Secure Function Extensions to Additively Homomorphic Cryptosystems

Mounika Pratapa[0000−0002−5991−016X] and Aleksander Essex[0000−0002−0228−0371]

Western University, Canada
{mpratapa,aessex}@uwo.ca

Abstract. The number-theoretic literature has long studied the question of distributions of sequences of quadratic residue symbols modulo a prime number. In this paper, we present an efficient algorithm for generating primes containing chosen sequences of quadratic residue symbols and use it as the basis of a method extending the functionality of additively homomorphic cryptosystems.
We present an algorithm for encoding a chosen Boolean function into the public key and an efficient two-party protocol for evaluating this function on an encrypted sum. We demonstrate concrete parameters for secure function evaluation on encrypted sums up to eight bits at standard key sizes in the integer factorization setting. Although the approach is limited to applications involving small sums, it is a practical way to extend the functionality of existing secure protocols built on partially homomorphic encryption schemes.

Keywords: Secure computation · Additive homomorphic encryption · Quadratic residues · Residue symbol sequences

1 Introduction

Ever since Yao's Millionaires' problem [29], distrusting parties have been computing things of mutual interest without sharing their respective inputs. Secure function evaluation (SFE) has many interesting applications in areas such as privacy-preserving machine learning [24], private information retrieval [10], similarity search in private databases such as genotype and other medical data [25], online voting [2], auctions [11] and private credit checking [18].

Despite recent advances in fully homomorphic encryption, *partially* homomorphic schemes (i.e., those offering homomorphic operations with respect to a single operation) still play an important role in secure computation. For example, Switzerland requires internet-based elections to be cryptographically verifiable[1] and the first certified implementation is based around mix nets built from additively homomorphic encryption.[2]

[1] Swiss Federal Chancellery Ordinance on Electronic Voting. Available: https://www.fedlex.admin.ch/eli/cc/2022/336/en

[2] The Swiss Post E-voting System. Available: https://gitlab.com/swisspost-evoting

In applications where partially homomorphic encryption is sufficient, such schemes can offer more clear-cut parameterizations, more mature hardness assumptions, more straightforward implementations, and faster executions relative to their fully homomorphic counterparts.

This paper presents a method for extending the functionality of additive homomorphic encryption schemes (specifically those with efficient full decryption) by working in groups containing sequences of quadratic residues and non-residues with a correspondence to a chosen Boolean function. Given an encrypted value $\mathsf{Enc}(x)$ and a Boolean function $f : \mathbb{Z}_t \to \{0, 1\}$, we present an efficient method for homomorphically evaluating $\mathsf{Enc}(f(x))$ in a single public-key operation across a short (but non-trivial) interval $0 \leq x < t$.

Let $\mathsf{QR} : \mathbb{Z} \times \mathbb{Z} \to \{0, 1\}$ be a function testing the quadratic residuosity of an integer $x \in \mathbb{Z}_p$, defined as

$$\mathsf{QR}(x, p) = \begin{cases} 0 & \text{if } x \text{ is a quadratic residue modulo } p. \\ 1 & \text{otherwise.} \end{cases}$$

Given $f(\cdot)$ and an integer sequence of the form $[\alpha x+\beta \mid 0 \leq x < t, \text{ and } \alpha, \beta > 0]$, our approach involves three components:

1. An efficient algorithm for finding a prime p for which
$$\mathsf{QR}(\alpha x + \beta, p) = f(x).$$
2. An additively homomorphic public-key cryptosystem embedding the required quadratic residue symbol sequence into the plaintext space, i.e., $\mathcal{M} \subset \mathbb{Z}_p$.
3. A public homomorphic operation that can blind the encryption of $\alpha x + \beta$ while preserving its quadratic residue symbol modulo p (and hence the output of the function $f(x)$).

Taken together, these components allow $f(x)$ to be securely evaluated on an encrypted sum in the range $0 \leq x < t$ for small (but non-trivial) values of t in a single public-key operation.

Previously, patterns in quadratic residues have been exploited for the evaluation of specific functions such as secure integer comparison [17], sign function evaluation [1,30], and threshold functions [16]. However, secure evaluation of *arbitrary* functions using quadratic residue patterns appears to be a novel direction. Our work extends the approach of [16] to the general case.

Contribution. We present an algorithm for generating primes with arithmetic sequences containing chosen quadratic residue symbols. These sequences extend the functionality of additively homomorphic cryptosystems and generalize the approach to secure evaluation of arbitrary functions by generating the candidate primes that facilitate the required quadratic residue symbol sequences. Using an additively homomorphic scheme with efficient full decryption (such as the schemes due to Paillier [22] and Okamoto-Uchiyama [21]), given an encrypted sum $\mathsf{Enc}(x)$, we present parameters for evaluating arbitrary functions $\mathsf{Enc}(f(x))$ for x up to $t = 256$ at the 4096-bit prime range and sums up to $t = 512$ where larger public-keys are acceptable.

2 Related Work

Finding patterns in quadratic residues and non-residues has been a subject in the number theory literature for a long time. Gauss posed the problem of finding the smallest quadratic non-residue n_p modulo a prime p [19], and papers over the past century have continued to refine this bound, most recently by Carella [8] showing $n_p \ll (\log p)(\log\log p)$.

Much of the subsequent literature has focused on distributions of consecutive runs of residues and non-residues, providing bounds on the size of a prime necessary to observe a specified run length. For some $a \geq \frac{1}{4}\sqrt{e}$, a sequence of length $p^{1/4}$ for a given p contains at least one residue and one non-residue according to Burgess [7]. Research has long explored primes with at least ℓ consecutive quadratic residues or non-residues. Studies by Brauer [5] and Davenport [14] examined arbitrary combinations of residues and non-residues, but these were restricted to very short sequences ($t < 10$).

Records for run lengths were improved with the rise of scientific computing in the 1980s. For example, Buell [6] experimentally studied the smallest primes exhibiting a consecutive sequence of ℓ residues followed by ℓ non-residues. For a residue symbol sequences of the form $\mathsf{QR}(a + x, p)$ for $0 \leq x < t$, they achieved $\ell = 9$ and thus $t = 18$ for $p = 414463$.

Since there are an equal number of residues and non-residues modulo an odd prime p, the probability that a particular integer will be a residue or non-residue would be $1/2$ across all primes if the distribution were uniform and random. Peralta [23] presented the probability of finding an arbitrary residue symbol sequence modulo p given the length of sequence t and found it deviates from random by a factor no more than $t(3 + \sqrt{p})/p$. Later research exploited the random-looking distributions of residue symbols for applications in watermarking [4] and pseudorandom bit generators [13, 26, 27], approximate pattern matching [15].

While there are several interesting applications exploiting the patterns in quadratic residues, Feige et al. [17] proposed a minimal model for secure integer comparison by exploiting the fact that, for $p = 7$, the Legendre symbols $\left(\frac{x}{p}\right) \forall x = a - b \mid a, b \in [-2, 2]$ coincide with the sign function of $x \in [-2, 2]$. Following this, improved secure function evaluation protocols were proposed [1, 30] to evaluate sign function by generating primes with the required quadratic residue symbol patterns modulo a prime number, specifically a Blum prime ($p \equiv 3 \bmod 4$). However, both these approaches [1, 30] rely on finding consecutive long runs of quadratic residues alone, thus only useful to perform secure comparison based on sign function. The work in [16] used brute force to search for consecutive sequences of ℓ residues followed by ℓ non-residues, where the quadratic residuosity function is calculated as $\mathsf{QR}(x+a, p)$. The maximum value attained in this approach is $t = 52$, for $a = 1134844$ and $p = 2269739$. Such runs were exploited to evaluate the threshold function, i.e., a Heaviside function $H(x)$, which is off until $x = c$.

