Secure Equality Testing Protocols in the Two-Party Setting

Majid Nateghizad
Cyber Security Group, Department of Intelligent Systems
Delft University of Technology, The Netherlands
m.nateghizad@tudelft.nl

Thijs Veugen*
Unit ICT, TNO
The Netherlands
thijs.veugen@tno.nl

Zekeriya Erkin
Cyber Security Group, Department of Intelligent Systems
Delft University of Technology, The Netherlands
z.erkin@tudelft.nl

Reginald L. Lagendijk
Cyber Security Group, Department of Intelligent Systems
Delft University of Technology, The Netherlands
r.l.lagendijk@tudelft.nl

ABSTRACT

Protocols for securely testing the equality of two encrypted integers are common building blocks for a number of proposals in the literature that aim for privacy preservation. Being used repeatedly in many cryptographic protocols, designing efficient equality testing protocols is important in terms of computation and communication overhead. In this work, we consider a scenario with two parties where party A has two integers encrypted using an additively homomorphic scheme and party B has the decryption key. Party A would like to obtain an encrypted bit that shows whether the integers are equal or not but nothing more. We propose three secure equality testing protocols, which are more efficient in terms of communication, computation or both compared to the existing work. To support our claims, we present experimental results, which show that our protocols achieve up to 99% computation-wise improvement compared to the state-of-the-art protocols in a fair experimental set-up.

CCS CONCEPTS

• **Theory of computation → Cryptographic protocols**;

KEYWORDS

Processing encrypted data, equality test, homomorphic encryption, privacy, efficiency.

ACM Reference Format:
Majid Nateghizad, Thijs Veugen, Zekeriya Erkin, and Reginald L. Lagendijk. 2018. Secure Equality Testing Protocols in the Two-Party Setting. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3230833.3230866

*He also works in Department of Cryptology, Centum Wiskunde & Informatica, The Netherlands

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3230866

1 INTRODUCTION

Processing encrypted data has been addressed in several fields, e.g. biometric data matching [26], recommender systems [15], data mining [32] and data aggregation [19], as it enables collaborating with an untrustworthy service provider to process privacy-sensitive data. The main idea is to provide only the encrypted version of the data to the service provider and invoke interactive cryptographic protocols with the decryption key owner to perform the desired algorithm. While being very secure regarding protecting the privacy-sensitive data without hampering the service, processing encrypted data introduces a significant amount of computational and communication overhead compared to performing the same algorithm with unencrypted data. In the literature, it is suggested to design custom-tailored cryptographic protocols, rather than applying generic solutions, to improve the efficiency of the privacy-preserving version of the algorithm. Building blocks with encrypted data for those algorithms like comparison, division, and equality checks [21] need to be designed with high efficiency. The main reason is that these core operations are repeated in large quantities in conventional data processing algorithms. For example, finding similar users in a system with millions of users to a particular one based on his or her taste for movies [15] requires comparison of similarity scores linear in the number of users in the system. Testing the equality of two encrypted integers is one of the widely-used operations, e.g. for searching in encrypted databases. Other applications for equality testing protocols also include, but are not limited to, secure pattern matching [13], secure linear algebra [7], and encryption switching protocols [4].

Yang et al. [31] introduced the first public key encryption that supports testing equality (PKwET). Their work allows checking whether two ciphertexts encrypted under the same or different keys are encryptions of the same value. Tong[28] introduced a protocol to enable equality testing for authorized users. Later works tried to improve the performance or functionality of PKwET [14, 18, 30]. However, existing PKwET proposals leak the result of the equality test to the service provider, which is usually a semi-trusted remote computation server.

Secure multi-party computation (MPC) is another approach to design algorithms for secure equality checking. In such protocols, two or more parties jointly compute an agreed function of their secret inputs. Many works have been introduced to show that any function can be computed securely using MPC [3, 5, 6]. Nishide and Ohta proposed a probabilistic constant-round equality test protocol

[23], where the Jacobi symbol is used to test quadratic residuosity of a value. Although the proposed protocol, (NO07), is efficient regarding computation, the result of the protocol is probabilistic: with a probability of 50%, the protocol returns a correct answer. The protocol is suggested to be executed φ times to minimize the error probability to $2^{-\varphi}$. As it is necessary to pick large values for φ to reduce the false positive rate, the protocol becomes computation and communication wise demanding.

Schoenmakers and Tuyls [27] has presented a method (ST06) to check the equality of two encrypted integers by using a protocol based on bit-decomposition. However, this protocol is expensive regarding the number of communication rounds. In [21], Lipmaa and Toft have introduced an equality test protocol (LT13) on top of an arithmetic black box [11]. The protocol uses Lagrange interpolation similar to [8]. A multiplicative masking is used in [8], a similar idea from [1], since its realization is easier than additive masking. However, multiplicative masking is not computationally efficient in a two-party setting because the size of the exponent can be very large. However, it is important to note that LT13 is more efficient when there are more than two parties in the setting. The works [27] and [21] are both secure against active adversaries.

There are also efficient secure equality testing protocols based on Garbled circuits (GC) [16, 17]. In [17], Kolesnikov et al. propose an efficient construction that enables XOR to be evaluated for free. Then, in [16], evaluation of MPC protocols (VAT09) using free XOR technique has improved the computational efficiency of garbled circuits up to 50%. However, computational efficiency was achieved at the cost of significant communication and pre-computation overhead. Moreover, using GC results in additional computation and communication overhead for converting encryptions to garbled circuit inputs as explained in [16].

In this paper, our aim is to check whether two encrypted values under an additively homomorphic encryption scheme such as Paillier [24] are equal or not. More precisely, party A has the encryption of two ℓ-bit integers, $[a]$ and $[b]$, where $[\cdot]$ denotes the encryption, and party B has the decryption key. Neither party is allowed to learn the outcome bit ϑ. The bit ϑ is one, exactly when $a = b$, and zero otherwise, similar to the other existing works [21, 23, 27]. We assume $0 \leq a, b < 2^{\ell}$.

We propose three secure equality testing protocols, namely EQT-1, EQT-2 and EQT-3. As it is shown in Figure 1, there is a trade-off between computation and communication cost. For the sake of clarity, we present a summary of our protocols:

- **EQT-1** is based on a coin toss where the results determines either performing a secure Hamming distance computation for the two inputs or invoking a secure joint function [10]. The resulting protocol has the least communication overhead among all other proposals. While it is computationally more expensive than our other two proposal, it is computationally 97% more efficient than the state-of-the-art and requires 24% less communication overhead.
- **EQT-2** relies on computing the Hamming distance and the secure comparison protocol [22]. The protocol has a balanced computational and communication overhead. EQT-2 is computationally 38% more efficient than EQT-1 with 25% more communication overhead.

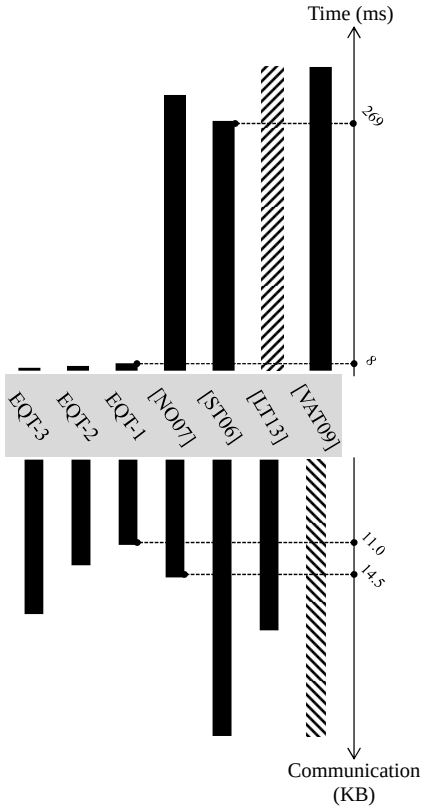


Figure 1: A summary of secure equality testing protocols' performance for 20-bit inputs. Dashed bar denotes that the value is very large and does not fit in the graph.

- **EQT-3** is using Lagrange interpolation that has the best overall computational performance and 99% more efficient than the state-of-the-art. EQT-3 is computation-wise over 30% more efficient than EQT-2, but it has 43% more communication overhead compared to EQT-2.

