Research Article

Ignacio Cascudo and Reto Schnyder*

A note on secure multiparty computation via higher residue symbols

https://doi.org/10.1515/jmc-2020-0013
Received Mar 17, 2020; accepted Dec 07, 2020

Abstract: We generalize a protocol by Yu 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 $(\cdot \mid p)$ agrees with the sign function for integers in a certain range $\{-N, \dots, N\} \subset \mathbb{Z}$. 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 $f\colon A \to \{\zeta_r^0, \dots, \zeta_r^{r-1}\}$ 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.

Keywords: secure multiparty computation, cyclotomic rings, power residue symbol

2010 Mathematics Subject Classification: 11T71, 94A60

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. as an expression consisting of nested sums and products over $\mathbb{F}_p$ 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 [10] 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 integer 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 cost, so there is much interest in more straightforward protocols for these operations. There are other protocols which do not rely on the decomposition of shared values, but still operate in a bitwise manner by using random pre-decomposed sharings [9].

Ignacio Cascudo: IMDEA Software Institute, 28223 Pozuelo de Alarcon, Madrid, Spain; Email: ignacio.cascudo@imdea.org. Much of this work was carried out while Ignacio was with the Department of Mathematical Sciences, Aalborg University, Denmark.
***Corresponding Author: Reto Schnyder:** Department of Mathematical Sciences, Aalborg University, 9220 Aalborg, Denmark; Email: reto@math.aau.dk

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 $\{1, \dots, N\}$ 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 $-N \le a \le N$. Two integers a and b can then be compared by computing $\operatorname{sgn}(a-b) = (a - b \mid p)$, assuming $|a-b| \le N$. 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 $f\colon A \to \{0, \dots, r-1\}$ 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:

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

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. We give a toy example in Section 6 to illustrate our ideas. Finally, in Section 7, we compare our method to other constant-round protocols for computing arbitrary functions.

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 coprime to 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 non-unit 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 $k \in \{0, \dots, r-1\}$ 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 a non-unit coprime to r and a. 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

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

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 $f\colon A \to \{0, \dots, r-1\}$, 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,

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

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 affine) function mapping to A. One can try many such encodings until one is found that is not contradictory. We will give considerations on the existence of good encodings in Section 5.3, and we will see an example in Section 6.

Let us note some initial requirements on the prime p. First, we want that (p) is a prime ideal of R, which is the case whenever p is different from r and does not split in R. 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.

In our protocol, we will be using the fields $\mathbb{F}_p$ and $F = R/(p)$. Note that an element in F can be represented by $r-1$ elements of $\mathbb{F}_p$ via the basis $\{1, \zeta_r, \dots, \zeta_r^{r-2}\}$. So given a linear secret sharing scheme over $\mathbb{F}_p$, we can consider a sharing of an element in F as a sharing of $r-1$ elements of $\mathbb{F}_p$. See Section 4 for details.

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 $f(a) \in \{0, \dots, r-1\}$ directly, we end up with the root of unity $\zeta_r^{f(a)} \in F$. If we decompose $(a \mid p)_r = a_0 + a_1\zeta_r + \dots + a_{r-2}\zeta_r^{r-2}$ with $a_i \in \mathbb{F}_p$, 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 $b_{r-1} = (1 - a_0 - \dots - a_{r-2})/r$ 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 $0b_0 + 1b_1 + \dots + (r-1)b_{r-1}$.

4 Secure computation of the residue symbol

We will show how to compute the power residue symbol based on any arithmetic black-box protocol over the field $\mathbb{F}_p$, for example based on secret sharing. This is analogous to how the Legendre symbol is computed in [12]. We note that only the upper argument of the residue symbol is secret, whereas the lower argument p is public. In order to explain how we do this, we first need to detail the model in which we will work.

Secure computation model

We consider the usual situation in secure multiparty computation: n parties $P_1, \dots, P_n$ want to jointly compute the output of certain (public) function on private inputs which belong to some of the parties, by means of some protocol in which the parties can communicate over point-to-point secure channels between each pair of parties.

We will consider a static, semi-honest adversary, that corrupts some subset of parties at the onset of the protocol, and gets to see all communication received by these parties, but cannot make them deviate from the protocol.[1] Security in this case means the adversary cannot infer more information about non-corrupted parties' inputs and outputs than what is already implied by the corrupted parties' inputs and outputs.

