Quantum-Resistant Network for Classical Client Compatibility

Te-Yuan Lin
Ph. D. Candidate of Department of Computer Science and Information Engineering, National Taiwan University; No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan; phone: + 886 2 33664888; fax: +886 2-23628167; e-mail: d03922002@ntu.edu.tw

Chiou-Shann Fuh
Professor of Department of Computer Science and Information Engineering, National Taiwan University; No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan; phone: + 886 2 33664888; fax: +886 2-23628167; e-mail: fuh@csie.ntu.edu.tw

Corresponding author: d03922002@ntu.edu.tw

Quantum computing is no longer a thing of the future. Shor’s algorithm proved that a quantum computer could traverse key of factoring problems in polynomial time. Because the time-complexity of the exhaustive key search for quantum computing has not reliably exceeded the reasonable expiry of crypto key validity, it is believed that current cryptography systems built on top of computational security are not quantum-safe. Quantum key distribution fundamentally solves the problem of eavesdropping; nevertheless, it requires quantum preparatory work and quantum-network infrastructure, and these remain unrealistic with classical computers. In transitioning to a mature quantum world, developing a quantum-resistant mechanism becomes a stringent problem. In this research, we innovatively tackled this challenge using a non-computational difficulty scheme with zero-knowledge proof in order to achieve repellency against quantum computing cryptanalysis attacks for universal classical clients.

KEYWORDS: Cloud Computing Security, Homomorphic Encryption, Quantum Computing, Zero-Knowledge.

1. Introduction

Computing can occur in any location and using a wide range of devices. The path to this accomplishment has passed through mainframe and personal computing, and then to Internet computing. Computation operates in an increasingly distributed manner; thus, data leakage threats have become ubiquitous. The current
approach to information security depends heavily on public key cryptography systems of computational security, where the security commitment comes from the time required for exhaustive key search exceeding that required for cryptography key validity, and the attacking resource cost outweighs the value of the message itself. The safety keeps assured until Shor’s algorithm [26] formally proved that a quantum computer could traverse key of factoring problems in polynomial time. As quantum computing power grows, computational complexity loses its intractability [16]. Eavesdroppers can be the Man-In-The-Middle (MITM), malicious communication counterparts, inadvertent/ intentional cloud providers, and even governments. Although owning a private quantum computer is unrealistic, attackers can leverage quantum computing services from cloud providers with quantum supremacy to conduct cryptanalysis attacks. Hence, it is imperative to identify different approaches to classical cryptography algorithms, which are known as Post-Quantum Cryptographies (PQC).

1.1. Paper Organization and Contributions

This paper is organized as follows:

Section 1 gives a brief overview of quantum cryptography and reviews research related to quantum resistance. Section 2 provides a detailed analysis of the important methods used for the proposed scheme. In Section 3, we further explain the methods in Section 2 and depict how we transform Zero-Knowledge Proof system concept to concrete proving/ verifying steps which suffice to support the idea of replacing public key exchange, to establish a quantum-resistant network. The algorithms comprising the scheme are given in Section 3, and the algorithm is summarized in Figure 4. The scheme extends the capability of traditional homomorphic encryption for addressing the vulnerability that quantum computers may bring. We offer a different perspective as an alternative to placing all bets on PKI improvements. We tested the real quantum computing provided by IBM Q experience of factorization and simulated the scheme algorithms in Section 4. The paper is summarized and concluded in Section 5.

Our main contributions are summarized below:

The novelty of the proposed scheme is how it achieves no public key-exchange during the entire protocol so that it greatly reduces the risk of quantum computing attacks.

We establish an initiative that combines ZKP and FHE methods to reduce possible breaches caused by third-party validation.

1.2. Quantum Cryptography

Similar to quantum computing from the principles of quantum mechanics, quantum cryptography takes advantage of qubit states with its un-trackable and no perfect cloning nature. Suppose there is an existing state of a quantum system A, denoted as $|\psi\rangle_A$, which we wish to clone without any prior knowledge. Then we take another independent quantum system, A', of initial state with identical Hilbert Space [17], denoted as $|e\rangle A'$. If we try to perform a measurement on A, the measurement immediately collapses the system into a certain eigenstate of the observable, totally corrupting the information contained in the original qubit. The alternative is to entangle the state of A and A' as a pair; these two systems can be seen as a composite object described by the tensor product $A \otimes A'$ of two vector spaces A and A', and its composite state is $|\psi\rangle A |e\rangle A'$. Next, to perform a unitary linear transformation ($U$) on quantum states to approximate the state $|\psi\rangle A$, this cloning can be denoted as

