Journal of Information Security, 2017, 8, 223-239
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Private Personal Information Verification

Hoang Giang Do, Wee Keong Ng

School of Computer Science and Engineering, Nanyang Technological University, Singapore City, Singapore
Email: do0004ng@e.ntu.edu.sg

How to cite this paper: Do, H.G. and Ng, W.K. (2017) Private Personal Information Verification. *Journal of Information Security*, **8**, 223-239.
https://doi.org/10.4236/jis.2017.83015

Received: June 21, 2017
Accepted: July 15, 2017
Published: July 18, 2017

Abstract

Physical document verification is a necessary task in the process of reviewing applications for a variety of services, such as loans, insurance, and mortgages. This process consumes a large amount of time, money, and human resources, which leads to limited business throughput. Furthermore, physical document verification poses a critical risk to clients' personal information, as they are required to provide sensitive details and documents to verify their information. In this paper, we present a systematic approach to address shortcomings in the current state of the processes used for physical document verification. Our solution leverages a semi-trusted party data source (*i.e.* a governmental agency) and cryptographic protocols to provide a secure digital service. We make use of homomorphic encryption and secure multi-party computation to develop a series of protocols for private integer comparison and (non-) membership testing. Secure boolean evaluation and secure result aggregation schemes are proposed to combine the results of the evaluation of multiple predicates and produce the final outcome of the verification process. We also discuss possible improvements and other applications of the proposed secure system of protocols. Our framework not only provides a cost-efficient and secure solution for document verification, but also creates space for a new service.

Keywords

Secure Computation, Homormophic Encryption, Multiparty Computation

1. Introduction

Recent advances in technology have led to the introduction of many digital and automated services, such as e-shopping, e-learning, and e-banking. These digitalised services not only reduce the cost of operation, but also increases throughput for businesses. However, several tasks in these services continue to involve a considerable amount of human effort. Physical document verification is a necessary task in the process of reviewing applications for many services, such as loans, insurances, and mortgages. This process consumes a large amount of time, money, and human resources. Consider the example of a loan or an insurance application. The applicants are usually required to provide numerous documents to certify their relevant personal information, such as birth certificate, statement of monthly income, marriage certificate, medical records, and so on. At the same time, the loan/insurance provider requires a considerable amount of human resource to verify and store these documents. This process can take several weeks to complete, and serves to limit business throughput.

Moreover, the process of physical document verification incurs a critical privacy risk for applicants. They provide to a third party (*i.e.* the service provider) many sensitive documents, such as birth certificates, IDs, health records, and so on. All these documents are stored in the provider's database. If the client applies for multiple schemes or subscriptions, multiple copies of his/her personal data are stored in different places. Since data can be leaked from the server, storing personal information in multiple third-party databases is not recommended. One source of such a leak is employees who do not follow the company's privacy policies, and may, intentionally or unintentionally, reveal sensitive client information. Even when the provider claims to enforce strict policies pertaining to privacy, there is still a chance that the database systems are vulnerable to malicious external attacks.

In this paper, we propose a systematic approach to address the abovementioned shortcomings of the current state of the process of physical document verification. We assume that there is a trusted data source that stores the certified personal information of clients. We also assume that a list of requirements (maybe involving the divulgence of private information) needs to be fulfilled by the applicant to qualify for a given scheme or subscription. We present a series of protocols that allow the verifier and the data keeper to communicate with each other and securely verify the applicant's information according to the requirements proposed by the verifier. The main contributions of this paper are as follows:

1) The proposed system digitalizes the process of document verification and hence enhances business throughput.

2) The system protects user confidentiality from both the verifier and the data keeper. The details of the requirements proposed by the verifier also remain hidden from the data keeper.

3) The proposed approach creates space for new services for information data storage and verification.

The remainder of the paper is organised as follows: In the next section, we review related work in the literature. Section 3 contains our problem formulation as well as the scenario we consider. Section 4 contains a discussion of our security model and assumptions as well as the underlying cryptographic techniques we leverage (*i.e.* Paillier's encryption scheme). The proposed solution to the problem of secure personal information verification is described in Section 5, which systematically discusses four stages of the solution. Section 6 presents experimental evaluations of the proposed sub-protocols. The final section discusses future work and our conclusions.

2. Related Work

The scenario we consider shares characteristics with the problem of zero-knowledge proof. Zero-knowledge proof systems, introduced by Goldwasser *et al.* [1], involve two parties—a prover and a verifier. The system allows the prover to convince the verifier of some fact without revealing information about the proof. In our proposed problem, a client has to prove that he/she poses several attributes that match the requirements provided by the verifier. Zero-knowledge proof systems have been well researched, and have a wide range of applications, including authentication [2], voting [3], and e-cash [4]. J. Camenisch [5] proposed a useful zero-knowledge proof scheme which allows the prover to convince the verifier that a digitally committed value is a member of a given public set. The scheme can be directly applied to our problem. The client is assigned a number of digital commitments for each attribute, such that he/she can prove that the commitments belong to certain public sets. However, a separate instance of proof and verification is needed for each independent verifier; and every time the client's attribute changes, he/she needs to be assigned a new commitment from a trusted server. Moreover, the scheme only presents solutions for simple membership and range predicates. In order to provide an efficient solution to the problem of personal information verification, where the predicates are much more complicated, we need to consider different approaches.

Another approach to consider is private set intersection [6] [7]. It allows the verifier to determine whether an applicant satisfies the relevant requirements given a threshold. However, this approach can only deal with exact attribute matching. Moreover, without a trusted party verifying the set of attributes, the applicant can use false information.

