Quantum Computers and Their Effects on Cryptography

Leonardo Mouta Pereira Pinheiro*, Ivan Guilhon Mitoso Rocha†
Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes, 50, São José dos Campos, Brazil
*leonardomppinheiro@gmail.com
†ivan.rocha@gp.ita.br

***Abstract*—Modern cryptography systems are generally based on the hypothesis of provable security, meaning that breaking the system is equivalent to solving a problem which is considered hard. Two such examples are the RSA and ElGamal cryptosystems, which are widespread in commercial applications and whose security is based, respectively, on the problems of integer factorization and discrete logarithm computing. We present the mathematical formalization of these two cryptography schemes and show how quantum computers are able to solve both these problems in polynomial time by means of Shor's algorithm, thus posing a serious threat to current secrecy standards. Fundamental concepts of quantum superposition, quantum Fourier transforms, and quantum phase estimations are discussed. As a countermeasure to quantum computers, however, it is possibility to implement quantum-based key exchange protocols, which can withstand attacks by both classical and quantum computing, thus providing an avenue for the reestablishment of secure communications.**

I. Introduction

This paper is intended as a final thesis for a minor degree in physics engineering. It aims to explore some introductory properties of quantum computing by following a thread of how this new technology is expected to change the field of cryptography.

In a general sense, the evolution of classical cryptography can be described as a gradual growth from the applications of cryptography through ancient, medieval and early-modern times, where it was based more on intuition than on mathematics [1], followed by the beginning of a mathematical formalization of cryptography ushered in by Shannon [2], down to current modern cryptography, which is fundamental for our Internet-based world [3].

Although the current cryptography standards have given the modern world a very high sense of security, this *status quo* risks being undermined by the rise of quantum computing. The advent of quantum computing in the 1980s has created a new way for us to think about machines, for contrary to classical computers, which run on deterministic bits of ones and zeroes, quantum computers are able to perform calculations on the superpositioned quantum states of "qubits". By manipulating all these superpositions in a parallel fashion, quantum computers are able to efficiently solve problems that remained unsolved by classical approaches. Since the security of most modern cryptosystems rests upon these problems, researchers soon found quantum algorithms, such as Shor's algorithm, that have the potential to render our current cryptography obsolete [4].

While such a scenario is still only theoretically possible, for real quantum computers suffer from implementation setbacks that are not easy to solve, it would be wise to prepare for the day when quantum computers are widespread. In that sense, some have started researching the field of "post-quantum cryptography" in order to create security protocols and cryptosystems that could withstand attacks from quantum computers, thus restoring the desired secrecy levels which are expected in day-to-day communication [5].

This work intends to explore all these points in order to provide an overall view of this shift from classical paradigms to post-quantum systems. Concerning its structure, it is as follows: section II explores the current paradigms of modern cryptography, focusing specifically on the RSA and ElGamal cryptosystems; section III provides an overview of the functioning and formalism of quantum computers, such as qubits, quantum state manipulation, quantum circuits, and so on. It also details Shor's algorithm, which is the fundamental object of quantum cryptanalysis, as well as the algorithm's building blocks, which are the quantum Fourier transform and quantum phase estimation circuits; section IV explains the general principles of post-quantum cryptography as well as one possible post-quantum alternative for secrecy, quantum key distribution, which is exemplified by the BB84 protocol; finally, section V provides a brief conclusion for this work.

II. Classical Public-Key Cryptography Schemes

A. Overview of Classical Cryptosystems

Although cryptography has existed for a fairly long time in human history, it is only recently that mathematicians have started to explore this field within the constraints of mathematical rigor. Modern cryptography rests on the notion of cryptosystem [3], which is defined as a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

1) $\mathcal{P}$ is the set of all possible plaintexts;
2) $\mathcal{C}$ is the set of all possible ciphertexts;
3) $\mathcal{K}$ is the keyspace, the set of all possible keys;
4) $\mathcal{E}$ is the set of encryption functions $e_k : \mathcal{P} \times \mathcal{C}$ and $\mathcal{D}$ is the set of all decryption functions $d_k : \mathcal{C} \times \mathcal{P}$;

5) For each key $k \in \mathcal{K}$, there exists a function $e_k \in \mathcal{E}$ and a function $d_k \in \mathcal{D}$ such that $d_k(e_k(x)) = x$ for every $x \in \mathcal{P}$.

It is worth mentioning that the conversion from human-readable plaintext and ciphertext into a form that is mathematically workable involves some sort of encoding mechanism, such as ASCII or UTF-8, for example.

The effectiveness of a cryptosystem is not stated in the preceding definition. A first attempt at defining which criteria make a cryptosystem safe was made by Auguste Kerckhoffs in the 19th century [1]. The six points originally listed are known as Kerckhoffs's principle, but only the first two are of substantial importance to this work [1]: 1) The system should be, if not theoretically unbreakable, unbreakable in practice; 2) compromise of the system should not inconvenience the correspondents. Starting with the second point, what it means is that the security of the system should not assume the secrecy of how to construct e_k and d_k from key k, resting instead with the inability of eavesdropping parties to access the key itself.

By having the security of the system residing with the key, we create the additional problem of how to secretly transmit said key between two parties. One possibility is assume the existence of a previously secured channel between the parties, such as in Fig. 1. In this case, Alice generates a key and sends it to Bob via this secure channel, which is then used to encrypt communications through the unsecured channel. This hypothesis, however, is generally unrealistic, but for cryptosystems where e_k is equal to d_k (or at least simply derivable from one another), the so-called "symmetric key" cryptosystems, there is no viable alternative to it.

On the other hand, one could pursue the alternative "public key" cryptosystems, such as shown in Fig. 2, where the fundamental assumption is that e_k and d_k are not easily derived from one another. With public key scheme, one does away with the need for a previously secure channel. This can be achieved in the following manner: Alice generates a key that consists of a public part and a private part. The public key is sent in the open and is used to encrypt messages, with the private key being used to decrypt the message.

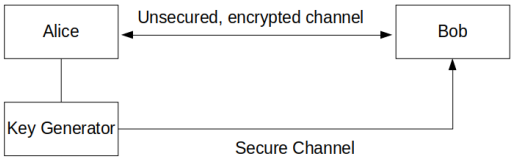


Fig. 1: Basic architecture of encrypted communications (using a symmetric key cryptosystem).

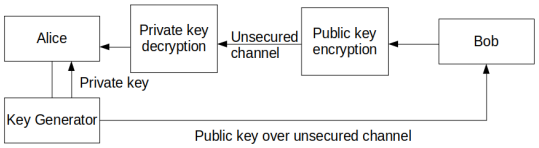


Fig. 2: Basic architecture of encrypted communications (using a public key cryptosystem).

Returning to the first of Kerckhoffs's principle, we must specify what we mean by "unbreakable". There are many ways to break a cipher: finding the key from the messages, using an alternative algorithm that allows decryption without knowledge of the key, alter a sent message, and so on. All these ideas hinge on finding a way around the scheme proposed by the cryptosystem. Shannon broadly defines three types of security [2] :

- Unconditional security, where a cryptosystem cannot be broken even with infinite computational power;
- Provable security, where breaking a cryptosystem is proven to be equivalent to solving a well-studied problem that is considered difficult;
- Computational security, where breaking the cryptosystem requires at least N operations, where N is a very large number.

The type of security depends on the type of "break" trying to be achieved. Furthermore, it also depends on the type of attack being made, such as ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, and so on.

Over the years, the sheer practicality of public-key systems over symmetric-key ones has made them the preferred industry in p2p communications. In the following section, we shall explore two notable examples of public-key cryptosystems, the RSA and the ElGamal.

B. RSA Cryptosystem

Before describing some tools of the RSA cryptosystem, some tools of modular arithmetic are required. The notation developed on this part shall be kept throughout the rest of this work.

Definition of modular arithmetic: for some integer $n > 0$, we define the set $\mathbb{Z}_n = \{0, 1, .., n-1\}$. Considering $a, b \in \mathbb{Z}_n$, one can define two operations $\oplus$, $\otimes$ such as (1), where $+$ and $\times$ are, respectively, the usual addition and multiplication. The algebraic structure $(\mathbb{Z}_n, \oplus, \otimes)$ thus satisfies the axioms of a commutative ring with unity[1] and is the basis for constructing modular arithmetic [6].

$$\begin{cases} a \oplus b = (a + b) \mod n \\ a \otimes b = (a \times b) \mod n \end{cases} \tag{1}$$

In (1), it is important to qualify the notation $\mod$, which will be used frequently. First we specify Euclidean division

[1]Closure, associativity, identity, commutativity and inverses for addition; closure, associativity, identity, and commutativity for multiplication; and left and right distributivity of multiplication over addition

for two integers a and b, with $b \neq 0$, as $a = b \times q + d$, where q is the quotient and d is the remainder, with the added requirement that $0 \leq d < |b|$. The notation $a \mod b$ means taking the remainder of the division of a by b: $a \mod b = d = a - b \times q$. On the other hand, considering a third integer c, the congruence notation $a \equiv c \pmod{b}$ means there exists some integer k such that $(a - c) = b \times k$.

Modular multiplicative inverses: taking the ring $(\mathbb{Z}_n, \oplus, \otimes)$ and an element $a \in \mathbb{Z}_n$, this element has an inverse $a^{-1} \in \mathbb{Z}_n$ such that $a \otimes a^{-1} = a^{-1} \otimes a = 1$ if and only if $\gcd(a, n) = 1$, where gcd means the "greatest common divisor" [6]. We shall notate the set $\mathbb{Z}_n^*$ as the set of all integers $0 \leq a < n$ such that $\gcd(a, n) = 1$. It follows that in $(\mathbb{Z}_n^*, \otimes)$, every element has a multiplicative inverse. Furthermore, for some prime number p, $(\mathbb{Z}_p, \oplus, \otimes)$ is promoted to field, since all non-zero elements now have a multiplicative inverse [6].

Exponentiation: the ring $(\mathbb{Z}_n, \oplus, \otimes)$ allows us to define an exponentiation notation such that $a^k = a \otimes a \otimes ... \otimes a$ (k terms), $a^0 = 1$, and $a^{-k} = (a^{-1})^k$ (if such inverse exists), for some integer k.

Extended Euclidean algorithm: this algorithm takes two inputs a and b and returns integers r, s and t such as in (2), running on time complexity $O(\log(\min(a, b)))$ on the worst case scenario [6].

$$\begin{cases} \gcd(a, b) = r \\ s \times a + t \times b = r \end{cases} \tag{2}$$

Finding multiplicative inverses: considering integer $a \in \mathbb{Z}_n^*$, running the extended Euclidean algorithm on a and n yields, as per (2), $s \times a + t \times n = \gcd(a, n) = 1$. Rearranging, we get $s \times a - 1 = -t \times n \rightarrow s \times a \equiv 1 \pmod{n}$. We can take $a^{-1} = s \mod n$ and thus find the multiplicative inverse of a in a time-efficient manner.

Euler's totient function: Euler's totient function $\phi(n)$ is defined for some integer $n > 0$ as the number of integers k such that $0 \leq k < n$ and $\gcd(k, n) = 1$. Theorem: considering the fundamental theorem of arithmetic, we can write n as a unique product of primes $n = \prod_i p_i^{e_i}$, where each p_i is a distinct prime and e_i a positive integer. With that decomposition, it can be proven that $\phi(n) = \prod_i (p_i^{e_i} - p_i^{e_i - 1})$ [6].

Euler's theorem: consider positive integers a and n such that $a \in \mathbb{Z}_n^*$. Euler's theorem postulates that $a^{\phi(n)} \equiv 1 \pmod{n}$ [6].

Given the previous definitions, it is now possible to describe the basics of the RSA cryptosystem.