$$U |\psi\rangle A |e\rangle A' = |\psi\rangle A |\psi\rangle A'. \quad (1)$$

The imperfection that the theoretical bounds were derived on the fidelity of cloned quantum states has been proved. In classical communication channels, we use public key/ private key pair and a trusted third party to promise the cryptographic key distribution is assured, based on computational intractability. In a quantum communication channel, the proven limitation that guarantees no eavesdropper cannot create an exact copy of a quantum cryptography key is critical. The feature can be useful in Quantum Key Distribution (QKD) protocol, by which a private key can be generated between two parties over a public channel. QKD is provably secure [4] because the eavesdropper, Eve, cannot gain the exact information from the qubits transmitted from Alice to Bob without interfering with their state. By calibrating the threshold of the error rate, it is easy for Alice and Bob to be aware of whether eavesdropping is happening. Once the error
rate is higher than pre-defined threshold, they can abandon the key and re-initialize the key-negotiation process until the low error rate is met. The shared key can then be used to encrypt and decrypt the classical information for communicating parties as we do now. The first famous QKD scheme, invented by Charles Bennett and Gilles Brassard in 1984, is known as BB84 [3, 4].

QKD is rigorously secure (many researchers consider it unconditionally secure) [1, 12, 18], but it requires quantum network infrastructure and quantum devices at both sides to transmit quantum bits and generate their shared key. QKD has emerged from the laboratory but remains in the preliminary implementation stage [8]. It is not widely adopted for most communicators due to noise interfering and performance and cost concerns [13, 19]. The aforementioned limits entail little compatibility with the current classical client running infrastructure.

**1.3. Previous Quantum Resistance Research**

In 2012, the Computer Security Resource Center in National Institute of Standards and Technology (NIST) initiated a project called Post-Quantum Cryptography standards [22], for reviewing contemporary technologies to develop effective new algorithms for protecting electronic information from attack by the computers of both tomorrow and today. The goal is simple—keeping existing public key infrastructure intact in a future era of quantum computing. The most promising replacements of public key infrastructure are proposed and analyzed; these fall into a couple of large algorithm families: lattices-based, code-based, multivariate-based, hash-based, and isogenies-based. Each has its own pros and cons. The comparison of each main family, today versus PQC effectiveness, is listed in the following table (see Table 1). Some other modernized implementations, such as Post-Quantum Yao and PQ-QT [6], still fall in the main family here. The common and noteworthy merit of these algorithms is compatibility with classical clients, without the need for augmented quantum preparation, qubits measurement, or quantum channel transmission. In contrast to quantum augmentations and quantum channels, the cost of the above algorithms is quite acceptable; however, except hash-based family, the security promises of the remaining rely heavily on the mathematical difficulty barrier, i.e., computational security. In a hash-based algorithm, instead of encrypting and decrypting messages directly, they are primarily used on digital signatures, so additional cryptographic complements would be necessary for the unhandled methods.

In this paper, we emphasize that a PQC scheme whose security commitment relies on none of three hard mathematical problems (i.e., the integer factorization problem, the discrete logarithm problem, and the elliptic-curve discrete logarithm problem) finding a path in the isogeny graph of super-singular elliptic curves.

**2. Methods Analysis**

### 2.1. Proof System for Authentication

The essential disparity between quantum computing and classical computing is computing speed. With that in mind, instead of complexity confrontation, devising a scheme that is radically independent of the mathematical difficulty barrier may be the best way to avoid quantum attacking. The Zero-Knowledge Proof (ZKP) method is one of the problem-solving candidates.

A ZKP is a system to prove the authenticity of a statement without leaking extra information of statement. A famous metaphor, Ali Baba’s Cave [25], is quite a good example to show the ZKP philosophy intuitively. The cave layout is illustrated as in Figure 1, with the entrance on one side (Point E) and the magic door blocking in the middle walkway at