While sharing similar purposes as the above approaches, our system considers a different setting in the context of zero-knowledge proof systems. The communication and verification processes are conducted by a verifier and a semi-honest data keeper, rather than by a verifier and a client. Our approach leverages the computation model of secure multi-party computation introduced by C. Yao [8]. Many follow-up studies have addressed different problems in this context. Some sub-protocols presented in this paper are inspired by [9] and modify [10] using studies along this line of research.

3. Problem Formulation

Definitions. Our proposed system involves three general parties—the client, the verifier, and the data keeper—as illustrated in **Figure 1**.

- The client. The client wishes to privately prove that his/her personal data satisfy the predicates predefined by the verifier.
- The verifier. The verifier (we call the verifier Bob) provides a series of predicates that need to be satisfied by the client.

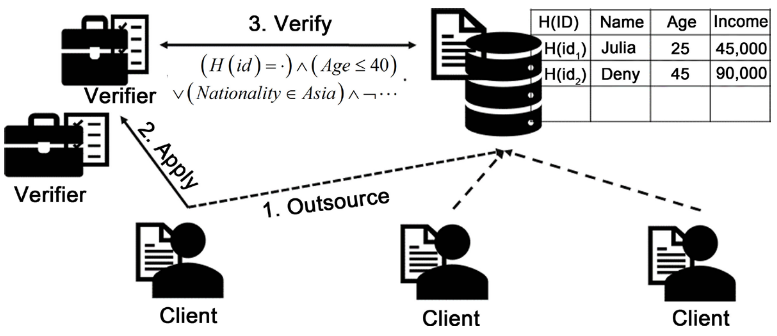

Figure 1. System Model.

Table 1. Sample personal data records.

ID	Name	Age	Sex	Income	Nationality	Marital Status
1	Julia	29	F	30,000	Singaporean	Married
2	Deny	32	M	35,000	Singaporean	Single
3	Christina	38	F	80,000	Myanmar	Single
4	Alwen	41	M	120,000	Indonesian	Married
5	Dino	37	M	90,000	Malaysian	Divorced

- The data keeper. The data keeper (whom we call Alice) stores the personal data of the client and provides a security guarantee for the data storage.

The database stored by the data keeper consists of *n* records. Each record describes a client by *m* attributes. Table 1 presents a simple example of data content maintained by the data keeper.

A *personal information verification scheme* is a Boolean function on a data record. The Boolean function is informally described by single and complex predicates. We assume that the single predicates are equality, inequality, membership, and non-membership.

- An equality predicate examines whether a variable x is equal to a certain value *a*: $x \overset{?}{=} a$.
- An inequality predicate inputs a variable *x* and a certain value *a*, and outputs 1 when $x \overset{?}{<} a$ (and 0 otherwise).
- The membership and non-membership predicates check whether variable *x* belongs (or does not belong) to a set *A* of elements: $x \overset{?}{\in} A,\ A=\{e_1,e_2,\cdots,e_n\}$.

A complex statement contains multiple single predicates and a set of logical expressions $\wedge,\vee,\neg$. A series of predicates (provided by the verifier) can be expressed as a complex predicate by combining them using the $\wedge$ operator. The personal data of clients are accepted by the verifier only if they satisfy the final complex predicate.

Workflow. To illustrate, consider the following example: A client first outsources his/her data to a data keeper called Alice. Alice can be a governmental agency. The client and Alice are responsible for ensuring the correctness of the information. The client can pay a small fee to Alice for keeping track of his/her data. The verifier Bob provides a subscription scheme. In order to subscribe to the scheme, the client needs to satisfy a number of statements for personal information, including age, income, nationality, health condition, etc. He/She wants to prove that he/she qualifies for the scheme, but does not want to reveal exact information. At the same time, he/she also wishes to hide the fact that he/she is applying the certain scheme through others (such as Alice). He/She should anonymously authenticate Bob to communicate with Alice. Bob interacts with Alice by our proposed approach. Finally, Bob should be able to decide whether the client qualifies for the given scheme.

4. Background

4.1. Security Assumptions

In this paper, the privacy/security of the proposed protocols is measured by the amount of information disclosed during execution. We adopt the security definitions and proof techniques from the literature on secure multi-party computation to analyse. The secure multi-party computation problem involves multiple parties collaboratively performing various types of computation without compromising the privacy of data. In the mid 1980s, C. Yao [8] introduced the idea of securely computing any two-party functionality in the presence of dishonest adversaries. Since then, various privacy-preserving protocols have been proposed to address different class of computation problems in the context of private data.

There are two common adversarial models under secure multi-party computation: semi-honest and malicious. In the malicious model, the adversary has the ability to arbitrarily deviate from the protocol specifications. On the other hand, in the semi-honest model, an attacker (*i.e.* one of the participating parties) is expected to follow the prescribed steps of the protocol. However, the attacker is subsequently free to compute additional information based on his or her private input, output and messages received during the execution of the secure protocol. Although the assumptions of the semi-honest adversarial model are weaker than those of the malicious model, we insist that this assumption is realistic under the problem settings. We assume that the data keepers are trusted (as they are governmental agencies) to ensure the confidentiality of sensitive client data. It is difficult to imagine them colluding with other companies to damage their own reputation. Moreover, it is often non-trivial for one party to maliciously deviate from a particular protocol which may be hidden in a complex process.