Suppose two parties, Alice and Bob, want to establish a secure, one-way communication from Bob to Alice in a hostile environment, such as shown in Fig. 2. An eavesdropper Eve can intercept any sort of communication transmitted between the parties over the unsecured channels, in ciphertext-only attack fashion. Eve can also access any public information available to the parties. The RSA cryptosystem aims to provide provable security based on the problem of integer factorization, for which there is no known polynomial-time classic algorithm [3].

The procedure for RSA can be divided in two parts: key generation and message exchange. These are described as follows [3]:

Key generation:

1) Alice generates a pair of two large random primes, called p and q. Alice then calculates $N = p \times q$;
2) Alice calculates $\phi(N) = (p - 1)(q - 1)$;
3) Alice picks a number a (random or not), such that $\gcd(a, \phi(N)) = 1$;
4) Alice calculates b such that $a \times b \equiv 1 \pmod{\phi(N)}$ using the extended Euclidean algorithm;
5) Alice sends the pair (N, a) over the network and keeps the values (p, q, b) private.

Message exchange:

6) Alice and Bob both know N, so they can build the ring $(\mathbb{Z}_N, \oplus, \otimes)$. All further operations in the exchange now refer to operations done within the ring's algebra;
7) Bob encodes his message into a number $x \in \mathbb{Z}_N$ representing the plaintext, using some previously agreed upon encoding scheme;
8) Bob builds a ciphertext $y \in \mathbb{Z}_N$ by taking $y = x^a$;
9) Bob sends y over the network;
10) Alice receives y and calculates $x' = y^b$;
11) Alice decodes x' using the same encoding scheme.

Proof of operation: in order to prove that the system works, we need to show that $x' = x$. The very construction of the system leads us to (3) by expanding the ring's algebra notation and applying the axioms of congruence algebra [6]. On the other hand, the construction of the modular inverse b leads to (4), which can be substituted into (3), resulting in (5). Euler's theorem[2], reduces (5) to (6). Furthermore, since $x \in \mathbb{Z}_N$ and the modulo operation in deriving x' also ensures $x' \in \mathbb{Z}_N$, the congruence becomes an equality such as in (6), thus proving that the cryptosystem successfully transmits the message so that $x' = x$.

$$\begin{cases} y \equiv x^a \pmod{N} \\ x' \equiv y^b \pmod{N} \end{cases} \rightarrow x' \equiv x^{a \times b} \pmod{N} \tag{3}$$

$$a \times b \equiv 1 \pmod{\phi(N)} \rightarrow \exists k \in \mathbb{Z} | a \times b = 1 + k\phi(N) \tag{4}$$

$$x' \equiv (x^{\phi(N)})^k x \pmod{N} \tag{5}$$

$$x' \equiv x \pmod{N} \rightarrow x' = x \blacksquare \tag{6}$$

Proof of security: the eavesdropper Eve has access to the pair (N, a), since this is public info. As per the second Kerckhoffs principle, she also knows the functioning of the system.

[2]The rare case where $\gcd(x, N) \neq 1$ is covered by combining Fermat's little theorem and the Chinese remainder theorem, resulting in the exact same result $x^{\phi(N)} \equiv 1 \pmod{N}$ for the case when N is the product of two primes. This proof is shown in [3].

Suppose Eve executes a ciphertext-only attack, capturing y. In order to obtain $x' = y^b$, she needs to find b, which is the missing link. Once she knows $\phi(N)$, b can be calculated quickly via the extended Euclidean algorithm. In order to find $\phi(N)$, Eve has two choices:

1) She can count k from 1 to N and efficiently check $\gcd(k, N)$. This guarantees computational security, for this method takes at least $O(N)$ steps, and N was constructed to be large;
2) She can use the prime decomposition of N to calculate $\phi(N)$. This ensures the system has provable security, since efficient integer factorization of semi-primes such as N is an open problem in mathematics [7]. At the time of writing (2022), the record for integer factorization rests with [8], who reported factoring RSA-240 and RSA[3]-250.

These facts prove the security of the system against ciphertext-only attacks for the ideal case. Other forms of exploits will be described later on.

Formalization: in order to complete our description of the RSA cryptosystem, a formal definition based on the scheme shown in II-A is provided:

The RSA Cryptosystem: Take an integer $N = p \times q$, where p and q are randomly generated primes, which are chosen to be large so that factoring N is a hard problem. Define the modular arithmetic ring $(\mathbb{Z}_N, \oplus, \otimes)$. Also define:

- $\mathcal{P} = \mathbb{Z}_N$
- $\mathcal{C} = \mathbb{Z}_N$
- $\mathcal{K} = \{(N, p, q, a, b) | a \times b \equiv 1 \pmod{\phi(N)}\}$
- Given a key $k \in \mathcal{K}$:
  - $e_k(x) = x^a, \ \forall x \in \mathcal{P}$
  - $d_k(y) = y^b, \ \forall y \in \mathcal{C}$

The tuple (N, a) constitutes the public key and the tuple (p, q, b) constitutes the private key.

Although we have proven that the RSA cryptosystem possesses provable security, there are some mistakes in implementation that render it unsafe, for example [3]:

- In the case where a and x are small, extraction of the original message could possibly be obtained by performing a simple a-root extraction. A padding scheme should be used to avoid this sort of problem;
- The attacker can, theoretically, try to encrypt different plaintexts using the public key until they find one whose encryption matches the ciphertext y, meaning RSA is not semantically secure without randomized padding [9];
- Some prime-generating libraries are liable to certain types of exploits which make determining p and q easier than factoring N. Care should be exercised when creating these random numbers, which is a whole field unto itself;

[3]The numbers when describing RSA refer to a notation proposed by the "RSA Factoring Challenge" and can either mean the number of digits of N, as is the case in the records mentioned, or the number of bits occupied by N, such as in RSA-2048.

There are other more effective attacks against RSA, especially if we also consider active adversaries. This means that the system, although theoretically robust, is not trivially implemented in real-world scenarios. It is recommended that standards such as PKCS [10] be rigorously followed when implementing RSA cryptosystems.

C. ElGamal Cryptosystem

While RSA was based on the problem of integer factorization, the ElGamal cryptosystem is based on the problem of computing discrete logarithms, which requires the definition of some additional concepts. As in the case with RSA, this notation will be kept constant throught this work.

Cyclic groups: considering a multiplicative group $G = (\mathbb{G}, \cdot)$, this group is a multiplicative cyclic group if and only if there is an element $g \in \mathbb{G}$ such that $\mathbb{G} = \{g^k | k \in \mathbb{Z}\}$, with exponentiation being defined as $g^k = g \cdot g \cdot ... \cdot g$ (k terms), $g^0 = 1_G$, and $g^{-k} = (g^{-1})^k$ [6]. The element g is called a generator of the group. It is worth mentioning the group $(\mathbb{Z}_n^*, \otimes)$ is multiplicative cyclic if and only if $n = 2, 4, p^k$, or $2p^k$, where p is an odd prime number and k is some positive integer [6] [11].

Order of elements: the order $ord(a)$ of an element a in multiplicative group G is the smallest positive integer n such that $a^n = 1_G$. If the group is cyclic and a is a generator of the group, its order is equal to the number of elements in set $\mathbb{G}$ [6]. Thus, $ord(G)$ and $ord(\mathbb{G}$ are also used to describe the number of elements in the set/group. For group $(\mathbb{Z}_p^*, \otimes)$ we have $ord(\mathbb{Z}_p^*) = \phi(p) = p - 1$.

Generating the group from the order: for some generator g of cyclic group $(\mathbb{G}, \cdot)$ such that $ord(g) = n$, we have that $\mathbb{G} = \{g^0, g^1, g^2, ..., g^{n-1}\}$ [6].

Finding generators: although there is no analytical form to find generators of a group, [12] describes an efficient algorithm for finding generators of multiplicative cyclic group $(\mathbb{G}, \cdot)$ by trial and error.

Discrete logarithm definition: let $G = (\mathbb{G}, \cdot)$ be a multiplicative cyclic group with some generator α such that $ord(\alpha) = n$. For some element $\beta \in G$, the discrete logarithm problem consists of finding x, with $0 \leq x < n$, such that $\alpha^x = \beta$. This is notated as $x = \log_\alpha \beta$, with group G kept implicit [12].

Given the previous definitions, it is now possible to describe the basics of the ElGamal cryptosystem.

The situation is kept the same: two parties, Alice and Bob, want to establish a secure, one-way communication from Bob to Alice in a hostile environment, such as shown in Fig. 2. The same eavesdropper Eve can intercept any sort of communication transmitted between the parties over the unsecured channels, in ciphertext-only attack fashion. Eve can also access any public information available to the parties. The ElGamal cryptosystem aims to provide provable security based on the problem of computing discrete logarithms, for which there is no known time-efficient classic algorithm [3].

The procedure for ElGamal can be divided in two parts: key generation and message exchange. These are described as follows [3]:

Key generation:

1) Alice generates a large prime (random or not) p, therefore obtaining the cyclic multiplicative group $(\mathbb{Z}_p^*, \otimes)$ (which will be used subsequent algebraic operations in the cryptosystem);
2) Alice chooses a generator α of $(\mathbb{Z}_p^*, \otimes)$;
3) Alice randomly chooses an element $r \in \mathbb{Z}_p^*$ and calculates $\beta = \alpha^r$;
4) Alice sends (p, α, β) over the network and keeps r private.

Message exchange:

5) Bob receives (p, α, β), thus constructing cyclic group $(\mathbb{Z}_p^*, \otimes)$;
6) Bob encodes his message into a number $x \in \mathbb{Z}_p^*$ representing the plaintext, using some previously agreed upon encoding scheme;
7) Bob chooses a random integer $m \in \mathbb{Z}_p^*$;
8) Bob computes $y_1 = \alpha^m$;
9) Bob computes $y_2 = x \otimes \beta^m$;
10) Bob sends (y_1, y_2) over the network;
11) Alice receives (y_1, y_2) and calculates $x' = y_2 \otimes (y_1^r)^{-1}$;
12) Alice decodes x' using the same encoding scheme.

Proof of operation: in order to prove that the system works, we need to show that $x' = x$. In order to achieve this end, we can start by expanding the notation of the modulo operations and explicitly showing the components of the ciphertexts, as in (7). The commutativity of modular exponentiation [6] yields (8), which can be combined with the fact that $\beta \equiv \alpha^r \pmod p$, resulting in (9). Here, it is evident that $\beta^m \times (\beta^m)^{-1} \equiv 1 \pmod p$, meaning (9) simplifies into (10). Moreover, since the modulo operation embedded in $\otimes$ ensures that both x and x' are in $\mathbb{Z}_p^*$, the congruence in (10) is actually an equality, thus proving that the cryptosystem successfully transmits the message so that $x' = x$.

$$x' = y_2 \otimes (y_1^r)^{-1} \rightarrow x' \equiv x \times \beta^m \times (\alpha^{m \times r})^{-1} \pmod p \quad (7)$$

$$\alpha^{m \times r} \equiv \alpha^{r \times m} \equiv (\alpha^r)^m \pmod p \quad (8)$$

$$x' \equiv x \times \beta^m \times (\beta^m)^{-1} \pmod p \quad (9)$$

$$x' \equiv x \pmod p \rightarrow x' = x \blacksquare \quad (10)$$

Proof of security: the eavesdropper Eve has access to the tuple (p, α, β), since this is public info. As per the second Kerckhoffs principle, she also knows the functioning of the system. Suppose Eve executes a ciphertext-only attack, capturing (y_1, y_2). In order to obtain $x' = y_2 \otimes (y_1^r)^{-1}$, Eve has two choices, which essentially boil down to the same issue of calculating a discrete logarithm [12]:

1) Knowing $y_1 = \alpha^m$ and $y_2 = x \otimes \beta^m$, Eve can try to find $m = \log_\alpha y_1$ and then substitute $y_2 = x \otimes \beta^{\log_\alpha y_1}$ to try and solve for x. If she executes a trial and error method, and assuming an even distribution on the choice of m, she will have to analyze $O(p)$ elements on average, which is infeasible for large p, thus proving computational security.
2) Eve could try to find $r = \log_\alpha \beta$ and just apply the decryption function directly. If she executes a trial and error method, and assuming an even distribution on the choice of a, she will have to analyze $O(p)$ elements on average, which is infeasible for large p, thus proving computational security.