| Table 1                                                                 | Main Family | Proposed Signatures | Key Exchange Size | Efficient |
|------------------------------------------------------------------------|-------------|---------------------|-------------------|-----------|
| Lattices-based                                                         | 2.7 kb      | 1 kb                | Y                 |
| Code-based                                                             | 192 kb      | 1 mb                | Y                 |
| Multivariate-based                                                    | Usually smaller than public key | 10-100 kb | Y         |
| Hash-based                                                             | 41 kb       | 1 kb                | Y                 |
| Other – Isogenies-based                                               | 1228 kb     | 330 bytes           | N                 |
the cave’s opposite side; only the person who knows the secret word of a magic spell can unlock the door. The verifier (V) wants to know whether the prover (P) holds the secret word (w); P also wants to prove his awareness of w, but he does not want to reveal his knowledge (i.e., w) to V or to reveal his knowledge to other observers (O). To achieve this with ZKP, they can label the left and right paths as A and B and set some ground rules. First, V waits outside the cave at E as P goes in. P chooses either path A or B. Second, V is not allowed to see which path P takes. Then, V enters the cave and stands at point M, shouting the name of the path via which V wants P to return, either A or B, chosen at random. Providing P really does know the magic word; this is easy: P opens the door using w, if necessary, and returns along the desired path. However, suppose P did not know the magic word, w, then P would only be able to return via the same path by which he had entered. Since V would choose A or B at random, P would have a 50 percent chance of guessing correctly. If they continued the round many times, say 20 times in a row, P’s chance of successfully anticipating all of V’s requests would become extremely low (i.e., 1/220, about one in a million). Thus, if P repeatedly appears at the exit V requests, then V can conclude that it is very likely that P does know w. From start to finish, w is unknown to the V and the other observers, O, and this embodies zero-knowledge.

2.2. Homomorphic Encryption System for Confidentiality

An intuitive approach for preserving the privacy of user information in cloud-based services is to encrypt everything before sending it to the cloud. This is secure, but the cloud service cannot operate on data to provide computing advantages before decryption. The dilemma is that we can never securely transmit a conventional decrypting key. Thus, we wonder whether there exist encryption schemes that allow some computation to be performed directly on encrypted data (without first decrypting it). The birth of homomorphic encryption (HE) was exactly to this end, computation on ciphertexts without decrypting sensitive data. This empowers the users to outsource their computing work to cloud service providers securely or to chain different services (secure multi-party computation) together. A general HE scheme, ε, is primarily characterized by four operating phases: KeyGen, Enc, Dec, and Eval, as shown in Figure 2.

Let plaintext \( m \in M \), ciphertext \( c \in C \), and a key pair \( k \in K \), where \( K \) is a secret key space for encryption and decryption. In KeyGen, we design a specific algorithm to generate a key pair used for the next encryption, decryption, and evaluation phases. The output of KeyGen can be denoted as \( k = (k_e, k_d) \). It is known as symmetric HE if \( k_e = k_d \); asymmetric HE if \( k_e \neq k_d \). Here, we use the symmetric key and deem it the same as P’s secret key (a simplified w). To avoid some sophisticated attacks, the actual \( k \) is a hashed mixture of a random oracle and an initial vector composed to-
gather as a dynamic key generating process. We will elaborate on the mechanism further in Section 3.

In phase \( \text{Enc}_\varepsilon \), the inputs are \( k \) and \( m \), so the output, \( c \), can be denoted as

\[
c = \text{Enc}_\varepsilon (k, m).
\]  

(2)

In phase \( \text{Dec}_\varepsilon \), the inputs are \( k \) and \( c \), so the output, \( m \), can be denoted as

\[
m = \text{Dec}_\varepsilon (k, c).
\]  

(3)

In the phase \( \text{Eval}_\varepsilon \) (a.k.a. re-encryption), it is correlated with a set of operating functions \( F \), and the output can be denoted as

\[
\text{Eval}_\varepsilon (k, f, c_1, c_2, \ldots, c_t) = f (m_1, m_2, \ldots, m_t)
\]  

(4)

for every Boolean function \( f \in F \varepsilon \) and arbitrary ciphertext \( c_1, c_2, \ldots, c_t \), where \( c_i = \text{Enc}_\varepsilon (k, m_i) \).

\[
\forall f \in F \varepsilon, \forall m \in M, \text{ciphertext } c \in C.
\]  

(5)

Thus, the correctness of the scheme is satisfied when

\[
C \leftarrow \text{Eval}_\varepsilon (k, f, c) \leftrightarrow f (m_1, m_2, \ldots, m_t) = \text{Dec}_\varepsilon (k, c).
\]  

(6)

In short, the operation \( F \varepsilon \) in ciphertext space, \( C \), which is constructed by the encryption after the completion of \( f \) on its corresponding plaintext space, \( M \), can be denoted as

\[
F \varepsilon (C) = \text{Enc}_\varepsilon (f (M)).
\]  

(7)

where \( f \) is usually a time-consuming/compute-intensive task, with HE, since \( f (M) \) and \( C \) are homomorphic, any \( F \varepsilon \) operation executing on \( C \) by the third-party service provider is equivalent to its counterpart, \( \text{Enc}_\varepsilon (f (M)) \).

As a result, decrypt the \( F \varepsilon (C) \), and we get \( f (M) \), so we are allowed to delegate \( \text{Enc}_\varepsilon (f (M)) \) to any third-party service providers securely.

