A Note on Secure Multiparty Computation via Higher Residue Symbol Techniques*

Ignacio Cascudo† Reto Schnyder‡

March 10, 2020

Abstract

We generalize a protocol by Yu [12] for comparing two integers with relatively small difference in a secure multiparty computation setting. Yu's protocol is based on the Legendre symbol. A prime number p is found for which the Legendre symbol in $\mathbb{F}_p$ agrees with the sign function for integers in a certain range $\{-N, \ldots, N\}$. This can then be computed efficiently.

We generalize this idea to higher residue symbols in cyclotomic rings $\mathbb{Z}[\zeta_r]$ for r a small odd prime. We present a way to determine a prime number p such that the r-th residue symbol $(\cdot \mid p)_r$ agrees with a desired function $f\colon A \to \{\zeta_r^0, \ldots, \zeta_r^{r-1}\}$ on a given small subset $A \subset \mathbb{Z}[\zeta_r]$, when this is possible. We also explain how to efficiently compute the r-th residue symbol in a secret shared setting.

1 Introduction

In secure multiparty computation (MPC), a group of parties, each of which has some secret data, wish to collaboratively compute a function on it without revealing their inputs. A common technique for this is to represent the inputs as elements of a finite field $\mathbb{F}_p$ and represent the function as an arithmetic circuit over that finite field, i.e. a number of sums and products over

*This work is supported by Aalborg University under the SECURE project.

†IMDEA Software Institute, Madrid, Spain. Email: ignacio.cascudo@imdea.org. Much of this work was carried out while Ignacio was with the Department of Mathematics, Aalborg University, Denmark.

‡Department of Mathematics, Aalborg University, Aalborg, Denmark. Email: reto@math.aau.dk

$\mathbb{F}_p$ involving the inputs and public values, and then process this circuit gate-by-gate. For example in secret-sharing based secure multiparty computation protocols, inputs are secret shared among the parties using a linear secret sharing scheme, such as Shamir's scheme [11] or additive secret sharing. In such a scheme, addition of two secrets and multiplication of a secret by a public value can be done by simply performing the operation locally. Multiplication of two secrets can be achieved with a small amount of additional communication between the parties. See for example the book [4] for details about how this plays out in a number of secret-sharing based MPC protocols. But many other operations are more complicated to perform in a secret shared setting, especially those that do not correspond to natural operations in a finite field. Examples for this are size comparison, integer division and modular reduction by values that are coprime to p. Protocols that compute such operations often rely on decomposing a shared value into its binary representation, after which the operation can be performed on secret shared bits [e.g. 5]. However, this decomposition involves a significant computational and communication overhead, so there is much interest in more straightforward protocols for these operations. There are other protocols which don't rely on the decomposition of shared values, but still operate in a bitwise manner by using random pre-decomposed sharings [9].

An alternative idea for comparison of secret values is given in [12]. Their protocol attempts to compute the sign of a secret shared value $[a]_{\mathbb{F}_p}$ by computing the Legendre symbol $(a \mid p)$. For this to work, they must choose the prime modulus p in such a way that all values $\{1, \ldots, N\}$ are quadratic residues modulo p, for a relatively large N, and that -1 is a quadratic non-residue. Then, it holds that $\operatorname{sgn}(a) = (a \mid p)$ for $-N \leq a \leq N$. Two integers a and b can then be compared by computing $\operatorname{sgn}(a - b) = (a - b \mid p)$, assuming $|a - b| \leq N$. The Legendre symbol can be computed relatively easily in a secret-shared setting using a few rounds of precomputation and a single online round. Yu shows that at any given order of magnitude, a prime p can be found which satisfies the desired properties with N being of size $\Omega(\log p)$. The inspiration for this protocol comes from [7], where the idea is first presented for the special case $N = 2$, $p = 7$.

In [1], the authors improve on Yu's method and extend the range where the comparison is valid by a factor of roughly three, for a given modulus size. They achieve this by computing the residue symbol on a small neighbourhood of the input value and performing a majority vote. They also provide another protocol that increases the valid range by a factor of five compared to Yu, but requires an additional online round.