2 PRELIMINARIES

The symbols and their description are listed in Table 1.

2.1 Security Setting

We consider the semi-honest security model [12], where both parties are assumed to be honest in following the protocol description, while they are curious to obtain more information than they are entitled to. In this setting, it is assumed that A and B do not collude.

2.2 Homomorphic Encryption

We use two additively homomorphic and semantically secure encryption schemes, namely Paillier [25] and DGK [10]. In an additively homomorphic encryption scheme, multiplying two ciphertexts $\mathcal{E}_{pk}(m_1)$ and $\mathcal{E}_{pk}(m_2)$ results in a ciphertext, whose decryption is the sum of two plaintexts m_1 and m_2: $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m_1) \cdot \mathcal{E}_{pk}(m_2)) = (m_1 + m_2) \bmod n$, where n is the encryption system modulus. Consequently, exponentiation of any ciphertext with a public integer

Table 1: List of symbols

Symbol	Description	Symbol	Description
a, b	input messages	δ	comparison result
ℓ	input bit size	n	crypto modulus
$\oplus$	exclusive or	κ	security parameter
r, ρ, s, w	randoms	φ	error controller [23]
$[.]$	Paillier cipher	$[\![.]\!]$	DGK cipher
A, B	stack holders	x_i	i^{th} bit of integer x
$\mathbb{Z}_u$	DGK plaintext space	$d(a,b)$	Hamming distance of a and b
p, q, v_p, v_q	primes	t	DGK parameter
ϑ	equality result	sk	private key
pk	public key	$\mathcal{E}_{pk}$	encryption
$\mathcal{D}_{sk}$	decryption	$\log \ell$	$\lceil \log_2 \ell \rceil$

value k yields the encrypted product of the original plaintext and the public value: $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)^k) = (k \cdot m) \bmod n$.

Paillier. The Paillier encryption [25] for a given message $m \in \mathbb{Z}_n$ is defined as $\mathcal{E}_{pk}(m, r) = g^m \cdot r^n \mod n^2$, where n is the product of two distinct large prime numbers p and q, ciphertext $\mathcal{E}_{pk}(m, r) \in \mathbb{Z}^*_{n^2}$, r is a random number from $\mathbb{Z}^*_n$, and g is a generator of $\mathbb{Z}^*_n$. The public key is (g, n), and the private key is (p, q). This encryption scheme is additively homomorphic. The security of Paillier is based on hardness of computing n^{th} residue classes. For the decryption operation, we refer readers to [25].

DGK. The DGK cryptosystem [10] is used in this work for two reasons: 1) it is more efficiency than Paillier in term of computation and communication since it has much smaller ciphertext size, 2) it enables checking whether a ciphertext is an encryption of zero without performing the expensive decryption operation, which can save computation. Note that DGK decryption is very expensive for large inputs since it uses a look-up table.

The process of generating the keys is as follows: 1) choose two distinct t-bit prime numbers v_p, v_q, 2) construct two distinct prime numbers p and q, where $v_p | (p-1)$ and $v_q | (q-1)$ such that $n = pq$ is a k-bit RSA modulus, 3) choose u, the smallest possible prime number but greater than $\ell + 2$, 4) choose a random r that is a $2.5t$-bit integer, and 5) choose g and h such that $\text{ord}(g) = uv_pv_q$ and $\text{ord}(h) = v_pv_q$, where $\ell < t < k$. The public and the private keys are $pk = (n, g, h, u)$ and $sk = (p, q, v_p, v_q)$, respectively.

In the rest of the paper, we denote the ciphertext of a message m by $[m]$ for the Paillier cryptosystem and $[\![m]\!]$ for the DGK. We also omit the modular reductions, when describing the computational steps, for simplicity.

3 OUR PROTOCOLS

3.1 Equality Testing Protocol (EQT)-1

In this protocol, described in Protocol 1, we use the idea that computes either the Hamming distance between two encryptions or performs a secure comparison based on the idea from the DGK comparison protocol [10], after a coin toss. The reason for this coin toss is as follows: calculating the Hamming distance is less expensive regarding computation and communication overhead. However, only using Hamming distance for equality check leaks information. Assume that only the Hamming distance is used for testing the equality of a and b. Then, party B learns whether $a = b$ after performing the DGK zero-check, since the Hamming distance is always zero, precisely when $a = b$. Party B acquiring this information is not desired since we do not want Party A and B to learn any information. Therefore, we toss a coin and we either compute the Hamming distance or perform secure comparison, hiding what is being computed from Party B.

Protocol 1 EQT-1

1: Party A generates a sufficiently large ($\ell + 1 + \kappa$ bits) random value r, computes $[x] \leftarrow [a - b + r]$, and sends $[x]$ to B.
2: Party B decrypts $[x]$, computes the first ℓ bits x_i, $0 \le i < \ell$, encrypts them separately with DGK (for efficiency reason), and sends $[\![x_i]\!]$ to A.
3: Party A computes $[\![r_i \oplus x_i]\!]$ for $0 \le i < \ell$, by distinguishing $r_i = 0$ and $r_i = 1$. If $r_i = 0$, $[\![r_i \oplus x_i]\!] \leftarrow [\![x_i]\!]$, else, $[\![r_i \oplus x_i]\!] \leftarrow [\![1]\!] \cdot [\![x_i]\!]^{-1}$.
4: Party A tosses a random coin $\delta_A \in \{0, 1\}$.
5: **if** $\delta_A = 0$ **then**
6: Party A computes $[\![c_0]\!] \leftarrow \prod_{i=0}^{\ell-1} [\![r_i \oplus x_i]\!]$, and multiplicatively blinds it with a large (in case of DGK, the random number ρ should contain $2t$ bits) random number ρ.
7: Party A generates $\ell - 1$ non-zero random integers c_i, $1 \le i < \ell$, and encrypts them.
8: **else**
9: Party A computes $[\![c_i]\!] \leftarrow [\![-1]\!] \cdot [\![r_i \oplus x_i]\!] \cdot (\prod_{j=i+1}^{\ell-1} [\![r_j \oplus x_j]\!])^2$, for $0 \le i < \ell$.
10: Party A generates ℓ large (in case of DGK, the random numbers ρ_i should be $2t$ bits) non-zero random numbers ρ_i, $0 \le i < \ell$, and uses them to multiplicatively blind c_i: $[\![c_i]\!] \leftarrow [\![c_i]\!]^{\rho_i}$.
11: **end if**
12: Party A randomly permutes the order of the $[\![c_i]\!]$, $0 \le i < \ell$, and sends them to party B.
13: Party B decrypts (in case of DGK, a DGK zero-check is sufficient) the numbers $[\![c_i]\!]$ to find whether one of them is 0. If (at least) one of them is 0, party B sets $\delta_B \leftarrow 1$, otherwise $\delta_B \leftarrow 0$.
14: Party B encrypts δ_B, and sends it to A.
15: Party A computes $[\vartheta]$, by distinguishing $\delta_A = 0$ and $\delta_A = 1$. If $\delta_A = 0$, $[\vartheta] \leftarrow [\delta_B]$, else, $[\vartheta] \leftarrow [1] \cdot [\delta_B]^{-1}$.

Correctness. Since $0 \le a, b < 2^\ell$, we have $r - 2^\ell < x = a - b + r < r + 2^\ell$, so it is sufficient to check whether the first (least significant) ℓ bits of x and r are equal. We know $\sum_{i=0}^{\ell-1} r_i \oplus x_i \ge 0$, with equality precisely when $a = b$. In case $\delta_A = 0$, we have $c_0 = \sum_{i=0}^{\ell-1} r_i \oplus x_i$, so the (blinded) numbers c_i, $0 \le i < \ell$ will have exactly one zero, precisely when $a = b$, and will all be non-zero, otherwise. Therefore, if $\delta_B = 1$, we have $\delta_A \oplus \delta_B = 1 = (a = b)$. The case $\delta_B = 0$ is similar. If $\delta_A = 1$, party A computes $c_i = -1 + r_i \oplus x_i + 2 \sum_{j=i+1}^{\ell-1} r_j \oplus x_j$. If all $r_i \oplus x_i = 0$, then all $c_i = -1$, and $\delta_B = 0$. Otherwise, precisely

one of the c_i will be zero [9], and $\delta_B = 1$. Therefore, in both cases $(a = b) \leftarrow \delta_A \oplus \delta_B$.