In the early version, \( \text{Enc}_\varepsilon \) is classified as a Partially Homomorphic Encryption (PHE) algorithm of \( f \), because \( f \) can only operate addition or multiplication under encryption, but not both; then, an advanced version scheme comes in and is classified as Somewhat Homomorphic Encryption (SHE) since \( f \) can operate both addition and multiplication with limited rounds. Finally, \( f \) can operate both addition and multiplication with no infinite rounds, so the \( \text{Enc}_\varepsilon \) scheme is deemed a Fully Homomorphic Encryption (FHE), which was implemented in 2009 based on PHE and SHE by Gentry [31].

FHE cryptosystems have better practical implications in the outsourcing of private data computations. An intuitive example is a line of assembling gold or diamonds into rings. A piece of gold (data) is locked inside a glovebox (encrypted by FHE) so that a worker (the cloud service provider) may transform it into a ring (the computed result). The ring is later taken out by the ring owner (data owner) when the glovebox is unlocked by the ring owner’s key (data owner’s secret key). This prevents the key from transmitting risk (hacking/embezzled) and the possibility of gold stolen (data stolen), shown as the following figure 3.

**Figure 3**
Glovebox idea and FHE [20]
3. Proposed Scheme

Based on the above-elaborated analysis on currently known methods, we can merge the two systems into a new scheme in a novel way.

Instead of relying on public key infrastructure, we transform the data privacy problem of outsourcing cloud computing into a zero-knowledge proof manner. The scheme mixes ZKP and FHE, acting as Ali Baba’s cave with an FHE glovebox in-between the classical client (Alice) and her cloud service provider (Bob). We name the scheme ZKP-Cave after this philosophy (as shown in Figure 4).

Figure 4
ZKP-Cave — a fully homomorphic encryption scheme that conforms with zero-knowledge proof and is compatible with classical parties

ZKP-Cave elements:
- The prover, P: Alice, the cloud service consumer who has her secret key, w, to encrypt her data and plans to outsource her computing tasks to the cloud service provider, Bob. Before Bob accepts Alice’s tasks, Alice must prove she really is the secret key owner who uses the key to encrypt the data within the tasks.
- The verifier, V: Bob, the cloud service provider who has sufficient computing and memory resources, even attainable to the quantum-grade. Bob verifies the consumer’s identity and ownership before processing the computing tasks.
- The eavesdropper, E: Eve, the eavesdropper who is spying on the classical channel between Alice and Bob. Eve can be an imposter and conducts MITM attacks. Furthermore, Eve can delegate her computing tasks to any cloud service provider, even attainable to the quantum grade.
- The statement, x: Since Alice uses w to encrypt the data to be processed, the statement is that Alice knows the valid data owner’s secret key w.
- The proof, π: It is related to the parameters x and w, i.e., π ← Prove (x, w); without revealing w to V, P has to convince V that her knowledge of w to hold the inequality 1 ← Verify (x, π) true.
- The private data set was pre-processed by a specific FHE algorithm, ε, before transmitting to the verifier, V. The encryption key, w, needs to be generated in the phase, KeyGen, and used to encrypt the data set.

For the entire proving protocol, V can learn nothing about P’s knowledge of w.

3.1. Pseudo-code of Two ZKP-Cave Algorithms

Algorithm 1. Avoid the public key exchange risk via the proving algorithm

Initialized inputs:
- rounds: the rounds of executing the dummy test task
- randNum : a random integer number generated from a Random class

```java
private static void main(String args[]) {
    int t = 0;
    int operand1, operand2;
    int[] operandType = new int[t];
    int[] operationResultSet = new int[t];
    while (t < rounds) {
        operand1 = randNum.nextInt();
        operand2 = randNum.nextInt();
        randomOperatorPickup();
    }
```
12: \( \text{operationResultSet}[t] = \text{calculateOperands}(\text{operand1}, \text{operand2}, \text{operandType}[t]); \)

13: \( t ++; \)

14: }

15: // Storing each encrypted pair of operands
16: for (int i = 0; i < operandSet.length; i++) {
17: \( \text{operandSet[i]} = \text{new OperandPair} (\text{Enc}(\text{operand1}, \text{w}), \text{Enc}(\text{operand2}, \text{w}) \));
18: }

19: // Shuffle the element sequence of each set and submit them to cloud provers
20: \( \text{submitShuffleSet(operandSet, operationResultSet);} \)

21:}

22:

23: \( \text{public static int calculateOperands(int operand1, int operand2, String operator)} \)
24: {
25: \( \text{int operationFHE, operationResult;} \)
26: \( \) // Each operational result must be processed by FHE with private key w
27: \( \text{operationFHE} = \text{EncFHE(String.valueOf(operand1)} + \text{operator.charAt(0)} + \text{String.valueOf(operand2)}); \)
28: \( \text{operationResult} = \text{(int)} \text{operationFHE.toString();} \)
29: \( \text{return operationResult;} \)
30: }