Our contribution. In this paper, we present a generalization of the idea of [12] by using the residue symbol in a cyclotomic ring $\mathbb{Z}[\zeta_r]$, for some odd prime r. The goal is to compute a chosen function $f\colon A \to \{0, \ldots, r-1\}$ in a limited domain $A \subset \mathbb{F}_p[\zeta_r]$. We develop a method for finding a prime number p such that the r-th power residue symbol coincides with f on the domain A, when this is possible:

$$\left(\frac{a}{p}\right)_r = \zeta_r^{f(a)} \text{ for } a \in A.$$

As in [12], our protocol requires an offline precomputation phase, but has an online phase consisting of a single round. As an added benefit, the output of f is obtained in a *one-hot* encoding, which can be helpful if it is e.g. used in a condition for a branching algorithm.

Unfortunately, we are not currently aware of a practical use of our protocol, since the limitations on the size of A are too strict. However, we hope that our new ideas motivate future work that can lessen these restrictions and find applications where our idea outperforms existing solutions.

Outline of the paper. In Section 2, we present basic definitions and facts about the power residue symbol in a cyclotomic ring. We then present the idea of our protocol in Section 3, and in Section 4 we explain in detail how to compute the residue symbol. Then, in Section 5, we present a method for finding an appropriate modulus p given a desired function to compute. Finally, to illustrate our ideas, we give a toy example in Section 6.

2 The Power Residue Symbol

Let r be an odd prime. Let $R = \mathbb{Z}[\zeta_r]$ be the r-th ring of cyclotomic integers, considered as a subring of $\mathbb{C}$. Here, ζ_r is a primitive r-th root of unity. If $\mathfrak{b}$ is an ideal of R, we denote by $N\mathfrak{b}$ its norm.

Definition 1 (See [8, Prop. 14.2.1])**.** Let $\mathfrak{p}$ be a nonzero prime ideal of R such that $r \notin \mathfrak{p}$, and let $a \in R \setminus \mathfrak{p}$. Then, there exists an integer s, unique modulo r, such that

$$\zeta_r^s \equiv a^{\frac{N\mathfrak{p}-1}{r}} \pmod{\mathfrak{p}}.$$

We define the r-th *power residue symbol* as

$$\left(\frac{a}{\mathfrak{p}}\right)_r = (a \mid \mathfrak{p})_r = \zeta_r^s.$$

If $a \in \mathfrak{p}$, we define $(a \mid \mathfrak{p})_r = 0$. It holds that a is an r-th power modulo $\mathfrak{p}$ if and only if $(a \mid \mathfrak{p})_r = 1$. Clearly, the power residue symbol is multiplicative in the first argument. That is, for $a, b \in R$, we have

$$\left(\frac{ab}{\mathfrak{p}}\right)_r = \left(\frac{a}{\mathfrak{p}}\right)_r \left(\frac{b}{\mathfrak{p}}\right)_r.$$

If $\mathfrak{b}$ is any proper ideal of R not containing r, we extend the power residue symbol multiplicatively. That is, if $\mathfrak{b}$ factors into prime ideals as $\mathfrak{b} = \mathfrak{p}_1 \cdots \mathfrak{p}_s$, then

$$\left(\frac{a}{\mathfrak{b}}\right)_r = \prod_{i=1}^{s} \left(\frac{a}{\mathfrak{p}_i}\right)_r.$$

Finally, if $b \in R$ is any element coprime to r, we simply define

$$\left(\frac{a}{b}\right)_r = \left(\frac{a}{(b)}\right)_r,$$

where (b) is the ideal generated by b.

Definition 2 (See [8, p. 206])**.** An element $a \in R$ is called *primary* if it is coprime to r, not a unit and congruent to a rational integer modulo $(1-\zeta_r)^2$.

If $a \in R$ is coprime to r and not a unit, then there is a unique $k \in \{0, \ldots, r-1\}$ such that $\zeta_r^k a$ is primary.

Theorem 3 (Eisenstein Reciprocity, [8, Theorem 1, p. 207])**.** *Let $b \in \mathbb{Z}$ be coprime to r and not a unit. Let $a \in R$ be primary and coprime to b. Then,*

$$\left(\frac{a}{b}\right)_r = \left(\frac{b}{a}\right)_r.$$

Theorem 4 (See [3, Theorem 4.9])**.** *The group of units R^* is the direct product of the group of roots of unity in R and the group of positive real units $(R \cap \mathbb{R}_{>0})^*$.*

Lemma 5. *Let $a, b \in R$, such that b is coprime to r and a is coprime to b. Suppose that each of a and b is either real or purely imaginary. Then, $(a \mid b)_r = 1$.*

Proof. First, note that for any prime ideal $\mathfrak{p}$ not containing r, we have

$$\left(\frac{-1}{\mathfrak{p}}\right)_r \equiv (-1)^{\frac{N\mathfrak{p}-1}{r}} \pmod{\mathfrak{p}},$$

which is 1 if $\mathfrak{p}$ lies above an odd prime number, and -1 if it lies above 2. But in the second case, $-1 \equiv 1 \pmod{\mathfrak{p}}$, so either way, $(-1 \mid \mathfrak{p})_r = 1$.