Optimization. Although the decryption of $[x]$ and the multiplicative blindings dominate the computational complexity, the number of multiplications for computing the $[\![c_i]\!]$ can be reduced further.

Instead of computing $c_i \leftarrow -1 + r_i \oplus x_i + 2\sum_{j=i+1}^{\ell-1} r_j \oplus x_j$, we can use the optimization introduced in [22]. Parties can compute $c_i \leftarrow -1 + r_i + x_i + \sum_{j=i+1}^{\ell-1} 2^j(r_j - x_j) = (-1 + r_i + \sum_{j=i+1}^{\ell-1} 2^j r_j) + (x_i - \sum_{j=i+1}^{\ell-1} 2^j x_j) = c_i^A + c_i^B$, where c_i^A can be computed in clear by party A, and c_i^B by party B. Then, for each i, $0 \le i < \ell$, it requires only one multiplication to compute $[\![c_i]\!] \leftarrow [\![c_i^A]\!] \cdot [\![c_i^B]\!]$, which party A has to perform in case $\delta_A = 1$. These modified c_i's retain the property that precisely one of them is zero, if and only if, $a \neq b$.

In case $\delta_A = 0$, party A can use the same $[\![c_0^B]\!]$ to compute an encryption of $c_0 \leftarrow (-r_0 + \sum_{j=1}^{\ell-1} 2^j r_j) + c_0^B$ instead of $c_0 \leftarrow \sum_{i=0}^{\ell-1} r_i \oplus x_i$, which requires computing $c_0^A \leftarrow -r_0 + \sum_{j=1}^{\ell-1} 2^j r_j$ in clear. Also for this c_0, we have $c_0 = 0$, if and only if $a = b$. This optimization avoids computing $[\![r_i \oplus x_i]\!]$ by party A, and reduces the computation of each $[\![c_i]\!]$ to one multiplication.

3.2 Equality Testing Protocol (EQT)-2

Our proposal also uses the Hamming distance of a and b, which is $d(a,b) \leftarrow \sum_{i=0}^{\ell-1} a_i \oplus b_i$, to determine ϑ. As described in Protocol 2, it works as follows: (1) party A computes $[x]$, (2) party A computes the encrypted Hamming distance between $x \bmod 2^\ell$ and $r \bmod 2^\ell$, and (3) party A and B use a secure comparison protocol to compute the encryption of $(d > 0)$. If $d > 0$, then $a \neq b$, and $a = b$, otherwise. Note that there is a fundamental difference with EQT-1, which also computes Hamming distance: EQT-2 computes the Hamming distance of the two encrypted inputs, invokes a secure comparison protocol, namely EPPCP from [22], which returns an encrypted result. Therefore, there is no information revealed to Party B. The choice of using EPPCP is based on its high performance. The DGK comparison protocol is not preferred as the message space too large to create a look-up table.

Protocol 2 EQT-2

1: Party A generates a sufficiently large ($\kappa + \ell + 1$ bits) random value r, computes $[x] \leftarrow [a - b + r]$, and sends $[x]$ to B.
2: Party B decrypts $[x]$, computes the first ℓ bits x_i, $0 \le i < \ell$, and their sum $X \leftarrow \sum_{i=0}^{\ell-1} x_i$, encrypts them all separately, and sends them to A.
3: Party A computes $[d] \leftarrow [\sum_{i=0}^{\ell-1} r_i \oplus x_i] = [\sum_{i=0}^{\ell-1} r_i] \cdot [X] \cdot (\prod_{i=0, r_i=1}^{\ell-1} [x_i])^{-2}$.
4: Two parties jointly run the comparison protocol (EPPCP) [22], where party A receives [1] if $[0 < d]$, [0] otherwise, while both parties learn nothing about the inputs and the relation between $[d]$ and [0]. The ones' complement of the result of EPPCP is simply the result of the equality test, $[\vartheta] \leftarrow [1] - EPPCP([0],[d])$.

Correctness. Since $x - r = a - b$, and $0 \le a, b < 2^\ell$, then $a = b$ if $d = d(x \bmod 2^\ell, r \bmod 2^\ell) = 0$. To check if $a = b$, party A and party B jointly run EPPCP, where *EPPCP* returns (encrypted) zero when $a = b$, and (encrypted) one when $a \neq b$.

Optimization. In EQT-2, we first securely compute $d \leftarrow d(a,b)$, $0 \le d \le \ell$, and afterward securely compute $\delta \leftarrow (d > 0)$. The complexity of EPPCP depends on the size of its inputs. To reduce this size, we can add an additional communication round to securely compute $\acute{d} \leftarrow d(d,0)$, $0 \le \acute{d} < \log_2 \ell$, and securely compare $\acute{d}$ with 0. This way of reducing the input size can be repeated many times, which reduces the computation effort at the cost of increasing the number of communication rounds.

3.3 Equality Testing Protocol (EQT)-3

Previously presented protocols rely on secure comparison or efficient zero-check of the DGK encryption scheme. Unlike the other two protocols, EQT-3, described in Protocol 3, is a protocol that does not require zero-checking or secure comparison. The main idea of EQT-3 is to first compute the Hamming distance e between $x \bmod 2^\ell$ and $r \bmod 2^\ell$ similar to the other protocols. Note that $0 \le e \le \ell$, where ℓ is the bit-length of the inputs a and b. To make the range of e smaller (later we show that the smaller range of e results in a significant more efficient protocol), we mask $[e]$ with a large number number w, $[y] \leftarrow [e + w]$. Afterward, we compute the Hamming distance d between $y \bmod 2^{\log_2 \ell}$ and $w \bmod 2^{\log_2 \ell}$, where $0 \le d \le \log_2 \ell$. Finally, we generate and compute a polynomial that maps $d = 0$ to 1, and $d \in \{1, 2, \ldots, \log_2 \ell\}$ to 0.

Protocol 3 EQT-3

1: Party A generates a sufficiently large ($\ell + 1 + \kappa$ bits) random value r, computes $[x] = [a - b + r]$, and sends $[x]$ to B.
2: Party B decrypts $[x]$, computes the first ℓ bits x_i, $0 \le i < \ell$, and their sum $X = \sum_{i=0}^{\ell-1} x_i$, encrypts them separately, and sends them to A.
3: Party A computes the first ℓ bits of r to derive $[e] = [\sum_{i=0}^{\ell-1} r_i \oplus x_i] = [\sum_{i=0}^{\ell-1} r_i] \cdot [X] \cdot (\prod_{i=0, r_i=1}^{\ell-1} [x_i])^{-2}$.
4: Party A generates a sufficiently large ($\log_2 \ell + \kappa$ bits) random value w, computes $[y = e + w]$, and sends $[y]$ to B.
5: Party B decrypts $[y]$, computes the first $\log_2 \ell$ bits y_i, $0 \le i < \log_2 \ell$, and their sum $Y = \sum_{i=0}^{(\log_2 \ell)-1} y_i$, encrypts them separately, and sends them to A.
6: Party A computes the first $\log_2 \ell$ bits of w to derive $[d] = [\sum_{i=0}^{(\log_2 \ell)-1} w_i \oplus y_i] = [\sum_{i=0}^{(\log_2 \ell)-1} w_i] \cdot [Y] \cdot (\prod_{i=0, w_i=1}^{(\log_2 \ell)-1} [y_i])^{-2}$.
7: Party A generates a sufficiently large ($\log_2 \log_2 \ell + \kappa$ bits) random value s, computes $[z = d + s]$, and sends $[z]$ to B.
8: Party B decrypts $[z]$, computes $\lambda = z \bmod \varrho$, where $\varrho = (\log_2 \ell) + 1$, and from that the integers γ_i, $0 \le i < (2 \log_2 \ell) + 1$, as specified bellow. Party B encrypts the γ_i, and sends them to A.
9: Party A computes $\sigma = s \bmod \varrho$, and $[\vartheta] = [f(\sigma - \lambda)] = [\sum_{i=0}^{2\log_2 \ell} \gamma_i \sigma^i] = (\ldots([\gamma_{2\log_2 \ell}]^\sigma \cdot [\gamma_{(2\log_2 \ell)-1}])^\sigma \ldots \cdot [\gamma_1])^\sigma \cdot [\gamma_0]$.