In both cases, the underlying problem is finding the discrete logarithm of some value, which has no known polynomial-time classical algorithm [12]. The current record for discrete logarithm computation is also held by [8], who computed a discrete logarithm modulo (RSA-240 + 49204). These facts demonstrate the provable security of the system against ciphertext-only attacks for the ideal case. Other forms of exploits will be described later on.

Formalization: in order to complete our description of the ElGamal cryptosystem, a formal definition based on the scheme shown in II-A is provided:

The ElGamal Cryptosystem: Consider a prime number p which is large enough so that the discrete logarithm problem in multiplicative cyclic group $(\mathbb{Z}_p^*, \otimes)$ is hard. Let $\alpha \in \mathbb{Z}_p^*$ be a generator of said group. Define:

- $\mathcal{P} = \mathbb{Z}_p^*$
- $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$
- $\mathcal{K} = \{(p, \alpha, r, \beta) | \alpha^r = \beta\}$
- Given a key $k \in \mathcal{K}$ and a random integer $m \in \mathbb{Z}_p^*$:
  - $e_k(x, m) = (y_1, y_2) | y_1 = \alpha^m, y_2 = x \otimes \beta^m, \forall x \in \mathcal{P}$
  - $d_k(y_1, y_2) = y_2 \otimes (y_1^r)^{-1}, \forall (y_1, y_2) \in \mathcal{C}$

The tuple (p, α, β) constitutes the public key and the value r constitutes the private key.

Although we have proven that the ElGamal cryptosystem possesses provable security, there are some mistakes in implementation that render it unsafe, for example [3]:

- In the case where α is small and p is large, the discrete logarithm could become a "normal" logarithm problem. A padding scheme should be used to avoid this sort of problem;
- The attacker can, theoretically, try to encrypt different plaintexts using the public key until they find one whose encryption matches the ciphertext y, meaning ElGamal is not semantically secure without randomized padding [9];
- If the private key r is too small, it might be discovered by trial and error;

More effective attacks against ElGamal exist, especially if we also consider active adversaries. This means that the system, although theoretically robust, is not trivially implemented in real-world scenarios. It is recommended that standards such

as PKCS [10] be rigorously followed when implementing ElGamal cryptosystems.

III. Quantum Computing and its Impact on Cryptography

A. Fundamentals of Quantum Computing

1) Single Qubit Systems and their Representations: While classical computers operate in terms of "bits", quantum computers are based on the concept "qubits" [13]. The difference between the two reflects a difference between the deterministic and the probabilistic. A classical computer could, for example, read an electric voltage and associate a certain bit value with it: if the voltage is below a certain threshold, it is assigned the bit "0"; if the voltage is higher the same threshold, it gets the bit "1". Since electrical signals (and other physical entities) follow classical laws, the signal can either be 0 or 1, meaning it is binary [5]. On the other hand, consider the spin of some electron, which is described by quantum mechanics. Say we assign the spin up to the state $|0\rangle$ and the spin down to the state $|1\rangle$. This way, any state $|\psi\rangle$ of the electron's spin can be expressed as in (11), where the normalization is required as per Born's rule. The range of all possible values of $|\psi\rangle$ is called its vector space or state space [5].

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \mid \alpha, \beta \in \mathbb{C},\ |\alpha|^2 + |\beta|^2 = 1 \tag{11}$$

We can use the electron's spin (or some other physical entity bound by quantum mechanics) as a means of transmitting information, therefore stepping into the world of quantum computing.

Any quantum signal whose state can be written as a superposition[4] of two binary states is a "quibt" [13]. In our example, the spin of an electron is a qubit. The pair $\{|0\rangle, |1\rangle\}$ represents an orthonormal basis, called the computational basis, which can be used to perform measurements. These measurements allow us to access the information stored in a qubit and transform it into a classical signal. This is generally done by assigning $|0\rangle \to 0$ and $|1\rangle \to 1$.

Some differences in relation to classical computing already start to appear. Consider, for example, the qubit whose state is given by $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. If we measure this signal with respect to the computational basis, we get $Pr[|0\rangle] = Pr[|1\rangle] = \frac{1}{2}$, meaning that the same signal, when converted to a classical bit as per the equivalence previously established, will yield bit 0 50% of the time and bit 1 50% of the time. Therefore, the same signal can result in different values when measured. This property is what makes quantum computing different than classical computing. While a classical bit can be either 0 or 1, a quantum bit exists as a whole range of superpositions, and its measured value is probabilistic.

We previously mentioned the computational basis $\{|0\rangle, |1\rangle\}$, but in reality any pair of states $\{|a_1\rangle, |a_2\rangle\}$ such that $\langle a_i|a_j\rangle = \delta_{ij}$, with δ_{ij} the Kronecker delta, can be used a basis for the qubit's state space. Although we have discussed measurements against the computational basis, in reality measurements can be made against any orthonormal basis [5]. Some special orthonormal state pairs are particularly noteworthy [5]:

- Basis $\{|+\rangle, |-\rangle\}$ (also called the Hadamard basis)
  - $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
  - $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- Basis $\{|+i\rangle, |-i\rangle\}$
  - $|+\mathrm{i}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \mathrm{i}|1\rangle)$
  - $|-\mathrm{i}\rangle = \frac{1}{\sqrt{2}}(|0\rangle - \mathrm{i}|1\rangle)$

There are several ways to represent a single qubit. One of them is the vector representation, which is particularly useful when we wish to represent operators/gates as matrices. In this representation, a state such as the one in (11) written as $|\psi\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$. Henceforth, whenever a vector notation is used, we'll assume it is with respect to the computational basis [5].

Another representation of a single qubit system relates to the concepts of global phase and relative phase. Two qubit states $|\psi\rangle$ and $|\psi'\rangle$ are considered equivalent ($|\psi\rangle \sim |\psi'\rangle$) if there exists a value $\alpha \in [0, 2\pi)$ such that $|\psi\rangle = e^{i\alpha}|\psi'\rangle$. The angle α is called the global phase of the state [5]. By manipulating the global phase of the state, it is possible to see that any qubit state can be written as $|\psi\rangle = \begin{bmatrix}\cos\frac{\theta}{2}\\ e^{\mathrm{i}\varphi}\sin\frac{\theta}{2}\end{bmatrix}$, with $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$. The angle φ is called the relative phase [5]. The angle pair (θ, φ) can be used to represent the qubit using the so-called "Bloch sphere" [14], which is shown in Fig. 3. In the Bloch sphere, we represent the state $|\psi\rangle$ as a unit vector from the origin, with θ being its polar angle and φ its azimuth angle. Another feature of the Bloch sphere is that it maps the special states previosuly described to key points on the sphere's surface [4].

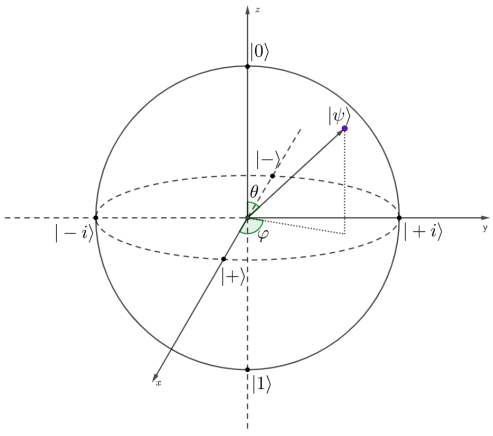


Fig. 3: The Bloch sphere is a useful method for representing single qubit systems.

2) Multiple Qubit Systems: Suppose we now have two qubit channels, A and B, each one containing a state: $|\psi_A\rangle_A = a_0|0\rangle_A + a_1|1\rangle_A$ and $|\psi_B\rangle_B = b_0|0\rangle_B + b_1|1\rangle_B$ respectively.

[4]Some definitions of superposition require that $\alpha\beta \neq 0$, but here the term is used more broadly. A signal which is equal to one of the binary states is still considered a qubit.

The full state $|\psi\rangle_{AB}$ of the system is given in (12)., where $\otimes$ represents the tensor product of the elements [5].

$$\begin{aligned}|\psi\rangle_{AB} = |\psi_A\rangle_A \otimes |\psi_B\rangle_B = a_0b_0|0\rangle_A \otimes |0\rangle_B \\ +a_0b_1|0\rangle_A \otimes |1\rangle_B \\ +a_1b_0|1\rangle_A \otimes |0\rangle_B \\ +a_1b_1|1\rangle_A \otimes |1\rangle_B\end{aligned} \tag{12}$$

We can simplify the notation on the right hand side of (12) by adopting the convention that $|a\rangle_A \otimes |b\rangle_B = |a\rangle_A|b\rangle_B = |ab\rangle_{AB}$, thus converting (12) into (13). The values $\{|00\rangle_{AB}, |01\rangle_{AB}, |10\rangle_{AB}, |11\rangle_{AB}\}$ form a basis for the complete system, also called the computational basis in analogy to the single qubit system. Using the vector notation proposed previously, we have $|\psi\rangle_{AB} = [a_0b_0,\ a_0b_1,\ a_1b_0,\ a_1b_1]^T$.

$$\begin{aligned}|\psi\rangle_{AB} = a_0b_0|00\rangle_{AB} \\ +a_0b_1|01\rangle_{AB} \\ +a_1b_0|10\rangle_{AB} \\ +a_1b_1|11\rangle_{AB}\end{aligned} \tag{13}$$

This notion can be extended. Each qubit is a complex vector space V_i of computational basis $\{|0\rangle_i, |1\rangle_i\}$. For a system of n qubits, the complete vector space is $V = V_{n-1} \otimes ... \otimes V_1 \otimes V_0$, with computational basis $B = \{|0\rangle_{n-1}...|0\rangle_1|0\rangle_0, |0\rangle_{n-1}...|0\rangle_1|1\rangle_0, ..., |1\rangle_{n-1}...|1\rangle_1|1\rangle_0\}$ yielding a total of 2^n elements [5]. We can both drop the subscripts and join the basis states in order to keep the notation for the computational basis more palatable $B = \{|0...00\rangle, |0...01\rangle, ..., |1...11\rangle\}$. It is important, however, to remember two things: 1) the number of qubits involved in the system, which should be clear from the context; and 2) the fact that the rightmost digit of each basis element refers to qubit 0, the second rightmost to qubit 1, and so on[5].

We can compress this notation even further by doing a binary-decimal conversion on the elements of the basis. By saying that qubit 0 is the least significant qubit and that qubit $n-1$ is the most significant, we can convert a basis element $|a_{n-1}a_{n-2}, ..., a_1, a_0\rangle$, with $a_i \in \{0, 1\}$, to $|k\rangle$, with $k = \sum_{i=0}^{n-1} a_i 2^i$. Using this notation, which shall persist throughout this work, the computational basis can be written as $B = \{|0\rangle, |1\rangle, ..., |2^n - 1\rangle\}$

Example: suppose we have a system with $n = 2$ qubits, q_0, q_1. The least significant qubit is q_0 and the most significant qubit is q_1. The computational basis for the system $V = V_1 \otimes V_0$ is $B = \{|00\rangle_{q_1q_0}, |01\rangle_{q_1q_0}, |10\rangle_{q_1q_0}, |11\rangle_{q_1q_0}\}$, which can be written as $B = \{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$.