Our approach. As we prove in the next section, the number of primes exhibiting a given residue symbol sequence is infinite. However, in contrast to previous work, instead of relying on the Legendre symbols of consecutive numbers modulo a prime p, we present parameters to find arithmetic sequences of the form $\alpha x + \beta$, which can be used to generate primes toward secure evaluation of an arbitrary function with an integer domain and Boolean range.

3 Cryptographic Preliminaries

Let f be a function where, $f : \mathbb{Z}_t \to \{0, 1\}$ is defined over an integer input x for $0 \leq x < t$. Our objective is to securely evaluate f without revealing its inputs. To achieve this, we present relevant notations and an algorithm that can provably generate primes embedding arithmetic residue symbol sequences that implement f. Such primes are useful to extend the functionality of additively homomorphic cryptosystems.

Definition 1 (Legendre Symbol). *The Legendre symbol is a function* $L : \mathbb{Z} \times \mathbb{Z} \mapsto \{-1, 0, 1\}$ *defined as:*

$$\left(\frac{x}{p}\right) \equiv \begin{cases} 1 & \textit{if } x \textit{ is quadratic residue} \bmod p \\ -1 & \textit{if } x \textit{ is quadratic non-residue} \bmod p \\ 0 & \textit{if } x \equiv 0 \bmod p. \end{cases}$$

It can be directly established that the quadratic residuosity function $\mathsf{QR} : \mathbb{Z} \times \mathbb{Z} \to \{0, 1\}$, as defined in the introduction:

$$\mathsf{QR}(x, p) = \begin{cases} 0 & \text{if } x \text{ is a quadratic residue modulo } p. \\ 1 & \text{otherwise.} \end{cases}$$

is a modification of the Legendre symbol's co-domain where

$$\mathsf{QR}(x, p) = \frac{\left(\frac{x}{p}\right) + 1}{2}.$$

Since the Legendre symbol is a completely multiplicative function of its top argument, so is the quadratic residuosity function.

3.1 Linear Embeddings of Boolean Functions in Residue Sequences

We deal with the following functions:

- A function $f : \mathbb{Z}_t \to \{0, 1\}$ that needs to be securely evaluated over an input x, for $0 \leq x < t$ and $t \in \mathbb{Z}^+$.
- The quadratic residuosity function $\mathsf{QR} : \mathbb{Z} \times \mathbb{Z} \to \{0, 1\}$ for secure evaluation of f.

– A mapping function $h : \mathbb{Z} \to \mathbb{Z}_p$, that maps an input x into $h(x) = (\alpha x + \beta) \bmod p$. Here, $\{\alpha, \beta\} \in \mathbb{Z}^+$ and are given as input parameters to facilitate the application of QR function.

We are interested in locating primes p which contain some residue symbol sequences modulo a prime p that imitate the range of f. In other words, given and integers $\{\alpha, \beta\}$ we are looking for some p that can compute:

$$\mathsf{QR}(h(x), p) = f(x)$$

for $0 \leq x < t$.

Approach to Secure computation. We can homomorphically evaluate f using an additive scheme as follows. Let $\mathsf{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ be an additively homomorphic public-key cryptosystem that display the following homomorphisms:

$$\mathsf{Enc}(x_1) \cdot \mathsf{Enc}(x_2) = \mathsf{Enc}(x_1 + x_2 \bmod p)$$

and

$$\mathsf{Enc}(x_1)^{x_2} = \mathsf{Enc}(x_1 x_2 \bmod p).$$

Given an encrypted value $\mathsf{Enc}(x)$ in the range $0 \leq x < t$, and an $\alpha, \beta > 0$, one can homomorphically compute $\mathsf{Enc}(h(x))$ as follows:

$$\mathsf{Enc}(h(x)) = \mathsf{Enc}(x)^{\alpha} \cdot \mathsf{Enc}(\beta) = \mathsf{Enc}(\alpha x + \beta \bmod (p)).$$

Applying the quadratic residue function to the decryption $\mathsf{Enc}(h(x))$ yields.

$$\mathsf{QR}(\mathsf{Dec}(\mathsf{Enc}(h(x))), p) = \mathsf{QR}(h(x), p) = f(x).$$

This demonstrates the basic mechanics of the secure evaluation of f. Clearly, however, the decrypter could recover x from seeing $h(x)$, and thus a homomorphic blinding function will be presented later in section 4.

3.2 Prime numbers with Chosen Residue Symbol Sequences

We begin with the Legendre symbol as a standard notation and later re-frame the discussion in terms of a quadratic residuosity function QR. Consider a list of t distinct primes $\{a_1, \ldots, a_t\}$ and a list of Legendre symbols $\{\ell_1, \ldots, \ell_t\}$ where $\ell_x \in \{-1, 1\}$. For all $1 \leq x \leq t$, a prime p can be generated using Algorithm 1 such that

$$\left(\frac{p}{a_x}\right) = \ell_x.$$

function ψ(Primes $\{a_1, \ldots, a_t\}$, target symbols $\{\ell_1, \ldots, \ell_t\}$, bit-length λ)
  while $True$ **do**
    $B \leftarrow [\,]$
    for $1 \leq x \leq t$ **do**
      if $\ell_x == 1$ **then**
        Choose $b_x \xleftarrow{R} QR_{a_x}$ ▷ Set of quadratic residues modulo a_x
      else
        Choose $b_x \xleftarrow{R} NR_{a_x}$ ▷ Set of non-residues modulo a_x
      end if
      $B.append(b_x)$
    end for
    Compute $p' \leftarrow \mathsf{CRT}\big([a_1, \ldots, a_t], [b_1, \ldots, b_t]\big)$
    Compute $A \leftarrow \prod_{x=0}^{t-1} [a_x]$
    Choose $k \xleftarrow{R} [k_{min}, k_{max}]$ ▷ Largest interval such that $|p| = \lambda$
    Compute $p \leftarrow kA + p'$
    if $isPrime(p) == True$ **then**
      return p
    end if
  end while
end function

Algorithm 1

Theorem 1. *The function ψ in Algorithm 1 is guaranteed to return a prime p that facilitates the chosen residue symbol sequences.*

Proof. See Appendix A for the proof.

Theorem 2. *For all $t \in \mathbb{Z}^+$ and all functions $f : \mathbb{Z}_t \to \{0, 1\}$ there exists a prime p and two integers $0 < \alpha, \beta < p$ such that for all $0 \leq x < t$*

$$\frac{\left(\frac{\alpha x + \beta}{p}\right) + 1}{2} = f(x)$$

where $\left(\frac{\alpha x+\beta}{p}\right)$ denotes the Legendre symbol of $\alpha x + \beta$ modulo p.

Proof. Let α, β, t be positive integers such that $\alpha x + \beta$ is prime for all $0 \leq x < t$. The existence of such an α, β is guaranteed for all $t > 0$ by a theorem due to Green and Tao [20], which proves the primes contain arbitrarily long arithmetic sequences, and, therefore, there exists an α, β for all $t > 0$ such that $\alpha x + \beta$ is prime for all $0 \leq x < t$. Given such a linear sequence of prime valued[3] $(\alpha x+\beta)$'s, Theorem 1 guarantees there exists a prime p such that for all $0 \leq x < t$,

$$\left(\frac{p}{\alpha x + \beta}\right) = 2f(x) - 1.$$

[3] Requiring all $(\alpha x+\beta)$ be prime is only done to facilitate the existence proof. In practice, Algorithm Gen (see Section 4.1) can generate suitable keypairs in the presence of composite $(\alpha x + \beta)$'s.