Our techniques work in fact on top of any secure computation protocol for arithmetic operations over $\mathbb{F}_p$. This means that any given party can create an encoding $[x]_{\mathbb{F}_p}$ of a value $x \in \mathbb{F}_p$ known to that party, so that: the adversary cannot obtain any knowledge about x from $[x]_{\mathbb{F}_p}$; there is a protocol that allows the set of all parties to recover $x \in \mathbb{F}_p$ from $[x]_{\mathbb{F}_p}$; and given sharings of elements $x, y \in \mathbb{F}_p$ (where x, y may not be known by the same party, in fact they may not be known by anybody) there are secure computation subprotocols that allow parties to compute encodings for $x + y$, xy and ax for public $a \in \mathbb{F}_p$ (denoted respectively $[x + y]_{\mathbb{F}_p} = [x]_{\mathbb{F}_p} + [y]_{\mathbb{F}_p}$, $[xy]_{\mathbb{F}_p} = [x]_{\mathbb{F}_p}[y]_{\mathbb{F}_p}$, $[ax]_{\mathbb{F}_p} = a[x]_{\mathbb{F}_p}$).

The customary example for this are secret-sharing based secure computation protocols, where the encoding $[\cdot]_{\mathbb{F}_p}$ is given by a secret sharing scheme. This scheme is furthermore often linear, which allows computing $[x + y]_{\mathbb{F}_p}$ (resp. $[ax]_{\mathbb{F}_p}$) given $[x]_{\mathbb{F}_p}$, $[y]_{\mathbb{F}_p}$ (resp. a and $[x]_{\mathbb{F}_p}$) by having each party operate locally on their shares, i.e., it requires no interaction among parties.

There exist many secret-sharing based arithmetic protocols which are secure against different types of adversaries, depending on which sets of parties the adversary can corrupt (the adversary structure) and whether we assume a restriction on the computational capabilities of the adversary or not (where we speak respectively of computational or information-theoretical security). For example, one can construct arithmetic black-box protocols which are information-theoretically secure against adversaries that can corrupt any set of t parties where $t < n/2$, and protocols which are computationally secure against adversaries corrupting any set of $n - 1$ parties.

Our protocols will inherit the security properties from the arithmetic black-box protocol over $\mathbb{F}_p$, with respect to the adversary structure and adversary computational capabilities it tolerates.

Preprocessing

A usual resource in secure multiparty computation protocols is preprocessing: the fact that parties can initiate and typically carry out the heavy part of the secure computation protocol even before the (encoded) inputs are provided. That is, we can split the computation in two phases: a preprocessing, or offline phase, which is the phase carried out before the inputs are given and where the parties compute encodings of random correlated information; and an online phase, which uses the input and is typically much lighter. We will use this approach in our protocols too.

From encodings over $\mathbb{F}_p$ to encodings over F

As mentioned before, an encoding (e.g. a sharing) $[x]_F$ of an element in F will simply be the encoding in $\mathbb{F}_p$ of each of its coordinates $x_i \in \mathbb{F}_p$ with respect to the basis $\{1, \zeta_r, \ldots, \zeta_r^{r-2}\}$, i.e. $([x_0]_{\mathbb{F}_p}, [x_1]_{\mathbb{F}_p}, \ldots, [x_{r-2}]_{\mathbb{F}_p})$. Both additions and products of F-encodings can be obtained by operating on the $\mathbb{F}_p$-encodings of their coordinates. Specifically, additions of two F-encodings require $r - 1$ additions of $\mathbb{F}_p$-encodings (by just adding coordinate-wise the vectors of $\mathbb{F}_p$-encodings), so in the case of a linear secret sharing scheme it can be done locally by the shareholders. Multiplying by a public constant in R can also be done locally, but it may involve up to $O(r^2)$ additions and multiplications of $\mathbb{F}_p$-encodings by $\mathbb{F}_p$-constants (depending on the public constant). Finally, products of two F-encodings require $O(r^2)$ products and $O(r^2)$ additions of $\mathbb{F}_p$-encodings.

Inversions of roots of unity

Our algorithms will also need to invert F-encodings of r-th roots of unity in R (seen as elements in F), that is the elements ζ_r^i, $i = 0, \ldots, r - 1$. Note that $(\zeta_r^i)^{-1} = \zeta_r^{r-i}$. Moreover, in the basis $\{1, \zeta_r, \ldots, \zeta_r^{r-2}\}$, ζ_r^i is given by

1 We note that many techniques are known to upgrade protocols which are secure against semi-honest adversaries to settings where corrupted parties may deviate from the protocol (where the adversary is called malicious or active).

the i-th unit vector (numbering from 0 to $r-2$) if $i < r-1$, and by the vector $(-1,-1,\ldots,-1)$ if $i = r-1$, since $\zeta_r^{r-1} = -1 - \zeta_r - \cdots - \zeta_r^{r-2}$.