Concerning the measurements for multiple qubit systems, they work in a similar fashion to that of single qubit systems. Suppose a state $|\psi\rangle = \sum_{i=0}^{2^n-1} a_i|i\rangle$ is measured against the computational basis. The probability of measuring state $|k\rangle$ is $Pr[|k\rangle] = |a_k|^2$. By expanding the notation $|k\rangle = |a_{n-1}...a_1a_0\rangle = |a_{n-1}\rangle_{n-1}...|a_1\rangle_1|a_0\rangle_0$, where $a_i \in \{0, 1\}$, we can see that measuring $|k\rangle$ equals measuring $|a_0\rangle$ on qubit 0, $|a_1\rangle$ on qubit 1, and so on. From here, it is merely a question of applying the same encoding to classical bits seen in section III-A1. Measurements on different basis work in a similar manner. Assuming an orthonormal basis $B' = \{|b_o\rangle, |b_1\rangle, ..., |b_{2^n-1}\rangle\}$ for the n-qubit vector space (where $n \geq 1$), the probability of measuring state $|b_i\rangle$ from state $|\psi\rangle$ is $Pr[|b_i\rangle] = |\langle b_i|\psi\rangle|^2$.

Even though the complex vector space for an n-qubit system was constructed by taking the tensor product of the vector spaces of the individual qubits $V = V_{n-1} \otimes ... \otimes V_1 \otimes V_0$, that does not mean that every state of the system can be written as a tensor product of states in the original qubits. When $\nexists\{|\psi_{n-1}\rangle_{n-1}, ..., |\psi_1\rangle_1, |\psi_0\rangle_0\} \mid |\psi\rangle = |\psi_{n-1}\rangle_{n-1} \otimes ... \otimes |\psi_1\rangle_1 \otimes |\psi_0\rangle_0$, it is said that the system state $|\psi\rangle$ is entangled [5].

Example: Consider a system of 2 qubits. The state $|\psi_1\rangle = [1/2, 1/2, 1/2, 1/2]^T$ can be written as $|\psi\rangle = [1/\sqrt{2}, 1/\sqrt{2}]_1^T \otimes [1/\sqrt{2}, 1/\sqrt{2}]_0^T$, meaning it is not entangled. On the other hand, the state $|\psi_2\rangle = [1/\sqrt{2}, 0, 0, 1/\sqrt{2}]^T$ cannot be written as a tensor product over the qubits, being, therefore, entangled.

The interest in entangled states lies in the fact that measurements on one qubit yield information about some other qubit. Rewriting the state $|\psi_2\rangle$ of the previous example in full state notation, we get $|\psi_2\rangle_{10} = \frac{1}{\sqrt{2}}(|0\rangle_1|0\rangle_0 + |1\rangle_1|1\rangle_0)$. If we perform a measure in the computational basis on qubit 0 and obtain $|0\rangle$, we immediately know that a measure on qubit 1 will yield $|0\rangle$ [5]. Likewise, a measurement on qubit 0 yielding $|1\rangle$ means we have to measure $|1\rangle$ on qubit 1, and so on.

Some entangled states are of special interest: the so-called "Bell states". These are entangled states obtained on 2-qubit systems that represent, at the same time, the simplest and maximum entanglement [4]. The Bell states are listed below [14]:

- $|\Psi^{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Psi^{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

As a general formula for the Bell states, we can write $|\Psi^{ij}\rangle = \frac{1}{\sqrt{2}}(|0j\rangle + (-1)^i|1\delta_{0j}\rangle)$, for $i, j \in \{0, 1\}$. It is also worth noting that the Bell states are orthonormal, meaning they can be used to generate proper state superpositions.

3) Operators and Quantum Ports: Operators will be defined as transformations from the state space of a quantum system to itself [5]. Not all operators imaginable are permissible, for they must satisfy the rules of quantum mechanics. Namely, the operators, once defined in their vector spaces, need to satisfy the requirements of linearity (14), for the principle of superposition to hold, and the preservation of the inner product (15), so that no contradictions arise in terms of measurement [5]. In these equations, U is an operator and $U^\dagger$ means the complex conjugate transpose of U. Operators can be represented as both matrices or bra-ket entities [4].

[5]Although different orders can be used, it is important to keep the same qubit order throughout the development of applications.

$$U\sum_{i=1}^{k} a_i|\psi_i\rangle = \sum_{i=0}^{k} a_i U|\psi_i\rangle \tag{14}$$

$$\langle\phi|U^\dagger U|\psi\rangle = \langle\phi|\psi\rangle \tag{15}$$

Equations (14) and (15) can be satisfied for all states if we have $U^\dagger = U^{-1}$, meaning operators have to be unitary [5].

One very important property of this definition is the no-cloning theorem: $\nexists U|U(|a\rangle|0\rangle) = |a\rangle|a\rangle,\ \forall|a\rangle$ [4]. What this means is that it is not possible to construct a "cloning" operator that copies a state to another state without altering the original. This will be fundamental when discussing the quantum key-exchange protocols. Another consequence of the restriction that operators are unitary is that every quantum operator can be reversed by applying its conjugate transpose.

When an operator is applied to a small number of qubits, it is also called a gate. Some gates are particularly important, for they are used in most quantum computing applications. We can cite, for example, the Pauli gates X (also called a "bit-flip"), Y and Z [5], which act upon 1-qubit systems. Their names are derived from the visual effect they entail on the Bloch sphere: applying an X gate on state $|\psi\rangle$ means rotating this state π radians around the x axis (and the same logic holds for gates Y and Z). Even more, given this geometrical reasoning, it is easy to verify that $\{|0\rangle, |1\rangle\}$ are eigenstates of gate Z, $\{|+\rangle, |-\rangle\}$ are eigenstates of gate X, and $\{|+\mathrm{i}\rangle, |-\mathrm{i}\rangle\}$ are eigenstates of gate Y

- $X = |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- $Y = -|1\rangle\langle 0| + |0\rangle\langle 1| = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
- $Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Another fundamental 1-qubit operator is the Hadamard gate:

- $H = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

The importance of this gate is shown in (16), where it maps $\{|0\rangle, |1\rangle\} \rightarrow \{|+\rangle, |-\rangle\}$. For an element $|i\rangle,\ i \in \{0, 1\}$ of the 1-qubit computational basis, we can write the Hadamard transform as $H|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^i|1\rangle)$.

Although the Hadamard gate was built for 1-qubit systems, it can be generalized for some specific applications, such as in (17), where we have an n-qubit system at state $|0\rangle_{n-1} \otimes ... \otimes |0\rangle_1 \otimes |0\rangle_0 = |0\rangle^{\otimes n}$ (the exponential notation means only a repetition of the tensor product). We can see that such an application takes the initial state, which can be obtained by setting all original qubits to state $|0\rangle$, into a homogeneous superposition of the computational basis's elements for the n-qubit state space [4].

$$\begin{cases} H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle \\ \\ H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle \end{cases} \tag{16}$$

$$\begin{aligned} H^{\otimes n}|0\rangle^{\otimes n} &= (H|0\rangle)^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}}(|0...0\rangle + |0...1\rangle + ... + |1...1\rangle) \end{aligned} \tag{17}$$

As for multiple-qubit gates, the most important one is the $CNOT$ gate which act on 2-qubit systems. Its use, which is illustrated in (18), means that it flips the state of the second qubit if the first qubit is in state $|1\rangle$, but conserves the state if the first qubit is in state $|0\rangle$. For superpositioned states, we can write the state's decomposition over the computational basis and then apply the $CNOT$ gate accordingly using the operator's linearity.

- $CNOT = |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11|$

$$\begin{cases} CNOT|00\rangle = |00\rangle \\ CNOT|01\rangle = |01\rangle \\ CNOT|10\rangle = |11\rangle \\ CNOT|11\rangle = |10\rangle \end{cases} \tag{18}$$

Any operator can be made into a controlled operator, meaning it only acts upon the state if the controller state is $|1\rangle$, doing nothing otherwise [15]. This notion can also be extended to create classically-controlled gates, which only apply an operator if it receives an input 1 from a classical bit.

Gates can be applied in sequence and in parallel to generate circuits, which form the core of quantum computation.

4) Circuits: Gates can be combined into circuits, which form the core of quantum computing. The circuits shown in this work, such as in Fig. 4, were obtained using Qiskit [14], which is a python-based package for quantum computing developed by IBM. The rules for understanding quantum circuits are:

1) Information flows from left to right;
2) Qubit registers, which are represented by single lines, are numbered, and the higher the value, the more signification the qubit;
3) Double lines represent classical bits, It is possible to have either many double lines, each one representing a bit, or a single double line representing a bit string;
4) Measurements, which are always done against the computational basis, take a 1-qubit state to a classical bit in accordance to the encoding in section III-A1. If measuring into a bit string, significance is preserved;
5) Gates are represented by squares. The list of most common gates can be found in Fig. 5.

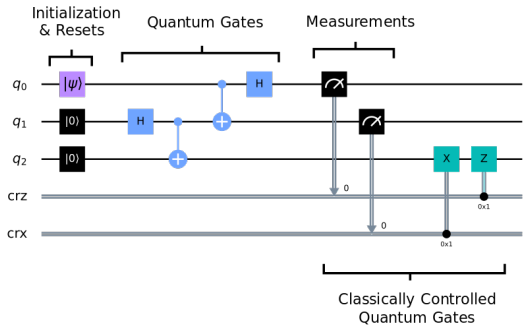


Fig. 4: An example quantum circuit, obtained from Qiskit [14].

Operator	Gate(s)	Matrix
Pauli-X (X)	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Fig. 5: The main gates used in quantum circuits [14].

To further solidify our understanding of quantum systems, some examples are shown here.

Consider the simple, 1-qubit circuit in Fig. 6. In it we have $|\psi_1\rangle = |0\rangle$. After passing the Hadamard gate, we get $|\psi_2\rangle = H|\psi_1\rangle = H|0\rangle = |+\rangle$. Measuring $|\psi_2\rangle = |+\rangle$ against the computational should entail $Pr[|0\rangle] = Pr[|1\rangle] = \frac{1}{2}$, meaning we should get bit 0 and bit 1 with equal probability. Qiskit allows us to run this circuit either on a classical computer[6] or on one of IBM's quantum computers. The results of running this circuit 1024 times are shown in Fig. 7 with these being in agreement with our predictions.

[6]The classical computer simulates the quantum computer by using randomness packages. Evidently, this is less efficient than using a real quantum computer [14].

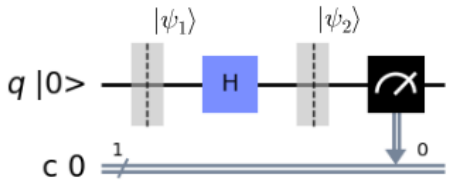


Fig. 6: A simple quantum circuit as an example.

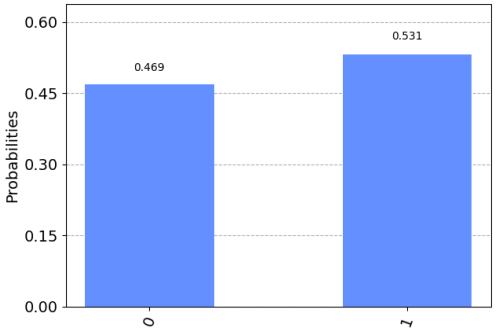


Fig. 7: Measurements obtained by running the circuit in Fig. 6 in Qiskit.