Let now $c \in R \setminus \mathfrak{p}$, and let $(c \mid \mathfrak{p})_r = \zeta_r^s$. We have

$$\begin{aligned} c^{\frac{N\mathfrak{p}-1}{r}} &= \zeta_r^s + k \quad \text{with } k \in \mathfrak{p}, \\ \overline{c}^{\frac{N\mathfrak{p}-1}{r}} &= \zeta_r^{-s} + \overline{k} \quad \text{with } \overline{k} \in \overline{\mathfrak{p}}, \end{aligned}$$

where $\bar{\cdot}$ denotes complex conjugation, and we see that $(\overline{c} \mid \overline{\mathfrak{p}})_r = \overline{(c \mid \mathfrak{p})_r}$. By multiplicativity, these two properties apply also if we replace the lower argument $\mathfrak{p}$ by any $b \in R$ coprime to r.

Hence, for a and b as in the statement of the lemma, we get

$$\overline{\left(\frac{a}{b}\right)_r} = \left(\frac{\overline{a}}{\overline{b}}\right)_r = \left(\frac{\pm a}{\pm b}\right)_r = \left(\frac{a}{b}\right)_r,$$

which therefore has to be 1. □

3 The Basic Idea

The basic idea of our protocol is the following. Suppose we are given a function $f\colon A \to \{0, \ldots, r-1\}$, where A is a subset of R. We hope to find a prime number p such that the function f corresponds to the r-th power residue symbol with lower argument p on A. That is,

$$\left(\frac{a}{p}\right)_r = \zeta_r^{f(a)} \text{ for } a \in A. \tag{1}$$

Then, we can securely compute $f(a)$ by computing the residue symbol, which can hopefully be done more efficiently.

We will see in Section 5 how we can find such a prime p. Note however, that condition (1) often contains internal contradictions or other impossible requirements. It is then necessary to adapt A and f to resolve these.

Remark 6. In practice, to avoid the aforementioned internal contradictions in our requirements, the actual input values will first be encoded via some (ideally linear) function mapping to A. One can try many such encodings until one is found that is not contradictory. We will see an example of this in Section 6.

Let us note some initial requirements on the prime p. First, we want that p does not split in R, so that (p) is a prime ideal. We need this to be the

case so that we can use basic facts about $(a \mid p)_r$, and so that $F = R/(p)$ is a finite field of size p^{r-1}. For reasons we will see later, we also wish for the value $(\zeta_r \mid p)_r$ to be ζ_r. It would be possible to fix it to another primitive root of unity, but that would complicate our analysis. We will show in Section 5 how to achieve this.

For our protocol, the parties will share their values as elements of the finite fields $\mathbb{F}_p$ and $F = R/(p)$. Note that a sharing of an element in F can be considered as a sharing of $r-1$ elements of $\mathbb{F}_p$ via the basis $\{1, \zeta_r, \ldots, \zeta_r^{r-2}\}$. Either additive sharing or Shamir sharing can be used for this. In the case of Shamir sharing, we require that the evaluation point of each party lies in the base field $\mathbb{F}_p$.

Since the residue symbol $(a \mid p)_r$ depends only on a modulo p, it is now possible to evaluate the function f at $a \in A$ by instead evaluating $(a \mid p)_r$. We will see in Section 4 how to do this efficiently.

Remark 7. In the above, instead of getting a sharing of the result $f(a) \in \{0, \ldots, r-1\}$ directly, we end up with the root of unity $\zeta_r^{f(a)} \in F$. If we decompose $(a \mid p)_r = a_0 + a_1\zeta_r + \cdots + a_{r-2}\zeta_r^{r-2}$ with $a_i \in \mathbb{F}_q$, this almost results in a one-hot encoding of $f(a)$. The only difference is that $r-1$ is represented by $a_i = -1$ for all i. A proper one-hot encoding can be computed locally by setting $b_{r-1} = (1 - a_0 - \cdots - a_{r-2})/r$ and $b_i = a_i + b_{r-1}$ for $i < r-1$. If we prefer to share the result as a single value, we can then easily compute that as $0b_0 + 1b_1 + \cdots + (r-1)b_{r-1}$.