In short, we assume that the verifier and the data keeper are semi-honest. They will correctly follow the protocol specifications. However, at the same time, they are also curious about the applicants' information. In general, secure personal information as described in Section 5 should meet the following privacy requirements:

- Client-to-verifier privacy. The verifier should not be able to gain any details concerning the client's personal data stored in the data keeper's database, except for those he can learn from the result (*i.e.* the client qualifies or not).
- Client-to-data-keeper privacy. At any point during protocol execution, the

identity of the applicant should not be revealed to the data keeper.

- End user's privacy. The verifier should not be able to obtain any information relating to other clients stored in the data keeper's database.
- Verifier-to-data-keeper privacy. The details of the predicates should not be leaked to the data keeper. This requirement is particularly applicable to private services where the selection criteria may be private to the provider.

4.2. Additive Homomorphic Encryption

An additive homomorphic encryption scheme is a cryptosystem that allows arithmetic (*i.e.* addition) operations to be performed on the ciphertext without decryption or knowing the actual values. Efficient additive homomorphic cryptosystems have been proposed, such as the Pallier cryptosystem [11], or the Damgard and Jurik cryptosystems [12], which are Paillier encryption scheme of flexible lengths. For simplicity, we assume that a Paillier cryptosystem is used for encryption and decryption throughout this paper.

The Paillier cryptosystem consists of three algorithms:

1) $KeyGen(1^{\lambda}) \to (pk, sk)$: Inputs a security parameter and produces the key pair (pk, sk).

2) $Enc(pk, m) \to c$: Inputs a public key pk and a message *m*, and outputs a ciphertext *c*.

3) $Dec(sk, c) \to m$: Inputs a private key sk and a ciphertext *c*, and outputs a message *m*, such as $Dec(sk, Enc(pk, m)) = m$.

The security of the Paillier encryption scheme relies on the computational hardness assumption of a novel mathematical problem called composite residuosity. The decision version of this problem class assumes that no polynomial-time algorithm can distinguish the *N-th* residues modulo N^2 with a non-negligible probability. $Enc(\cdot)$ and $Dec(\cdot)$ denote the Paillier encryption and decryption algorithms, respectively.

The Paillier cryptosystem is additive homomorphic encryption. If we consider two operators $\times$ and $+$ in the ciphertext and the plaintext domains, respectively, m_1 and m_2 are two plaintext elements. The Paillier encryption scheme Enc satisfies the followings properties:

- Additive Homomorphism:

$$Enc(m_1) \times Enc(m_2) = Enc(m_1 + m_2).$$

- Homomorphic Multiplication:

$$Enc(k \times m_1) = Enc(m_1)^k.$$

- Semantic Security: Informally, a semantically secure [13] encryption scheme is a probabilistic, polynomial-time algorithm such that given the ciphertext, an adversary cannot deduce any additional information about the plaintext.

The above computation is performed modulo N^2. We refer the reader to [11] for more details. We also note that any additive homomorphic encryption scheme that satisfies the above properties can be utilized to implement our proposed framework.

5. Secure Information Verification

5.1. Overview

Our proposed approach to the secure information verification problem consists of four stages:

1) *Setup*—During this phase, the client goes through an anonymous authentication process so that the verifier is authenticated to communicate with the data keepers for the verification stage. In addition, the data keeper and the verifier generate an encryption key pair and exchange the public key of the homomorphic cryptosystem. These public keypairs are utilized for secure communication and computation at the later stages.

2) *Single Predicate*—In this stage, the data consumer evaluates a predicate for each entity in the dataset of the data keeper. The output of this stage is the encryption of either 1 or 0, depending on whether the entity satisfies the predicate.

3) *Secure Complex Predicate Evaluation*—Based on the results of the previous stage, the verifier collaborates with the data keeper to compute the result of the complex logical combination of Boolean predicates. Again, the output of this stage is the encryption of either 1 or 0 depending on whether the entity satisfies the predicate.

4) *Aggregation of Output Data*—At this stage, the final result is aggregated, decrypted and shown to the verifier. Since the data keeper computes the decryption, we propose a secure protocol to generate the outcome so that the data keeper cannot obtain any information concerning the final result.

5.2. Setup

In the setup phase, the client is first required to complete anonymous authentication with the data keeper Alice, who then allows the verifier Bob to initiate the secure information verification process on the records of Alice's database. When the clients agree to their personal information being stored in Alice's database, she issues to each client a credential to be used for authentication. Each time a client subsequently requests access to Alice's database, he/she uses her credentials for verification with Alice, who begins communication with Bob for the information verification process.

Traditional password-based authentication systems expose the identity of the client to the data keeper Alice. Hence, they violate the client-to-data-keeper privacy requirement. To satisfy this, it is desirable to have an authentication scheme that promises unlinkability, *i.e.* the server should not be able to link user requests such that access to the same user cannot be recognised as such.

As the anonymous authentication process is not our main contribution here, we only briefly review possible approaches to satisfy this requirement. The most feasible solution is anonymous credentials introduced by D. Chaum [14]. This allows a user to prove that he/she has obtained a credential issued by an organisation without revealing anything regarding his/her identity other than the credential. J. Camenisch [15] proposed a protocol that allows an organisation to issue a credential by obtaining a signature on a committed value. The client can then prove with zero knowledge that she has a signature under the organisation's public key on the given value.

Applied to our problem setting, the client first generates a non-interactive zero-knowledge proof (*i.e.* applying the Fiat--Shamir transform) of his/her credentials with Alice. He/She transfers the proof to Bob, who submits the proof to Alice. Finally, Alice authenticates Bob to communicate and verify the client's information.