We can use quantum circuits for more interesting applications. One such possibility is generating bell states using the circuit in Fig. 8, where $i, j \in \{0, 1\}$, meaning we can start the qubit registers at any one of the 1-qubit basis elements. This way, we get (19). After applying the Hadamard gate at qubit 1, we arrive at (20). We can use the linear property of the $CNOT$ operator to get to (21). The $CNOT$ gate only acts if the controlling qubit is in state $|1\rangle$, meaning (21) is identical to (22), which matches our previous equation for Bell states in section III-A2, thus proving that the system is successful in its objective.

$$|\psi_1\rangle = |i\rangle_1|j\rangle_0 \tag{19}$$

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2}}(|0\rangle_1 + (-1)^i|1\rangle_1)|j\rangle_0 \\ &= \frac{1}{\sqrt{2}}(|0\rangle_1|j\rangle_0 + (-1)^i|1\rangle_1|j\rangle_0) \end{aligned} \tag{20}$$

$$\begin{aligned} |\psi_3\rangle &= CNOT|\psi_2\rangle \\ &= \frac{1}{\sqrt{2}}(CNOT(|0\rangle_1|j\rangle_0) + (-1)^i CNOT(|1\rangle_1|j\rangle_0)) \end{aligned} \tag{21}$$

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1|j\rangle_0 + (-1)^i|1\rangle_1|\delta_{0j}\rangle_0) = |\Psi^{ij}\rangle \tag{22}$$

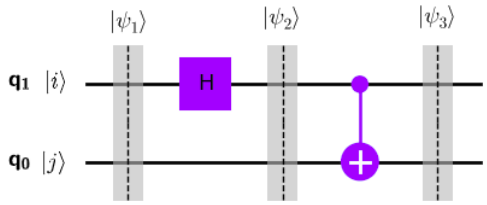


Fig. 8: Quantum circuit for generating Bell states.

Also, we can build a circuit to solve the inverse problem of finding i, j given a Bell state $|\Psi^{ij}\rangle$. This problem is called "Bell measurement". Since both the $CNOT$ and H operators used in Fig. 8 are real and symmetric, they are their own inverses. Therefore, all we need to do is run the circuit in the "other direction", as in Fig. 9 [14].

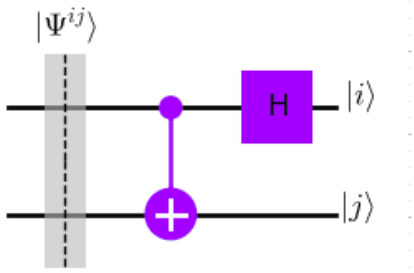


Fig. 9: Quantum circuit for performing Bell measurements.

In the following sections, we shall show some particular circuits that serve as building blocks for Shor's algorithm.

B. Quantum Fourier Transform (QFT)

Before moving on to the quantum Fourier transform (QFT), which is a fundamental part of Shor's algorithm, we should recall that, as shown in III-A2, the state $|\psi\rangle$ of an n-qubit system can be written as in (23), where the compressed decimal notation for the computational basis hides behind it the binary computational basis that exists in each individual qubit [14].

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i|i\rangle \mid a_i \in \mathbb{C}, \sum_{i=0}^{2^n-1} |a_i|^2 = 1 \tag{23}$$

It is also wise to recap the behavior of the classical discrete Fourier transform (DFT): let $\vec{x} = [x_0, x_1, ..., x_{N-1}]^T$ be a vector. The DFT maps vector $\vec{x}$ to vector $\vec{y} = [y_0, y_1, ..., y_{N-1}]$ according to the formula in (24) [14][7]. It is possible to shorthand $e^{\frac{2\pi i}{N}jk} = \omega_N^{jk}$. Computational implementations of the DFT follow an algorithm dubbed the fast Fourier transform (FFT). When $N = 2^n$ for some $n \in \mathbb{Z}$, the FFT runs on time $O(Nn)$ [16].

[7]Other definitions exist for the DFT, but this is the most convenient for our future applications.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i}{N}jk} \tag{24}$$

In a similar manner, let $|X\rangle$ be a state in an n-qubit system. As per (23) and writing $2^n = N$, let $|X\rangle = \sum_{j=0}^{N-1} x_j|j\rangle$. The QFT maps $|X\rangle$ to state $|Y\rangle = \sum_{k=0}^{N-1} y_k|k\rangle$ in the same vector space using equation (25) [14]. It is also worth mentioning that the notation $|\tilde{X}\rangle = QFT|X\rangle$, with $|\tilde{X}\rangle = \sum_{k=0}^{N-1} \tilde{x}_i|k\rangle$ is also used to express quantum Fourier transformations.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \tag{25}$$

We can manipulate (25) so that it can be used in a quantum gate formulation $QFT|X\rangle = |Y\rangle$, which is not yet the case. By analyzing the particular case $|X\rangle = |j\rangle$ in (25), where $|j\rangle$, $0 \leq j < N$ is one of the elements of the computational basis for the n-qubit system, (25) simplifies to $y_k = \frac{1}{\sqrt{N}}\omega_N^{jk}$. By summing this result over $0 \leq k < N$, we arrive at (26), which expresses what happens if we apply QFT to an element of the computational basis.

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk}|k\rangle,\ 0 \leq j < N, j \in \mathbb{Z} \tag{26}$$

Considering the notation in (23) and the linear property of quantum operators, we can deduce (27), which shows one possible way how one might calculate the full QFT for state $|Y\rangle$ from a general state $|X\rangle$. By substituting (26) into (27), we arrive at (28), which allows us to isolate the QFT operator in (29).

$$|Y\rangle = QFT\left(\sum_{j=0}^{N-1} x_j|j\rangle\right) = \sum_{j=0}^{N-1} x_j QFT|j\rangle \tag{27}$$

$$\begin{aligned} |Y\rangle &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \omega_N^{jk} x_j|k\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \omega_N^{jk}|k\rangle\langle j|X\rangle \\ &= \left(\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \omega_N^{jk}|k\rangle\langle j|\right)|X\rangle \end{aligned} \tag{28}$$

$$QFT = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} \omega_N^{jk}|k\rangle\langle j| \tag{29}$$

Returning to (26), we can expand the decimal notation into binary notation, arriving at (30), where it is important to remember: 1) that basis elements are not entangled and thus can be decomposed in the tensor product; and 2) that the most significant qubit is the one with the highest index.

$$
\begin{aligned}
QFT|j\rangle &= \frac{1}{\sqrt{N}} \sum_{\substack{k_{n-1},\ldots,k_0 \\ \in\{0,1\}}} \omega_N^{j(2^{n-1}k_{n-1}+\ldots+2^0 k_0)} |k_{n-1}\rangle \ldots |k_0\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{\substack{k_{n-1},\ldots,k_0 \\ \in\{0,1\}}} \bigotimes_{p=n-1}^{0} \omega_N^{j2^p k_p} |k_p\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{p=n-1}^{0} \sum_{\substack{k_{n-1},\ldots,k_0 \\ \in\{0,1\}}} \omega_N^{j2^p k_p} |k_p\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{p=n-1}^{0} (|0\rangle + \omega_N^{j2^p} |1\rangle)
\end{aligned}
\tag{30}
$$

Expanding (30) and also noting that $|j\rangle = |j_{n-1}\rangle \otimes \ldots \otimes |j_0\rangle$, we arrive at (31), which contains a very important piece of information about the QFT: while the basis element $|j\rangle$ was written in terms of elements $\{|0\rangle, |1\rangle\}$, which belong to the z axis in the Bloch sphere, the qubit components of $|\tilde{j}\rangle$ are in the plane $x - y$ of the Bloch sphere. This process can be understood as encoding information about the binary form of a number into the relative phase of the qubit state vectors (and vice-versa).

$$
\begin{aligned}
QFT(|j_{n-1}\rangle \otimes \ldots |j_1\rangle \otimes |j_0\rangle) = \\
&\frac{1}{\sqrt{2^n}}(|0\rangle + e^{\frac{2\pi \mathrm{i}}{2}\, j}|1\rangle) \otimes \\
&\ldots \\
&\otimes (|0\rangle + e^{\frac{2\pi \mathrm{i}}{2^{n-1}}\, j}|1\rangle) \\
&\otimes (|0\rangle + e^{\frac{2\pi \mathrm{i}}{2^n}\, j}|1\rangle)
\end{aligned}
\tag{31}
$$

We can illustrate this idea with an example using Qiskit: consider a 4-qubit system. We can write the basis element $|7\rangle = |0\rangle|1\rangle|1\rangle|1\rangle$. Representing these individual qubits in separate Bloch spheres, we get the result in shown in Fig. 10a. The QFT of this element is, as per (31), $|\tilde{7}\rangle = \frac{1}{4}(|0\rangle + e^{\mathrm{i}7\pi}|1\rangle)(|0\rangle + e^{\mathrm{i}\frac{7\pi}{2}}|1\rangle)(|0\rangle + e^{\mathrm{i}\frac{7\pi}{4}}|1\rangle)(|0\rangle + e^{\mathrm{i}\frac{7\pi}{8}}|1\rangle)$, which is shown in Fig. 10b. Thus, we went from a binary encoding of number 7 on the z axis to a relative phase encoding of the same number.

In terms of the circuit implementation of the QFT, a detailed description is beyond the scope of this work, but it can be achieved through a combination of Hadamard, rotation, and swap gates [14]. We shall limit ourselves to considering the QFT as a single, black-box gate, such as the one in Fig. 11, which implements the previous example, for $|\psi_1\rangle = |7\rangle$ and $|\psi_2\rangle = |\tilde{7}\rangle$. The quantum implementation QFT over an n-qubit system runs on time $O(n^2)$, thus giving it a significant edge over the classical methods [14].

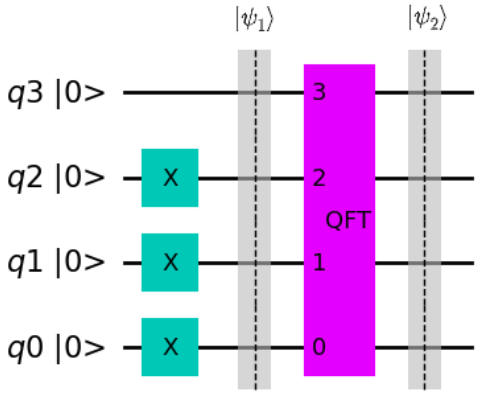


Fig. 11: An example application of QFT: this circuit calculates $QFT|7\rangle$ on 4 qubits.

C. Quantum Phase Estimation (QPE)

Another key element of Shor's algorithm is the quantum phase estimation (QPE) circuit. Let U be a quantum operator with some eigenstate $|\psi\rangle$ on n-qubits. Since U is unitary, all its eigenvalues must be of modulus 1, therefore allowing us to write (32), where θ is generally restricted to $\theta \in [0, 1)$ to avoid redundancy [14].

$$
U|\psi\rangle = e^{2\pi \mathrm{i}\theta}|\psi\rangle \tag{32}
$$

As mentioned in section III-A3, any operator can be transformed into a controlled operator, and that is what will be done with U, resulting in operator CU, such as in Fig 12, where the register for $|\psi\rangle$, called the "eigenstate register" contains n qubits, and the gate is controlled by a single-qubit register, called "evaluation register" (for reasons that will become clear later).