The polynomial f, which is specified below, can be computed beforehand in the clear. Each integer γ_i can be computed by only

one multiplicative inverse and one multiplication. The exponentiation to the power −2 in step 3 is implemented by one multiplicative inverse, and one square.

Computation of γ_i. As shown before, $x = a - b + r$, $e = d(x,r)$, $y = e + w$, $d = d(y,w)$, $z = d + s$, and finally $\vartheta = (z = s)$, where $0 \le d \le \log \ell$. The idea is to compute the Lagrange polynomial $f(x)$ such that it maps 0 to 1, and maps x to 0, where $0 < |x| \le \log_2 \ell$. Then, we can compute $\delta = f(s \bmod \varrho - z \bmod \varrho) = f(\sigma - \lambda)$.

The Lagrange polynomial f is easily found as $f(x) = \prod_{i=-\log_2 \ell, i\neq 0}^{\log \ell} \frac{x-i}{-i} = (-1)^{\log \ell}(\log \ell!)^{-2} \prod_{i=-\log \ell, i\neq 0}^{\log \ell}(x-i) = (-1)^{\log \ell}(\log \ell!)^{-2} \sum_{i=0}^{2\log \ell} f_i x^i$, where $f_i \in \mathbb{Z}$ can be derived.

The binomial expansion of $x^i = (\sigma - \lambda)^i$, for $0 \le i \le 2\log \ell$, gives $\sum_{j=0}^{i} \binom{i}{j} \sigma^j (-\lambda)^{i-j}$. Therefore, we can write

$$f(\sigma - \lambda) = (-1)^{\log \ell}(\log \ell!)^{-2} \sum_{i=0}^{2\log \ell} f_i \sum_{j=0}^{i} \binom{i}{j} \sigma^j (-\lambda)^{i-j},$$

which reduces to $\sum_{j=0}^{2\log \ell} \gamma_j \sigma^j$, where $\gamma_j = (-1)^{\log \ell}(\log \ell!)^{-2} \sum_{i=j}^{2\log \ell} f_i \binom{i}{j} (-\lambda)^{i-j}$ and $0 \le j \le 2\log \ell$. In order to compute the γ_j, party B computes the multiplicative inverse of $(\log \ell!)^2$ modulo n, and multiplies this with the integer $\sum_{i=j}^{2\log \ell} f_i \binom{i}{j} (-\lambda)^{i-j}$.

Correctness. In EQT-3, the Hamming distance e between x mod 2^ℓ and r mod 2^ℓ is computed. For efficiency purpose (we will discuss this later), then, the second Hamming distance d between the first $\log \ell$ low significant bits of $e + w$ and w is computed. Afterward, party B computes encrypted γ_i values, which are required from party A to compute a polynomial that outputs the equality testing result. Actually, a Lagrange polynomial $f(\sigma - \lambda) = \prod_{i=-\log \ell, i\neq 0}^{\log \ell} \frac{(\sigma-\lambda)-i}{-i}$ is generated in this equality testing protocol, where it maps $(\sigma - \lambda = 0)$ to 1 and other values to 0. Recall that $\sigma = s \bmod \varrho$, $\lambda = z \bmod \varrho$, and $z = d + s$; therefore, if and only if $d = 0$, then $f(\sigma - \lambda)$ outputs 1.

Optimisation. Similar to ETQ-3, adding one more round reduces the communicational and computational costs. Besides that, it is also possible to lessen the number of rounds in EQT-3 if there is a limit in a particular application setting. However, decreasing one round makes the polynomial more complicated and the protocol more expensive.

4 SECURITY ANALYSIS

In this section, we provide proofs to show that our three secure equality testing protocols are simulation secure in the semi-honest security model. Informally, we mean that the probability that an adversary can learn private information from truly generated data by the parties in our protocols is at most negligibly more than the probability that an adversary can learn from given randomly generated data. We use the simulatability paradigm [20] in our proofs, where the adversary takes the control of the network and try to obtain the final result of the protocol by itself as the only party in the protocol. In this paradigm, security is defined as a comparison of computation work-flow in "real world" and "ideal world".

In real world, a protocol can be broken into sub-protocols or computations that are carried out by each party throughout the protocol. Let us denote π as one of EQT protocols; we can split π into two parts: $\pi = \pi_A$ and π_B, which are performed in parties A and B, respectively. π_A takes $[a]$ and $[b]$, which are the inputs, and outputs $[\vartheta]$, $[\vartheta] \leftarrow \pi_A([a],[b];\phi)$. And π_B decrypts the given encryptions from party A, processes them, and sends their encrypted versions to party B. Thus, to perform secure equality testing the encrypted messages flow from one party to another party and together they generate the $[\vartheta]$ as the result of EQT. Assuming party A is corrupted by an adversary $\mathcal{A}$, then $\mathcal{A}$ has access to $[a]$, $[b]$, and $[\vartheta]$. Similarly, when party B is corrupted, the adversary has access to the intermediate computation results.

In an ideal world, it is assumed that one of the parties is corrupted by an adversary. Then, he uses a simulator to generate the outputs of the other party. This would be similar to performing EQT with just one party, which is corrupted. In the ideal world, an adversary $\acute{\mathcal{A}}$, who has control over party A, has only access to her inputs $[a]$, $[b]$, and the garbage inputs given from simulated party B instead of the correct result of π_B. The goal is to show that $\mathcal{A}$ can learn equal or negligibly more than $\acute{\mathcal{A}}$, meaning that they are computationally indistinguishable, then we can claim that EQT is a simulation secure protocol.

Definition 4.1. Let $a \in \{0,1\}^*$ represents the parties' inputs, $n \in \mathbb{N}$ to be a security parameter, and $X = \{X(a,n)\}_{a\in\{0,1\}^*; n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*; n\in\mathbb{N}}$, two infinite sequences of random variables, are probability ensembles. Then, X and Y are computationally indistinguishable, denoted as $X \stackrel{c}{\equiv} Y$, if there is a polynomial $p(.)$ for every non-uniform polynomial-time probabilistic algorithm (nuPPT) D such that:

$$|Pr[D(X(a,n)) = 1] - Pr[D(Y(a,n)) = 1]| < 1/p(n) \quad (1)$$

4.1 Security of EQT-1

Let denote the computation of $[c_i]$ as A_{f_1}, δ_B as B_{f_1}, and $[\vartheta]$ as A_{f_2} in EQT-1. Let $f = (A_f, B_f)$, where $A_f = (A_{f_1}, A_{f_2})$ and $B_f = (B_{f_1})$, the f to be the PPT functionality for EQT-1. The view of the i^{th} party, $i \in \{A,B\}$, during the execution of EQT-1 on $([a],[b];\phi)$ and security parameter n is denoted by $view_i^{EQT-1}([a],[b];\phi;n) = (w, r_i; m_1^i, \cdots, m_t^i)$, where $w \in \{[a],[b],\phi\}$ based on the value of i, r^i are the i^{th} party internal random numbers, and m_j^i represents the j^{th} message that is received by i^{th} party. Recall that party B does not have any initial input, thus its inputs is denoted as ϕ. $output_i^{EQT-1}([a],[b];\phi;n)$ represents the output of each party in EQT-1. To represent the joint output of both parties, we denote

$$\begin{aligned} output^{EQT-1} = (&output_1^{EQT-1}([a],[b];\phi;n),\\ &output_2^{EQT-1}([a],[b];\phi;n)). \end{aligned} \quad (2)$$

Definition 4.2. EQT-1 securely computes $f = (A_f, B_f)$ in the semi-honest security setting if there exits PPT algorithms Sim_A and Sim_B such that:

$$\begin{aligned} \{(Sim_A(1^n,[a],[b],A_f,f))\} \stackrel{c}{\equiv} \{(&view_A^f\\ &([a],[b];\phi;n), output^f([a],[b];\phi;n))\} \end{aligned} \quad (3)$$

and

$$\{(Sim_B(1^n,\phi,B_f,f))\} \stackrel{c}{\equiv} \{(view_B^f \\ ([a],[b];\phi;n),output^f([a],[b];\phi;n))\} \tag{4}$$

