Quantum algorithms for discrete logarithms with applications to Diffie Hellman and ElGamal

Research Project Report

Introduction to Quantum Computing CS 599

Fall 2025

Student: Asya Dente

Professor: Alexander Poremba

Contents

Abstract

A new technology is emerging, and it's putting in danger our security protocols, data and privacy. What could this threat be? It turns out this *new* technology is not actually so new. Quantum Computing is not exactly new as its been around for a while now, but its advancements have been rather slow mainly because of interference, stability and isolation issues. Once its potential is unleashed, it could cause quite a few problems in relation to today's systems. Quantum Computing is based on the principles of quantum physics, and its development could bring new opportunities, but with that, new risks.

This project will focus on the impact that quantum algorithms have on some of our current systems. More specifically systems that have security and privacy as their main objective. We will project ourselves into the future for a while and understand the implications that quantum computing will have on programs like these that define security in today's world. Not too surprisingly, we will see that cryptography needs a new plan for the future in order not to fall behind too quickly. Unfortunately, as secure as these systems are today, it only takes a powerful quantum computer to render them useless.

Chapter 1

Introduction

1.1 Motivation and Context

This topic is interesting to me because it focuses on a subject related to cybersecurity, which is one of the most fundamental sections of technology advancements. But, it sadly is also often overlooked. Today, people are so excited to jump into the next big thing, that they often tend to leave out its less "glamorous", attention-grabbing aspects. Up to this day, public key cryptography systems like Diffie-Hellman key exchange and ElGamal encryption rely on the computational difficulty of the discrete logarithm problem.

While post quantum cryptography is constantly being developed and tested, in the hopes to protect our current systems, it's necessary to understand what specifically is at risk.

Through this project, we will dive deeper into the complexities of quantum computing in hopes of better understanding the areas posing threats and how exactly quantum computers can make something like this possible.

One of the main works that have influenced this area is Shor's algorithm which has been widely studied. Without much background knowledge, it is still possible to develop a good research project looking into sources such as the one mentioned above and the book "Quantum Computation and Quantum Information" by Nielsen and Chuang just to mention two.

1.2 Project Objective & Structure of Report

This project will first analyze the literature review and share the necessary knowledge in order to answer the bigger question: how can quantum algorithms find vulnerabilities in Diffie Hellman and ElGamal, thus potentially putting our data at risk?

This report will use the sources mentioned previously, as well as some others, in order to answer to the following research questions:

1. Where does this threat come from and why is quantum computing the problem here?
2. What are discrete logarithm problems, and why are they pertinent to this research? How is quantum already one step ahead?
3. How will quantum affect certain applications such as Diffie Hellman and ElGamal?
4. Why is this an important topic to talk about today?

Chapter 2 will introduce the report with some background information and situate us for the research starting with the first few questions.

Chapter 3 will answer the second questions regarding discrete logarithms, their importance, as well as how to solve this problem using quantum algorithms.

Chapter 4 and Chapter 5 will respond to the third questions developing on the two schemes Diffie Hellman and ElGamal and explaining their implications in this research project.

Chapter 6 will finally conclude the research with a small word about post quantum cryptography and the importance of this subject in today's world; especially concerning what the project will have shared.

Chapter 2

Background: Quantum Computing and Cryptography

2.1 Quantum Computation

Quantum Computing uses the principles of linear algebra and complex vector spaces as foundational tools. Unlike classical computers, which make use of bits, quantum computers use qubits. At a physical level, a classical bit can only exist in one of two distinct states (like high/low voltage) representing either 0 or 1. A qubit, on the other hand, is a quantum system (like the spin of an electron or the polarization of a photon) that can exist in a **superposition** of both 0 and 1 at once, collapsing to one outcome only when measured (a problem today in the development of powerful quantum computers). Mathematically, this behavior is described by modeling a qubit as a normalized vector in a two-dimensional Hilbert space, typically represented as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \text{with} \qquad |\alpha|^2 + |\beta|^2 = 1$$

where $|0\rangle$ and $|1\rangle$ form an orthonormal basis and $\alpha, \beta \in \mathbb{C}$. Quantum gates are linear operators that act on such vectors and are represented by complex-valued matrices. Perhaps the most essential gate in quantum computing is the **Hadamard** gate which gives way to superposition.