4 Secure Computation of the Residue Symbol

We can compute the power residue symbol in a secret shared setting analogously to how the Legendre symbol is computed in [12]. We note that only the upper argument of the residue symbol is secret shared, whereas the lower argument p is public. Let $P_1, \ldots, P_n$ be the parties involved in the protocol. We assume that the parties are semi-honest (also known as passive), i.e. they will follow the specified instructions from the protocol. Many techniques are known to upgrade such protocols to settings where parties may deviate from the protocol (called malicious or active). Possible collusions of cheating parties are modeled by means of an adversary that corrupts those parties and sees all the information they know.

We consider a linear secret sharing scheme that allows to distribute a secret value among the n parties, by giving each one a share, so that an adversary corrupting up to some number t of parties can obtain no information

about the secret given those shares.

Therefore, any party can secret share an element of her choice $x \in F$, thereby creating a sharing $[x]_F$. As mentioned in the introduction, given $[x]_F, [y]_F$, parties can compute shares for $x+y$, i.e. $[x+y]_F$ without interaction (so these operations are considered essentially "free"). We denote this $[x+y]_F = [x]_F + [y]_F$. Likewise, given a public value $a \in F$ and $[x]_F$, parties can compute without interaction $[ax]_F = a[x]_F$. Parties can also compute shares for the product $[xy]_F$ (which we denote $[xy]_F = [x]_F[y]_F$), although this requires some additional communication, that depends on the specific protocol we use. Parties can open a jointly shared value $[x]_F$, so that every party learns x. As explained below, parties can jointly create a sharing $[r]_F$ of an element r that in the view of any party (and even in the joint view of any subset of $n-1$ parties) is uniformly random in F. Given $[x]_F, [r]_F$, where r is uniformly random in the sense above, opening $[x+r]_F$ gives no additional information about x apart from the a priori knowledge parties might have about x. If parties open $[xr]_F$ and find out that $xr \neq 0$, the only new information parties may learn about x is that $x \neq 0$, but all other information about x is protected.

Computing random elements of F. The parties can compute a uniformly random element of F in the usual way, described in Algorithm 1. Each party P_i chooses and shares a uniformly random value $[x_i]_F$. These values are then summed up: $[x]_F = \sum_{i=0}^{n} [x_i]_F$. In this way, even with an active adversary, the value x is uniformly random and secret if at least one party is honest.

Computing a random solved instance. In a preprocessing phase, the parties need to compute a random solved instance of the power residue symbol. That is, they want a pair of shared values $([x]_F, [x']_F)$, where $x' = (x \mid p)_r$ and x is uniformly random and unknown to the parties. To do this, we proceed as in [12], and described in Algorithm 2. The parties first select two uniform random shared values $[a]_F$ and $[b]_F$ and multiply them: $[d]_F = [a]_F[b]_F$. They then compute and open $f = d^r$. If this is zero, they abort. Otherwise, they know that a and d are uniformly random and independent elements of F^*, since F is a finite field. The parties then compute an r-th root $\hat{d}$ of f in the clear. We see that $d/\hat{d}$ is a uniformly random r-th root of unity. Hence,

$$[x]_F = \frac{[a]_F^r [d]_F}{\hat{d}}, \quad [x']_F = \left[\left(\frac{x}{p} \right)_r \right]_F = \frac{[d]_F}{\hat{d}}$$

constitute a uniformly random solved instance. Recall that we require $(\zeta_r \mid p)_r = \zeta_r$, so that $(d/\hat{d} \mid p)_r = d/\hat{d}$.

Computing the residue symbol. In the online phase, described in Algorithm 3 the parties then wish to compute the residue symbol of a secret shared value $[a]_F$. We assume that a is known to be nonzero. This is achieved by using a suitable encoding, as in Remark 6. Suppose we have a fresh random solved instance $([x]_F, [x']_F)$. The parties compute and open ax, which is uniformly random in F^* and hence doesn't reveal any information about a. They can then compute the residue symbol $(ax \mid p)_r$ in the clear, and finally obtain

$$\left[\left(\frac{a}{p}\right)_r\right]_F = \left(\frac{ax}{p}\right)_r \cdot [x']_F^{-1}.$$

Remark 8. Note that since x' is an r-th root of unity, computing its inverse is the same as computing its complex conjugate. This can be done locally on each party's share for additive sharing as well as Shamir sharing (if the evaluation points lie in the base field $\mathbb{F}_p$).

