård and C. Orlandi. Multiparty computation for dishonest majority: from passive to active security at low cost. In *Proc. 30th CRYPTO*, pages 558–576, 2010.

[17] I. Damgård and R. Thorbek. Non-interactive proofs for integer multiplication. In *Proc. 26th EUROCRYPT*, pages 412–429, 2007.

[18] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. 41st STOC*, pages 169–178, 2009.

[19] R. L. Graham and J. H. Spencer. A constructive solution to a tournament problem. *Canad. Math. Bull.*, 14(1):45–48, 1971.

[20] J. Guajardo, B. Mennink, and B. Schoenmakers. Modulo reduction for paillier encryptions and application to secure statistical analysis (extended abstract). In *Financial Cryptography '10*, volume 6052, pages 375–382, 2010.

[21] S. Halevi and C. Gentry. Implementing gentry's fully-homomorphic encryption scheme. In *Eurocrypt' 11*, volume 6632, pages 129–148, 2011.

[22] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *Proc. 6th TCC*, pages 294–314, 2009.

[23] H. W. Lenstra and C. Pomerance. Primality testing with Gaussian periods. In *preliminary version, http://www.math.dartmouth.edu/ carlp/PDF/complexity12.pdf*, July 2005.

[24] Y. Lindell and B. Pinkas. Privacy-preserving data mining. *Journal of the Cryptology*, 15(3):177–206, 2002.

[25] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of the ACM*, 1(1):59–98, 2009.

[26] C. Ning and Q. Xu. Multiparty computation for modulo reduction without bit-decomposition and a generalization to bit-decomposition. In *Asiacrypt*, page 487, 2010.

[27] T. Nishide and K. Ohta. Multiparty computation for interval, equality and comparison. In *10th PKC*, pages 343–360, 2007.

[28] R. PERALTA. On the distribution of quadratic residues and nonresidues modulo a prime number. *Mathematics of Computation*, 58(197):433–440, 1992.

[29] T. Reistad and T. Toft. Linear, constant-rounds bit-decomposition. In *ICICS*, pages 245–257, 2009.

[30] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math*, 6(1):64–94, 1962.

[31] M. Rubinstein and P. Sarnak. Chebyshev's bias. *Experiment. Math.*, 3(3):173–197, 1994.

[32] W. M. Schmidt. *Equations over finite fields: An elementary approach*. Springer-Verlag, 1976.

[33] A. Shamir. How to Share a Secret. In *CACM, 22*, pages 612–613, 1979.

[34] T. Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In *CT-RSA '09*, pages 357–371, 2009.

A Numerical Data

Definition A.1. For a positive number d, let $\bar{t}(d)$ be the first prime that is qualified for $\pm d$-CQRN.

Definition A.2. For a positive number ℓ, we define $\overline{n}(\ell) = max\{d \mid \lceil \log_2 \bar{t}(d) \rceil = \ell\}$. This notation stands for the maximum effective mapping range $\{-d, ..., d\}$ in $\mathcal{Z}_p$ that we can obtain when p is restricted to $2^{\ell-1} < p < 2^{\ell}$.

We show the growth of $\bar{t}$ and $\overline{n}$ in Tables 2 and 3 respectively and make the following claim.

Claim A.1. $\overline{n}(\ell) \geq 2\ell + 1, \forall \ell \geq 24$

From Lemma 4.1, we have $\overline{n}(\ell) = \Omega(\ell \log \ell)$. This implies that there is a constant C such that for all $\ell \geq C$, $\overline{n}(\ell) \geq 2\ell + 1$. In Table 3, we observe that $\overline{n}(\ell) \geq 2\ell + 1$ is true for all $\ell \geq 24$, and specifically, $\overline{n}(\ell) \geq 3\ell$ when $\ell = 32$. In addition, from the proof of Theorem 4.1, we can expect that $N(2^{\ell}, 2\ell + 1) - N(2^{\ell-1}, 2\ell + 1) >> 1$ when ℓ is sufficiently large, and from Table 3, this number grows to 131 when $\ell = 32$. This implies that this claim is true, although a strict proof would rely on the analysis of the error term of $N(2^{\ell}, 2\ell + 1) - N(2^{\ell-1}, 2\ell + 1)$.

d	$\bar{t}(d)$	d	$\bar{t}(d)$	d	$\bar{t}(d)$	d	$\bar{t}(d)$	d	$\bar{t}(d)$	d	$\bar{t}(d)$
1	3	18	5711	35	366791	52	12537719	69	120293879	86	2929911599
2	7	19	10559	36	366791	53	30706079	70	120293879	87	2929911599
3	23	20	10559	37	366791	54	30706079	71	120293879	88	2929911599
4	23	21	10559	38	366791	55	30706079	72	120293879	89	2929911599
5	71	22	10559	39	366791	56	30706079	73	131486759	90	2929911599
6	71	23	18191	40	366791	57	30706079	74	131486759	91	2929911599
7	311	24	18191	41	366791	58	30706079	75	131486759	92	2929911599
8	311	25	18191	42	366791	59	36415991	76	131486759	93	2929911599
9	311	26	18191	43	366791	60	36415991	77	131486759	94	2929911599
10	311	27	18191	44	366791	61	82636319	78	131486759	95	2929911599
11	479	28	18191	45	366791	62	82636319	79	131486759	96	2929911599
12	479	29	31391	46	366791	63	82636319	80	131486759	97	7979490791
13	1559	30	31391	47	4080359	64	82636319	81	131486759	98	7979490791
14	1559	31	366791	48	4080359	65	82636319	82	131486759	99	7979490791
15	1559	32	366791	49	4080359	66	82636319	83	2929911599	100	7979490791
16	1559	33	366791	50	4080359	67	120293879	84	2929911599		
17	5711	34	366791	51	12537719	68	120293879	85	2929911599		

Table 2: The growth of $\bar{t}(d)$.

ℓ	$\overline{n}(\ell)$	the first instance of $\overline{n}(\ell)$	$N(2^\ell, 2\ell+1) - N(2^{\ell-1}, 2\ell+1)$	ℓ	$\overline{n}(\ell)$	the first instance of $\overline{n}(\ell)$	$N(2^\ell, 2\ell+1) - N(2^{\ell-1}, 2\ell+1)$
1	0	0	0	17	28	95471	0
2	1	3	2	18	30	250799	0
3	2	7	2	19	42	366791	2
4	0	0	0	20	40	701399	0
5	4	23	0	21	42	1579751	0
6	4	47	0	22	46	4080359	2
7	6	71	0	23	46	5154551	0
8	6	191	0	24	52	12537719	3
9	12	479	0	25	58	30706079	9
10	10	719	0	26	60	36415991	5
11	16	1559	0	27	82	131486759	24
12	16	2999	0	28	72	139191191	32
13	18	5711	0	29	70	302794631	37
14	22	10559	0	30	78	546655511	27
15	30	31391	0	31	82	1147846391	83
16	28	35279	0	32	96	2929911599	131

Table 3: The growth of $\overline{n}(\ell)$. 0 indicates that there is no such instance.