Quantum systems also express **entanglement**, a correlation between qubits so strong that the state of one qubit cannot be described independently of the other. They become linked so that, when measured, the state of one instantly determines the state of the other, no matter how far apart they are. This property allows for many protocols in quantum information, including quantum teleportation and parts of Shor's algorithm[1].

[1] class notes

This leads us to another popular quantum term: **interference**. Interference happens when different quantum states combine with each other, sometimes strengthening and sometimes canceling each other out. Quantum algorithms use interference to amplify correct answers and cancel out wrong ones. One of the most famous examples of this is Shor's algorithm, which uses interference to factor integers and solve discrete logarithms faster than any other way (classically at least). This is partially why quantum computing represents such a serious threat to cryptography.

2.2 Cryptography

Cryptography is what allows individuals to secure information using mathematical algorithms that will convert readable data (which is also referred to as plaintext) to unreadable data (ciphertext). The goals of cryptography are simple: to allow for confidential, secure and authenticated communication. Classical cryptography is able to do this, but only with certain computational assumptions such as the idea that factoring large integers is hard, and computing discrete logarithms is hard. These assumptions are what make these problems not computationally feasible for classical computers.

THere are two popular approaches in cryptography: symmetric and asymmetric schemes. In symmetric cryptography, users need to have met or somehow exchanged a secret key beforehand; this quickly becomes complicated since most communications don't happen like this. Asymmetric cryptography solves this issue by having users share one key (their public key), while keeping the other one secret (their private key). The security of this system depends on the fact that finding the private key from the public key would require solving a hard mathematical problem. One such problem is the Discrete Logarithm Problem (DLP), and it is the foundation of some of the most popular asymmetric schemes like Diffie-Hellman and ElGamal which will be further analyzed in the next few chapters. As long as this problem remains computationally difficult, these schemes are secure against classical attacks.

However, with quantum computation in the picture, new algorithms become computationally feasible. Quantum algorithms, like Shor's algorithm, can solve certain mathematical problems that classical cryptography depends on (in a timely manner). By computing discrete logarithms in polynomial time (instead of exponential for classical computers), a powerful quantum computer would be able to break both Diffie–Hellman and ElGamal, making them insecure.

Chapter 3

Discrete Logarithm Problem (DLP)

3.1 Definition

The Discrete Logarithm Problem (DLP) is a fundamental problem in cryptography. It appears simple, but it has a crucial property that makes it computationally difficult for classical computers to solve. The problem is defined as follows:

$$g^x \equiv h \ (mod \ p)$$

where

- p is a large prime number [1]
- g is a primitive root (generator) of $\mathbb{Z}_p$ [2]
- h is a known element from group $\mathbb{Z}_p$

The task is to find the integer x such that raising g to that power gives us h modulo p. Here, x is defined as the discrete logarithm of h. Essentially, we want to compute the following:

$$x \equiv log_g(h) \ (mod \ p)$$

While the first equation was simple to compute, finding x is the real challenge that DLP poses. Consider this small example where we are given a base g, a modulus p, and a result h; our task is to find x such that $5^x \equiv 8 \ (mod \ 23)$:
So, we know that $p = 23$, $g = 5$, $h = 8$[3]. Since here the numbers are rather small, we would

[1] for a prime p, the multiplication group $\mathbb{Z}_p$ is all the numbers from 1 to $p-1$ using multiplication mod p.
[2] this means powers of g give every non zero number modulo p. If not primitive root, then DLP could have no solution for an h
[3] I asked ChatGPT to find me three values that would work

be able to use brute force and just try all the powers of 5 mod 23:

$x = 1$	$5 \; mod \; 23 = 5$	$\neq 8$
$x = 2$	$25 \; mod \; 23 = 2$	$\neq 8$
$x = 3$	$125 \; mod \; 23 = 10$	$\neq 8$
$x = 4$	$625 \; mod \; 23 = 4$	$\neq 8$
$x = 5$	$3125 \; mod \; 23 = 20$	$\neq 8$
$x = 6$	$15625 \; mod \; 23 = 8$	$= 8$ so we can stop

We can now see that $5^6 \equiv 8 \; (mod \; 23)$, thus the discrete logarithm is $x = 6$.