Computational cost. An invocation of Algorithm 1 costs n sharings, which can be done in a single round. Here, n is the number of parties.

The offline phase (Algorithm 2) costs 2 invocations of Algorithm 1, as well as 2 multiplications, 2 exponentiations to the power of r, and 1 opening. The round complexity can be minimized as follows: Both random values $[a]_F$ and $[b]_F$ can be generated in parallel in a single round. Then, $[d]_F = [a]_F[b]_F$, $[d]_F^r = [a]_F^r[b]_F^r$ and $[a]_F^r[d]_F = [a]_F^{r+1}[b]_F$ can be computed, and $[d]_F^r$ opened, in three rounds using an unbounded fan-in multiplication protocol [e.g. 2]. This gives a total round complexity of 4 rounds.

The online phase costs a single multiplication and one opening, which can be done in a single round. Recall from Remark 8 that computing the inverse of a root of unity can be done locally.

It should be noted that multiplication of elements in F is more expensive than multiplication in $\mathbb{F}_q$, a naïve implementation taking $O(r^2)$ multiplications in the base field, which however can be done in parallel in a single round. Also, unbounded fan-in multiplication precludes the use of square-and-multiply methods for computing exponentiation. However, since the exponent r is typically very small, that is not necessary.

Algorithm 1 Choosing a random element of F.

Each party P_i selects and shares a uniformly random $[x_i]_F \in F$
$[x]_F \leftarrow \sum_{i=1}^n [x_i]_F$
return $[x]_F$

Algorithm 2 Offline phase: Find a random solved instance $([x]_F, [x']_F)$ of the residue symbol, i.e. $x' = (x \mid p)_r$.

$[a]_F \leftarrow_R F$
$[b]_F \leftarrow_R F$
$[d]_F \leftarrow [a]_F[b]_F$
$f \leftarrow [d]_F^r$
if $f = 0$ **then**
  abort
end if
$\hat{d} \leftarrow \sqrt[r]{f}$
$[x']_F \leftarrow [d]_F/\hat{d}$
$[x]_F \leftarrow [a]_F^r[x']_F$
return $([x]_F, [x']_F)$

Algorithm 3 Online phase: Compute the residue symbol of $[a]_F$, given a solved instance $([x]_F, [x']_F)$.

$b \leftarrow [a]_F[x]_F$
$c \leftarrow (b \mid p)_r$
return $c \cdot [x']_F^{-1}$

5 Finding the Modulus

We are looking for a suitable prime modulus p. Recall from Section 3 that we want (p) to be a prime ideal of R. This is equivalent to p being a generator of the multiplicative group $\mathbb{F}_r^*$, by [8, Theorem 2, p. 196]. Hence, this condition depends only on p modulo r. Furthermore, we wish to fix $(\zeta_r \mid p)_r = \zeta_r$. Since

$$\left(\frac{\zeta_r}{p}\right)_r = \zeta_r^{\frac{p^{r-1}-1}{r}},$$

by definition, this is equivalent to having $p^{r-1} \equiv r + 1 \pmod{r^2}$. This gives us the first equation for p: let

$$M_0 = \{q \in \mathbb{Z}/r^2\mathbb{Z} \mid q \text{ is a generator of } \mathbb{F}_r^* \text{ and } q^{r-1} = r + 1\}.$$

Then, we require that $p \in M_0 \pmod{r^2}$.

We now wish to impose a condition of the form $(a \mid p)_r = \zeta_r^\ell$ for some $a \in R \setminus \{0\}$ and $\ell \in \mathbb{Z}$. We need to distinguish multiple cases.

Case 1: a is a unit. By Theorem 4, we can write $a = \pm\zeta_r^k u$, where $k \in \mathbb{Z}$ and u is a positive real unit. Applying Lemma 5, we see that

$$\left(\frac{a}{p}\right)_r = \left(\frac{\pm u}{p}\right)_r \left(\frac{\zeta_r^k}{p}\right)_r = \zeta_r^k,$$

which is independent of p. Requirements of this form are hence satisfied either for all primes under consideration or for none. If it is never satisfied, the requirements need to be adjusted.