Following the authentication process, Alice and Bob generate two Paillier key pairs using the *KeyGen* algorithms and agree on two key public pairs for communication during verification execution. We denote by $Enc_A(\cdot)$ and $Enc_B(\cdot)$ the Paillier encryption under Alice's public key and that under Bob's public key, respectively.

5.3. Single Individual Predicate Evaluation

In the single predicate evaluation stage, for each data record and each attribute that needs to be verified, the verifier Bob and the data keeper Alice together perform one of the following protocols: equality predicate evaluation, inequality predicate evaluation and (non-) membership predicate evaluation. The output of each protocol is an encrypted bit maintained by Bob. The resulting bit is encrypted under the data keeper's public key so that Bob cannot obtain any information relating to the other entities in the database. We now describe the three protocols to securely evaluate the results of these predicates.

5.3.1. Equality Predicate

A secure equality predicate evaluation tests whether two private inputs x and y are equal: $x \overset{?}{=} y$. We use the protocol presented by C. Gentry *et al.* [10] to develop the protocol for secure equality predicate evaluation. Gentry's equal-to-zero protocol [10] allows the comparison between a private value and zero. To be able to apply the equal-to-zero protocol, we execute a transformation (as presented in Protocol 1) on the two private inputs.

```
Algorithm 1: Secure Equality Evaluation
  Input: Alice holds integer $x$, Bob holds integer $y$
  Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $x = y$, and
          0 otherwise.
1 Bob computes and sends $Enc_B(y)$ to Alice;
2 Alice computes $Enc_B(a) = Enc_B(x - y)$ using the homomorphism;
3 Alice chooses a random $r$ and uses the homomorphism to compute
    $c \to Enc_B(a + r)$;
4 Denote the bit representation of r by $r_n \cdots r_2 r_1$. Alice encrypts the bits $r_i$
    under her key to obtain $c_i = Enc_A(r_i)$;
5 Alice sends to Bob both $c$ and all $c_i s$;
6 Bob decrypts c to obtain $a' = a + r$. Let denote $a'_n \cdots a'_2 a'_1$ be the binary
    representation of $a'$;
7 For each $i$, Bob computes $c'_i = Enc_A(1 - r_i)$ if $a'_i = 0$; otherwise, keep $c_i$
    unchanged;
8 Bob computes $rc = \bigwedge_i c'_i$, and outputs $Enc_A(b) = 1 \ominus rc$;
```

The computation in Steps 1 - 2 transforms the problem into a secure equal-to-zero protocol. In this protocol, Alice holds an encrypted message with value *a*. The message is encrypted under Bob's key; hence, neither Alice nor Bob has information concerning the value *a*. The remaining part of the protocol involves compare *a* with 0. In the last step, Bob is required to compute an *AND* operator on the ciphertext space. In binary setting, the *AND* operator is exactly a multiplication scheme. We describe a secure multiplication scheme as in Protocol 4. The protocol allows Bob to compute the product of two ciphertexts where he does not know the decryption key.

Algorithm 2: Secure Multiplication

```
  Input: Bob holds $(Enc_A(x), Enc(y))$, and Alice holds private key $sk_A$
  Output: Bob holds $Enc_A(x \times y)$
1 Bob generates two random number $r, s$;
2 Bob computes $Enc_A(x+r), Enc_A(y+s)$ and sends them to Alice;
3 Alice decrypts and obtains $x+r,\ y+s$;
4 Alice computes $(x+r)(y+s)$ and sends $Enc_A((x+r)(y+s))$ to Bob;
5 Bob computes
    $Enc_A(x \times y) = Enc_A((x+r)(y+s)) - Enc_A(x \times s) - Enc_A(y \times r) - Enc_A(r \times s)$;
```

The computations on lines 2 and 5 of Protocol 2 are performed by using the homomorphic property of Paillier encryption. During the protocol, Bob only works on encrypted data while the server receives two random numbers. Hence, no information regarding *x* and *y* is obtained by Bob and *S*. The correctness of the protocol is trivial, as $(x+r)(y+s) = x \times y + x \times s + y \times r + r \times s$. The protocol requires two encrypted integer transfers for communication. Bob needs to perform five multiplication operations and five exponentiation operations in the ciphertext space.

Analysis. We now analyse the correctness and security of the equality predicate evaluation protocol (Protocol 1). Due to the transformation in Steps 1 - 2, we only need to examine the remaining parts, where the two parties together compare the encrypted value *a* with 0.

We note that in Step 7, Alice reserves bit c_i when $a_i' = 0$. This means that we perform the *XNOR* operation on bit a_i' and ciphertext c_i for all $i = \overline{1, n}$. Remember that $c_i = Enc_A(r_i)$, so at that step, we actually compute $c_i' = Enc_A(f_i) = Enc_A(r_i \text{ } XNOR \text{ } a_i')$. Moreover, since $a' = a + r$, all $r_i \oplus a_i'$ s are zero if and only if $a = 0$. Therefore, all $f_i = 1$ if $a = 0$; the last step concludes the protocol, where Bob computes the AND of all bits f_i and outputs the inverted result.

The security of the two parties follows the semantic security properties of the employed encryption scheme—the Paillier cryptosystem. Alice only obtains the encryption version of y. On the other hand, Bob receives a randomized value $a' = a + r$ and an array of encrypted bits $\{Enc_A(r_n), \cdots, Enc_A(r_1)\}$. Hence, no more information is leaked to either party.