Therefore, given $[x]_F = ([x_0]_{\mathbb{F}_p}, [x_1]_{\mathbb{F}_p}, [x_2]_{\mathbb{F}_p} \ldots, [x_{r-2}]_{\mathbb{F}_p})$ where $x = \zeta_r^i$ for some i, it is easy to see that

$$[x]_F^{-1} := \left[x^{-1}\right]_F = ([x_0 - x_1]_{\mathbb{F}_p}, [-x_1]_{\mathbb{F}_p}, [x_{r-2} - x_1]_{\mathbb{F}_p}, \ldots, [x_2 - x_1]_{\mathbb{F}_p}).$$

In particular, if the $\mathbb{F}_p$-encoding is given by a linear secret sharing scheme, these inversions can be computed locally.

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, the value x is uniformly random and secret.

Arithmetic operations with random elements

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 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 $([x]_F, \left[x'\right]_F)$, where $x' = (x \mid p)_r$ and x is uniformly random in $F^\star$ and unknown to the parties. To do this, we proceed as in [12], and in Algorithm 2. The parties first select two uniformly 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 $F^\star$, 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,

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

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, which can be guaranteed by using a suitable encoding, as in Remark 6. Suppose we have a fresh random solved instance $([x]_F, \left[x'\right]_F)$. The parties compute and open ax, which is uniformly random in $F^\star$ and hence does not reveal any information about a. They can then compute the residue symbol $(ax \mid p)_r$ in the clear, and finally obtain

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

Computational and communication cost

The offline phase (Algorithm 2) can be optimized using the unbounded fan-in multiplication of [9], based on [2], which takes 3 rounds, $2l$ invocations of Algorithm 1 and $3l-1$ multiplications to secretly multiply l values. Since the first two rounds of the unbounded fan-in multiplication protocol are independent of the inputs, the values $[a]_F$ and $[b]_F$ can be generated and $[d]_F = [a]_F[b]_F$ computed at the same time. Then, $[d]_F^r$ and $[a]_F^r[d]_F$ are computed, and $[d]_F^r$ opened, in the third round. This gives a total cost of 3 rounds, as well as $2r + 3$ invocations of Algorithm 1 and $6r + 2$ multiplications of elements in F.

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

Recall that multiplication of elements in F is more expensive than multiplication in $\mathbb{F}_p$, 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 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 by checking the conditions (1) for many primes p. While it is possible to do this by simply computing $(a \mid p)_r$ for all $a \in A$, the computation of the residue symbol is relatively expensive in practice. In the following, we instead translate the conditions into equations of the form $p \in M_a \pmod{N_a}$ for $a \in A$, where $M_a \subseteq \mathbb{Z}/N_a\mathbb{Z}$ and $N_a > 1$, which are much faster to check.

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 $\mathbb{F}_r^\star$, 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

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

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 by choosing a new encoding as in Remark 6.

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

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

the last equation holds because the value of the residue symbol $(p \mid \hat{a})_r$ depends only on p modulo $N(\hat{a})$. 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 $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^\star$. 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 $q \in (\mathbb{Z}/N(\hat{a})\mathbb{Z})^\star$ 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

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

is a subgroup of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^\star$. If H_a is the entirety of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^\star$, 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 start over with a new encoding of the inputs (see Remark 6).

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 $g_1, \dots, g_n$ of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^\star$, of orders $k_1, \dots, k_n$, which induce an isomorphism

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

The Sage computer algebra system [11] contains a function that provides these generators, with the same complexity as factoring $N(\hat{a})$.

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 $k_1, \dots, k_n \in \mathbb{Z}_{>1}$. 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 full rank 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

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

Let B now be such a basis matrix for the lattice $\tilde{H}$. 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 $0 \le a_1, a_2 < r$.

To find the basis $h_1, \dots, h_n$ of $\tilde{H}$, we now proceed as in Algorithm 4. Let $e_1, \dots, e_n$ 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 $a \in \{0, \dots, r-1\}$ 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 $e_1, \dots, e_n$.