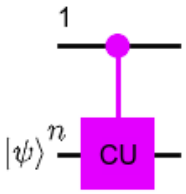
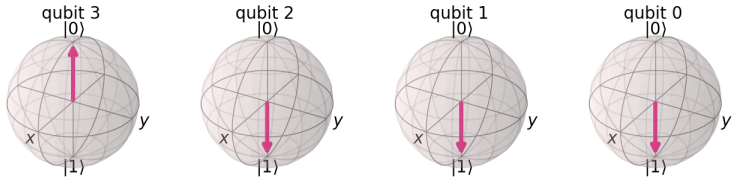


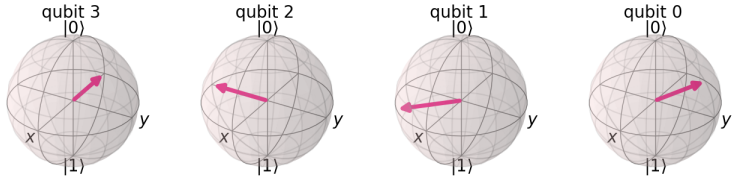
Fig. 12: Controlled application of gate U, which is controlled by a single qubit and targets the eigenstate register.

Consider the circuit in Fig. 13, where the notation CU^x means x applications of gate CU. If we start the evaluation register at $|0\rangle$, we can write (33), where $|\phi\rangle$ is some eigenstate[8] of operator U. After the application of the Hadamard gate, (33) becomes (34). Application of the controlled gate on (34) yields (35), which shows a remarkable property: both the value θ and

[8]The eigenstate notation was changed to $|\psi\rangle$ from the previous $|\psi\rangle$ to avoid confusion.



(a) State $|7\rangle$ - Before QFT



(b) State $|\tilde{7}\rangle$ - After QFT

Fig. 10: Example of QFT for element $|7\rangle$ in a 4-qubit system, which can be obtained using the circuit in Fig. 11. Each qubit state, both before and after the QFT, is represented using separate Bloch spheres.

the number of applications of U have been encoded into the relative phase of the evaluation qubit without changes to the eigenstate register.

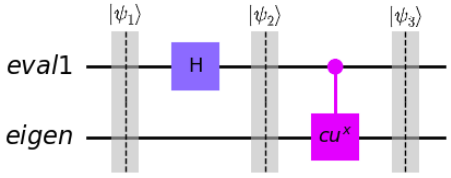


Fig. 13: Building block of QPE Circuit: Hadamard gates followed by a controlled application of U a total of x times.

$$|\psi_1\rangle = |0\rangle|\phi\rangle \tag{33}$$

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\phi\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle|\phi\rangle + |1\rangle|\phi\rangle) \end{aligned} \tag{34}$$

$$\begin{aligned} |\psi_3\rangle &= \frac{1}{\sqrt{2}}(CU^x|0\rangle|\phi\rangle + CU^x|1\rangle|\phi\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle|\phi\rangle + |1\rangle e^{2\pi i x\theta}|\phi\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i x\theta}|1\rangle)|\phi\rangle \end{aligned} \tag{35}$$

It is possible to generalize this circuit to one such in Fig. 14, known as the QPE circuit, where the evaluation register has n qubits and the eigenstate register has the necessary number of qubits to generate the eigenstate $|\phi\rangle$. After applying the several Hadamard gates, we arrive at (36). The application of the several controlled gates is identical to what was done in (34)-(35), thus allowing us to write (37), where $N = 2^n$. The form inside the parenthesis is exactly the same as (30) for the case when $|2^n\theta\rangle$ is an element of the computational basis for the n-qubit state space. It is possible, in that case, to write (37) as (38), from which (39) naturally follows. Performing a measurement from the evaluation register into the n classical bits, we find the value of $2^n\theta$ in binary, from which it is easy to calculate θ [14].

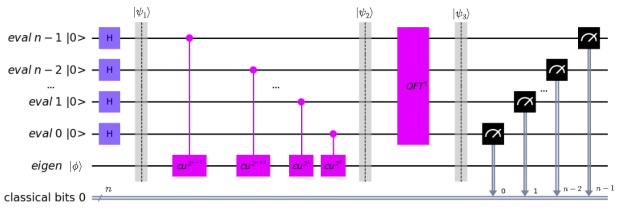


Fig. 14: QPE Circuit, which consists of parallel Hadamard gates on the evaluation qubits, followed by CU^x gates controlled by the evaluation registers and targeting the eigenstate registers. Finally, an inverse QFT is applied to the evaluation qubits, followed by measurements. The value of x for evaluation qubit q_n is $x = 2^n$.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\phi\rangle \tag{36}$$

$$
\begin{aligned}
|\psi_2\rangle &= \frac{1}{\sqrt{2^n}} \left(\bigotimes_{p=n-1}^{0} (|0\rangle + e^{2\pi \mathrm{i}\ 2^p \theta}|1\rangle) \right) |\phi\rangle \\
&= \frac{1}{\sqrt{2^n}} \left(\bigotimes_{p=n-1}^{0} (|0\rangle + e^{\frac{2\pi \mathrm{i}}{2^n}\ 2^n 2^p \theta}|1\rangle) \right) |\phi\rangle \\
&= \left(\frac{1}{\sqrt{N}} \bigotimes_{p=n-1}^{0} (|0\rangle + \omega_N^{(2^n\theta)\ 2^p}|1\rangle) \right) |\phi\rangle
\end{aligned} \tag{37}
$$

$$
|\psi_2\rangle = QFT(|2^n\theta\rangle)|\phi\rangle \tag{38}
$$

$$
\begin{aligned}
|\psi_3\rangle &= QFT^\dagger QFT(|2^n\theta\rangle)|\phi\rangle \\
&= |2^n\theta\rangle|\phi\rangle
\end{aligned} \tag{39}
$$

Of course, some hypotheses were implicit in the development of our equations. For example, we assume that $2^n\theta \in \mathbb{Z}_+$, which we shall challenge later on in this section. Moreover, for $|2^n\theta\rangle$ to be an element of the computational basis, we need $0 \leq \theta \leq 1 - 2^{-n}$ [14]. It is possible to see, therefore, that adding qubits to the evaluation register increases the range of values of θ that can be effectively measured. We can test this idea by applying the QPE circuit to the operator Y, which has an eigenvalue $e^{2\pi \mathrm{i}\ \frac{1}{4}}$ for eigenstate $|+\mathrm{i}\rangle$. The circuit will be built with 2 qubits on the evaluation register, therefore, we should measure $M(|2^2\frac{1}{4}\rangle) = M(|1\rangle) = 01$ on the classical register for all cases. This is precisely the behavior shown in Fig. 15, which was obtained with Qiskit for 1024 runs.

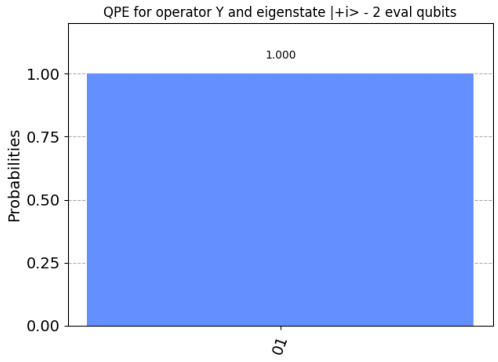


Fig. 15: An example application of QPE circuit for when $2^n\theta \in \mathbb{Z}_+$ - $(Y, |+\mathrm{i}\rangle)$. Only one value is measured.

On the other hand, it is possible that $2^n\theta \notin \mathbb{Z}_+$. Nevertheless, the algorithm can still yield the value of θ with reasonable precision. Specifically, using an evaluation register with n qubits, in order to obtain θ accurate to p bits with probability $1-\epsilon$, we must have $n = p + \left\lceil \log_2\left(2 + \frac{1}{2\epsilon}\right) \right\rceil$, with $\lceil . \rceil$ the ceiling function [4]. The chance that we'll obtain the closest value possible to $2^n\theta$ when measuring is generally higher than 40% [14]. We can illustrate this notion with an example: suppose we use operator $U = P(2\pi/3) = \begin{bmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\frac{2\pi}{3}} \end{bmatrix}$, which has eigenvalue $e^{2\pi \mathrm{i}\frac{1}{3}}$ for eigenstate $|\phi\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$. Using Qiskit to run the QPE algorithm on pair $(U, |1\rangle)$ with $n = \{3, 4\}$ evaluation qubits 1024 times, we can see the results in Fig. 16, which are also summarized TABLE I, demonstrating the impact that the increase on the number of qubits has on effectively running the QPE algorithm, and also that the algorithm works even if $2^n\theta \notin \mathbb{Z}_+$.

Finally, the problem of how to construct the eigenstate $|\phi\rangle$ for the QPE algorithm has been largely ignored, but will be taken into account in our subsequent discussion of Shor's algorithm.

Eval Qubits	Expected	Closest Measured	Difference	Pr[M(Closest)]
3	$\frac{8}{3} = 2.\bar{6}$	$0b011 = 3$	12.5%	0.693
4	$\frac{16}{3} = 5.\bar{3}$	$0b0101 = 5$	-6.25%	0.701

TABLE I: Results for $QPE((P(2\pi/3), |1\rangle))$ - 1024 runs with Qiskit

D. Shor's Algorithm

1) Algorithm Overview: Shor's algorithm is essentially an order finding algorithm. We may recall the definition of the order of an element from section II-C: the order $ord(a)$ of an element a in multiplicative group $(\mathbb{G}, \cdot)$ is the smallest positive integer n such that $a^n = 1_G$ [6].

Taking the group $(\mathbb{Z}_N, \otimes)$, with N a positive integer, the order r of element $a \in \mathbb{Z}_N$ such that[9] $\gcd(a, N) = 1$ is the smallest integer value that satisfies (40). The problem of finding r is considered hard if N is sufficiently large [12].

$$
a^r \mod N = 1 \tag{40}
$$

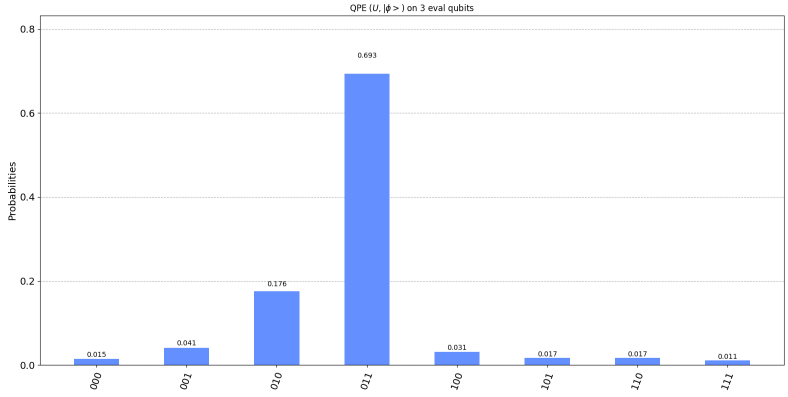
The function $f(x)$ in (41) is periodic of period r for $x \in \mathbb{Z}_+$ [4]. This function can be transformed to an analogous gate for quantum computation. Consider a system of n qubits such $2^n \geq N$, and a state $|y\rangle$ such that $y \in \mathbb{Z}, 0 \leq y < 2^n$, which makes $|y\rangle$ and element of the computational basis of the system. We can thus create an operator U such as in (42), where we can see that the result of the operation will also be an element of the computational basis. For the case where $|y\rangle = |1\rangle$, x applications of U leads to (43), which is analogous to (41) and shares with it the same properties, including period r [5]. Computationally, the gate U can be made using a series of swaps, cnots, ccnots, and X gates on the qubits [14].

$$
f(x) = a^x \mod N \tag{41}
$$

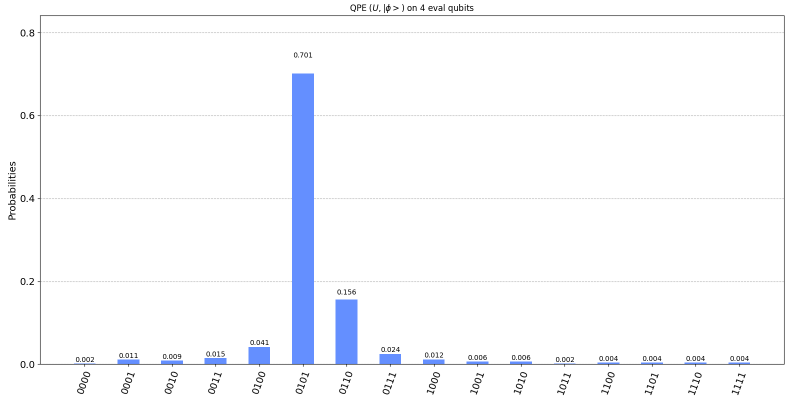
$$
U|y\rangle = |ay \mod N\rangle \tag{42}
$$

$$
U^x|1\rangle = |a^x \mod N\rangle \tag{43}
$$

[9]Restricting a and N to being co-primes guarantees a will have finite order [5].