5.3.2. Inequality Predicate

The inequality predicate considers two parties that pose two private integral values x and y and wish to evaluate the predicate $x \overset{?}{<} y$. This problem is known as secure comparison. The first solution to it was proposed by A. Yao [8] in the 1980s, and pioneered research on secure multi-party computation. Since then, extensive research has been conducted to address the problem of secure comparison. Di Crescenzo [16] proposed a secure comparison protocol with $O(n^2 \log N)$ complexity, where n is the length in bits of the compared numbers, and N is the group size of the plaintext of the employed encryption scheme. Fischlin [17] and Blake [9] reduced the complexity of the solutions to $O(n \log N)$.

We propose a variant of Blake's protocol [9] as a building block to compare two private inputs. In our problem setting, at this stage, it is expected that no information relating to the results of the evaluation are known to the verifier or the data keeper. Therefore, we cannot directly apply the protocol proposed by Blake [9], as it leaks the comparison results to one of the parties. In order to prevent such information leakage, we propose a mechanism, as an extension to the original scheme [9] as shown in Protocol 3.

Blake's protocol allows us to obliviously transfer one over two secrets depending on the result of the secure comparison. The protocol considers the scenario where there are two parties holding two private inputs x and y. The second party holds two secrets (s_0, s_1) (in addition to private input y). Blake's protocol allows the two parties obliviously transfer s_0 when $x > y$ and s_1 in the other case. Our modification adds one more step which is the secure equality evaluation protocol to determine the secret that has been sent. At the end of the protocol, Bob obtains an encrypted bit that indicates the result of the inequality comparison. To present the protocol, we follow Blake [9] and denote by D_S a set of integers agreed by the two parties before executing the protocol.

Algorithm 3: Secure Inequality Evaluation

```
Input: Alice holds integer $x$ and Bob holds integer $y$; common integer set $D_s$
Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $x < y$, and 0 otherwise
1 Bob draws two random number $s_0 \neq s_1$ from the set $D_s$;
2 Denote the bit representation of $x$ by $x_n \cdots x_2 x_1$; Alice encrypts each bit $x_i$ under her key and sends $(Enc_A(x_1), \ldots, Enc(x_n))$ to Bob;
3 For each $i = 1, \ldots, n$, Bob does the following:
    $(a)$ Compute $Enc_A(d_i) = Enc_A(x_i - y_i)$
    $(b)$ Compute the encryption of the XOR between $i - th$ bits $Enc_A(f_i) = Enc_A(x_i \oplus y_i)$ using homomorphism.
    $(c)$ Compute an encryption of vector $\gamma$, where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$
    $(d)$ Compute an encryption of vector $\delta$, where $\delta_i = d_i + r_i(\gamma_i - 1)$, where $r_i$ is a random number in $Z_n$
    $(e)$ Compute a random encryption of vector $\mu$, where $\mu_i = \frac{s_1 - s_0}{2}\gamma_i + \frac{s_1 + s_0}{2}$ and send a random permutation $\pi(Enc_A(\mu))$ to Alice.;
4 Alice obtains $\pi(Enc_A(\mu))$, decrypts it and determines that $\mu$ contains a single value $v \in D_S$;
5 Alice and Bob perform secure equality evaluation on inputs $v, s_0$;
6 Bob outputs the encrypted result $Enc_A(b)$;
```

In Step 3.b, Bob is required to compute the *XOR* of two encrypted bits x_i, and y_i. Since $f_i = x_i \oplus y_i = x_i + y_i - 2x_i y_i$, we can evaluate the result with the help of the secure multiplication protocol (Protocol 2). To compute the encryption of vector γ, δ, μ, Bob only needs to apply the homomorphic property of the Paillier cryptosystem as discussed in Section 4.2.

Analysis. We first show that the protocol correctly computes the desired functionality. The flag vector $f = \{f_i = x_i \oplus y_i\}$ is a binary vector, where the i-th bit indicates whether $x_i \neq y_i$. Therefore, vector γ as constructed in Step 2c is a vector with the following structure: it starts with one or more 0s followed by a 1, and then a sequence of non-1s.

Let *k* be the first position where x_i and y_i differ, which implies that $\gamma_k = 1$ and d_k determine the result of predicate $x < y$. δ randomizes the value of γ but keeps $\delta_k = d_k$. We note that with a large probability, γ_i is statistically close to uniformly random in Z_N. Finally, the transformation in Step 2e is a permutation of Z_n where set $-1 \rightarrow s_0$ and $1 \rightarrow s_1$. The final step, where a random permutation $\pi(\mu)$ is sent back to Alice, hides information concerning index *k*.

Since there is a negligible minority of elements of D_S in a group of size *N*, with overwhelming probability, there is exactly one element in vector μ belonging to set D_S. In Step 3, Alice can output either s_0 or s_1 depending on whether $x < y$. The last two steps conclude our construction of the protocol, where $v = s_0$ only if $x < y$ as desired.

We now prove the security of the protocol. Due to the universal security of the secure equality evaluation protocol, we only need to consider the first part (*i.e.* Steps 1 - 3). Privacy for Alice trivially holds because of the semantic security properties of the employed encryption scheme—the Paillier cryptosystem. Bob only receives from Alice a list of encryption messages, and obtains no more information about Alice's private input.