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 $\tilde{H}$.

```
$J \leftarrow 0$
for $i$ from $n$ to 1 do
    if $J = 0$ then
        if $\tilde{\chi}(e_i) = 1$ then
            $h_i \leftarrow e_i$
        else
            $h_i \leftarrow r \cdot e_i$
            $J \leftarrow i$
        end if
    else
        for $a$ from 0 to $r - 1$ do
            if $\tilde{\chi}(e_i + ae_J) = 1$ then
                $h_i \leftarrow e_i + ae_J$
            end if
        end for
    end if
end for
```

Remark 8. 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 9. 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.

5.2 Complexity of finding the modulus

Given the equations we determined above, what size can we expect for the smallest prime p that satisfies them, assuming that they are indeed satisfiable? We have seen in Section 5.1 that M_a for $a \in A$ is a subset of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$ of size at least $\varphi(N(\hat{a}))/r$, if it is not empty, where φ is the Euler phi function. Since in practice p is a prime larger than any $N(\hat{a})$, its reduction modulo $N(\hat{a})$ will always lie in $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$. Let us assume that the reductions of p in $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$ for $a \in A$ and in $(\mathbb{Z}/r^2\mathbb{Z})^*$ are uniformly random and independent, for random p.[2] Then we have $p \in M_a \pmod{N(\hat{a})}$ with probability at least $1/r$. Further, $p \in M_0$ is satisfied with

2 This assumption is reasonable if the $N(\hat{a})$ for $a \in A$ and r are pairwise coprime. If this is not the case, the dependence may be helpful or harmful.

probability $\varphi(r-1)/\varphi(r^2) > 1/r^2$. We can hence estimate the smallest prime satisfying the equations to have size at most $O(r^{|A|+2}(|A|+2)\log r)$ by the prime number theorem.

To check if a given prime integer p satisfies the conditions (1), we could simply compute $(a \mid p)_r$ for all $a \in A$. A single such test can be done by raising a to the $(N(p)-1)/r$-th power in $R/(p)$, which takes $O(\log N(p)) = O(r \log p)$ multiplications in $R/(p)$, or $O(r^3 \log p)$ multiplications in $\mathbb{F}_p$, giving a time complexity of $\tilde{O}(r^3 \log^2 p)$ using Schönhage-Strassen multiplication. By the previous paragraph, we expect that we need to test around $\tilde{O}(r^{|A|+2})$ primes, taking time around $\tilde{O}(r^{|A|+2} \cdot |A| \cdot r^3 \log^2(r^{|A|+2})) = \tilde{O}(r^{|A|+5}|A|^3)$. We also need to test $\tilde{O}(r^{|A|+2}|A|)$ integers for primality, which takes time $\tilde{O}(r^{|A|+2}|A| \log^2 r^{|A|}) = \tilde{O}(r^{|A|+2}|A|^3)$ using Miller-Rabin.

Instead, in the method described in the beginning of Section 5, we first list the sets M_a for all $a \in A$. Using the idea described in Section 5.1, we expect the dominating cost to be the enumeration of M_a after the generators have been found. If n is an upper bound on the norms of the elements of A, we can then estimate the time *and memory* cost of the first step to be $\tilde{O}(n|A|)$. Then, checking a single condition from (1) consists of a modular reduction of integers and a set membership test, taking time $\tilde{O}(\log p)$. In this case the cost of the Miller-Rabin primality test dominates, giving an estimated total time complexity for the second step of $\tilde{O}(r^{|A|+2}|A|^3)$. Alternatively, the Sieve of Eratosthenes can be used to reduce this to $\tilde{O}(r^{|A|+2}|A|^2)$ at the cost of $\tilde{O}(r^{|A|+2}|A|)$ memory.

Although the asymptotic estimates for the two methods are quite similar, our experiments indicate that the second method is significantly faster when the norms of the elements of A are not too large. However, we will see in Section 5.3 that the case of large norms is interesting as well.

5.3 Existence of a good encoding

As mentioned in and before Remark 6, for a given choice of A and $f\colon A \to \{0,\dots,r-1\}$ it is often impossible to satisfy (1) due to internal contradictions. So in order to compute a desired function $f'\colon A' \to \{0,\dots,r-1\}$, we need to find some values of A and f that do not lead to such contradictions, together with an encoding (an affine map) $E\colon A' \to A$ satisfying $f \circ E = f'$.

For small choices of A', it may be possible to find an encoding using trial and error, as we have done in the example of Section 6. We do not know of a guaranteed way to find good encodings, but we give some considerations on the matter.