31: \( \text{public static String randomOperatorPickup()} \) { char[] Operators= new char[]{'+','*'};
32: \( \) if(randNum == 0 || randNum % 2 == 0) Operators [0] = '+';
33: \( \) else Operators [0] = '*';
34: \( \text{return String.valueOf(Operators[0]);} \)
35: }

36:

Output: Shuffled operandSet and operationResultSet pair

---

**Algorithm 2.** The verifying algorithm (Processing the privacy data only after the prover passed verification)

**Initialized inputs:**
- confidence: the minimum level to accept P as true
- hitRateLevel: an adjustable threshold that considers reasonable network noise

1: \( \text{private static void main(String args[])} \) {
2: \( \text{int t = 0;} \)
3: \( \text{long operand = 0;} \)
4: \( \text{long operandResult = 0;} \)
5: \( \text{Long returningResult = 0;} \)
6: \( \text{int confidence = 0;} \)
7: \( \text{float hitRateLevel = 99.0;} \)
8: }
9: // Initialize the operand object of shuffleSet for verification
10: \( \text{OperandPair op = OperandPair.initInstance(shuffleSet);} \)
11: }
12: // Supply the encrypted operand pair object randomly and compare the result
13: \( \text{while (t < rounds)} \) {
14: \( \text{operand = op.getRandomOperand();} \)
15: \( \text{operandResult = op.getRandomOperandResult();} \)
16: }
17: // Collect Prover’s returning answer
18: \( \text{returningResult = op.send(operand);} \)
19: \( \text{if(returningResult == operandResult) \{ confidence ++;} \)
20: \( \)
21: \( \text{t ++;} \)
22: }
23: \( \text{if(confidence / t >= hitRateLevel)} \) { \text{if(confidence / t >= hitRateLevel)} {
24: \( \text{\} else \{ // Hacker detected, invalidate the session} \)
25: }
26: // Proving process passed, the session is secure
27: \( \text{\} else \{ // Hacker detected, invalidate the session} \)
28: }
29: }

Output: Pass or fail the verifying session
3.2. Incorporate with Improved Fully Homomorphic Symmetric Encryption

The version of Fully Homomorphic Encryption (FHE) has two main classes: asymmetric and symmetric. In the early version of the asymmetry-based approach, the size of encrypted data proliferates rapidly. It is inevitable for a bootstrapping framework, which accumulates noise of a large number of keys per computation [9]. Most as-is schemes are devoted to decreasing growing data size and processing overhead; the papers [5, 7, 20, 23, 29] present the improved version of FHE without Gentry’s bootstrapping prerequisite.

However, almost all asymmetry-based approaches are based on the same assumption (i.e., the large integer factorization remains a hard problem). Given the evolving maturity of quantum computing, the exposure of a public key can lead to the compromise of its private key used in the first place.

Our scheme sets the symmetric version FHE as the basis for encryption/decryption-sensitive data to become immune to the above attacking problems in a quasi-quantum cloud computing world. Per the KeyGen process discussed in the section of methods analysis, caution taken with the use of symmetric FHE in our scenario should design to be resistant to chosen and known-plaintext attacks from on-premises network interceptors, although the threat has nothing to do with the initial proposition of quantum-power attacks from the cloud or internet eavesdroppers. Our solution to this type of attack is incorporating every time dynamic key generation and dynamic block encryption while executing homomorphic encryption. Likewise, the extra inversed process should be involved during the decryption process of symmetric FHE, referring to P’s secret key (i.e., w). Hariss et al. [9, 10] have proposed the detailed implementation of dynamic key generation and dynamic block encryption.

4. Test and Evaluation

To analyze as-is quantum attacking force and the proposed scheme capability, we conducted several tests and evaluations to evaluate its effectiveness.

4.1. Factorization Attack Test on IBM Quantum Cloud Computers

The implementations are done with Python, Jupyter Notebooks, and IBM QISKit [11, 24], which allow developers to explore IBM Q Experience [14, 15] – a real cloud-enabled platform of quantum processors. In this work, the Shor Factorizing algorithm has been executed on IBM’s quantum computer, as shown in Figure 5.