This example shows how easy DLP is to solve for a small prime; nevertheless, in real cryptographic systems, the prime p is usually 2048 bits long meaning it has around $2^{2048} \approx 10^{616}$ possible values. Brute force becomes unfeasible, even with the best classical algorithms (it would take way too long). This is exactly why systems that use DLP are secure on classical computers.

3.2 Mathematical Structure

But how is this problem so complex when its reverse is so easy? To understand this, we need to look at the structure of the group in which it's defined. When p is a prime number, the set

$$\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$$

forms a *cyclic* group under multiplication modulo p. A cyclic group means that there exists some element g (called a generator) whose powers

$$g^1, \; g^2, \; g^3, \; \ldots, \; g^{p-1} \pmod{p}$$

produce every number in the group exactly once before repeating. So, if g is chosen as a generator, then for any $h \in \mathbb{Z}_p^*$ there has to be an exponent x such that $g^x \equiv h \pmod{p}$. This happens for sure if we chose correct values that respect our constraints since g would go through each value possible. If g is not a primitive root, then DLP may not find a solution for a certain h.

As we know, we can compute $g^x \pmod{p}$ very easily, even when p is extremely large, because it's simply a matter of squaring repeatedly. However, computing the reverse requires about $O(\sqrt{p})$ operations using the best algorithms on classical computers. For a 2048 bit prime, this would be around 2^{1024} steps, which is much more than any classical computers can handle.

3.3 Threat to DLP: Shor's Algorithm

With the presence of quantum computers, discrete logarithm problems can be solved in *polynomial time*. In lecture 12, we saw how Shor's algorithm was able to factor an integer N efficiently using *period finding*. This consisted in two parts: the classical part, which took care of turning the original problem into one that could be solved using period finding, and the quantum part, which took care of finding that period using superposition, interference and the Quantum Fourier Transform (QFT). Here below is a quick overview of how Shor's works:
In order to factor N, we choose a random x such that $1 < x < N$[4] and consider the function $f(j) = x^j \ mod \ N$, which has a period r (with $a^r \equiv 1 \ mod \ N$). The quantum portion of the algorithm creates a superposition over many values of j, evaluates $f(j)$ in parallel, and applies the QFT to extract the periodic structure of f. Once we have the period r, we can compute a nontrivial factor of N using classical arithmetic.

This same idea is used to resolve the discrete logarithm problem. Instead of trying to find the exponent x directly from the equation $g^x \equiv h \pmod{p}$, the quantum algorithm constructs a function whose periodic behavior finds the unknown value of x. One such function, discussed in Nielsen and Chuang (Section 5.4.2), is

$$f(x_1, x_2) = a^{sx_1+x_2} \bmod N$$

where s is the discrete logarithm we want to compute. This function is still periodic, except now, it's periodic in two dimensions, since for any integer ℓ we have

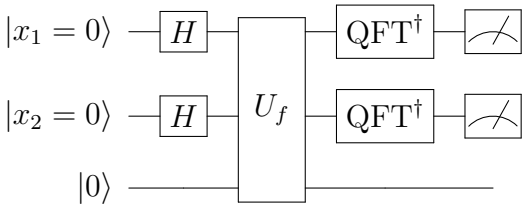
$$f(x_1 + \ell, x_2 - \ell s) = f(x_1, x_2)$$

which means that shifting the input pair (x_1, x_2) by the vector $(\ell, -s\ell)$ leaves the function value unchanged.

A quantum computer can use this and create a superposition over many pairs (x_1, x_2), evaluating f on all of them at the same time by applying a black box oracle U whcih takes care of the unitary transform $U|x_1\rangle|x_2\rangle|y\rangle \to |x_1\rangle|x_2\rangle|y \oplus f(x)\rangle$. Applying the QFT to the first two registers transforms this superposition so that the hidden two-dimensional period, which was hidden, is now visible through interference.

Measuring the two first registers then gives us information about the period from which the value of s can be recovered using the continued fractions algorithm mentioned in the book.