Suppose there existed a p such that $p \equiv 1 \bmod 4$. By the law of quadratic reciprocity,

$$\left(\frac{\alpha x+\beta}{p}\right) = \left(\frac{p}{\alpha x+\beta}\right) = 2f(x) - 1, \tag{1}$$

and therefore,

$$f(x) = \frac{\left(\frac{\alpha x+\beta}{p}\right)+1}{2}.$$

In the alternate case where all such primes p were congruent to 3 mod 4, Theorem 1 also guarantees there exists a prime p such that

$$\left(\frac{p}{\alpha x+\beta}\right) = \begin{cases} 2f(x)-1 & \text{if } \alpha x+b \equiv 1 \bmod 4 \\ 1-2f(x) & \text{if } \alpha x+b \equiv 3 \bmod 4. \end{cases}$$

For all $\alpha x + \beta \equiv 1 \bmod 4$, quadratic reciprocity again gives us

$$\left(\frac{\alpha x+\beta}{p}\right) = \left(\frac{p}{\alpha x+\beta}\right) = 2f(x) - 1.$$

Finally, for all $\alpha x + \beta \equiv 3 \bmod 4$,

$$\left(\frac{\alpha x+\beta}{p}\right) = -\left(\frac{p}{\alpha x+\beta}\right) = -(1-2f(x)) = 2f(x) - 1.$$

Therefore

$$f(x) = \frac{\left(\frac{\alpha x+\beta}{p}\right)+1}{2} \tag{2}$$

for all $0 \leq x < t$. □

4 Our Cryptosystem

Let $\mathsf{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Smul}, \mathsf{Eval}\}$ be an additively homomorphic public-key cryptosystem. Let $\mathcal{M}$ be the plaintext space and $m \in \mathcal{M}$ be a message. Without loss of generality and for the sake of a concrete description, we build CS based on the cryptosystem due to Okomoto and Uchiyama [21], which has a message space cardinality $|\mathcal{M}| = p$ for a large prime p. Given pre-computed sequence parameters α, β, and a Boolean function $f : \mathbb{Z}_t \to \{0, 1\}$ we define CS with the following functionalities:

- $\mathsf{Gen}(1^\rho, \alpha, \beta, f)$: Outputs secret key $\mathcal{SK} = \{p, q\}$ and public key $\mathcal{PK} = \{n\}$ where $n = p^2q$. Here p is chosen such that $\mathsf{QR}(\alpha x + \beta, p) = f(x)$ for $0 \leq x < t$. To facilitate efficient generation for non-trivial values of t, p is generated using the algorithm presented in Section 4.1. By contrast, q is randomly chosen using standard methods. Both $|p| = |q| = \lambda$, where λ is a standard length at the ρ-bit security level in the integer factorization setting.

– $\mathsf{Enc}(\mathcal{PK}, m)$: Encryption function accepting a public key $\mathcal{PK}$, plaintext m and outputting a ciphertext $c = [\![m]\!]$.
– $\mathsf{Dec}(\mathcal{SK}, c)$: Decryption function accepting a private key $\mathcal{SK}$, ciphertext $c = [\![m]\!]$ and outputting plaintext m.
– $\mathsf{Add}(c_1, c_2)$: Homomorphic addition accepting two encrypted messages $c_1 = [\![m_1]\!]$ and $c_2 = [\![m_2]\!]$ and outputting a ciphertext $c' = [\![(m_1 + m_2) \bmod p]\!]$.
– $\mathsf{Smul}(s, c)$: Scalar homomorphic multiplication accepting a ciphertext $c_1 = [\![m_1]\!]$ and a scalar m_2, outputting a ciphertext $c' = [\![(m_1 m_2) \bmod p]\!]$.

The properties of CS can be further used to securely evaluate f on an encrypted plaintext. Toward that end, a sixth functionality Eval, is defined:

– $\mathsf{Eval}(\mathcal{PK}, \alpha, \beta, c)$: Secure evaluation of f on an encrypted plaintext $c = [\![m]\!]$. First, a non-zero blinding parameter r_c is uniformly sampled from the message space $\mathcal{M}$. Since $\mathcal{M} = \mathbb{Z}_p$ and p is a private value, we sample uniformly from the public interval $r_c \leftarrow [1, 2^\lambda]$.[4] The function computes:

$$\mathsf{Smul}(r_c^2, \mathsf{Add}(\mathsf{Smul}([\![m]\!], \alpha), \mathsf{Enc}(\beta)) = [\![r_c^2 \cdot (\alpha m + \beta) \bmod p]\!].$$

Note that decrypting $[\![(\alpha m + \beta) \bmod p]\!]$ directly would reveal m, as α, β are public. Hence, a blinding operation is applied to randomize this value while preserving its residuosity. The result of Eval is the encryption of uniform quadratic residue in $\mathbb{Z}_p$ if $\mathsf{QR}(\alpha m + \beta) = 1$, and a uniform non-residue otherwise. The output of Eval can then be decrypted by the private key holder and the quadratic residuosity of the plaintext tested to reveal the outcome of $f(m)$.

4.1 Key Generation

To securely evaluate a function of the form $f : \mathbb{Z}_t \mapsto \{0, 1\}$, we work in an additive group modulo a prime p which contains an arithmetic sequence $S \subset \mathbb{Z}_p^*$ such that for each $s_m \in S$, $\mathsf{QR}(s_m) = f(m)$, or, in Legendre symbol form, where:

$$\left(\frac{s_m}{p}\right) = 1 - 2 \cdot f(m).$$

This section describes a method for generating a prime p containing such a sequence.[5]

Step 1: Let $S = \{s_m \mid s_m = \alpha m + \beta, 0 \leq m < t\}$ be an odd sequence for some $\alpha, \beta \in \mathbb{Z}^+$.

[4] If implementing CS based on an additive cryptosystem in which $|\mathcal{M}| = n$ is a public value, such as in the case of DGK [12] or Paillier [22], blinding factor r_c can be chosen from $\mathbb{Z}_n$.

[5] See our Python3 implementation of the key generation algorithm with example parameters: https://anonymous.4open.science/r/SFEpapercode-85C2

Step 2: For each $s_m \in S$, let $s_{m,0}^{(e_{m,0})}, \ldots, s_{m,\rho_m}^{(e_{m,\rho_m})}$ represent all of the prime factors of s_m for which $e_{m,j}$ is an odd power.[6] The multiplicative properties of the Legendre symbol give us

$$\left(\frac{s_m}{p}\right) = \left(\frac{s_{m,0}^{(e_{m,0})} \cdot \ldots \cdot s_{m,\rho_m}^{(e_{m,\rho_m})}}{p}\right) = \left(\frac{s_{m,0}}{p}\right) \cdot \ldots \cdot \left(\frac{s_{m,\rho_m}}{p}\right) = 1 - 2 \cdot f(m)$$

This can be rewritten in terms of applications of QR by replacing multiplications with additions when expressing the factorization of individual sequence elements:

$$\mathsf{QR}(s_m, p) = \mathsf{QR}(s_{m,0}, p) + \ldots + \mathsf{QR}(s_{m,\rho_m}, p) \equiv f(m) \bmod 2. \tag{3}$$

Expressing residues and non-residues in this form (i.e., as an addition modulo 2) instead of Legendre symbols (as a multiplication of signs) allows us to obtain a system of equations capturing the relationship between the unique prime factors of a sequence element and the function evaluated at that position.