(a) QPE for $(P(2\pi/3), |1\rangle)$ with 3 evaluation qubits.



(b) QPE for $(P(2\pi/3), |1\rangle)$ with 4 evaluation qubits.

Fig. 16: An example application of QPE circuit for when $2^n\theta \notin \mathbb{Z}_+$ - $(P(2\pi/3), |1\rangle)$. This illustrates the case of when the QPE circuit is not exact and how the number of evaluation qubits affects measurements.

Consider the state $|u_s\rangle$ such as in (44), with $s \in \mathbb{Z}$, where we can use k successive applications of $U|1\rangle$ to create the terms $|a^k \mod N\rangle$, which are themselves members of the orthonormal computational basis, thus creating a valid superposition. In (45), we can prove that $|u_s\rangle$ are eigenstates of U, with eigenvalues $e^{2\pi \mathrm{i} \frac{s}{r}}$. Furthermore, in (46), we prove that the states $|u_s\rangle$ are orthonormal.

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi \mathrm{i} \frac{sk}{r}} |a^k \mod N\rangle \tag{44}$$

$$\begin{aligned} U|u_s\rangle &= U\left(\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi \mathrm{i} \frac{sk}{r}} |a^k \mod N\rangle\right) \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi \mathrm{i} \frac{sk}{r}} U|a^k \mod N\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi \mathrm{i} \frac{sk}{r}} |a^{k+1} \mod N\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k'=1}^{r} e^{-2\pi \mathrm{i} \frac{s(k'-1)}{r}} |a^{k'} \mod N\rangle \\ &= e^{2\pi \mathrm{i} \frac{s}{r}} \left(\frac{1}{\sqrt{r}} \sum_{k'=1}^{r} e^{-2\pi \mathrm{i} \frac{sk'}{r}} |a^{k'} \mod N\rangle\right) \\ &= e^{2\pi \mathrm{i} \frac{s}{r}} |u_s\rangle \end{aligned} \tag{45}$$

$$\begin{aligned}\langle u_p|u_s\rangle &= \frac{1}{r}\sum_{k_1=0}^{r-1}\sum_{k_2=0}^{r-1} e^{2\pi \mathrm{i}\frac{(pk_1-sk_2)}{r}}\langle a^{k_1} \ mod N|a^{k_2} \ mod N\rangle \\ &= \frac{1}{r}\sum_{k=0}^{r-1} e^{2\pi \mathrm{i}k\frac{(p-s)}{r}} \\ &= \frac{1}{r} r\delta_{ps} \\ &= \delta_{ps}\end{aligned} \tag{46}$$

We can see that inputting a state $|u_s\rangle$ into the previously shown QPE algorithm would allows us to discover the value of $\frac{s}{r}$. However, since the value of r is not known *a priori*, it is not possible to construct eigenstate $|u_s\rangle$ for the eigenstate register. However, something remarkable happens when we attempt to superpose these eigenstates over a period, as in (47): the superposition of the eigenstates in a period is equal to the state $|1\rangle$, which can be obtained with a simple bit-flip in the least significant qubit of the eigenstate register.

$$\begin{aligned}|\phi\rangle &= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle \\ &= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi \mathrm{i}\frac{sk}{r}}|a^k \mod N\rangle \\ &= \frac{1}{r}\sum_{k=0}^{r-1}\left(\sum_{s=0}^{r-1} e^{-2\pi \mathrm{i}\frac{sk}{r}}\right)|a^k \mod N\rangle \\ &= \frac{1}{r}\sum_{k=0}^{r-1} r\delta_{0k}|a^k \mod N\rangle \\ &= |a^0 \mod N\rangle \\ &= |1\rangle\end{aligned} \tag{47}$$

The power of this manipulation can be seen when analyzing the circuit on Fig. 17, there the previous QPE circuit has been condensed into a single block. The circuit in Fig. 17 has $n = \lceil \log_2 N \rceil$ qubits in the eigenstate register in order to ensure all integers from 0 to $N-1$ are elements of the computational basis for the register [5]. As for the evaluation register, it has t qubits, with $N^2 \leq 2^t < 2N^2$, for reasons that will become more apparent later on [5].

From (47), it is possible to write (48) for the states in Fig. 17. After executing the QPE circuit, which distributes linearly over the superposition of the eigenstates, we arrive at (49).

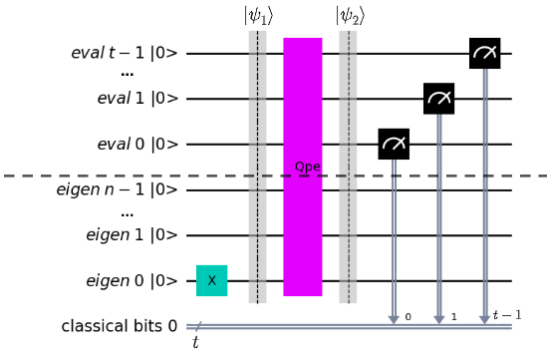


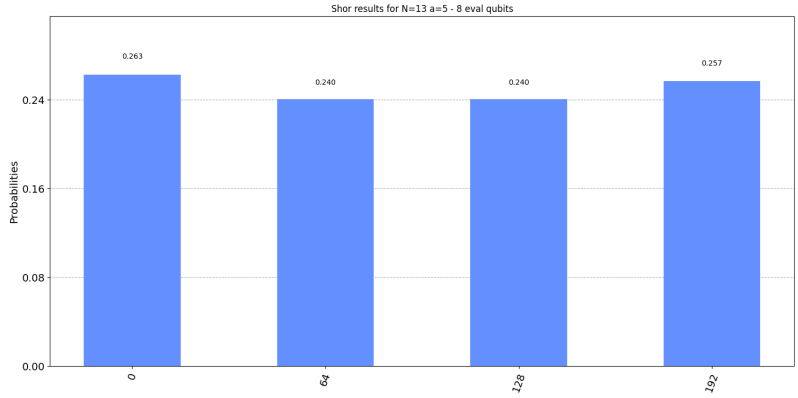
Fig. 17: A circuit to run Shor's order finding algorithm. It is built on top of the QPE circuit and makes use of the linearity principle by superposing eigenstates.

$$\begin{aligned}|\psi_1\rangle &= |0\rangle^{\otimes t}|1\rangle_n \\ &= |0\rangle^{\otimes t}\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle_n \\ &= \frac{1}{\sqrt{r}}\left(\sum_{s=0}^{r-1}|0\rangle^{\otimes t}|u_s\rangle_n\right)\end{aligned} \tag{48}$$

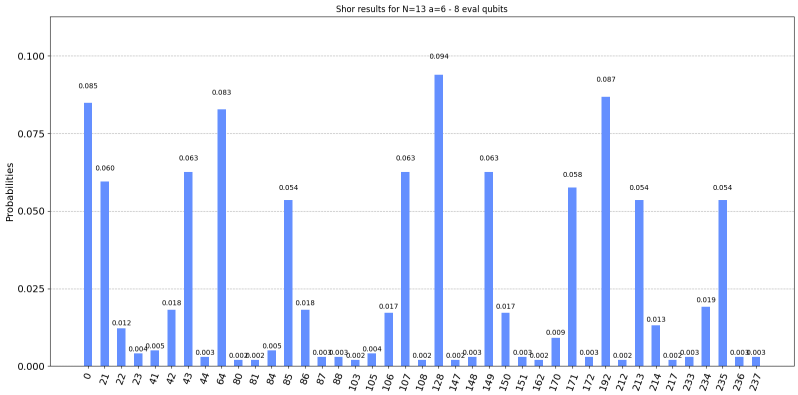
$$\begin{aligned}|\psi_2\rangle &= \frac{1}{\sqrt{r}}\left(\sum_{s=0}^{r-1} QPE(|0\rangle^{\otimes t}|u_s\rangle_n)\right) \\ &= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|2^t\frac{s}{r}\rangle_t|u_s\rangle_n\end{aligned} \tag{49}$$

By performing a measurement on the t evaluation registers, we can find the value of order r. In the case where $\frac{2^t}{r} \in \mathbb{Z}$, all elements $|s\frac{2^t}{r}\rangle$ are in the computational basis of the evaluation register, thus only these states will be measured, in very marked fashion. On the other hand, similarly to what happens with the QPE algorithm, when $\frac{2^t}{r} \notin \mathbb{Z}$ measurements will be distributed in an oscillatory fashion, with peaks in basis elements which are the closest to $s\frac{2^t}{r}$. We can see this behavior in the example of Fig. 18, where $N = 13$ and $a = \{5, 6\}$, which have orders 4 and 12 respectively. As expected, the case where $a = 5, r = 4$ yields measurements precisely on $\{0, 64, 128, 192\}$. When $a = 6, r = 12$, however, measurements will be dispersed, but with higher counts on integers closest to $\{0, 21.\bar{3}, 42.\bar{6}, ...\}$.

Even when the measurements are not in exact multiples, such as in Fig. 18b, it is still possible to estimate the order r of a by using continued fraction expansions (CFE). In the case where $N^2 \leq 2^t < 2N^2$, there is only one fraction with denominator N that is within $\frac{1}{N^2}$ of $\frac{v}{2^t}$, where v is an output measurement of the algorithm. When r and s are co-prime, the denominator of said fraction is equal to r [5]. In the example of Fig. 18b, suppose the algorithm has yielded $v = 21$. The CFE outputs $\frac{1}{12}$, resulting in $r = 12$, which is precisely the value desired. In can be quickly verified if the denominator



(a) $a = 5$



(b) $a = 6$

Fig. 18: Example of Shor's algorithm - $N = 13$.

outputted by the continued fraction expansion is in fact the order of a. Running Shor's algorithm multiple times increases the chance of correctly estimating r [4].

2) Shor Against RSA: Shor's order finding algorithm can be used to effectively factor an integer N as per Algorithm 1, thus breaking the RSA cryptosystem [5].

To compare Algorithm 1 with the RSA cryptosystem, we can assume that N is already in non-trivial, semi-prime format. By picking a random a, it is possible we might "get lucky" and find a factor of N. Even though this is rare, the algorithm covers this case. Much more likely is the case where a shares no factors with N, thus meaning we can use Shor's algorithm as described previously to efficiently find the order of a in $(\mathbb{Z}_N, \otimes)$. From the definition of the order of a, we get (50) for the case when r is even, with k being some integer. If neither term on the left hand side of (50) is a multiple of N, both terms share nontrivial factors with N. Upon obtaining p and q from the algorithm, it can be quickly verified if $N \mod p = 0$ and $N \mod q = 0$. If not, we can run the algorithm as many times as are needed.