Theorem 4.3. *The protocol EQT-1 is simulation secure and securely computes the functionality f, when the party A is corrupted by adversary $\mathcal{A}$ in the presence of semi-honest adversaries.*

We need to show that party A cannot computationally distinguish between generated messages and outputs from $\mathcal{S}_2$ that is the simulation of party B, and randomly generated data. Party A receives an output from $\mathcal{S}_2$, $[\delta_B]$. Given $[a]$, $[b]$, and 1^n (security parameter), party A works as follow:

(1) Party A chooses uniformly distributed random number r, δ_A, and r_i, $i \in \{0,\cdots,\ell-1\}$ for A_f.
(2) Party A executes A_{f_1} to obtain $[c_i]$, and sends them to S_2.
(3) S_2 tosses a random coin $\acute{\delta}_B \in \{0,1\}$ and sends $[\acute{\delta}_B]$ to party A.
(4) Party A performs A_{f_2} based on given $[\acute{\delta}_B]$ to get $[\acute{\vartheta}]$.

The output of the simulation can be written as: $Sim_A(1^n,[a],[b],A_f,f) = ([a],[b],r,\delta_A,r_i;[\acute{\delta}_B];([\acute{\vartheta}],\phi))$. The real view of part A can be presented as $view_A^f([a],[b]) = ([a],[b],r,\delta_A,r_i;[\delta_B])$. And the output of the real view is $output^f([a],[b]) = ([\vartheta],\phi)$. It can be observed that the encryption pairs $([\acute{\delta}_B],[\acute{\vartheta}])$ and $(\delta_B,[\vartheta])$ are computationally indistinguishable, since the crypto-scheme used in EQT-1 is semantically secure. Therefore, we can claim that

$$Sim_A(1^n,[a],[b],A_f,f) \stackrel{c}{\equiv} \{view_A^f([a],[b];\phi),output^f([a],[b];\phi)\}. \tag{5}$$

Theorem 4.4. *The protocol EQT-1 is simulation secure and securely computes the functionality f, when the party B is corrupted by adversary $\mathcal{A}$ in the presence of semi-honest adversaries.*

(1) S_1 chooses ℓ uniformly distributed random integers $\acute{c}_i$ and encrypts them, $[\acute{c}_i]$.
(2) S_1 tosses a random coin $r_1 \in \{0,1\}$. If $r_1 = 1$, then S_1 chooses a uniformly distributed random number $r_2 \in \{0,\cdots,\ell-1\}$ and sets $[\acute{c}_{r_2}] \leftarrow [0]$.
(3) S_1 sends $[\acute{c}_i]$ to party B.
(4) Party B executes B_{f_1} and sends $[\delta_B]$ back to S_1.

The simulation and the real view can be written as:

$$Sim_B(1^n,\phi,B_{f_1},f) = (\phi;[\acute{c}_i];([\vartheta],\phi)) \\ view_B^f([a],[b],\phi,n) = (\phi;[c_i];([\vartheta].\phi)) \tag{6}$$

Since party A has the decryption key, we should show that $\mathcal{A}$ cannot distinguish between $\acute{c}_i$ and c_i. To do so, we analyze two possible types of information leakage:

(1) Existence of zero in c_i: in EQT-1, $\mathcal{A}$ cannot learn extra information if, for any i, $c_i = 0$. The reason is that δ_A decides whether $c_i = 0$ means the equality or inequality of a and b, and party B has no access to δ_A. Moreover, location i does not leak any information, because party A permutes the c_i values before sending them to party B.
(2) Information about a and b if $c_i \neq 0$: in case of $c_i \neq 0$, party B is still cannot learn any extra information, since the c_i values are multiplicatively masked in party A.

Therefore, $\mathcal{A}$ cannot distinguish between c_i and $\acute{c}_i$, which means:

$$Sim_B(1^n,\phi,B_{f_1},f) = \{view_B^f([a],[b],\phi,n),output^f([a],[b];\phi)\} \tag{7}$$

4.2 Security of EQT-2

Denoting computation of $[x]$ as A_{f_1}, $([X],[x_i])$ as B_{f_1}, $[g]$ as A_{f_2}, $([\hat{g}],[t_i],[g.2^{-\log_2 \ell}])$ as B_{f_2}, $[e_i]$ as A_{f_3}, $[\hat{\lambda}]$ as B_{f_3}, and $[\vartheta]$ as A_{f_4}, we have $A_f = (A_{f_1},A_{f_2},A_{f_3},A_{f_4})$, $B_f = (B_{f_1},B_{f_2},B_{f_3})$, and $f = (A_f,B_f)$.

Theorem 4.5. *The protocol EQT-2 is simulation secure and securely computes the functionality f, when the party A is corrupted by adversary $\mathcal{A}$ in the presence of semi-honest adversaries.*

(1) Party A chooses uniformly random numbers r, $\hat{r}$, s, and h_i.
(2) Party A executes A_{f_1} to obtain $[x]$ and sends it to S_2
(3) S_2 generate ℓ random one-bit values $\hat{x}$ and another random integer $\hat{X}$. S_2 sends $[\hat{x}_i]$ and $[\hat{X}]$ to party A.
(4) Party A calls A_{f_2} to get $[g]$ and sends it to S_2.
(5) S_2 generate three random numbers $[\acute{g}]$, $[\hat{t}_i]$, and $[g^{''}]$ to party A.
(6) Party A executes A_{f_3}, computes $[e_i]$, and sends $[e_i]$ to S_2.
(7) S_2 tosses a random coin $\hat{\lambda}$ and sends it to party A.
(8) Party A executes A_{f_4} to obtain $[\vartheta]$.

Based on the same reason in Theorem 4.3, clearly, party A cannot distinguish between $([\hat{x}],[\hat{X}],[\acute{g}],[\hat{t}_i],[g^{''}],[\acute{\lambda}])$ and $([x],[X],[g],[t_i],[g\cdot 2^{-\log_2 \ell}],[\hat{\lambda}])$. Therefore,

$$Sim_A(1^n,[a],[b],A_f,f) \stackrel{c}{\equiv} \{view_A^f([a],[b];\phi),output^f([a],[b];\phi)\}. \tag{8}$$

Theorem 4.6. *The protocol EQT-2 is simulation secure and securely computes the functionality f, when the party B is corrupted by adversary $\mathcal{A}$ in the presence of semi-honest adversaries.*

Party A receives three outputs from $\mathcal{S}_1$ $[x]$, $[g]$, and $[e_i]$. Given $[1]$, $[b]$, and 1^n (security parameter), party A works as follow:

(1) S_1 chooses a $(\kappa+\ell+1)$-bit random value $\hat{x}$, a random value $o \in \{-\ell,\cdots,\ell\}$ and sends $[\hat{x}] \leftarrow [\hat{x}+o]$ it to party B.
(2) Party B executes F_{f_1} and sends $[x_i]$ and $[X]$ back to S_1.
(3) S_1 generates a $(\kappa+\log_2 \ell+1)$-bit random value $\acute{g}$ and sends it to party B.
(4) Party B call F_{f_2} and sends $[\hat{g}]$, $[t_i]$, and $[g\cdot 2^{-\log_2 \ell}]$ to S_1.
(5) S_1 chooses i random number $\hat{e_i} \in \mathbb{Z}_u^*$, $i \in \{0,\cdots,\log_2 \ell-1\}$. Then, S_1 tosses another coin δ, and if $\delta = 1$, then chooses a random i and $\hat{e}_i = 0$. Afterwards, S_1 sends $[\hat{e}_i]$ to party B.
(6) Party B executes B_{f_3} to get $\acute{\lambda}$, and sends it to S_1.