Figure 5
Shor Factorizing Test on IBM Q Experience, as of early 2020

In earlier research, the theoretical circuits of Shor’s algorithm use 2n+3 qubits for factoring [2], yet the circuits consume 4n+2 qubits for factoring with IBM Q Experience in practice, according to our observation. We infer that the overhead comes from auxiliary quantum registers used in addition and multiplication of algorithm implementation. Furthermore, our test result shows that burst time in quantum processors consumes rapidly when the slow rise of small integer factorization. The trend has self-explained enough, even with IBM’s state-of-the-art 53-qubit quantum computer [21], still hard to deal with the length of 13-bit integer factorization in a flash time, not to say threaten 1024-bit or 2048-bit key length in which we found commonly used by cryptographic algorithms.
today. However, should Moore’s Law or even Neven’s Law [28] be applied to the number of quantum computing power qubits growth, in less than two decades, the factoring run chart result would be quite different.

4.2. Scheme Effectiveness Simulation

Referring back to the definitions of the ZKP-Cave elements in the section of the proposed scheme, we can simulate the interactions of proving steps and verifying steps.

ZKP-Cave Proving Steps:
- A cloud service consumer P first generates a dummy test task that includes random plaintext numbers (operands). P chooses a small number of plaintext numbers and calculates them with a random Boolean operator, either multiplication or addition.
- P encrypts the chosen plaintext operands and their Boolean operational result, respectively, with P’s secret key (i.e., w). P keeps these ciphertexts as a set of dummy test task results.
- P repeats the above process for t rounds and hence gets t sets of dummy task results. Within the dummy task results, there are two kinds of groups; one is the operand set, another is the operational result set. Every operand set has a corresponding operational result in the operational result sets.
- Next, P submits ciphertexts of operand sets and operational result in scrambled order to V over a classical channel.

ZKP-Cave Verifying Steps:
A cloud service provider V receives and stores the ciphertexts of operand and operational result sets from P for later processing and verification.
- V verifies P, who claims to be authentic, by sending an encrypted operand pair that is randomly picked up out of the stored operand sets and asks for the correct encrypted operational result repeatedly. Once V gets the response, regardless of whom, he immediately compares whether that matches anything in the operational result sets sent from the authentic P in the very beginning.
- When an eavesdropper, E, tries to trick V into believing that E = P since E has no idea of which operand set (encrypted) is corresponding to which operational result set (encrypted), she can only respond the challenge by guessing one from the set pairs earlier sent by P. As the examining rounds

| Table 2 |
|----------------|
| **Dummy Test Tasks Simulations** |

| Round | 1 | 2 | ... | t |
|-------|---|---|-----|---|
| **Operand Set (in plaintext)** |
| Operand 1 | 5 | 42022 | ... | 77 |
| Operator (* or +) | * | + | ... | + |
| **Operational Result Set (in plaintext)** |
| Operand 2 | 3 | 998 | ... | 32 |
| Operation Result | 15 | 43020 | ... | 109 |
| **Dummy Test Task Set** |
| 1 | 2 | ... | t |
| **Operand Set (in ciphertext)** |
| Operand 1 (encrypted) | 70994920...7074870821 | 62090366...7954462948 | ... | 62312036...7582581582 |
| Operand 2 (encrypted) | 70994920...7774870819 | 62090366...7954421924 | ... | 62312036...4782581537 |
| **Operation Result Set (in ciphertext)** |
| Operation Result (encrypted) | 50402786...7687472399 | 12418073...08884872 | ... | 12462407...5165163119 |
grow, V will find the hit rate abnormally decreased
as the number of dummy test rounds increases.
From the scene, V knows there must be someone
in the middle who is trying a hack engineering;
thus, V can discard the rest of the verifying process
and invalidate any request by the session. The
soundness can be achieved since the cheating P
can always be rejected by the honest V.
- After t rounds of operand sets have been traversed
with a high hit rate, V would have a high confidence
level that $1 \leftarrow \text{Verify}(x, \pi)$ holds true to accept
as P; otherwise, V invalidates the session. The
completeness can thereby be achieved.

The evaluation result is summarized in Table 2. With
several rounds of dummy test execution generated
by cloud service consumers, we can observe that the
scope for a cloud service provider is merely execut-
ing massive instructions on the operation result in
ciphertext with pre-defined FHE evaluate algorithm
of ZKP-Cave. Notice that the data-in-process always
stays in a ciphered manner during the protocol. ZKP-
Cave prevents malicious cloud service providers from
spying on sensitive content. The scheme promises ze-
ro-knowledge commitment against its secret knowl-
dge processors and any other eavesdroppers.
The effectiveness of our scheme maintains a simu-
lator to produce a scrambled and indistinguishable
output to avoid linkage with any meaning during the
interaction between the prover and the verifier. In
the case of a malicious adversary/service provider
armed with quantum computing power, since there
is no key-exchange during ZKP-Cave protocol, there
is no PKI computational vulnerability that can be
breached by quantum computation.