Case 2: a is coprime to r and not a unit. There is some $k \in \mathbb{Z}$ such that $\hat{a} = \zeta_r^k a$ is primary. So we want that p is coprime to a and

$$\zeta_r^\ell \stackrel{!}{=} \left(\frac{a}{p}\right)_r = \left(\frac{\zeta_r}{p}\right)_r^{-k} \left(\frac{\hat{a}}{p}\right)_r = \zeta_r^{-k} \left(\frac{p}{\hat{a}}\right)_r = \zeta_r^{-k} \left(\frac{p \bmod N(\hat{a})}{\hat{a}}\right)_r,$$

the last equation holds because the value of the residue symbol $(p \mid \hat{a})_r$ depends only on p modulo $\hat{a}$. This gives us another modular equation for p: let

$$M_a = \left\{ q \in \mathbb{Z}/N(\hat{a})\mathbb{Z} \;\middle|\; \left(\frac{q}{\hat{a}}\right)_r = \zeta_r^{\ell+k} \right\}.$$

Then, we require that $p \in M_a \pmod{N(\hat{a})}$.

Case 3: a is not coprime to r. Let first $\mu = 1 - \zeta_r^2$, which is a prime element of R. Note that

$$\left(\frac{\mu}{p}\right)_r = \left(\frac{\zeta_r}{p}\right)_r \left(\frac{\zeta_r^{-1} - \zeta_r}{p}\right)_r = \zeta_r$$

by Lemma 5, since $\zeta_r^{-1} - \zeta_r$ is purely imaginary. The ideal (r) factors as $(\mu)^{r-1}$ in R, so we can write $a = \mu^m \tilde{a}$ for some $m > 0$ and $\tilde{a}$ coprime to r. (m is the valuation of a at (μ).) We proceed as before with $\tilde{a}$ instead of a. If $\tilde{a}$ is a unit, then as in Case 1, the requirement is either always satisfied or never. If $\tilde{a}$ is not a unit, let $k \in \mathbb{Z}$ such that $\hat{a} = \zeta_r^k \tilde{a}$ is primary. We end up with the set

$$M_a = \left\{ q \in \mathbb{Z}/N(\hat{a})\mathbb{Z} \;\middle|\; \left(\frac{q}{\hat{a}}\right)_r = \zeta_r^{\ell+k-m} \right\},$$

and we require that $p \in M_a \pmod{N(\hat{a})}$.

5.1 Computing the Conditions

How do we find the elements of the set M_a? First, we note that M_a is contained in $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$. Since it appears inevitable that our procedure takes at least polynomial time in $N(\hat{a})$, the brute force method of simply computing $(q \mid \hat{a})_r$ for each $q \in (\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$ seems viable. However, computation of the residue symbol is relatively expensive in practice, so we use a method that requires only few invocations of the residue symbol.

Since the residue symbol is multiplicative, the set

$$H_a = \left\{ q \in (\mathbb{Z}/N(\hat{a})\mathbb{Z})^* \;\middle|\; \left(\frac{q}{\hat{a}}\right)_r = 1 \right\}$$

is a subgroup of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$. If H_a is the entirety of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$, the condition $p \in M_a \pmod{N(\hat{a})}$ is satisfied for all values or no value of p. In the latter case, we need to adjust our parameters.

If however H_a is a proper subgroup, it follows that it has index r, and that the set M_a is a coset of H_a. We use the fact that the index is known to efficiently find a set of generators for H_a, after which the entirety of M_a can easily be computed.

For this, we first need to find a set of generators $g_1, \ldots, g_n$ of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$, of orders $k_1, \ldots, k_n$, which induce an isomorphism

$$\mathbb{Z}/k_1\mathbb{Z} \times \cdots \times \mathbb{Z}/k_n\mathbb{Z} \xrightarrow{\sim} (\mathbb{Z}/N(\hat{a})\mathbb{Z})^*.$$

The Sage computer algebra system [10] contains a function that efficiently provides these generators.

We describe the algorithm for computing a set of generators for the subgroup H_a in the following setting, which is related to the hidden subgroup problem.

Finding a subgroup with known index. Suppose we are given an abelian group of the form $G = \mathbb{Z}/k_1\mathbb{Z} \times \cdots \times \mathbb{Z}/k_n\mathbb{Z}$, where $k_1, \ldots, k_n \in \mathbb{Z}_{>1}$. Suppose furthermore that we have access to the characteristic function $\chi\colon G \to \{0,1\}$ of a subgroup $H \subseteq G$ of index dividing r, where r is a known prime number. That is, $\chi(x) = 1$ if and only if $x \in H$. The goal is to find a set of generators of H, using only a small number of invocations of χ.