Bob's privacy against the semi-honest party Alice is proven by constructing a simulator $Sim_A(x, v)$, where *x* is the private input of Alice and *v* the value obtained in the part of the protocol that is examined. $Sim_A(x, v)$ needs to generate a distribution statistically close to the view of Alice in real execution. The simulator generates a random vector μ': for $i = 1, \cdots, n$, a random element $\mu'_i \in Z_n$ is chosen. It then replaces the randomly chosen element of μ' with *s* (*i.e.* $\mu'_i \leftarrow s$), and outputs $\{x, Enc(\mu')\}$. As discussed above, due to the randomization in Step 2.d, vector μ is statistically close to being uniformly random in Z_N (except one element in D_S).

5.3.3. (Non-) Membership Predicate

A membership predicate allows the verifier to examine whether an attribute of the client falls into certain categories. A simple example is the case where the verifier wishes to know if an applicant works in the education industry (e.g. teacher, student, librarian, school counsellor, etc.). A non-membership predicate is the complement of the membership query, and tests whether a particular value is excluded from a set.

The membership predicate evaluation protocol is presented in Protocol 4. The non-membership predicate can be easily derived from Protocol 4 by applying the *NOT* operator discussed in Section 5.4.

In the protocol, Alice is required to evaluate the encrypted polynomial $P(x)$ at point $x = a$ (line 3). She can do so due to the homomorphism of the cryptosystem. She first computes a^i in plaintext, and then computes $Enc_B(c_i \times a^i) = Enc_B(c_i)^{a^i}$ by the homomorphic multiplication property of the Paillier cryptosystem. Finally, she calculates $P(a) = \sum c_i \times a^i$ in encrypted form by applying the homomorphic addition property.

```
Algorithm 4: Secure Membership Evaluation
  Input: Alice holds an integer $a$, Bob holds a set if integer
         $S = \{x_1, x_2, \cdots, x_n\}$
  Output: Bob holds an encrypted bit $Enc_A(b)$ such that $b = 1$ if $a \in S$ and
          0 otherwise
1 Bob computes the characteristic polynomial of the set:
    $P(x) = c_n x^n + \cdots + c_1 x + c_0 = (x - x_1)(x - x_2) \cdots (x - x_n)$;
2 Bob encrypts the coefficients of a polynomial $\{c_0, c_1, \ldots, c_n\}$ under his own
    key, obtaining $\{Enc_B(c_0), Enc_B(c_1), \ldots, Enc_B(c_n)\}$, sends to Alice the
    encrypted set;
3 Alice securely evalutes the encrypted value of the polynomial at $x = a$,
    denote $v \to P(a)$;
4 Alice generates random $r$, and compute $Enc_B(v + r)$ and send to Bob;
5 Bob decrypts to obtain $v + r$;
6 Alice and Bob performs Secure Equality Evaluation to with inputs $r$,$v + r$;
7 Bob outputs the encrypted result Enc(b);
```

Analysis. We first analyse the correctness of the protocol. If $a \in S$, there exists one x_i such that $x = x_i$. This implies $v = P(a) = 0$. This observation leads to the final step of the protocol, where Bob and Alice collaboratively evaluate the equality predicate with inputs r and $v + r$. Hence, the encrypted bit $Enc(b)$ is an indicator of whether value *a* belongs to set *S*.

The security the protocol can be proven with two simulators that generate the views of the two parties, Alice and Bob. For Alice, a simulator that generates and sends *n* random encrypted values is a valid simulator. Due to semantic security, she cannot distinguish the simulator from a real-world scenario. Similarly for Bob, a random number $v + r$ can be easily simulated. Finally, the security of Protocol 4 concludes the proof of security of the secure membership evaluation protocol.

5.4. Complex Predicate Evaluation

At this stage, Bob holds the encrypted result of the evaluation for each data record, with each attribute in a complex predicate that needs to be verified. This sub-section discusses three basic primitives that operate on the encrypted inputs at this stage. With these primitives, Bob has the capability to compute the results of the encryption of the desired bit to evaluate each data record. The output of this stage is an encrypted bit for each data record. This bit indicates whether the given record satisfies the complex statement.

The inputs of the three primitives are either one encrypted bit (NOT operation) or two encrypted bits (AND and OR operations). They are described as follows:

1) $\neg$ (*NOT*)—It is easy to derive the formula for bit negation operation: $Enc(\neg x) = Enc(1) - Enc(x)$. Clearly, the operation leaks no information regarding the encrypted bit x to either Alice or Bob. It requires one exponentiation operation and one multiplication operation. Alice receives no more data, whereas Bob only works on his inputs, which are encrypted data.

2) $\wedge$ (*AND*)—Because $x \wedge y = x \times y$ for any two bits x, y, the primitive is identical to the description of SecMul (Protocol 2). The protocol requires five multiplication operations and five exponentiation operations in ciphertext space. The security of the protocol follows the analysis of secure multiplication (*i.e.* Protocol 2).

3) $\vee$ (*OR*)—Since $x \vee y = x + y - x \times y$, we can derive the definition of the OR primitive as in Protocol 5. The protocol requires seven multiplication operations and six exponentiation operations in ciphertext space. During the protocol, data that Bob and Alice receive are identical to those received during Protocol 2; hence, Bob and Alice gain nothing following protocol execution.

Algorithm 5: Secure OR Operation

```
  Input: Bob holds $(Enc(x), Enc(y))$, and Alice holds the private key $sk$
  Output: Bob holds $Enc(x \vee y)$
1 Bob and Alice collaboratively compute $Enc(x \times y)$ using Protocol 2 ;
2 Bob computes $r = Enc(x) + Enc(y) - Enc(x \times y)$;
3 Bob outputs r;
```

5.5. Aggregation of Output Data

As the input of this stage, for each entity in Alice's database, Bob holds an encrypted bit that determines whether the data record qualifies the complex statement. In order to ensure there is exactly one qualified data record in case the application is successful, we introduce one special attribute to the final complex predicate. The attribute is the secret identification of the client in the database.