Step 3: Let $A = \{a_0, \ldots, a_{u-1}\}$ represent the set of u unique prime factors from the combined set of all sequence factors $s_{m,j}$ across all sequence elements s_m. That is, $a_i \in A$ if there is some s_m such that $a_i \mid s_m$ and $a_i \nmid s_j$ for all other $j \neq m$. Non-unique factors will not be utilized in this calculation and are assigned a fixed, implicit target residue symbol of 1. For each sequence element $s_m \in S$ and each unique prime factor $a_j \in A$, we define a function $d(a_j, s_m)$ such that

$$d(a_j, s_m) = \begin{cases} 1 & \text{if } a_j \mid s_m \\ 0 & \text{otherwise.} \end{cases}$$

Step 4: Define a $(t \times u)$ matrix M. Let the last column represent function f evaluated at m. Form an augmented matrix representing the system of equations arising from Equation (3):

$$\begin{array}{c} \\ s_0 \\ s_1 \\ \vdots \\ s_{t-1} \end{array} \begin{array}{c} \begin{array}{cccc} a_0 & a_1 & \ldots & a_{u-1} \end{array} \\ \left(\begin{array}{cccc|c} d(a_0, s_0) & d(a_1, s_0) & \ldots & d(a_{u-1}, s_0) & f(0) \\ d(a_0, s_1) & d(a_1, s_1) & \ldots & d(a_{u-1}, s_1) & f(1) \\ \vdots & \vdots & & \vdots & \vdots \\ d(a_0, s_{t-1}) & d(a_1, s_{t-1}) & \ldots & d(a_{u-1}, s_{t-1}) & f(t-1) \end{array}\right) \end{array}$$

Step 5: Using Gaussian elimination, convert M into reduced row echelon form, i.e., compute $M' \leftarrow \mathsf{RREF}(M)$. If the system of equations implied by M' is consistent and exactly determined, each $a_j \in A$ implies a residue value $\sigma_j \in \{0, 1\}$ which will satisfy the overall requirement that $\mathsf{QR}(s_m) = f(m)$ for $0 \leq m < t$.

[6] Factors $s_{m,j}^{e_{m,j}}$ of even power will have a fixed residue symbol of 1 and are excluded from the generation algorithm (having no bearing on the outcome).

If the system is consistent and *underdetermined*, select a single valid solution uniformly at random and proceed to Step 6. Otherwise repeat the same process from Step 1 with a different α, β.

Step 6: For each factor $a_j \in A$ and each residue value $\sigma_j \in \{0,1\}$ computed in the previous step, select b_j uniformly from $[0, a_j)$ such that $\mathsf{QR}(b_j, a_j) = \sigma_j$.

Step 7: For each pair a_j, b_j, apply Chinese remaindering to compute a p' satisfying the following system of congruences:

$$\begin{aligned} p' &\equiv b_0 \bmod a_0 \\ p' &\equiv b_1 \bmod a_1 \\ p' &\equiv b_2 \bmod a_2 \\ &\vdots \\ p' &\equiv b_{u-1} \bmod a_{u-1}. \end{aligned}$$

Step 8: Compute:

$$p \leftarrow k\Big(\prod_{j=0}^{u-1} a_j\Big) + p'$$

for $k \xleftarrow{R} [k_{min}, k_{max}]$ sampled uniformly from the largest interval such that $|p| = \lambda$.

Step 9: If $p \equiv 1 \bmod 4$ and p is prime, continue to the next step, otherwise repeat Steps 6–8 until such a p is found.

Step 10: Generate a random prime q of length λ using a standard generation method suitable for the integer factorization setting.

Step 11: Output p, q.

4.2 Encryption and Decryption

We recall the Okamoto-Uchiyama cryptosystem [21], which is the basis for CS.

Key generation: Run the key generation algorithm in Section 4.1 to obtain large primes p, q. Compute $n = p^2 q$. Select a uniform $g \in \{2, \ldots, n-1\}$ such that:

$$g^{p-1} \not\equiv 1 \bmod p^2.$$

Set $h \leftarrow g^n \bmod n$. Return $\mathcal{SK} \leftarrow \{p, q\}$ and $\mathcal{PK} \leftarrow \{n, g, h\}$.

Observe: $\mathbb{Z}_n^*$ is a cyclic group of order $p(p-1)(q-1)$. Let us define two subgroups: $\mathbb{G}_p \subset \mathbb{Z}_n^*$, the subgroup of order p, and $\mathbb{G}_\phi \subset \mathbb{Z}_n^*$, the subgroup of order $\phi = (p-1)(q-1)$. An isomorphism $\mathbb{Z}_n^* \cong \mathbb{G}_p \times \mathbb{G}_\phi$ exists such that $g \in \mathbb{Z}_n^*$ can be rewritten as $g_p g_\phi \bmod n$, for some $g_p \in \mathbb{G}_p$ and $g_p \in \mathbb{G}_\phi$ respectively. Thus g has order $p \cdot \phi$ whereas $h = g^n = g^{p^2 q} = g_p^{p^2 q} g_\phi^{p^2 q} = g_p^0 g_\phi^{p^2 q} \equiv g_\phi^{\hat{r}} \bmod n$ has order ϕ.

Encryption: A plaintext $0 \leq m < p$ is encrypted as follows. Uniformly sample $r \leftarrow \mathbb{Z}_n^*$. Output:

$$c \leftarrow g^m h^r \bmod n.$$

Observe: Following our notation, $c = (g_p g_\phi)^m (g_\phi^{\hat{r}})^r = g_p^m g_\phi^{m+\hat{r}r} \equiv g_p^m g_\phi^{\bar{r}} \bmod n$. In other words, m is captured in the subgroup of order p and $g_\phi^{\bar{r}}$ is indistinguishable from a uniform element in $\mathbb{G}_\phi$.

Decryption: A ciphertext c is decrypted as follows. First compute

$$\hat{c} \leftarrow c^\phi \bmod n.$$

Observe: $\hat{c} = c^\phi = (g_p^m g_\phi^{\bar{r}})^\phi = (g_p^m)^\phi (g_\phi^{\bar{r}})^\phi = g_p^{m\phi} g_\phi^0 = g_p^{m\phi} \bmod n$.

Compute the discrete logarithm of $g_p^{m\phi}$ to recover $m\phi$. An efficient algorithm exists for computing discrete logarithms in $\mathbb{G}_p \subset \mathbb{Z}_{p^2q}^*$ given knowledge of p (which we omit for space). Finally, compute

$$(m\phi) \cdot \phi^{-1} \equiv m \bmod p$$

and return m.

4.3 Correctness of the Evaluation Function

The Eval function defined in Section 4 is used to homomorphically evaluate $[\![f(m)]\!]$ on encrypted plaintext $[\![m]\!]$.

Theorem 3. *Given* $c = \mathsf{Enc}(m)$, $\mathsf{QR}(\mathsf{Dec}(\mathcal{SK}, \mathsf{Eval}(\mathcal{PK}, \alpha, \beta, c)), p) = f(m)$ *in the range* $0 \leq m < t$.

Proof. Given ciphertext $c = \mathsf{Enc}(m)$, a blinding factor r_c sampled from the public interval $r_c \leftarrow [1, 2^\lambda]$, and public sequence parameters $\alpha,\ \beta$, Eval computes:

$$\begin{aligned} c' = \mathsf{Eval}(\mathcal{PK}, \alpha, \beta, c) &= (c^\alpha \cdot [\![\beta]\!])^{r_c^2} \bmod n \\ &= ([\![m]\!]^\alpha \cdot [\![\beta]\!])^{r_c^2} \\ &= [\![(\alpha m + \beta) \cdot r_c^2]\!]. \end{aligned}$$

Recall from the definitions in Section 3.1

$$\mathsf{QR}(\alpha m + \beta, p) = f(x)$$

and

$$\mathsf{QR}(r_c^2, p) = 1.$$

Decrypting c' returns $(\alpha m + \beta) \cdot r_c^2 \bmod p$. Therefore applying the QR-function to the decryption result we have

$$\begin{aligned} \mathsf{QR}((\alpha m + \beta) \cdot r_c^2, p) &= \mathsf{QR}(\alpha m + \beta, p) \cdot \mathsf{QR}(r_c^2, p) \\ &= f(m) \cdot 1 \\ &= f(m). \end{aligned}$$

□

5 Semantic Security of CS

We prove that CS is semantically secure under common hardness assumptions. While CS partially depends on the semantic security of the underlying Okamoto-Uchiyama [21] cryptosystem, the key-generation algorithm has been modified from the general case by the quadratic residuosity function. Here we argue that the modified key-generation function does not affect the semantic security of the underlying cryptosystem. First, we recall some definitions.