First, we note that by pulling H back along the projection homomorphism $\pi\colon \mathbb{Z}^n \to G$, we get a lattice $\tilde{H}$ containing $k_1\mathbb{Z} \times \cdots \times k_n\mathbb{Z}$. Let $\tilde{\chi} = \chi \circ \pi$ be the characteristic function of $\tilde{H}$. We now find a basis for $\tilde{H}$, which maps to a set of generators of H. Every lattice contained in $\mathbb{Z}^n$ has a unique basis

given by the columns of a full rank $n \times n$ integer matrix B in Hermite normal form [6]. That is, $B = (B_{ij})$ satisfies

$$\begin{aligned} B_{ij} &= 0 \text{ for } 1 \leq i < j \leq n, \\ B_{ii} &> 0 \text{ for } 1 \leq i \leq n, \\ 0 \leq B_{ij} &< B_{ii} \text{ for } 1 \leq j < i \leq n. \end{aligned}$$

The determinant of B is equal to the index of the subgroup H in G, and hence divides the prime r by assumption. This means that B has at most one diagonal entry equal to r, with all others being 1.

For example, the matrix B might look like this:

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a_1 & a_2 & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

with $0 \leq a_1, a_2 < r$.

To find the basis $h_1, \ldots, h_n$ of $\tilde{H}$, we now proceed as in Algorithm 4. Let $e_1, \ldots, e_n$ be the standard basis vectors of $\mathbb{Z}^n$. We let i decrease from n to 1, and so go through the columns of B from right to left.

1. For as long as $\tilde{\chi}(e_i) = 1$, we simply set $h_i = e_i$.
2. If $\tilde{\chi}(e_i) = 0$, we know that we have reached the column with r in the diagonal, so we set $h_i = re_i$ and fix $J = i$.
3. For each of the remaining values of i, we search for the unique $a \in \{0, \ldots, r-1\}$ such that $\tilde{\chi}(e_i + ae_J) = 1$, and set $h_i = e_i + ae_J$.

This way, we can compute the basis of $\tilde{H}$ using at most $(n-1)r + 1$ invocations of χ. Note that if the index of H is 1, the algorithm simply returns the original basis $e_1, \ldots, e_n$.

Remark 9. In the case relevant to this paper, χ is given by the residue symbol, which not only tells us whether an element is in H, but in which coset of H it lies. In this case, the value a in step 3 above can be computed with a single invocation of χ, which reduces the total number of invocations needed to just n.

Remark 10. The algorithm can easily be generalized to the case where r is not prime, in which case there may be multiple diagonal entries not equal to 1. It requires at most nr invocations of χ.

Algorithm 4 Computing a basis of a sublattice $\tilde{H} \subseteq \mathbb{Z}^n$ of prime index r, given the characteristic function $\tilde{\chi}$ of $\tilde{H}$.

```
J ← 0
for i from n to 1 do
    if J = 0 then
        if χ̃(e_i) = 1 then
            h_i ← e_i
        else
            h_i ← r · e_i
            J ← i
        end if
    else
        for a from 0 to r − 1 do
            if χ̃(e_i + a e_J) = 1 then
                h_i ← e_i + a e_J
            end if
        end for
    end if
end for
```

6 Toy Example

We present an example, in which we compute reduction modulo 3 for integers $x \in \{0, \ldots, n\}$ for some small n. We pick $r = 3$. This example was constructed with the help of the Sage computer algebra system [10].

Setting $A = \{0, \ldots, n\}$ and $f(x) = x \bmod 3$ cannot work for this, since $(x \mid p)_r = 1$ for all valid primes p and $x \in \mathbb{Z}$, $p \nmid x$, by Lemma 5. Instead, we encode the problem as follows. Let $n = 18$, and

$$A = \{11 + x\zeta_r \mid 0 \leq x \leq 18\}$$
$$f(11 + x\zeta_r) = x \bmod 3.$$

This encoding was found by trial and error. Then, following our procedure from Section 5, we get

$$M_0 = \{q \in \mathbb{Z}/9\mathbb{Z} \mid q \text{ is a generator of } \mathbb{F}_3^* \text{ and } q^2 \equiv 4\} = \{2\}.$$