We assume that when the client registers his/her data with Alice the data keeper, Alice generates a secret random number r_c to identify the client. The number is stored in the database as an attribute of the client. We introduce additional steps to address the requirement:

1) The client encrypts the random secret under Bob's key, obtain $Enc_B(r_c)$.

2) The client anonymously sends the encryption of secret value to Alice.

3) Alice and Bob perform secure equality evaluation (starting from step 2) and get the result $Enc_A(b)$.

4) Bob applies *AND* operation with $Enc_A(b)$ and the current result of evaluation process.

With the additional step, now Bob holds an array of encrypted bits with all 0s and at most one bit 1. Bob uses a homomorphism to compute the encrypted sum of these bits; the result is the encryption of either 1 or 0. He can send it to Alice for decryption and obtain the final result to determine whether the applicant qualifies. However, this may compromise Bob's privacy, especially when he wants to hide his business progress. In order to maintain his privacy, we introduce one step for the randomization of the decryption process as follows:

1) Bob computes the encrypted sum using a homomorphism to obtain $Enc_A(s)$.

2) Bob generates a random number r, and computes $c = Enc_A(s+r)$ and sends to Alice.

3) Alice decrypts c to obtain $s+r$ and sends it back to Bob. Note that Alice only receives a random number so that she learns nothing about the result of the application.

4) Bob computes the result $s = s+r-r$.

Finally, Bob is able to decide the result of the verification process by bit s.

5.6. Discussion

We first consider the security of the entire system, since all intermediate results revealed to Alice and Bob are either random or semantically secure encryptions of numbers. Furthermore, the outputs of all sub-protocols (only seen by Bob) are always encrypted under Alice's key. Under the assumptions of the semi-honest model, we claim that the sequential composition of these sub-protocols leaks no details of the client or the predicates proposed by the verifier.

The second issue we consider is the practical implementation of the system. Since the same procedure is applied for all the data entries, the verification results for each data record can be computed in parallel. That means we are able to construct multiple verification threads, each one is corresponding to one data entry. By the batch verification approach, we can improve the running time of the whole process by a factor of n/m, where n is the number of data records and m is the number of threads.

While the same procedure is applied for each data record, the data keeper is not able to know who is the applicant. In practice, there are some cases that the data keeper (e.g. a governmental agency) is allowed to know the identity of the applicant, where this rigorous security feature is then not required. The proposed solution can be modified, and inherently improves performance. Specifically, the client can perform a simple authentication rather than an anonymous solution to allow the verifier to communicate with the data keeper. The verification process only needs to be performed on the only one data record identified by the client. Hence, the cost of the proposed solution is reduced by a factor of n where n is the number of data records in the data keeper's database.

In our proposed solution, an applicant qualifies only if he/she satisfies all criteria specified by a single predicate or complex predicates. Hence, we can define a complex predicate to cover all criteria using the AND operation. We also can extend our protocol to adapt to threshold criteria, where the applicant qualifies only if he/she satisfies more than k criteria. The idea is to compute the sum of each predicates evaluation (in encrypted form) and apply a slightly modified version of Protocol 3 to compare the encrypted value with threshold k.

6. Implementation

We implemented our proposed method, and calculated the CPU time required to run our sub-protocols from Section 5. Our experiments were conducted on a Windows 10.0 machine with a 3-GHz processor and 16 GB of RAM. We used the Paillier cryptosystem as the underlying additive homomorphic encryption scheme and implemented the proposed sub-protocols in Java.

We first examined the operation of the secure equality evaluation and the secure inequality evaluation protocols. Two factors affect the performance of these protocols: the Paillier key size and the domain size of the input. Table 2 shows the processing times of Protocols 1 & 3 with different settings of bit size and key size. We performed the experiment with bit lengths of 32, 64 and 160. The latter was the size of the output of the SHA-1 hash function we used for the secret identification described in Section 5.5. The result showed that these protocols require twice the time for double-bit size of inputs; the time needed increased by a factor of nearly 7 when the Paillier key size was doubled.

The third single-predicate evaluation building block was the (non-) membership predicate. The run time of the building block depends on three factor: the Paillier key size, the number of elements in the set and the bit size of the inputs, where bit size only affects the final step of Protocol 4, which is the secure equality evaluation protocol. Figure 2 show the relationship between the

Table 2. Run times of secure equality and inequality evaluation protocols (ms).

1024 bit Key size		
Size	Prtcl.1	Prtcl.3
32	796	1769
64	1472	3542
160	3277	8623
2024 bit Key size		
Size	Prtcl.1	Prtcl.3
32	4477	12,047
64	9983	24,755
160	22,569	57,393

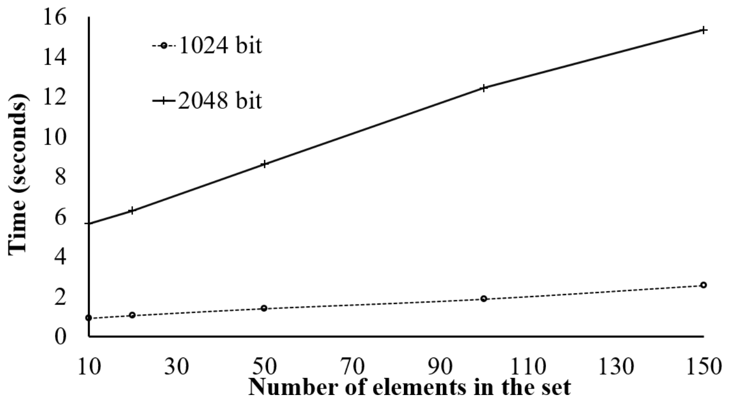

Figure 2. Running time of Secure Membership Evaluation.