Definition 2. *A function $f : \mathbb{N} \mapsto \mathbb{R}$ is negligible with respect to $n \in \mathbb{N}$, if for all positive polynomials p, there exists some $M \in \mathbb{Z}$ such that it holds $f(n) < \frac{1}{p(n)}$ every time $n > M$.*

Definition 3. *Given an encryption function $\mathsf{Enc_k}(m)$, message length $|m|$, random number r and a security parameter 1^n, a cryptosystem CS is semantically secure if there exists a pair of probabilistic polynomial time algorithms $\mathcal{A}$ and $\mathcal{A}'$ such that, for every such pair:*

$$Pr[\mathcal{A}(1^n, \mathsf{Enc_k}(m), r) = m] - Pr[\mathcal{A}'(1^n, |m|, r) = m] < \frac{1}{p(n)}.$$

Note that $\mathcal{A}$ has access to the ciphertext $\mathsf{Enc}_k(m)$, where as $\mathcal{A}'$ has access only to the message length $|m|$, while the rest of the information both the algorithms have access to remain equal. Intuitively, the above definition thus implies that the ciphertext $\mathsf{Enc_k}(m)$ does not reveal any additional information about the underlying plaintext message.

Now, we proceed to establish the security proof of CS involving quadratic residuosity function by proving that the semantic security of the cryptosystem CS reduces to deciding the quadratic residuosity of a plaintext message m modulo a prime p, i.e., $\mathsf{QR}(m, p)$.

Definition 4. *Given $a \in \mathbb{Z}_n^*$ and $n = p^2q$ for unknown p, q, the p-th residue decision problem, denoted as $PRDP$, is the problem of deciding if there exists a b such that $a \equiv b^p \bmod n$.*

The semantic security of Okamoto-Uchiyama cryptosystem can be stated in terms of the $PRDP$. Particularly, for a message $m = 0$, the ciphertext $c = \mathsf{Enc}(0)$ is a p-th residue modulo n, since $c = g^0 h^r = h^r \bmod n$. Recall from Section 4.2 that $h = g^n = g^{p^2 q} \bmod n$ is a p-th residue.

Definition 5. *Given $\mathsf{Enc}(m)$ and an unknown p, the quadratic residuosity* mod *p decisional problem, denoted as $QRPDP$, is the computational problem of determining whether m is a quadratic residue modulo p, computing $\mathsf{QR}(m, p)$.*

Theorem 4. *The p-th residue decision problem is polynomially reducible to the quadratic residuosity decision problem, i.e., $PRDP \leq_p QRPDP$.*

Proof. Let $\mathcal{A}$ be an algorithm computing the quadratic residuosity of message $\mathsf{QR}(m, p)$. For a key-size λ and a ciphertext $c = \mathsf{Enc}(m)$, we construct an algorithm $\mathcal{B}(c)$ that returns $True$ if c is a p-th residue modulo in a polynomial factor of $\mathcal{A}$'s runtime with non-negligible advantage.

function $\mathcal{B}$(c)
    Uniformly sample $\gamma \xleftarrow{R} [1, 2^\lambda]$
    Compute $c' \leftarrow c \cdot \mathsf{Enc}(\gamma^2)$
    if $\mathcal{A}(c') == False$ **then**
        return $False$
    end if
    return $True$
end function

Algorithm: $\mathcal{B}$

function $\mathcal{A}$(c)
    Given $c = \mathsf{Enc}(m)$
    Compute $y \leftarrow \mathsf{QR}(m, p)$
    return y
end function

Algorithm: $\mathcal{A}$

Consider the case when $m = 0$. $\mathsf{QR}(0 + \gamma^2) = \mathsf{QR}(0) \cdot \mathsf{QR}(\gamma^2) = 1 \cdot 1 = 1$ for all γ. Therefore

$$Pr[\mathcal{B}(c) = True \,|\, m = 0] = 1.$$

For any other value of $m > 0$, the output of $\mathcal{B}(c)$ depends on the residuosity of $(m + \gamma^2)$, which in turn depends on the distribution of quadratic residues modulo p, giving us

$$Pr[\mathcal{B}(c) = True \,|\, m > 0] = \left(\frac{1}{2}\right) + \epsilon.$$

The overall probability

$$\begin{aligned} Pr[\mathcal{B}(c) = True \mid m \leftarrow [0, \lambda]] &= (Pr[\mathcal{B}(c) = True \,|\, m = 0] \cdot Pr[m = 0]) \\ &\quad + (Pr[\mathcal{B}(c) = True \,|\, m > 0] \cdot Pr[m > 0]) \\ &= \left(1 \cdot \frac{1}{2^\lambda}\right) + \left(\left(\frac{1}{2} + \epsilon\right) \cdot \left(1 - \frac{1}{2^\lambda}\right)\right) \geq \epsilon \end{aligned}$$

is non-negligible. This implies that as long as the semantic security of underlying encryption scheme holds, the $\mathsf{QR}()$ function does not reveal any additional information about the underlying plaintext message. □

Security in the presence of factor base $[a_x]$. Since the input parameters α, β and the range of $f(x)$ are public, it is easy to determine the sequence and factor base $[a_x]$. The array $[b_x]$ used for CRT is chosen randomly and it remains hidden. Since the chosen $[b_x]$ and k varies with each iteration of the key-generation algorithm, this adds additional randomness to the choice of primes.

Leaking the quadratic residuosity of elements in $\mathbb{Z}_{p^2}$. The public nature of α, β and the function $f(x)$ creates access to a limited oracle that provides information about quadratic residuosity $\mathsf{QR}(x, p)$ for t elements, where t is the function size. Due to this reason, the security of our system relies on a slightly weaker assumption than factoring $n = p^2q$. The alternate hardness assumption we propose here is factoring $n = p^2q$ in the presence of the quadratic residuosity oracle $\mathsf{QR}(x, p)$. Although the information provided by such an oracle is highly restricted relative to $\mathbb{Z}_n$, whether this information can be exploited to factorize n remains an open question and requires further cryptanalytic efforts.

6 Protocol

Protocol π is conducted between two semi-honest parties P_A and P_B. Let P_A input a vector of plaintexts $X = \{x_1, \ldots, x_a\}$ and P_B input a vector of plaintexts $Y = \{y_1, \ldots, y_b\}$ for $x_i, y_j \in \mathcal{M}$. Let $\mathsf{CS} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Smul}, \mathsf{Eval}\}$ be an additively homomorphic cryptosystem with the functionalities defined in Section 4. Let P_A be the holder of the private-key $\mathcal{SK}$.

Since our protocol is framed as an *extension* of existing protocols based on additive schemes (e.g., vector addition, weighted sums etc.). We begin by defining a sub-protocol π_{sub}, which capturing the existing protocol conducted on X and Y using the conventional functionalities $\{\mathsf{Enc}, \mathsf{Add}, \mathsf{Smul}\}$. Suppose the output of π_{sub} results in P_A receiving a single ciphertext $[\![m]\!]$ where m represents the nominal outcome of π_{sub}. Protocol π extends π_{sub} with the Eval functionality to homomorphically compute $f(m)$ given $[\![m]\!]$. The full protocol π is presented in Figure 1.