[4] if $gcd(x, N) > 1$, then algorithm already stops because we found a non trivial factor of N



The circuit above gives a simplified view of the discrete logarithm algorithm. The first two registers have the variables (x_1, x_2), while the third register stores the value of the function

$$f(x_1, x_2) = a^{sx_1+x_2} \bmod N$$

whose hidden period $(\ell,\ -s\ell)$ encodes the discrete logarithm s. After initializing the registers, the two Hadamard gates create the superposition

$$\frac{1}{2^t} \sum_{x_1=0}^{2^t-1} \sum_{x_2=0}^{2^t-1} |x_1\rangle|x_2\rangle|0\rangle$$[5]

Applying the oracle U_f computes $f(x_1, x_2)$ in parallel, giving us the amplitude

$$e^{2\pi i(s\ell_2 x_1+\ell_2 x_2)/r}$$

To extract the hidden period, the algorithm then applies the inverse Quantum Fourier Transform (QFT†) to the first two registers. The QFT turns periodicity in the computational basis into "peaks" in terms of frequency, which then makes the measurements reach values that make sense in terms of the hidden period vector $(\ell,\ -s\ell)$. Measuring these first two registers finally gives us values that satisfy the constraint from the book

$$\sum_{k=0}^{r-1} e^{2\pi ik(\ell_1/s-\ell_2)/r} = r$$[6]

As mentioned before, we can now use the generalized continued fractions algorithm that will then recover the discrete logarithm s.

Because the security of schemes like Diffie Hellman and ElGamal rely on the classical complexity of computing discrete logarithms, these quantum algorithms expose vulnerabilities which make these schemes no longer secure. The next chapters will show precisely how these protocols depend on DLP and how quantum algorithms destroys their security.

[5] alg step 2, Nielsen and Chuang (Section 5.4.2)
[6] equation in section 5.71, Nielsen and Chuang (Section 5.4.2)

Chapter 4

Application to Diffie Hellman

4.1 Diffie Hellman

Diffie Hellman key exchange is a method used generate a symmetric key, to secure communications, over a public channel. The idea behind Diffie Hellman (6.1) is that two users have a secret key each, which only they can see, and some public parameters which everyone can see. In order to secure a connection, they will both use the public parameters on their own key, and send the computed exponentiations to the other user. When they receive the computation from their partner, they will both use their own secret key to get the final resulting key. If the procedure is done correctly, they will end up with the same resulting key as it contains a "mixture" of both of their private keys along with the public parameters. Since this all seems very abstract, take Alice and Bob as an example:
Alice has a secret key $a = 4$, and Bob has his secret key $b = 6$. They publicly agree on a large prime modulus p and a generator g. To keep it simple, take $p = 23$ and $g = 5$. Alice then computes

$$A = g^a \bmod p = 5^4 \bmod 23 = 4$$

and Bob computes

$$B = g^b \bmod p = 5^6 \bmod 23 = 8$$

They exchange the values A and B over the public channel (everyone can see these values as they're nothing readable). Alice now raises Bob's value to her secret exponent:

$$s = B^a \bmod p = 8^4 \bmod 23 = 2$$

and Bob raises Alice's value to his secret exponent:

$$s = A^b \bmod p = 4^6 \bmod 23 = 2$$

And we can that both computations give out the same shared key $s = g^{ab} \bmod p = 5^{6*4}\ mod\ 23 = 2$, which can now be used as a symmetric key for secure communication.

If an eavesdropper was to look at this exchange, they would be able to see the public values A, B and the parameters (p, g); however, the private exponents a and b remain hidden. Recovering the shared secret key

$$s = g^{ab} \bmod p$$

from the available information is believed to be difficult because of its assumptions:

1. The Discrete Logarithm Problem (seen before)
2. The Computational Diffie–Hellman Problem (CDH)[1]

It would require solving the Computational Diffie Hellman (CDH) problem or, computiing one of the discrete logarithms using DLP which we've discussed in this last chapter. This is exactly why the application of DLP to Diffie Hellman can cause problems in the future, Diffie Hellman is based off the assumption that DLP is computationally hard.