The simulation and the real view can be written as:

$$sim_B(1^n,\phi,B_f,f) = (\phi;[\hat{x}],[\acute{g}],[\hat{e}_i];([\vartheta,\phi])) \\ view_B^f([a],[b],\phi,n) = (\phi,[x],[g],[e_i];([\vartheta,\phi])) \tag{9}$$

To provide simulation security, party B should not be able to distinguish between $(\hat{x},\acute{g},\hat{e}_i)$ and (x,g,e_i). Recall that party B has access to the decryption key and can see the data in the clear. 1) party A masks $[a-b]$ with a large enough random value r to hide the difference from party B. Thus, $\mathcal{A}$ cannot distinguish between x and

$\hat{x}$. 2) $[z]$ values are also additively masked with another random number, which makes $\acute{g}$ and g indistinguishable for party B. 3) party A also multiplicatively masks c_i values which also makes $\hat{e}$ indistinguishable from e to party B. 4) party B cannot learn about the relation between a and b be seeing a zero in one of the e_i values, since it is calculated based on random number s. Thus, if any of $e_i = 0$ then still party B does not know whether $a = b$ or $a \neq b$. Based on the four stated properties, we can claim that:

$$Sim_B(1^n, \phi, B_f, f) = web\{view_B^f([a],[b],\phi,n), output^f([a],[b];\phi)\} \tag{10}$$

4.3 Security of EQT-3

Let us denote computation of $[x]$ as A_{f_1}, $[x_i]$ and $[X]$ as B_{f_1}, $[y]$ as A_{f_2}, $[y_i]$ and $[Y]$ as B_{f_2}, computation of $[z]$ as A_{f_3}, $[\gamma]$ as B_{f_3}, and computation of $[\vartheta]$ as A_{f_4}.

Theorem 4.7. *The protocol EQT-3 is simulation secure and securely computes the functionality f, when the party A is corrupted by adversary $\mathcal{A}$ in the presence of semi-honest adversaries.*

S_2 works as follow:

(1) Party A generate uniformly distributed random numbers r, w and s.
(2) Party A executes A_{f_1} to obtain $[x]$ and sends it to S_2.
(3) S_2 generate uniformly distributed random numbers $[\hat{x_i}]$ and $[\hat{X}]$, and sends them to party A.
(4) Party A calls A_{f_2} and sends $[\hat{y}]$ to S_2.
(5) S_2 generate random number $[\hat{y}_i]$ and $[\hat{Y}]$, and sends them to party A.
(6) Party A performs A_{f_3} to get $[z]$ and sends it S_2.
(7) S_2 chooses a random number $\hat{\gamma}$ and sends $[\hat{\gamma}]$ to party A.
(8) Party A executes A_{f_4} to obtain $[\vartheta]$.

Because of semantical security of the crypto-schemes used in this work, $\mathcal{A}$ cannot distinguish between $([\hat{x_i}],[\hat{X}],[\hat{y}_i],[\hat{Y}],[\hat{\gamma}])$ and $([x_i],[X],[y_i],[Y],[\gamma])$. Thus, we can claim that

$$Sim_A(1^n, [a],[b], A_f, f) \stackrel{c}{\equiv} \{view_A^f([a],[b];\phi), output^f([a],[b];\phi)\}. \tag{11}$$

Theorem 4.8. *The protocol EQT-3 is simulation secure and securely computes the functionality f, when the party A is corrupted by adversary $\mathcal{B}$ in the presence of semi-honest adversaries.*

S_1 works as follow:

(1) S_1 chooses a random number $[\hat{x}]$.
(2) Party B receives $[\hat{x}]$, decrypts it, executes B_{f_1} to obtain x_i and X, and sends them to S_1 in encryption form.
(3) S_1 generates random number $\hat{y}$ and sends $[\hat{y}]$ to party B.
(4) Party B executes B_{f_2} and sends $[y_i]$ and $[Y]$ back to S_1.
(5) S_1 chooses a random number $\hat{z}$ and sends $[\hat{z}]$ to party B.
(6) Party B calls B_{f_3} to get γ and sends it to S_1 in encryption form.

Recall that party B is keeping the private key and is able to decrypts the encryption given from party A. Thus, we need to show $\mathcal{A}$ cannot learn any private information by distinguishing given messages. First, party B cannot distinguish between $\hat{x}$ and x, since x is additively masked with a $\kappa + \ell + 1$-bit value. Second, distinguishing between $\hat{y}$ and y is not computationally possible for party B, since y is additively masked with large enough random value. Similarly, party B cannot distinguish between $\hat{z}$ and z, since z value is additively masked. Therefore, we can conclude that

$$Sim_B(1^n, \phi, B_f, f) = \{view_B^f([a],[b],\phi,n), output^f([a],[b];\phi)\} \tag{12}$$

5 PERFORMANCE ANALYSIS

5.1 Complexity Analysis

Communication. Table 2 presents the number of communication rounds and the amount of data transmitted. Table 2 shows that, except ST06 and NO07, they are all constant-round protocols. ST06 uses bit decomposition, which results in more number of communication rounds.

Table 2: Number of communication rounds and amount of data transmission ($\ell = 20, \varphi = 12, u = 31$).

Protocols	Rounds		Transmitted data (KB)	
[LT13]	2	2	$\ell + 1$	21
[ST06]	$2\ell + 2\lceil\log_2 \ell\rceil + 2$	52	$(6\ell - 5)/2$	57.5
[NO07]	3φ	36	$(2\varphi + \lceil\log_2 \varphi\rceil + 1)/2$	14.5
EQT-1	2	2	$(\ell + 2)/2$	1 1
EQT-2	3	3	$(\ell + \lceil\log_2 \ell\rceil)/2 + 2$	14.5
EQT-3	3	3	$(\ell + 3\lceil\log_2 \ell\rceil + 6)/2$	20

According to Table 2, using EQT-1 requires the least data transmission among the other protocols, which is mainly due to the use of DGK encryption scheme. In contrast, EQT-3 is the least communication-wise efficient protocol because of the large ciphertext space of Paillier and transmission of the Lagrange polynomial coefficients.

Computation. Table 3 presents the overall computational cost given in the number of modular multiplications. The cost of the DGK zero-check function can be represented as $3t/4$ multiplications modulo n [10] and we can show the complexity of a ciphertext modulo n with an x-bit exponent as $3x/2$ multiplications modulo n. According to Table 3, LT13 is the most computation-wise expensive protocol because of having ℓ exponentiations with $(2\kappa)i$ bits exponents, for $1 \leq i \leq \ell$. There are also $(6\ell)_{-1}$ exponentiations in ST06, but the exponents are -1. From the Table 3, we can see that EQT-3 has the least number of multiplication among the others. The main reason for having this outstanding efficiency in EQT-3 is performing less expensive exponentiations. In EQT-3, there are $(2\lceil\log_2 \ell\rceil)$ exponentiations with $\lceil\log_2 \ell\rceil/2$-bit exponents.

Table 4 presents the numbers of encryption and decryption. According to Table 4, ST06 and NO07 require a higher number of decryption to be performed compared to the other protocols. Since Paillier decryption is an expensive operation, a large number of decryption in a protocol is not desirable.

Our proposed protocols show different performances regarding communication and computation. For a system with limited communication resources, EQT-1 is a suitable choice, since it has only two communication rounds and the lowest data transmission cost.

Table 3: Overall computational cost and the complexity ($\ell = 20$).

Protocols	Multiplication		Complexity
[LT13]	$\ell(768(\ell+1)+7/2)+3079$	331169	$O(\ell^2)$
[ST06]	$45\ell+\ell/2-1$	902	$O(\ell)$
[NO07]	$(3\ell\lceil\log_2 \ell!\rceil)/2+\ell+20672$	22522	$O(\ell \log_2 \ell!)$
EQT-1	$504\ell+\ell/2+345$	10435	$O(\ell)$
EQT-2	$\ell/2+841\lceil\log_2 \ell\rceil+25$	4240	$O(\ell)$
EQT-3	$\ell/2+3/2(\lceil\log_2 \ell\rceil)^2+5/2\lceil\log_2 \ell\rceil+14$	74	$O(\ell)$

Table 4: Number of encryption and decryption per protocol ($\ell = 20, \varphi = 12, u = 31$).