For e.g. $a = 11 + 5\zeta_r$, we have $f(a) = \ell = 2$, so we want $(a \mid p)_r = \zeta_r^2$. We have $N(a) = 91$, so a is coprime to 3. Furthermore, $\hat{a} = \zeta_r a = 6\zeta_r - 5$ is

primary, so $k = 1$. We hence get

$$\begin{aligned} M_a &= \left\{ q \in \mathbb{Z}/91\mathbb{Z} \;\middle|\; \left(\frac{q}{\hat{a}}\right)_r = \zeta_r^{\ell+k} = \zeta_r^3 = 1 \right\} \\ &= \{1, 2, 4, 8, 16, 17, 23, 27, 32, 34, 37, 45, 46, \\ &\quad 54, 57, 59, 64, 68, 74, 75, 83, 87, 89, 90\}. \end{aligned}$$

Similarly, we find M_a for all other $a \in A$.

Finally, we use brute force to find a prime p which lies in M_0 and in each M_a after the appropriate modular reduction. The smallest one is

$$p = 26\,403\,527.$$

We conclude that

$$\left(\frac{11 + x\zeta_r}{p}\right)_r = \zeta_r^{x \bmod 3}$$

for $0 \leq x \leq 18$.

One can show that it is not possible to extend the example above to $a = 11 + 19\zeta_r$.

7 Conclusion

We have introduced a protocol for secure multiparty computation which allows the evaluation of certain desired functions $f\colon A \to \{0, \ldots, r-1\}$ on secret shared values, for a small subset $A \subset \mathbb{Z}[\zeta_r]$ of the r-th cyclotomic ring, where r is a small prime. Our protocol is a generalization of a protocol by Yu [12], and makes use of the residue symbol of $\mathbb{Z}[\zeta_r]$, by getting it to agree with the desired function on A. It uses only a single round in the online phase, and a constant number of offline preprocessing rounds.

We can then use this idea to compute a function g over a more "natural" domain, like $A' \subset \mathbb{Z}$, by first encoding it as a function $f\colon A \to \{0, \ldots, r-1\}$ in a suitable way. As we have shown in the example, there may be different ways of doing this encoding, and the feasibility and performance of our technique may depend on the chosen encoding.

It is an open question to find concrete applications where our protocol has significant advantages over alternative solutions, such as polynomial interpolation. While our method requires fewer online rounds than polynomial interpolation, it is also more restrictive in which functions and domains it allows, and requires the use of a specific prime modulus p. It is therefore also of interest to improve and formalize the methods of encoding a desired function in such a way as to be compatible with the residue symbol, and so

that p remains reasonably small, which so far we have mostly done by trial and error.

References

[1] Mark Abspoel et al. "Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks". In: *Cryptographers' Track at the RSA Conference*. Springer. 2019, pp. 453–472.

[2] Judit Bar-Ilan and Donald Beaver. "Non-cryptographic fault-tolerant computing in constant number of rounds of interaction". In: *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*. ACM. 1989, pp. 201–209.

[3] Yuri F. Bilu, Yann Bugeaud, and Maurice Mignotte. *The problem of Catalan*. Vol. 9. Springer, 2014.

[4] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation and secret sharing*. Cambridge University Press, 2015.

[5] Ivan Damgård et al. "Unconditionally secure constant-rounds multiparty computation for equality, comparison, bits and exponentiation". In: *Theory of Cryptography Conference*. Springer. 2006, pp. 285–304.

[6] Cynthia Dwork. "Lattices and their application to cryptography". In: *Lecture Notes, Stanford University* (1998).

[7] Uri Feige, Joe Killian, and Moni Naor. "A minimal model for secure computation". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 554–563.

[8] Kenneth Ireland and Michael Rosen. *A classical introduction to modern number theory*. Vol. 84. Springer Science & Business Media, 1990.

[9] Takashi Nishide and Kazuo Ohta. "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol". In: *International Workshop on Public Key Cryptography*. Springer. 2007, pp. 343–360.

[10] William A. Stein et al. *Sage Mathematics Software (Version 7.5.1)*. The Sage Development Team. 2017. URL: `http://www.sagemath.org`.

[11] Adi Shamir. "How to share a secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[12] Ching-Hua Yu. "Sign Modules in Secure Arithmetic Circuits." In: *IACR Cryptology ePrint Archive* (2011).