4.2 Quantum Implications

A classical adversary cannot really recover a or b, since this would require solving DLP. A quantum adversary, however, can apply Shor's algorithm for discrete logarithms to compute

$$a = \log_g(A) \qquad \text{or} \qquad b = \log_g(B)$$

in polynomial time.

To solve Diffie Hellman, we can use the same algorithm as DLP; the attacker would just use the public value $A = g^a \bmod p$ (or $B = g^b \bmod p$) as an instance of the DLP algorithm and apply the same quantum procedure described in the previous chapter. The attacker would construct the same function, only with a since that's the value we're after this time:

$$f(x_1, x_2) = g^{ax_1 + x_2} \bmod p,$$

[1] given g^a, g^b, p, compute $g^{ab}\ mod\ p$. So, could an eavesdropper compute the shared secret without knowing the private values a or b at all?

Once either one of the exponents is known, getting the shared secret is pretty simple. For example, if the attacker learns a, then

$$s = g^{ab} \bmod p = (g^b)^a \bmod p = B^a \bmod p.$$

Thus, a quantum attacker can get the same key that Alice and Bob create, and can decrypt any encrypted communication that relies on this key.

Since recovering a or b breaks the Computational Diffie Hellman (CDH) assumption, Diffie Hellman cannot be considered secure in a post-quantum world. Any system built on Diffie Hellman for key creation becomes vulnerable once an adversary can run Shor's algorithm.

Chapter 5

Application to ElGamal

5.1 ElGamal

ElGamal is a public-key encryption algorithm based on the Diffie Hellman key key exchange that uses a private and public key pair for secure communication and digital signatures. In order to explain clearly the idea behind ElGamal (6.2), let us use Alice and Bob again as two users wishing to communicate securely. Alice wishes to send a message to Bob. The process can be broken up into three parts:

1. Bob creates a private key b only he can see, and a public key $h = g^b \ mod \ p$ which everyone is able to see. So far, this scheme uses the same properties as Diffie Hellman.

2. Alice chooses a random number k that is part of $\mathbb{Z}_p$, and uses g^k as some sort of a one-time pad generator. SHe then computes two things:

$$c_1 = g^k \ mod \ p \qquad and \qquad c_2 = m \cdot h^k \ mod \ p$$

   which she will send back to Bob. The reason why she sends a pair of values is becasue the value of c_1 will help Bob decrypt the message from c_2.

3. Bob now has everything he needs in order to decrypt the message. He received (c_1, c_2), and has his secret key b. So, he can now compute $c_1^b = g^{kb} \ mod \ p = h^k \ mod \ p$ which gets him the same number Alice multiplied her message by. Bob can now remove the encryption on the message by doing

$$m = c_2 \cdot (h^k)^{-1} \ mod \ p$$

   And Bob has the message decrypted only to himslef.

Looking at the following example could help make this clearer:
Bob's secret key is $b = 6$, the public parameters are $p = 23$ and $g = 5$ (like previous example), thus his public key is $h = g^b \ mod \ p = 5^6 \ mod \ 23 = 8$. Now, let's say Alice wants to encrypt the message $m = 15$; she chooses a new random number $k = 9$ and encrypts the message as follows:

$$c_1 = g^k \ mod \ p = 5^9 \ mod \ 23 = 11 \qquad and \qquad c_2 = m{\cdot}h^k \ mod \ p = 15{\cdot}8^9 \ mod \ 23 = 20$$

Once Bob gets it back, he can decrypt it by first computing $c_1^b = g^{kb} \ mod \ p = 11^6 \ mod \ 23 = 9$ (the value k chosen by Alice), and then with the values of k,c_2 and his public key h:

$$m = c_2 \cdot (h^k)^{-1} \bmod p = 20 \cdot (8^9)^{-1} \bmod 23 = 20 \cdot 9^{-1} \bmod 23 = 20 \cdot 18 \bmod 23 = 15$$

Therefore, we find that Bob has decrypted the correct message!
This encryption scheme is considered safe based off a few assumptions:

1. Like Diffie Hellman, ElGamal makes use of the difficulty of the DLP.
2. It also assumes that it is safe under the CDH.
3. Laslty, it considers itself safe under the Decisional Diffie Hellman (DDH) assumption.[1]

However, the issue here remains the same; these assumptions all fall to the ground if DLP is broken since both CDH and DDH depend on DLP.