Table 3. Running times of proposed sub-protocols.

Key Size	Secure Negation	Secure AND	Secure OR
512	4 ms	20 ms	22 ms
1024	17 ms	73 ms	86 ms
2048	81 ms	517 ms	558 ms

run times of the two remaining factors and the performance of the building block.

We had made a similar observation earlier: the cost of the secure membership evaluation protocol when the key size was 1024 bits was roughly six to seven times more efficient than with a length of 2048 bits for the Paillier key. The computational cost of the protocol also increased linearly with the size of the set. Finally, the run time of the three protocols that evaluated the Boolean functions are shown in Table 3. The same characteristics concerning the effect of the Paillier key size held for these building blocks.

In order to verify the feasibility of the whole proposed system, we conducted an experiment on a simulated dataset. We consider a complex statement verification comprising of 10 single predicates linking together by two boolean operations *AND*, *OR*. The running time for verifying single data record was 25 seconds, and it took approximately 1 hour to verify one thousand data record in the parallel mode of 10 threads running simultaneously.

7. Conclusion

In this paper, we proposed a framework for privacy-preserving verification of personal information. We used the secure multi-party computation model and homomorphic encryption to develop a systematic solution to the problem in four stages. We showed that the proposed scheme can protect the clients privacy from both the verifier and the data keeper, and at the same time provides privacy to the former. Different ways to further enhance the performance of the proposed method and a scheme extension for threshold verification were discussed. The experimental results highlighted the efficiency and feasibility of our proposed scheme under different security settings.

References

[1] Goldwasser, S., Micali, S. and Rackoff, C. (1985) The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). *Proceedings of the* 17*th Annual ACM Symposium on Theory of Computing*, Providence.

[2] Yang, Y.J., Zhou, J.Y., Weng, J. and Bao, F. (2009) A New Approach for Anonymous Password Authentication. *Twenty-Fifth Annual Computer Security Applications Conference*, Honolulu.

[3] Groth, J. (2005) Non-Interactive Zero-Knowledge Arguments for Voting. *Third International Conference on Applied Cryptography and Network Security*, New York. https://doi.org/10.1007/11496137_32

[4] Camenisch, J., Hohenberger, S. and Lysyanskaya, A. (2006) Balancing Accountability and Privacy Using e-Cash (Extended Abstract). 5*th International Conference on*

Security and Cryptography for Networks, Maiori.
https://doi.org/10.1007/11832072_10

[5] Camenisch, J., Chaabouni, R. and Shelat, A. (2008) Efficient Protocols for Set Membership and Range Proofs. *Advances in Cryptology-ASIACRYPT* 2008*,* 14*th International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne.

[6] Kissner, L. and Song, D. (2005) Privacy-Preserving Set Operations. *Advances in Cryptology-CRYPTO* 2005: 25*th Annual International Cryptology Conference*, Santa Barbara. https://doi.org/10.1007/11535218_15

[7] De Cristofaro, E., Gasti, P. and Tsudik, G. (2012) Fast and Private Computation of Cardinality of Set Intersection and Union. *Cryptology and Network Security*, 11*th International Conference, CANS* 2012*,* Darmstadt.

[8] Yao, A.C.-C. (1982) Protocols for Secure Computations (Extended Abstract). 23*rd Annual Symposium on Foundations of Computer Science*, Chicago.

[9] Blake, I.F. and Kolesnikov, V. (2009) One-Round Secure Comparison of Integers. *Journal of Mathematical Cryptology*, **3**.

[10] Gentry, C., Halevi, S., Jutla, C.S. and Raykova, M. (2015) Private Database Access with He-Over-Oram Architecture. 13*th International Conference on Applied Cryptography and Network Security*, New York.
https://doi.org/10.1007/978-3-319-28166-7_9

[11] Paillier, P. (1999) Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Advances in Cryptology—EUROCRYPT*99, *International Conference on the Theory and Application of Cryptographic Techniques*, Prague.
https://doi.org/10.1007/3-540-48910-x_16

[12] Damgård, I. and Jurik, M. (2001) A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. *Public Key Cryptography*, 4*th International Workshop on Practice and Theory in Public Key Cryptography*, Cheju Island. https://doi.org/10.1007/3-540-44586-2_9

[13] Goldreich, O. (2004) The Foundations of Cryptography: Vol. 2, Basic Applications. Cambridge University Press, Cambridge.

[14] Chaum, D. (1985) Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, **28**, 1030-1044.
https://doi.org/10.1145/4372.4373

[15] Camenisch, J. and Lysyanskaya, A. (2002) A Signature Scheme with Efficient Protocols. 3*rd International Conference on Security in Communication Networks*, Amalfi.

[16] Di Crescenzo, G. (2000) Private Selective Payment Protocols. *Financial Cryptography*, 4*th International Conference*, Anguilla.

[17] Fischlin, M. (2001) A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires. *Topics in Cryptology*, *The Cryptographer's Track at RSA Conference*, San Francisco.