5. Discussions
When dealing with issues such as secure multi-party
computation, the approach of simulation is most com-
monly used. The critical key of simulation is to build
an indistinguishable carrier to an adversary that in-
teracts with all parties involved. In our scheme, we
see the nature of quantum computing, i.e., the result
is a measuring state with higher probability; in other
words, there is always a chance of lower probability
to get wrong answers. Our scheme promises data pri-



6. Conclusions
In this paper, after introducing quantum comput-
ing, quantum cryptography, and discussing various
post-quantum cryptography schemes and limits, we
propose an original scheme to include a zero-knowl-
dge system and FHE symmetric encryption. The
proposed method is an instance to combine the au-
thentication of ZKP and the confidentiality of FHE
in a communicating system to preserve data privacy
in the scenario of immature quantum computation.
Quantum computers will eventually be helpful to
solve innovation problems as well as giving rise to
various trusted computing issues. Convergence of
technology is one direction for achieving this, which
deserves to long, hard contemplation among academ-
ics.

We look forward to motivating more possibilities in
the quantum resistance field and continual improve-
ments for cloud computing security. Coupling differ-
ent measures to a certain extent of secure diversity
are our suggested way to sustain safety under uncer-
tain quantum computing attacks.

Acknowledgement
We sincerely thank National Taiwan University-IBM
Q Hub, who sponsored the state-of-the-art quantum
computers and application interfaces to contribute
this research as a great testing platform for simulat-
ing the factorization with the Shor algorithm.
References

1. Al-Janabi, S. T. F. Unconditionally Secure Authentication in Quantum Key Distribution. i-Manager’s Journal on Software Engineering, 2007, 1(3), 30-42. https://doi.org/10.26634/jse.1.3.720

2. Beauregard, S. Circuit for Shor’s Algorithm Using 2n+3 Qubits. Quantum Information & Computation, 2003, 3(2), 175-185. https://doi.org/10.26421/QIC3.2-8

3. Bennett, C. H., Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing. In Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, 1984, 175-179.

4. Bennett, C. H., Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing. Theoretical Computer Science. Theoretical Aspects of Quantum Cryptography- Celebrating 30 Years of BB84, 2014, December 4, 560, Part 1: 7-11. https://doi.org/10.1016/j.tcs.2014.05.025

5. Bonnoron, G. A Journey Towards Practical Fully Homomorphic Encryption. Cryptography and Security [cs.CR]. Ecole Nationale Supérieure Mines-Télécom Atlantique, English, 2018, ffNNT: 2018IMTA0073ff. ftel-02011668

6. Büscher, N., Demmler, D., Karvelas, N. P., Katzenbeisser, S., Krämer, S., Krämer, J., Rathee, D., Schneider, T., Struck, P. Secure Two-Party Computation in a Quantum World. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (Eds.) Applied Cryptography and Network Security. ACNS 2020. Lecture Notes in Computer Science, 2020, 12146, 461-480. Springer, Cham. https://doi.org/10.1007/978-3-319-76687-4_9

7. Chao, F., Yang, X. Fast Key Generation for Gentry-Style Homomorphic Encryption. The Journal of China Universities of Posts and Telecommunications, 2014, 21(6), 37-44. https://doi.org/10.1016/S1005-8885(14)60343-5

8. Diamanti, E., Lo, H.-K, Qi, B., Yuan, Z. Practical Challenges in Quantum Key Distribution. Quantum Information, 2016, 2, 16025. https://doi.org/10.1038/nqi.2016.25

9. Hariss, K., Noura, H., Samhat, A. E. An Efficient Fully Homomorphic Symmetric Encryption Algorithm. Multimed Tools and Applications, 2020, 79, 12139-12164. https://doi.org/10.1007/s11042-019-08511-2

10. Hariss, K., Noura, H., Samhat, A. E., Chamoun, M. Design and Realization of a Fully Homomorphic Encryption Algorithm for Cloud Applications, 2018, 127-139. Springer, Cham. https://doi.org/10.1007/978-3-319-76687-4_9

11. Hartnett, K. Getting Started with Qiskit. Retrieved from https://www.quantamagazine.org/does-nevens-law-describe-quantum-computings-rise-20190618/#:~:text=Neven's%20law%20states%20that%20quantum supremacy%20is%20around%20the%20corner. Accessed on 2019 July 18

12. Horace, P. Yuen. Unconditional Security. In Quantum Key Distribution. Quantum Physics, 2012, arxiv.org/abs/1205.5065v2

13. Huang, D., Huang, P., Lin, D., Zeng, G. Long-Distance Continuous-Variable Quantum Key Distribution by Controlling Excess Noise. Scientific Reports, 2016, 6, 19201. https://doi.org/10.1038/srep19201

14. IBM. Get Started with IBM Quantum Experience. Retrieved from https://quantum-computing.ibm.com/docs. Accessed on 2019 May 8.