Public: Public-key $\mathcal{PK}$, sequence parameters $\{\alpha, \beta\}, f : \mathbb{Z}_t \mapsto \{0, 1\}$, the description of a secure sub-protocol π_{sub} invoking conventional additively homomorphic functionalities $\mathsf{Enc}, \mathsf{Add}, \mathsf{Smul}$.
Private Input (P_A): Plaintexts $X = \{x_1, \ldots, x_a\}$. Private key $\mathcal{SK} = \{p, q\}$.
Private Input (P_B): Plaintexts $Y = \{y_1, \ldots, y_b\}$.
Output: Given $[\![m]\!] \leftarrow \pi_{sub}(X, Y)$, P_A learns $f(m)$. P_B learns $\perp$.

- $\boldsymbol{P_A, P_B}$: Run $\pi_{sub}(X, Y)$. P_B receives $[\![m]\!]$ as output. P_A receives $\perp$.
- $\boldsymbol{P_B}$ **:** Compute:
$$c' \leftarrow \mathsf{Eval}(\mathcal{PK}, \alpha, \beta, [\![m]\!]).$$
- $\boldsymbol{P_B \rightarrow P_A}$ **:** P_B sends c' to P_A.
- $\boldsymbol{P_A}$ **:** Decrypt $c' \leftarrow \mathsf{Dec}(\mathcal{SK}.\mathsf{c}')$ to obtain $m' \leftarrow (\alpha \cdot m + \beta) \cdot r_c^2$.
- $\boldsymbol{P_A}$ **:** Compute $\mathsf{QR}(m', p) = f(m)$.

Fig. 1: Secure Function Evaluation Protocol π

Theorem 5. *The secure evaluation of function $f : \mathbb{Z}_t \mapsto \{0,1\}$ by the protocol π is correct.*

Proof. From Theorem 3 we previously established that $\mathsf{QR}(\mathsf{Dec}(\mathcal{SK}, c^{'}))$ returns $f(m)$ as

$$\mathsf{QR}(\mathsf{Dec}(\mathcal{SK}, c^{'}), p)) = f(m).$$

6.1 Participant privacy during the protocol π

Participant privacy while running the protocol π in two party setting guarantees that there is no inadvertent leakage of information in the presence of semi-honest adversaries. Such adversaries follow the protocol exactly but try to learn more information than allowed based on their respective inputs and any intermediate transcripts during the protocol execution. To formalize this idea, we adopt privacy by simulation approach by creating the view of the parties without the knowledge of any keys. Security proof is established by constructing a simulator S that generates a view for adversary that is computationally indistinguishable from its real view. Privacy by simulation requires certain notations to proceed further.

- $f = (f_{P_A}, f_{P_B})$ is the two-party functionality that is computed by the protocol π.
- The view of P_A and P_B denoted as $View^{\pi}_{P_A}$ and $View^{\pi} P_B$ are defined as:

$$\mathsf{View}^{\pi}_{P_A}(x) = (x_{P_A}, r_{P_A}, m_{P_B}),$$

$$\mathsf{View}^{\pi}_{P_B}(x) = (x_{P_B}, r_{P_B}, m_{P_A}).$$

  where, $x = (x_{P_A}, x_{P_B})$ represent the inputs of the participants to the protocol, $r = (r_{P_A}, r_{P_B})$ are the random values generated during the transaction and $m = (m_{P_A}, m_{P_B})$ are the messages sent by the respective parties during the protocol.
- We say that π securely computes f in the presence of a semi-honest adversary if we are able to construct the algorithms S_{P_B} to simulate P_B's view to establish P_A's privacy and S_{P_A} to simulate P_A's view to establish P_B's privacy.

***P_A*'s Privacy** For P_A's privacy, we need to establish that P_B's view is simulatable given P_B's input x_{P_B} and output $f_{P_B}(x_{P_A}, x_{P_B}) = \perp$. We have output by P_A denoted as $\mathsf{Output}^{\pi}_{P_A}$, which is a result of the execution of protocol π on the combined input from both the parties.

Theorem 6 (P_A's privacy). *There exists a probabilistic polynomial time algorithm S_{P_B} such that*

$$\big[S_{P_B}(x_{P_B}, \perp), f(x_{P_A}, x_{P_B})\big] \stackrel{c}{\equiv} \big[\mathsf{View}^{\pi}_{P_B}, \mathsf{Output}^{\pi}_{P_A}(x)\big].$$

where $\stackrel{c}{\equiv}$ indicates ciphertext indistinguishability.

Proof. The proof of P_A's privacy is simple because P_B only receives the ciphertext $c = \mathsf{Enc}(x)$, i.e., $m_{P_A} = c$. To simulate P_B's view the simulator just needs to sample the messages of the form $c \leftarrow \mathbb{Z}_n^*$. Due to the semantic security of the CS, it is easy to establish that a ciphertext $c = \mathsf{Enc}(x)$ is computationally indistinguishable from an element of $\mathbb{Z}_n^*$. S_{P_B} can now compute the output using public key $\mathcal{PK}$, c and P_B's input x_{P_B} and computing $f(x_{P_A}, x_{P_B})$ homomorphically, thereby simulating $View_{P_B}$. □

6.2 P_B's Privacy

We rely on the similar approach to the proof of P_A's privacy. We establish that View_{P_A} is simulatable given P_A's input denoted as x_{P_A}. Note that, the privacy proof relies on the hiding properties of blinding factor b. Let $QR, NR \subset \mathbb{Z}_n$ respectively denote the subsets of quadratic residues and non-residues modulo prime n. P_A decrypts ciphertext c', this results in random looking message $m' = (f(x_{P_A}, x_{P_B})\alpha + \beta) \cdot r_c^2 \bmod n$ where r_c is a uniform element modulo n and thus r_c^2 is uniform in QR.

Theorem 7 (P_B's privacy). *There exists a probabilistic polynomial time algorithm S_{P_A} such that*

$$\left[S_{P_A}(x_{P_A}, f(x_{P_A}, x_{P_B}))\right] \stackrel{c}{\equiv} \left[\mathsf{View}_{P_A}^{\pi}, \perp\right]$$

Proof. Algorithm S_{P_A} will begin by directly computing encryptions using $\mathcal{PK}$ and the P_A's input x_{P_A} and outputs $f(x_{P_A}, x_{P_B})$. S_{P_A} now computes the quadratic residuosity using $\mathsf{QR}(f(x_{P_A}, x_{P_B}))$ to check for the output, for 0 the decrypted plaintext would be of the form $m' \in NR$. S_{P_A} samples $m' \leftarrow_R NR$ for each $0 \leq j < k - 1$ and computes the corresponding ciphertexts $c = \mathsf{Enc}(m')$ using $\mathcal{PK}$. If $f(m) = 1$, then, plaintext would be of the form $m \in QR$. If $(f(x_{P_A}, x_{P_B})\alpha + \beta)$ is a quadratic residue (resp. non-residue), then the term $(f(x_{P_A}, x_{P_B})\alpha + \beta) \cdot r_c^2$ is indistinguishable from a random element in QR (resp. NR) as proved in [16], implying m' values are indistinguishable from a real-world plaintext $(f(x_{P_A}, x_{P_B})\alpha + \beta) \cdot r_c^2 \bmod p$. □

7 Results and discussion

7.1 Experimental setup

The following subsections describe the process of finding appropriate sequence parameters α, β, generation of prime p, and implementing the secure evaluation of a generic function using the protocol π. We implemented all the experiments in Python3 on a local Intel i7 quad-core processor @ 1.8GHz with 8 GB RAM.