Protocols	Encryption		Decryption	
[LT13]	$2\ell+1$	41	2	2
[ST06]	5ℓ	100	$3\ell-1$	59
[NO07]	$\ell+2\varphi+1$	45	$2\varphi+1$	25
EQT-1	$2\ell+2$	42	1	1
EQT-2	$\ell+2\lceil\log_2 \ell\rceil+5$	35	2	2
EQT-3	$\ell+3\lceil\log_2 \ell\rceil+8$	43	3	3

Although EQT-1 has a very low communication cost, its computational cost is twice more than EQT-2 and hundred times more than EQT-3. For a system with very limited computational resources, EQT-3 is a good choice, since it has significantly low computational cost. However, EQT-3's data transmission cost is twice more than EQT-1 and also higher than EQT-2.

5.2 Experimental Results

The protocols are implemented using C++ and external libraries: MPIR, Boost, and SeComLib on a single Linux machine running Ubuntu 14.04 LTS, with a 64-bit microprocessor and 8 GB of RAM, ignoring network latency. The cryptographic key lengths of the Paillier and DGK cryptosystems are chosen according to NIST standards [2], which are valid until 2030. Table 5 shows the parameters used for the implementation.

Table 5: Parameters used in the implementation.

Parameter	Symbol	Value
Bit size of inputs	ℓ	2-30
Security parameter	κ	112 bits
Paillier message space	n	2048 bits
DGK message space	u	31
DGK security parameter	t	224 bits
Error controller in NO07	φ	12

As [23] does not provide an analysis on φ, we implemented and analyzed their proposal. Figure 2 shows various values of φ with their corresponding error rates. Furthermore, it presents run-time of the NO07 equality testing protocol with 25-bit inputs, and different φ values. As it is shown in Figure 2, choosing $\varphi = 12$ makes the error probability negligible.

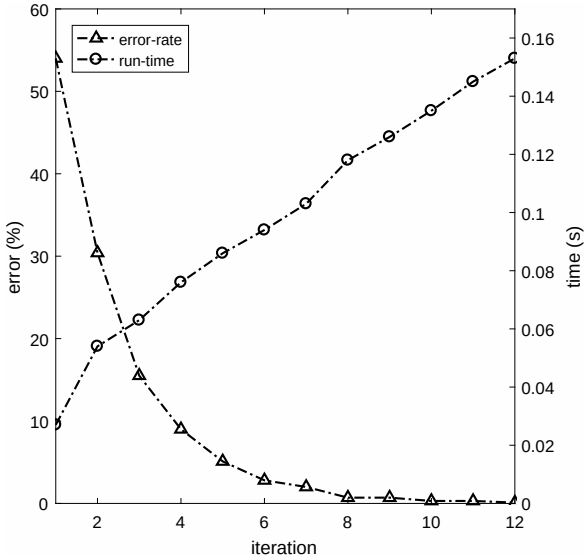


Figure 2: The error-rates and run-times of NO07 for different values of φ.

Figure 3 shows the run-times of all the described secure equality testing protocols. Since VAT09, LT13, ST06, and NO07 are much more expensive regarding run-time than our protocols; we present their run-times separately in Figure 3b. As it is shown in Figure 3a, EQT-1 has the lowest run-time for inputs smaller than 20 bits. Figure 3b shows the proposed protocols are computationally much more efficient than the state-of-the-art, as they outperform NO07, ST06, VAT09, and LT13 by 95%, 96%, 97%, and 99%, respectively for 25-bit inputs.

Notice that Figure 3a shows the total run-time, which involves run-times of all operations including encryption and decryption. However, Table 3 only presents the complexities of multiplication, exponentiation, and DGK zero-check, and it does not take into account the encryption and decryption costs.

5.3 Applying data packing

We observed that Paillier decryption dominates a significant portion of the total run-times of the protocols. For instance, it is shown in Table 3 that EQT-3 is the most efficient protocol. However, in Figure 3a, EQT-3 does not demonstrate the same efficiency due to the cost of Paillier decryption. Data packing [29] can be used to mitigate such effect of Paillier decryption cost. Data packing reduces decryption cost since multiple messages packed in one ciphertext can be decrypted at once. However, data packing is applicable in the cases where there are multiple equality tests to be performed. This condition is realistic since in existing applications such as search algorithms many equality tests are needed. Since data packing uses the plaintext space of the encryption scheme, Paillier, efficiently, it also reduces the communication cost. Figure 4 shows that the run-time and total data transmission of our equality testing protocols are reduced significantly after applying data packing. Notice that

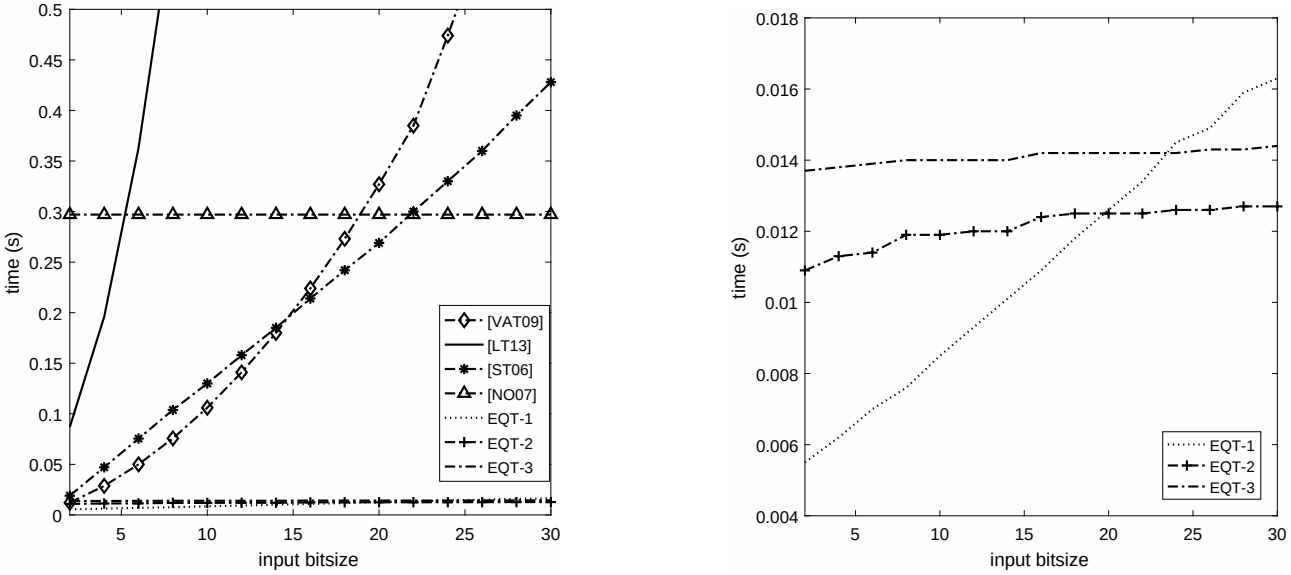


Figure 3: Run-times of the equality testing protocols without data packing.

the results in Figure 4a are matching our analysis in Table 3, which does not take encryption and decryption into account. According to Figure 4a, EQT-3 after data packing outperforms VAT09, LT13, ST06, and NO07 by 99% for the inputs larger than 20 bits.

Table 6 compares performance of the protocol based on run-time, communicational round, and total data transmission for the inputs size of 20 bits. Table 6 clearly shows the trade-off between communication and computation costs in all protocols.

Table 6: Comparing EQT's performances ($\ell = 20, \varphi = 12, u = 31$).

Protocols	Run-time (sec)	Rounds	Data transmission (KB)
EQT-1	0.008	2	10
EQT-2	0.0048	3	13
EQT-3	0.0033	3	19

6 CONCLUSIONS

Testing equality of encrypted values is a building block in a number of cryptographic protocols such as searching in encrypted databases. In this work, we have investigated the state-of-the-art protocols and propose three new cryptographic protocols, which are significantly more efficient than the existing work regarding communication and computation. However, each protocol presented in this paper has its own advantages and disadvantages on run-time, bandwidth, and the number of rounds. Nevertheless, our analysis and experimental results support our claims in terms of efficiency compared to the state-of-the-art.

REFERENCES

[1] Judit Bar-Ilan and Donald Beaver. 1989. Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*. 201–209. DOI: http://dx.doi.org/10.1145/72981.72995

[2] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. 2007. Nist sp800-57: Recommendation for key management part 1: General (revised). *NIST, Tech. Rep* (2007).