Algorithm 1: Expansion of the random tree $\mathcal{T}$

```
Function Shor RSA(N)
    a = Random integer between 2 and N − 1
    if gcd(a, N) == 1 then
        r = Order of a in (Z_N, ⊗) from Shor-CFE
        if r is even then
            p = gcd(a^(r/2) + 1, N)
            q = gcd(a^(r/2) − 1, N)
        end
    end
    else
        return a
    end
end
```

$$a^r \equiv 1 \pmod{N} \rightarrow (a^{\frac{r}{2}} + 1) \times (a^{\frac{r}{2}} - 1) = k \times N \quad (50)$$

Finally, it is fundamental to mention that Algorithm 1 runs polynomially on the size of N, that is on $O(poly(\log N))$,

which is significantly faster than any existing classical algorithm with the same end [17], meaning its implementation would break the RSA cryptosystem.

3) Shor Against ElGamal: Shor's period finding algorithm can also be used effectively against the ElGamal cryptosystem, albeit with some modifications.

In fact, both the integer factorization and the discrete logarithm problem can be formulated as hidden subgroup problems on Abelian groups, which can be solved efficiently by quantum computers [4]. Although the proof of this statement and its adaptability to discrete logarithm are beyond the scope of this work, we shall enumerate the steps that lead to this conclusion.

Hidden subgroup problem (HSP): let $G = (\mathbb{G}, \cdot)$ be a group. If a subgroub $H < G$ can be implicitly defined by a function f on G such that f is constant on every coset of H but distinct for each coset, the problem of finding a generating set of H is called the hidden subgroup problem [6]. When the group G is Abelian, this problem can be solved by a quantum computer using $O(poly(\log ord(G)))$ operations and one call to a black-box oracle, thus meaning this problem is solved in a reasonable time [4].

The discrete logarithm as a hidden subgroup problem: given group $G = (\mathbb{Z}_p^*, \otimes)$ where p is a prime, a generator $g \in G$ and an arbitrary element $y \in G$, the discrete logarithm problem consists in finding $x \in G$ such that $g^x = y$. If we consider the function $f : G \times G \to G$ such that $f(a, b) = g^{-a}y^b$, the set of elements such that $f(a, b) = 1$ is the hidden subgroup H of $G \times G$ of tuples of the form (kx, k). By solving the HSP for H, the element $(x, 1)$ can be computed, thus solving the discrete logarithm problem by yielding x [5]. This formulation is effective against other kinds of discrete logarithm formulations, including elliptic curves, which are also widely used in modern crypography [18].

4) Difficulties in Implementing Shor's Algorithm: While Shor's algorithm (both in its original formulation and general HSP adaptations) can be used to undermine modern cryptography systems, its practical implementation is still far from complete. Considering RSA factoring, for example, so far only the factorization of very small numbers has been achieved with quantum computers running Shor's algorithm [19]. The main problems with effectively running Shor's algorithm consist in 1) providing a sufficient number of qubits [4]; and 2) avoiding decoherence during the execution of the algorithm [20]. Most current experimental verifications of Shor's algorithm have actually relied upon previous knowledge of the answer [19].

It is worth mentioning that there are approaches other than Shor's algorithm that are more efficient when it comes to breaking certain encryption patterns. As an example, [21] has managed to achieve factorization of 56153 with only 4 qubits, which is a remarkable feat. On the whole, however, quantum algorithms still under-perform most classical algorithms, and some time is still needed for quantum computers to pose a serious threat to modern cryptography standards.

IV. Post-Quantum Key Exchanges

A. Principles of Quantum Key Distribution

Even though quantum computers are not advanced enough to pose a threat to communications security, that may be the case in the near future. Therefore, several systems have been proposed which ensure secrecy in the post-quantum world. One option (though not the only one) is to use the principles of quantum mechanics in order to securely distribute cryptographic keys. These methods, which are collectively called "quantum key distribution" (QKD), rely both on the probabilistic nature of quantum physics and on the no cloning theorem. The first method of this type to be proposed was the BB84 protocol, which still serves as a kind of "blueprint" for QKD [5].

B. BB84 Protocol

The BB84 protocol was proposed as a means for two parties, named Alice and Bob, to exchange a cryptographic key through a channel where eavesdropper Eve can intercept and read information. This is not a cryptosystem, but just a means of transmitting a key, which can then be used to establish secure communications between the parties via some symmetric key system such as AES, for example.

The procedure is as follows [5]:

1) Alice randomly generates n-bit key $k = \{0, 1\}^n$;
2) Alice randomly generates n-bit string $q = \{0, 1\}^n$;
3) Alice encodes (k, q) as $|\psi\rangle$, where $|\psi\rangle$ is an n-qubit state as expressed in (51) and using the encoding presented in (52). As (52) shows, the bit q_i determines the basis of the encoding while the bit k_i determines the value of the 1-qubit state in the basis chosen;
4) Alice sends state $|\psi\rangle$ over the network to Bob;
5) Bob randomly generates string $q' = \{0, 1\}^n$;
6) Bob calculates string k' as shown in (53), which means that the value of q'_i determines if Bob will use basis $\{|0\rangle, |1\rangle\}$ or $\{|+\rangle, |-\rangle\}$ for measuring the qubits sent by Alice. Once the measurement is made, the bit assigned to k'_i is straightforward from the encoding in (52). If $q_i = q'_i$, both Alice and Bob must have the same key-bit $k_i = k'_i$;
7) Alice and Bob compare the values of q_i and q'_i over the network. If $q_i = q'_i$, they keep the bits k_i and k'_i. If $q_i \neq q'_i$, they discard the bits k_i and k'_i. After comparing these strings and discarding all diverging bits, we shall have for the reduced keys $\bar{k}$ and $\bar{k}'$ that $\bar{k} = \bar{k}'$.

$$|\psi\rangle = \bigotimes_{i=n-1}^{0} |\psi_i\rangle = \bigotimes_{i=n-1}^{0} |\psi_{k_i q_i}\rangle \tag{51}$$

$$\begin{cases} |\psi_{00}\rangle = |0\rangle \\ |\psi_{10}\rangle = |1\rangle \\ |\psi_{01}\rangle = |+\rangle \\ |\psi_{11}\rangle = |-\rangle \end{cases} \tag{52}$$

$$k'_i = \delta_{0q'_i}\langle 1|\psi_i\rangle + \delta_{1q'_i}\langle -|\psi_i\rangle \tag{53}$$

We must have $\bar{k} = \bar{k}'$ from the fact that if Alice and Bob used the same basis for encoding and measurement respectively, then they must have the same key-bits, with all others being discarded [5].

As for security, Eve can intercept both the state $|\psi\rangle$ and the values of q and q' during the comparison phase. If Eve measures the values of $|\psi_i\rangle$, she will have a 50% chance of choosing the same basis as Alice, meaning that for long keys she will know 50% of k, which she can verify by intercepting the values of q sent for comparison. However, Bob also has to choose the same basis in order for the bit to kept in the reduced key, which Eve can also verify by looking at q'. The chance of Eve having chosen the same basis as Alice and Bob is 50%, meaning for long keys she will know approximately 50% of $\bar{k}$. This value is independent of the computational power possesed by Eve, meaning quantum computers are useless to improve information collection [5].

While this attack by Eve poses a security risk, because of the no-cloning theorem Eve's measurement of the states $|\psi\rangle$ will necessarily cause disruption if she has chosen the wrong basis, which will happen approximately 50% of the time. Therefore, it might so happen that the values of $\bar{k}_i \neq \bar{k}'_i$ for some bit because of Eve's disruption. For this disruption to happen, Eve must have used a different basis than Alice and Bob, which happens with probability 50% in the context of the reduced key, and Bob's measurement from this qubit must result in a different encoding, which happens with probability 50%, meaning that for each bit i in the reduced key, there is probability 25% that $\bar{k}_i \neq \bar{k}'_i$. If Alice and Bob sacrifice some bits of the reduced key by comparing them over the network and them discarding them, the comparison of 9 bits suffice to detect tampering with over 90% chance, with tampering being recognized by having $\bar{k}_i \neq \bar{k}'_i$. By detecting tampering, Alice and Bob can simply move to a different channel [5].

For all its advantages, the BB84 has some liabilities which render it impractical, such as vulnerabilities to man-in-the-middle attacks [22], decoherence of the quantum states and noise over long distance transmission [5], among others [23]. For these reasons, BB84 has been generally superseded by some of its spin-offs, such as E91 and B92, but these too suffer from their own problems [5], meaning QKD still has to surmount many challenges before being commercially implementable.

V. Conclusion

As has been shown, Shor's order finding algorithm, which is an application of the quantum phase estimation circuit, could theoretically be successfully employed against both the RSA and ElGamal cryptosystems, undermining the security of modern communications. In practice, however, some the difficulties involved in managing multiple-qubit systems mean that it will still take some time before this type of attack poses an imminent threat. In order to hedge this eventual scenario, schemes that can potentially provide secrecy in the post-quantum computers world are already being developed, one such example being the BB84 protocol. Such schemes, despite their shortcomings, can be improved upon before the use of quantum computers become widespread. In summary, as was the case in the past, the constant race between cryptography and cryptanalysis is still very much a reality.

References

[1] David Kahn. *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996.

[2] Claude E Shannon. "Communication theory of secrecy systems". In: *The Bell system technical journal* 28.4 (1949), pp. 656–715.

[3] Douglas R Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005.

[4] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[5] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011.

[6] Kenneth H Rosen. *Handbook of discrete and combinatorial mathematics*. CRC press, 2017.

[7] Arjen K. Lenstra. "Integer Factoring". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 611–618. ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_455. URL: https://doi.org/10.1007/978-1-4419-5906-5_455.

[8] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, et al. "Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 62–91.

[9] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption & how to play mental poker keeping secret all partial information". In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. 1982, pp. 365–377.

[10] Yongge Wang. "Public key cryptography standards: Pkcs". In: *arXiv preprint arXiv:1207.5446* (2012).

[11] OEIS Foundation Inc. *Entry A033948 in the on-line encyclopedia of integer sequences*. 2022. URL: https://oeis.org/A033948.

[12] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[13] N David Mermin. *Quantum computer science: an introduction*. Cambridge University Press, 2007.

[14] MD SAJID ANIS, Abby-Mitchell, Héctor Abraham, et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[15] Xiao-Qi Zhou, Timothy C Ralph, Pruet Kalasuwan, et al. "Adding control to arbitrary unknown quantum operations". In: *Nature communications* 2.1 (2011), pp. 1–8.

[16] Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992.

[17] David Beckman, Amalavoyal N Chari, Srikrishna Devabhaktuni, et al. “Efficient networks for quantum factoring”. In: *Physical Review A* 54.2 (1996), p. 1034.

[18] John Proos and Christof Zalka. “Shor’s discrete logarithm quantum algorithm for elliptic curves”. In: *arXiv preprint quant-ph/0301141* (2003).

[19] Unathi Skosana and Mark Tame. “Demonstration of Shor’s factoring algorithm for N = 21 on IBM quantum processors”. In: *Scientific Reports* 11.1 (2021), pp. 1–12.

[20] GP Berman, DI Kamenev, and VI Tsifrinovich. “Collective decoherence of the superpositional entangled states in the quantum Shor algorithm”. In: *Physical Review A* 71.3 (2005), p. 032346.

[21] Nikesh S Dattani and Nathaniel Bryans. “Quantum factorization of 56153 with only 4 qubits”. In: *arXiv preprint arXiv:1411.6758* (2014).

[22] Jinxiang Huang, Yong Wang, Huadeng Wang, et al. “Man-in-the-middle attack on BB84 protocol and its defence”. In: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. 2009, pp. 438–439. DOI: 10.1109/ICCSIT.2009.5234678.

[23] Rahul Aggarwal, Heeren Sharma, and Deepak Gupta. “Analysis of various attacks over BB84 quantum key distribution protocol”. In: *International Journal of Computer Applications* 20.8 (2011), pp. 28–31.