Finding sequence parameters α, β Recall as part of the key generation algorithm in Section 4.1, our goal is to find some linear sequence defined by $s_m = \alpha m + \beta$ and some prime p for which the Legendre symbol of s_m modulo p matches a given Boolean function evaluated at $f(m)$ over $0 \leq m < t$.

For this to be true for all possible Boolean functions f, we require the residue symbols of each s_m to be, in essence, independently programmable. This is not possible for most sequences. For example, the factorization of elements of the sequence $3m + 2$ is: $2, 5, 2^3, 11, \ldots$. Here, no matter what prime p is chosen, the Legendre symbol of s_0 is always the same as s_2. Conversely, the factorization of $4m + 3$ is: $3, 7, 11, 3 \cdot 5, \ldots$. Here, no matter the Legendre symbol of s_0, the symbol for s_3 can be set independently by choosing a prime p for which 5 has the necessary symbol to give s_3 the required symbol to match $f(3)$.

In general, we can independently "program" each symbol of the sequence so long as each s_m contains at least one unique odd-powered prime factor. This condition is simple and sufficient, although not strictly necessary (cf. steps 3–5 of Section 4.1)

We took a brute-force approach to finding sequence parameters α, β for different values of t. We began with the smallest step size (α) and starting point (β), incrementing β across a heuristically chosen intervals at each function size t. For example, at $t = 512$, we searched in the range $\alpha < 6000, \beta < 100$ and recorded the parameters yielding the minimal bit-length $|p|$. These parameters are not optimal. The goal was to demonstrate practical, concrete values of α, β. Finding more efficient algorithms and computing optimal bounds on $|p|$ is left to future work.

For each candidate α/β, we generated the sequence $[\alpha x + \beta]$ for $0 \leq x < t$. We factored each sequence element, and for each prime factor $(s_m)^{e_m}$ for which e_m is odd, we added s_m to the set factors A as defined in Step 3 of Section 4.1. Let the product of this set be $\prod A$. Since $p > \prod A$ (see step 8 in Section 4.1), we set $|p| = \lceil \log_2(\prod A) \rceil$ representing the lower bound of $|p|$. For each domain cardinality t, we report the α, β leading to the smallest $|p|$ found in our search. For example, the linear sequence formed by $\alpha = 342, \beta = 787$ contains unique factors sufficient to produce sub 3000-bit primes for evaluating 8-bit Boolean functions. See Table 1 for our experimentally found sequence parameters.

Generating prime p Once suitable α, β are found, the steps to find p are implemented according to the key-generation algorithm described in Section 4.1. We ran our implementation of the key generation function Gen for various function sizes. Our results are displayed in Table 2. As we can see, the most time among all the steps is taken by iteratively finding the right set of b_x's to generate the prime. To speed up this step, we built a look up table containing sets of all quadratic residues and non-residues with respect to each a_x and for each iteration, the suitable b_x is randomly chosen. Additionally, the CRT step has to be computed each time we find a new set of b_x's. CRT has complexity $O((S_1+S_2)^2)$, where S_i denotes the number of digits in the modulus we are trying to solve, which is a product of all the moduli present in the system of congruences. So the

Domain cardinality t	α	β	$\lvert p \rvert$ (bits)
8	2	27	46
16	6	29	97
32	20	53	263
64	84	305	719
128	90	197	1218
256	342	787	2858
512	1938	31	7066

Table 1: Sample sequence parameters α, β.

larger the number of congruences to be solved, the greater is its time taken and computational complexity. The time taken for CRT as displayed in the table 2 is for a single round of CRT computation once the right set of b_x's are found.

Function size (domain cardinality t)	512	256	128	50
Gaussian Elimination	0.236	0.078	0.015	0.004
Test for consistency	0.016	0.009	0.002	0.001
Finding the right b_x	87.30	21.00	3.900	0.560
CRT	25.24	3.6	0.142	0.062

Table 2: Run time for various steps in the the key generation in seconds

We also performed the comparison of our approach with that of the secure function evaluation system introduced in [16]. Whereas the previous work focused on a specific function class (thresholds), our approach works across the entire class of Boolean functions, and at larger domain sizes. In fact, due to the usage of pre-determined α, β values, generating the right prime for the largest function size of 512 took less than 2 minutes, including all the steps. The search-based approach introduced in [16] takes more than 20 minutes to produce a sequence of size 26.

7.2 An example case of secure function evaluation using π

Our scheme has several potential applications that involve secure function evaluation. Specifically, our scheme aims to eliminate the need to display intermediate computations to either party involved in the transaction. For example: in case of similar patients query, the state of the art approaches [3, 9, 25, 28, 31] using

either homomorphic encryption or other multiparty computation techniques require several communication rounds to retrieve the records of patients sharing similar genetic makeup. To reduce the communication overhead, they display the similarity score for each record directly to the querying party leading to regression based database re-identification attacks. There is a scope for such attacks in other applications such as secure machine learning inference, especially in classification problems. Our scheme can be applied to display the class labels while hiding intermediate scores. To test the performance of our parameters, we implemented a simple secure function evaluation on threshold functions using the protocol described in Section 6. The threshold function denoted as $\tau_t(x)$ is similar to that of the one implemented in [16]. Note that we implemented the same threshold function to demonstrate the efficiency of our protocol. However, we can implement any function with a boolean co-domain using the same protocol. We adopt the definition for $\tau_t(x)$ from [16] where

$$\tau_t(x) = \begin{cases} 1 & \text{if } x \geqslant t \\ 0 & \text{Otherwise.} \end{cases}$$

The range of a threshold function can be represented as $\{0, 0, 0, ..., 1, 1, 1\}$. In other words, all the values below the threshold are mapped to 0 and above the threshold are mapped to 1. Using this range with a maximum length of k and a fixed threshold value t, the inputs to Gen would be

$$\alpha,\ \beta,\ f(x) = \{0\}_{x=0}^{t} \parallel \{1\}_{x=t+1}^{k}.$$

Once the prime number p is generated based on these parameters, we use this to build our cryptosystem CS as follows:

- We generate q such that q is a large prime and compute $n = p^2q$,
- Compute $g \in 2, \cdots, n-1$ such that $g^{p-1} \not\equiv 1 \bmod p^2$ and $h = g^n \bmod n$,
- $\mathcal{PK} = (\alpha, \beta, n, g, h)$ and $\mathcal{SK} = (p, q)$.

Similar to the work in [16], the protocol uses Dice coefficient as a similarity metric to perform linkage between two datasets that consist of user names. The records are linked approximately to address any variations or errors in the strings being compared. For such an approach to work, the records will be considered a match if the Dice coefficient between two strings is above a threshold value. The protocol is performed between two parties P_A, P_B and is described below briefly. The details for sub-protocols 1 and 2 can be referred from [16] as the steps are the same. The difference with our protocol is during computing the evaluation function, which is in Step 6 of sub-protocol 2 in [16]. Particularly, Protocol 1 in [16] is modified as follows:

- **Public parameters**: $\mathcal{PK}, \alpha, \beta$ and for a threshold value t maximum set cardinality μ, where $\mu = t$
- **Private parameters**: Party P_A holds a list of strings $[a_1, \ldots, a_n]$ and private keys. Party P_B holds a list of strings $[b_1, \ldots, b_n]$

– **Protocol**:
  - P_A, P_B produce set intersection cardinalities between the private inputs using sub-protocol 1 from [16]
  - By modifying the final step in sub-protocol 2 in [16] both parties compute threshold dice coefficient d_{ij} such that $d_{ij} = \mathsf{Eval}(\mathcal{PK}, \alpha, \beta, [\![\theta_{\ell_b}]\!]) = (([\![\theta_{\ell_b}]\!])^{\alpha} \cdot \beta)^{r_c^2}$
– **Output:** For all threshold dice coefficient values, if $\mathsf{QR}(\mathsf{Dec}(\mathcal{SK}, d_{ij}), p) = 1$, P_A outputs the index.