5.2 Quantum Implications

It should be rather clear by now... a quantum computer would be able to break ElGamal exactly the same way it breaks DLP and Diffie Hellman: with Shor's algorithm to compute discrete logarithms. If an attacker sees Bob's public key $h = g^b \ mod \ p$, they would simply need to run Shor's algorithm to recover the private key $b = log_g(h) \ mod \ p$. Once the attacker knows b, decrypting the cipher texts become easy since they now know everything Bob knows. The attacker would then just run the same steps seen in the previous section using (c_1, c_2) and b in order to decrypt the message. Any communication encrypted with classical ElGamal could be thus decrypted by a powerful enough quantum computer, which is why moving onto post-quantum encryption schemes is essential.

[1] if someone is given (g^a, g^k, z), they wouldn't know if $z = g^{ak}$ or if z is just random. It is DDH that makes sure that c_2 really hides the message under $m \cdot h^k$

Chapter 6

Conclusion and PQC

6.1 Post Quantum Cryptography

Post-quantum cryptography (PQC) wishes to replace current systems with new algorithms that remain secure even against quantum attacks. PQC protocols don't necessarily require quantum hardware in order to be implemented. This cryptography simply relies on mathematical problems which are considered to be safe (for now at least) from both classical and quantum attacks. A few schemes are already being tested such as some lattice-based schemes[1] (Kyber and Dilithium), some code-based systems[2] as well as some hash-based signatures[3].

The US National Institute of Standards and Technology (NIST) has selected and released standards for several PQC algorithms to replace existing public-key systems. THis is just the beginning of a major transition ensuring long-term security for digital communications.

6.2 Conclusion

As shown throughout this report, schemes such as Diffie Hellman and ElGamal become insecure once a quantum computer can run Shor's algorithm, since their security relies fully on the difficulty of the discrete logarithm problem. This vulnerability has some major implications as was described in the report because many protocols rely on this same assumption. By removing their foundation, they're left with no security. This topic is important and needs attention today as we may not even yet realize the full potential of quantum computers; and once they're here, there's nothing stopping them. The simple fact that so many systems rely on this one assumption that can be broken so easily says a lot about our quantum readiness.

[1] uses multi-dimensional grids of points as the foundation for encryption and digital signatures
[2] based on the difficulty of decoding error-correcting codes
[3] relies on the security of one-way hash functions

Bibliography

[1] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. Relevant sections: Section 5.4.2 (p. 238), Section 10.10.2 (p. 432), and Appendix 5 (p. 640).

[2] A. M. Childs. *Lecture Notes on Quantum Algorithms - Chapter 5*. Available at: https://www.cs.umd.edu/~amchilds/qa/qa.pdf.

[3] A. Mandl. *Quantum Algorithms for the Discrete Logarithm Problem*. Bachelor's Thesis, TU Wien, 2021. Available at: https://repositum.tuwien.at/bitstream/20.500.12708/18830/1/Mandl%20Alexander%20-%202021%20-%20Quantum%20algorithms%20for%20the%20discrete%20logarithm%20problem.pdf.

[4] Lecture Notes:
Lecture 10 — Quantum Fourier Transform;
Lecture 11 — Quantum Phase Estimation;
Lecture 12 — Shor's Algorithm.

[5] W. Diffie and M. E. Hellman. "New Directions in Cryptography," *IEEE Transactions on Information Theory*, IT-22(6): 644–654, 1976. Available at: https://ieeexplore.ieee.org/document/1055638.

[6] ChatGPT (OpenAI). Used as a supplementary explanation tool for clarifying difficult concepts and verifying intermediate steps where needed.

[7] *Diffie–Hellman Key Exchange*. Available at: https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange.

[8] *Diffie–Hellman Key Exchange and Perfect Forward Secrecy*, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/computer-networks/diffie-hellman-key-exchange-and-perfect-forward-secrecy/.

[9] *Discrete Logarithm Problem Explained*, YouTube. Available at: https://www.youtube.com/watch?v=za9azzh4v9A.