One common cause of contradictions is that different elements of A can have prime factors in common (or prime factors that are conjugates of each other under an automorphism of R). Trying to satisfy some of the requirements in (1) can fix the residue symbol on the prime factors, which may end up fixing the residue symbol of some other element of A to an undesirable value. To avoid this, we could try to find an encoding such that no two distinct elements of A have prime factors that are conjugates of each other.

Proposition 10. *Let $A' \subset R$ be a finite subset containing no two distinct elements that are conjugate to each other under an automorphism of R.*

Let s' be the product of all nonzero elements of the form $a - \sigma(b)$ for $a, b \in A'$ and $\sigma \in \operatorname{Aut}(R)$. Define $s \in \mathbb{Z}$ as the product of all (integer) prime numbers dividing $N(s')$, and let $q \in \mathbb{Z}$ be coprime to s.

Then, no two distinct elements of $A = \{q + as \mid a \in A'\}$ have prime factors that are conjugate under an automorphism of R.

Proof. Let p be a prime element of R dividing s. Since p does not divide q, it cannot divide $q + as$ for any $a \in A'$.

Suppose now that we have $a, b \in A'$, $\sigma \in \operatorname{Aut}(R)$ and a prime element $p \nmid s$ such that $p \mid q + as$ and $\sigma(p) \mid q + bs$. Hence,

$$p \mid (q + as) - \sigma^{-1}(q + bs) = (a - \sigma^{-1}(b))s.$$

Since p is prime and does not divide s, it must divide $a - \sigma^{-1}(b)$. However, if $a - \sigma^{-1}(b)$ is nonzero, its prime factors are also factors of s by definition. Hence $a = \sigma^{-1}(b)$, and so $a = b$ by the condition on A'. □

In this case, the norms $N(a)$ for $a \in A$ are pairwise coprime. The norms $N(\hat{a})$ are furthermore coprime to r by construction of $\hat{a}$ (see Section 5). Under the assumption that the equations $q \in M_a \pmod{N(\hat{a})}$ and $q \in M_0$ $\pmod{r^2}$ are individually satisfiable, it then follows by the Chinese remainder theorem and Dirichlet's prime number theorem that a prime solution p exists that satisfies all at once.

However, it is not guaranteed that the individual equations are satisfiable. For example, the residue symbol $(\hat{a} \mid p)_r$ for $\hat{a}$ a unit, or an r-th power, or real or purely imaginary is independent of p, so M_a may be empty. Since these types of elements are relatively uncommon, it may nonetheless be possible to avoid them with a good choice of A. In fact, if A' is chosen as a subset of $\mathbb{Z}$, the construction in Proposition 10 will always result in $A \subset \mathbb{Z}$. It is therefore necessary to choose an A' that does not consist of integers, for example by finding a suitable "pre-encoding" $E' : A'' \to A'$.

The downside of this approach is that it results in $a \in A$ of very large norm, which can make the sets M_a unmanageably large. It may therefore be preferable to simply test the conditions (1) directly instead of applying the method described in the beginning of Section 5.

When the norms $N(\hat{a})$, $a \in A$, and r are pairwise coprime, there is another approach to finding a solution p that does not require enumerating M_a: Let $N = r^2 \prod_{a \in A} N(\hat{a})$, and pick a random element $b_a \in M_a$ for each $a \in A$, as well as $b_0 \in M_0$. It should be feasible to find these if they exist, since at least one in r elements of $(\mathbb{Z}/N(\hat{a})\mathbb{Z})^*$ is in M_a. Then, use the Chinese remainder theorem to translate $(b_a)_{a \in A \cup \{0\}}$ to $b \in (\mathbb{Z}/N\mathbb{Z})^*$, and test if b is prime. If not, multiply b by a random r-th power in $(\mathbb{Z}/N\mathbb{Z})^*$ congruent to 1 modulo r^2 and test again. This way, we expect to find a prime in around $\log(N)$ steps. Unfortunately, the resulting prime will be of size $O(N)$, which may be too large for practical purposes.

6 Toy example

We present an example, in which we compute reduction modulo 3 for integers $x \in \{0, \dots, n\}$ for some small n. We pick $r = 3$. This example was constructed with the help of the Sage computer algebra system [11].

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

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

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

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

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

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

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 $0 \le x \le 18$.