For the particular case of threshold functions, due to the increase in domain size of the function evaluated that is leading up to 8-bit numbers, we can compute the dice-coefficient for more precise threshold values. The results of our implementation are summarized in the Table 3 in comparison with other similar studies that rely on quadratic residue symbols for secure function evaluation. It can be observed that, due to the ability of our key generation algorithm to generate primes as per the $f(x)'$s range, we can compute any kind of functions securely unlike the approaches in [1, 16, 30].

Performance Indicator	Noisy Legendre Symbol [1]	Yu's Protocol [30]	Residue PHE [16]	Our Protocol
Domain cardinality (t)	623	$\Omega(log(p))$	26	512
Residue symbol sequence type	$\{1\}^t$	$\{1\}^t$	$[0]^t \mid\mid [1]^t$	$\{0,1\}^t$
Secure function evaluation type	Specific (sign functions)	Specific (sign functions)	Specific (thresholds)	General (Boolean)

Table 3: Comparison between secure function evaluation protocols that rely on the runs of quadratic residues.

8 Conclusion

This paper discusses a method to extend the functionality of additively homomorphic schemes in applications where the encrypted sum is below a threshold t based on chosen patterns on quadratic residues modulo a prime. We developed a novel algorithm to encode such patterns into the private keys of cryptosystems in the integer factorization setting. We presented a protocol with concrete parameterizations for efficiently evaluating arbitrary Boolean functions on encrypted sums up to $t = 512$.

Future work will seek to push the domain cardinality t to higher values and will also explore the possibility of integrating of this technique in the fully homomorphic setting.

References

1. Abspoel, M., Bouman, N.J., Schoenmakers, B., de Vreede, N.: Fast secure comparison for medium-sized integers and its application in binarized neural networks. In: Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings. pp. 453–472. Springer (2019)
2. Adida, B.: Helios: Web-based open-audit voting. In: USENIX security symposium. vol. 17, pp. 335–348 (2008)
3. Asharov, G., Halevi, S., Lindell, Y., Rabin, T.: Privacy-preserving search of similar patients in genomic data. Proceedings on Privacy Enhancing Technologies **2018**(4), 104–124 (2018)
4. Atallah, M.J., Wagstaff Jr, S.S.: Watermarking with quadratic residues. In: Security and Watermarking of Multimedia Contents. vol. 3657, pp. 283–288. International Society for Optics and Photonics (1999)
5. Brauer, A.: Combinatorial methods in the distribution of k-th. power residues. Combinatorial mathematics and its applications pp. 14–37 (1969)
6. Buell, D.A., Hudson, R.H.: On runs of consecutive quadratic residues and quadratic nonresidues. BIT Numerical Mathematics **24**(2), 243–247 (1984)
7. Burgess, D.A.: The distribution of quadratic residues and non-residues. Mathematika **4**(2), 106–112 (1957)
8. Carella, N.: Consecutive quadratic residues and quadratic nonresidue modulo p. arXiv preprint arXiv:2011.11054 (2020)
9. Cheng, K., Hou, Y., Wang, L.: Secure similar sequence query on outsourced genomic data. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp. 237–251 (2018)
10. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of IEEE 36th Annual Foundations of Computer Science. pp. 41–50. IEEE (1995)
11. Damgard, I., Geisler, M., Kroigaard, M.: Homomorphic encryption and secure comparison. International Journal of Applied Cryptography **1**(1), 22–31 (2008). https://doi.org/10.1504/IJACT.2008.017048
12. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for online auctions. In: Australasian conference on information security and privacy. pp. 416–430. Springer (2007)
13. Damgård, I.B.: On the randomness of legendre and jacobi sequences. In: Conference on the Theory and Application of Cryptography. pp. 163–172. Springer (1988)
14. Davenport, H.: On the distribution of quadratic residues (mod p). Journal of the London Mathematical Society **1**(1), 49–54 (1931)
15. Egidi, L., Manzini, G.: Better spaced seeds using quadratic residues. Journal of Computer and System Sciences **79**(7), 1144–1155 (2013)
16. Essex, A.: Secure approximate string matching for privacy-preserving record linkage. IEEE Transactions on Information Forensics and Security **14**(10), 2623–2632 (2019)
17. Feige, U., Killian, J., Naor, M.: A minimal model for secure computation. In: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. pp. 554–563 (1994)
18. Frikken, K., Atallah, M., Zhang, C.: Privacy-preserving credit checking. In: Proceedings of the 6th ACM conference on Electronic commerce. pp. 147–154 (2005)
19. Gauss, C.F.: Disquisitiones arithmeticae, vol. 157. Yale University Press (1966)

20. Green, B., Tao, T.: The primes contain arbitrarily long arithmetic progressions (2004)
21. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31–June 4, 1998 Proceedings 17. pp. 308–318. Springer (1998)
22. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
23. Peralta, R.: On the distribution of quadratic residues and nonresidues modulo a prime number. Mathematics of Computation **58**(197), 433–440 (1992)
24. Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A hybrid secure computation framework for machine learning applications. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. p. 707–721. ASIACCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3196494.3196522, https://doi.org/10.1145/3196494.3196522
25. Salem, A., Berrang, P., Humbert, M., Backes, M.: Privacy-preserving similar patient queries for combined biomedical data. Proceedings on Privacy Enhancing Technologies **2019**(1), 47–67 (2019)
26. Sárközy, A.: On finite pseudorandom binary sequences and their applications in cryptography. Tatra Mt. Math. Publ **37**, 123–136 (2007)
27. Sárközy, A., Stewart, C.: On pseudorandomness in families of sequences derived from the legendre symbol. Periodica Mathematica Hungarica **54**(2), 163–173 (2007)
28. Schneider, T., Tkachenko, O.: Episode: Efficient privacy-preserving similar sequence queries on outsourced genomic databases. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 315–327 (2019)
29. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167. IEEE (1986)
30. Yu, C.H.: Sign modules in secure arithmetic circuits. Cryptology ePrint Archive (2011)
31. Zhu, R., Huang, Y.: Efficient privacy-preserving general edit distance and beyond. IACR Cryptology ePrint Archive **2017**, 683 (2017)

A Proof of Theorem 1

We present the proof of Theorem 1 below:

Proof. The proof proceeds in two parts. In the first part, we prove the existence of an integer p' containing the chosen residue symbol sequence. In the second part, we prove the existence of p' implies the existence of a prime p with the same properties. Because each a_x is prime, each $0 < b_x < a_x$ is to be co-prime to a_x, Chinese Remainder Theorem guarantees the existence of a unique solution p' to the system of congruences formed by $[a_x], [b_x]$:

$$\begin{aligned} p' &\equiv b_0 \bmod a_0 \\ &\vdots \\ p' &\equiv b_t \bmod a_t. \end{aligned}$$

Since

$$\left(\frac{b_x}{a_x}\right) = \ell_x, \text{ and } p' \equiv b_x \bmod a_x,$$

for each $1 \leq x < t$ we have

$$\left(\frac{p'}{a_x}\right) = \ell_x.$$

Now we show the existence of an integer p' implies the existence of a prime p with the same congruences. Since $p \equiv p' \bmod A_{prod}$, and therefore $p \equiv b_x \bmod a_x$, then

$$\left(\frac{p}{a_x}\right) = \ell_x.$$

Finally, since p' is relatively prime to A_{prod}, Dirichlet's theorem guarantees there are infinitely many primes of the form $kA_{prod} + p'$. □