15. IBM. Quantum Computing at IBM. Retrieved from https://www.ibm.com/quantum-computing/learn/what-is-ibm-q. Accessed on 2019, March 2.

16. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A. K., Thomé, E., Bos, J. W., Gaudry, P., Kruppa, A., Montgomery, P. L., Osvik, D. A., Riele, H. T., Timofeev, A., Zimmermann, P. Factorization of a 768-Bit RSA Modulus. In: Rabin, T. (Eds.) Advances in Cryptology: Lecture Notes in Computer Science, 2010, 6223, 333-350. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14623-7_18

17. Levitan, B. M. Hilbert Space. In Hazewinkel, M. (Eds.) Encyclopedia of Mathematics, Springer Science+Business Media B. V / Kluwer Academic Publishers, 1994, XL, 5402. ISBN 978-1-55608-010-4

18. Lo, H., Chau, H., Ardehali, M. Efficient Quantum Key Distribution Scheme and a Proof of Its Unconditional Security. Journal of Cryptology, 2005, 18, 133-165, https://doi.org/10.1007/s00145-004-0142-y

19. Mailoux, L. O., Grimaila, M. R., Hodson, D. D., McLaughlin, C. V., Baumgartner, G. B. Quantum Key Distribution: Bloon or Bust? CSIAC, Retrieved from https://www.csiac.org/journal-article/quantum-key-distribution-bloon-or-bust. Accessed on 2018 July 21.

20. Martins, P., Sousa, L., Mariano, A. A Survey on Fully Homomorphic Encryption: An Engineering Perspective. ACM Computing Surveys, 2017, 50(6), Article 83, 33. https://doi.org/10.1145/3124441
21. MIT Technology Review. Retrieved from https://www.technologyreview.com/2019/09/18/132956/ibms-new-53-qubit-quantum-computer-is-the-most-powerful-machine-you-can-use. Accessed on 2019 September 18.

22. NIST. Post-Quantum Cryptography. Retrieved from http://csrc.nist.gov/groups/ST/post-quantum-crypto. Accessed on 2018, March 7.

23. Ogura, N., Yamamoto, G., Kobayashi, T., Uchiyama, S. An Improvement of Key Generation Algorithm for Gentry’s Homomorphic Encryption Scheme. In: Echizen, I., Kunihiro, N., Sasaki, R. (Eds.) Advances in Information and Computer Security. IWSEC 2010, Lecture Notes in Computer Science, 2010, 6434. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16825-3_6

24. Qiskit Development Team. Getting Started with Qiskit. Retrieved from https://qiskit.org/documentation. Accessed on 2019, August 21.

25. Quisquater, J.-J., Guillou, L. C., Berson, T. A. How to Explain Zero-Knowledge Protocols to Your Children. Advances in Cryptology - CRYPTO ’89: Proceedings. Lecture Notes in Computer Science, 1990, 435, 628-631. https://doi.org/10.1007/0-387-34805-0_60

26. Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 1997, 26(5), 1484-1509. https://doi.org/10.1137/S0036144598347011

27. Wallden, P., Kashefi, E. Cyber Security in the Quantum Era. Communications of the ACM, 2019, 62, 120-120. https://doi.org/10.1145/3241037

28. Yoshida, H. Moore’s Law Is Replaced by Neven’s Law for Quantum Computing. Retrieved from https://community.hitachivantara.com/s/article/moores-law-is-replaced-by-nevens-law-for-quantum-computing. Accessed on 2019 June 25.

29. Zhang, Y., Liu R., Lin D. Improved Key Generation Algorithm for Gentry’s Fully Homomorphic Encryption Scheme. In: Kim H., Kim, D. C. (Eds.) Information Security and Cryptology - ICISC 2017. ICISC 2017. Lecture Notes in Computer Science, 2018, 10779, 93-111. Springer, Cham. https://doi.org/10.1007/978-3-319-78556-1_6

30. Zoufal, C., Lucchi, A., Woerner, S. Quantum Generative Adversarial Networks for Learning and Loading Random Distributions. Npj Quantum Information, 2019, 5(1), 103. https://doi.org/10.1038/s41534-019-0223-2

31. Gentry, C. Fully Homomorphic Encryption Using Ideal Lattices. In Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC ’09, New York, NY, USA, ACM, 2009, 169-178. https://doi.org/10.1145/1536414.1536440

This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 (CC BY 4.0) License (http://creativecommons.org/licenses/by/4.0/).