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 1–10. DOI: http://dx.doi.org/10.1145/62212.62213

[4] Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. 2017. Encryption Switching Protocols Revisited: Switching Modulo p. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*. 255–287. DOI: http://dx.doi.org/10.1007/978-3-319-63688-7_9

[5] David Chaum, Claude Crépeau, and Ivan Damgård. 1988. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 11–19. DOI: http://dx.doi.org/10.1145/62212.62214

[6] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press. http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053

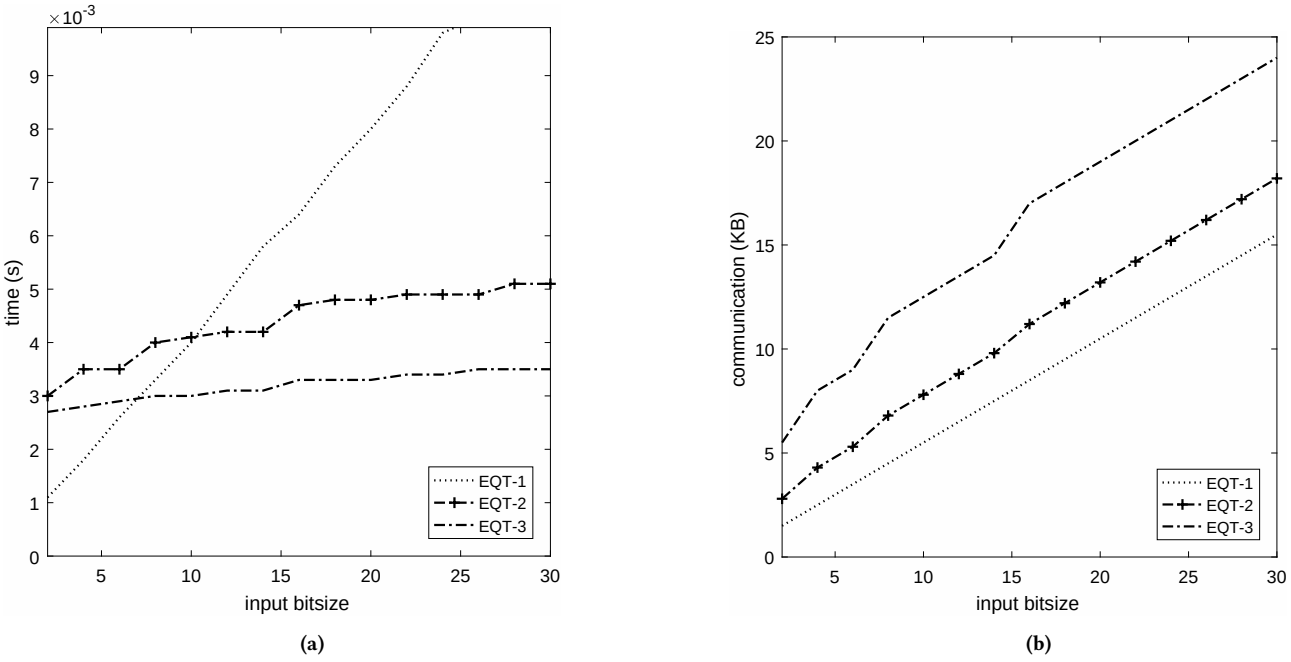
[7] Ronald Cramer, Eike Kiltz, and Carles Padró. 2007. A Note on Secure Computation of the Moore-Penrose Pseudoinverse and Its Application to Secure Linear Algebra. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*. 613–630. DOI: http://dx.doi.org/10.1007/978-3-540-74143-5_34

[8] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. 2006. Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. 285–304. DOI: http://dx.doi.org/10.1007/11681878_15

[9] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. 2008. Homomorphic encryption and secure comparison. *IJACT* 1, 1 (2008), 22–31.

[10] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. 2009. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT* 1, 4 (2009), 323–324. DOI: http://dx.doi.org/10.1504/IJACT.2009.028031

[11] Ivan Damgård and Jesper Buus Nielsen. 2003. Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference,*



(a) **(b)**

Figure 4: Run-time and data transmission of the equality testing protocols after data packing.

Santa Barbara, California, USA, August 17-21, 2003, Proceedings. 247–264. DOI: http://dx.doi.org/10.1007/978-3-540-45146-4_15

[12] Oded Goldreich. 2004. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press.

[13] Carmit Hazay and Tomas Toft. 2014. Computationally Secure Pattern Matching in the Presence of Malicious Adversaries. *J. Cryptology* 27, 2 (2014), 358–395. DOI: http://dx.doi.org/10.1007/s00145-013-9147-8

[14] Kaibin Huang, Raylin Tso, and Yu-Chi Chen. 2017. Somewhat semantic secure public key encryption with filtered-equality-test in the standard model and its extension to searchable encryption. *J. Comput. Syst. Sci.* 89 (2017), 400–409.

[15] Arjan Jeckmans, Andreas Peter, and Pieter Hartel. 2013. Efficient privacy-enhanced familiarity-based recommender system. In *Computer Security–ESORICS 2013*. Springer, 400–417.

[16] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. 2009. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*. 1–20. DOI: http://dx.doi.org/10.1007/978-3-642-10433-6_1

[17] Vladimir Kolesnikov and Thomas Schneider. 2008. Improved Garbled Circuit: Free XOR Gates and Applications. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*. 486–498. DOI: http://dx.doi.org/10.1007/978-3-540-70583-3_40

[18] Hyung Tae Lee, San Ling, Jae Hong Seo, and Huaxiong Wang. 2016. Semi-generic construction of public key encryption and identity-based encryption with equality test. *Inf. Sci.* 373 (2016), 419–440.

[19] Chen Li, Rongxing Lu, Hui Li, Le Chen, and Jie Chen. 2015. PDA: a privacy-preserving dual-functional aggregation scheme for smart grid communications. *Security and Communication Networks* 8, 15 (2015), 2494–2506.

[20] Yehuda Lindell. 2017. How to Simulate It - A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*. 277–346.

[21] Helger Lipmaa and Tomas Toft. 2013. Secure Equality and Greater-Than Tests with Sublinear Online Complexity. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*. 645–656. DOI: http://dx.doi.org/10.1007/978-3-642-39212-2_56

[22] Majid Nateghizad, Zekeriya Erkin, and Reginald L Lagendijk. 2016. An efficient privacy-preserving comparison protocol in smart metering systems. *EURASIP Journal on Information Security* 2016, 1 (2016), 1–8.

[23] Takashi Nishide and Kazuo Ohta. 2007. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography–PKC 2007*. Springer, 343–360.

[24] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. 223–238. DOI: http://dx.doi.org/10.1007/3-540-48910-X_16

[25] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. 223–238. DOI: http://dx.doi.org/10.1007/3-540-48910-X_16

[26] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2009. Efficient Privacy-Preserving Face Recognition. In *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*. 229–244. DOI: http://dx.doi.org/10.1007/978-3-642-14423-3_16

[27] Berry Schoenmakers and Pim Tuyls. 2006. Efficient Binary Conversion for Paillier Encrypted Values. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. 522–537. DOI: http://dx.doi.org/10.1007/11761679_31

[28] Qiang Tang. 2012. Public key encryption supporting plaintext equality test and user-specified authorization. *Security and Communication Networks* 5, 12 (2012), 1351–1362.

[29] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, Mehmet Celik, and Aweke Lemma. 2007. A secure multidimensional point inclusion protocol. In *Proceedings of the 9th workshop on Multimedia & security*. ACM, 109–120.

[30] Libing Wu, Yubo Zhang, Kim-Kwang Raymond Choo, and Debiao He. 2017. Efficient and secure identity-based encryption scheme with equality test in cloud computing. *Future Generation Comp. Syst.* 73 (2017), 22–31.

[31] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. 2010. Probabilistic Public Key Encryption with Equality Test. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*. 119–131.

[32] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. 2005. Privacy-Preserving Classification of Customer Data without Loss of Accuracy. In *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*. 92–102. DOI: http://dx.doi.org/10.1137/1.9781611972757.9