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

7 Alternatives

There are several alternatives for computing an arbitrary function $f\colon A \to \{0, \dots, r-1\}$ in a secret shared manner using a constant number of rounds.[3]

For example, one may use Lagrange interpolation to find a degree $|A| - 1$ polynomial $F \in R[x]$ that agrees with f on A. The polynomial can then be evaluated in three rounds using unbounded fan-in multiplication [2, 9], the first two of which can be done in precomputation. This compares favourably to our protocol, which has three offline and one online round. Furthermore, polynomial interpolation works for any prime modulus larger than $\max\{|A|, r\}$, whereas our protocol requires a specific and much larger prime modulus. On the other hand, our protocol has the advantage of requiring fewer multiplications, namely $O(r)$ instead of $O(|A|)$.

One could also consider decomposing f into a sequence of two-valued functions $f_1, \dots, f_k\colon A \to \{0, 1\}$ and computing each using the Legendre symbol. It would then be necessary to find k different (affine) encodings $E_i\colon A \to \mathbb{Z}$, $1 \le i \le k$, and a prime p such that $f_i(a)$ agrees with $2(E_i(a) \mid p) - 1$ simultaneously for all i. Such encodings could be found using similar ideas to those presented in Section 5.3 and [12]. If we assume that the E_i have disjoint images, and that the residue symbol on the elements of these images are uniformly random and independent, this would give us a p of expected size $O(2^{k|A|}k|A|)$. If we use a one-hot encoding to represent $f(x) = 1f_1(x) + \cdots + (r-1)f_{r-1}(x)$, we get $k = r - 1$, which results in a much larger estimate for p than in our method. One could instead choose $k = \lceil \log_2 r \rceil$ and encode f as a Boolean circuit in the f_i. This would result in a similar expected size for p as in our protocol, but may require extra rounds to evaluate the circuit, especially if a one-hot encoding of the output is desired. Compared to our protocol, this method takes the same number of offline rounds, but either gives a larger expected p or requires extra online rounds. In some cases, the estimate for p can be improved by choosing E_i with overlapping images. For example, using a one-hot encoding for the function $f(x) = x \bmod 3$ of our toy example, one may notice that $f_2(x) = f_1(x-1)$ and so use two mostly overlapping encodings.

8 Conclusion

We have introduced a protocol for secure multiparty computation which allows the evaluation of certain desired functions $f\colon A \to \{0, \dots, r-1\}$ 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 $f\colon A \to \{0, \dots, r-1\}$ 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. It is 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.

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

We wish to thank an anonymous reviewer for the alternative idea of encoding a function in terms of several two-valued functions, see Section 7.

3 Note that the alternatives we mention require sending messages of size at least linear in the size of the domain of the function $|A|$, which we expect to be the case for our protocol too, given the estimate for p in Section 5.2.

References

[1] M. Abspoel, N. J. Bouman, B. Schoenmakers and N. de Vreede, Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks, in: *Topics in Cryptology – CT-RSA 2019* (M. Matsui, ed.), Springer, Cham, pp. 453–472, 2019.

[2] J. Bar-Ilan and D. Beaver, Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction, in: *Proc. 8th ACM Symp. on Principles of Distributed Computing*, PODC '89, pp. 201–209, ACM, New York, 1989.

[3] Y. F. Bilu, Y. Bugeaud and M. Mignotte, *The Problem of Catalan*, Springer, Cham, 2014.

[4] R. Cramer, I. B. Damgård and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, Cambridge University Press, New York, 2015.

[5] 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: *Theory of Cryptography* (S. Halevi and T. Rabin, eds.), Springer, Berlin, Heidelberg, pp. 285–304, 2006.

[6] C. Dwork, *Lattices and Their Application to Cryptography*, Lecture Notes, Stanford University, 1998.

[7] U. Feige, J. Killian and M. Naor, A Minimal Model for Secure Computation, in: *Proc. 26th ACM Symp. on Theory of Computing*, STOC '94, pp. 554–563, ACM, New York, 1994.

[8] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, 2nd ed, Graduate Texts in Mathematics 84, Springer-Verlag, New York, 1990.

[9] T. Nishide and K. Ohta, Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol, in: *Public Key Cryptography – PKC 2007* (T. Okamoto and X. Wang, eds.), pp. 343–360, Springer, Berlin, Heidelberg, 2007.

[10] A. Shamir, How to Share a Secret, *Commun. ACM* **22** (1979), 612–613.

[11] W. A. Stein et al., *Sage Mathematics Software (Version 9.1)*, The Sage Development Team, 2020.

[12] C.-H. Yu, *Sign Modules in Secure Arithmetic Circuits.*, Cryptology ePrint Archive, Report 2011/539, 2011.