[10] *Shor's Algorithm and Discrete Logs*, YouTube. Available at: https://www.youtube.com/watch?v=yKNTfeqSEhc.

[11] *ElGamal Encryption Algorithm*, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/computer-networks/elgamal-encryption-algorithm/.

[12] *DDH*, Available at: https://en.wikipedia.org/wiki/Decisional_Diffie%E2%80%93Hellman_assumption.

Appendix

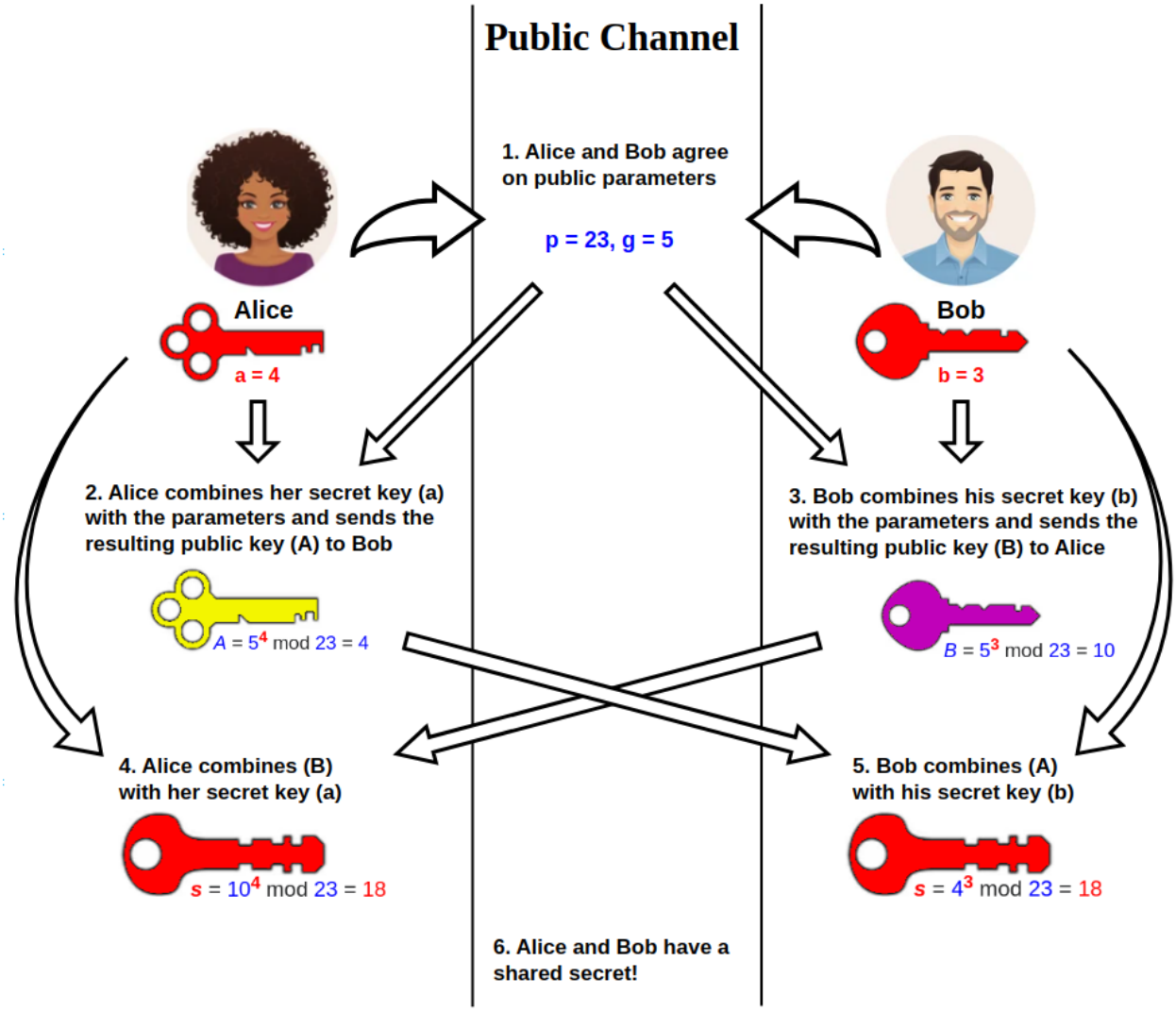


Figure 6.1: Diffie Hellman

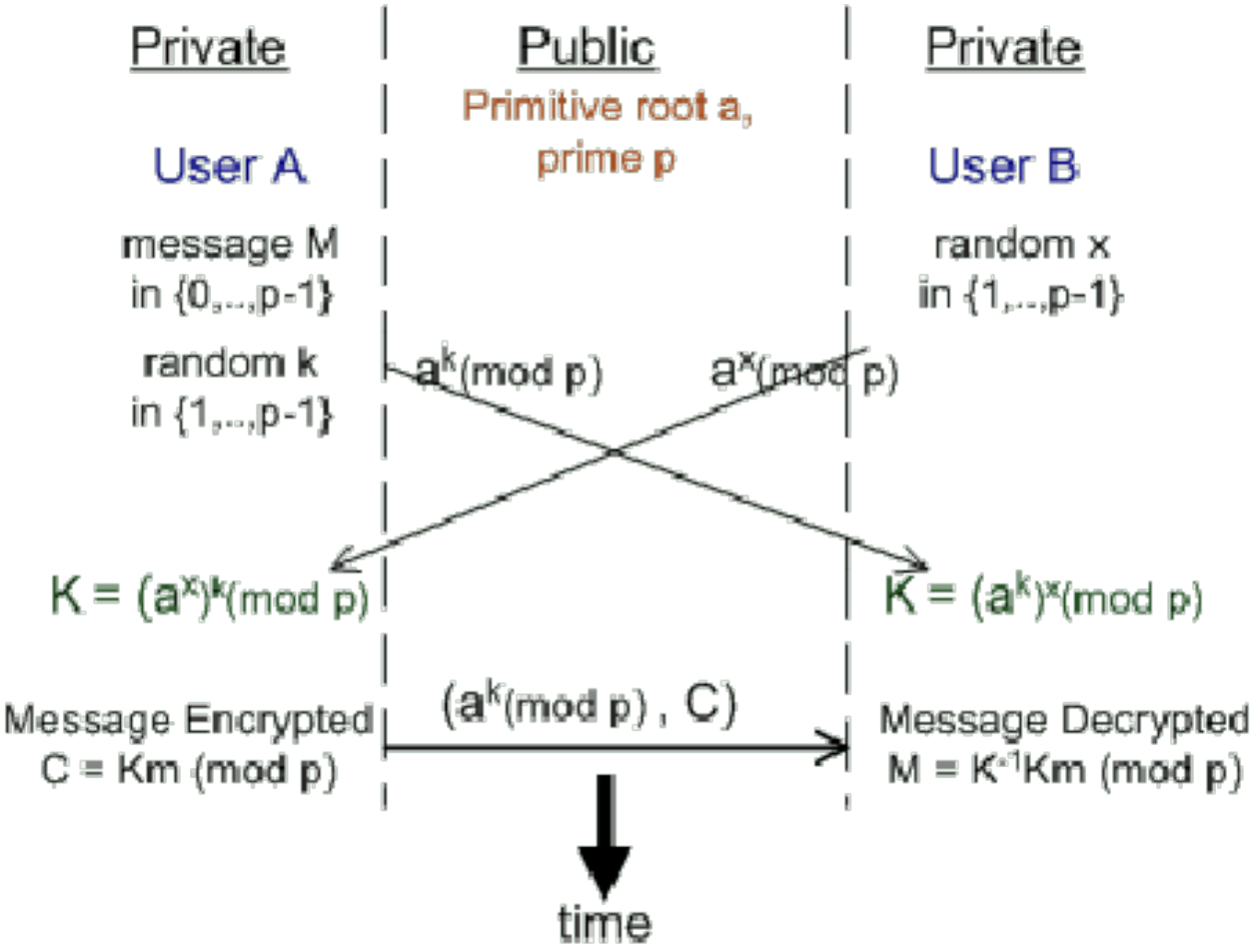


Figure 6.2: ElGamal