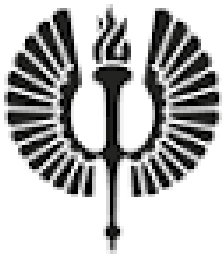

UNIVERSITY OF TURKU

# The Impact of Quantum Computing on Cryptographic Protocols

**Millie Adamson**

**Msc Thesis**

**Supervisors:**
**Professor Valtteri Niemi**
**Professor Ion Petre**

June 27, 2025

DEPARTMENT OF MATHEMATICS AND STATISTICS

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

ADAMSON, MILLIE: The Impact of Quantum Computing on Cryptographic Protocols
MSc Thesis, 52 pages
Mathematics
June 2025

---

This thesis explores the impact of quantum computing on the security of e-voting, using Estonia's e-voting system as a case study. Chapters 1 and 2 provide insight into the development of quantum computing and why this threatens contemporary cryptography. Chapter 3 reviews the current e-voting protocol used in Estonia. Chapter 4 considers the challenges and urgency of migrating to post quantum e-voting. Then, chapters 5 and 6 explore potential quantum safe alternatives to contemporary cryptography, in particular exploring the use of lattice based hard problems. Chapters 7 and 8 provide insight into proposed post quantum solutions specific to e-voting and the current challenges faced with their implementations.

Keywords: cryptography, post-quantum cryptography, Shor's Algorithm, e-voting, ML-KEM.

# Contents

# List of Figures

# 1 Introduction

Quantum computing is an emerging technology which poses an interest to the cryptography community due to the potential threat it poses to modern cryptosystems.

Quantum computing utilises two key principles of quantum mechanics; superposition and entanglement. Whilst a classical computer uses the binary bits 0 and 1, quantum computers use qubits. Similar to classical bits, qubits can take the value 0 or 1, however they are able to exist in a superposition, which is a feature of quantum mechanics. This means that they can exist in a 0-state or a 1-state, or in a state that is a linear combination of these. As a result, we have a much larger problem space and thus computation is sped up exponentially [1].

When a qubit is measured, it will collapse into a basis state, i.e., either to a 0-state or a 1-state. The concept of entanglement means that two or more qubits may be connected in a way that measuring one effects the measurement of the other, irrespective of the physical distance between them, aiding the transfer of qubits in the network [1]. These two ideas will be discussed more concretely in Subsection 2.2.

Given that many cryptosystems widely used in practice, such as RSA, are reliant on the difficulty of solving a particular problem, it is imperative to explore how quantum computers will effect the difficulty of such problems. In particular, the two most common difficult problems that public key cryptosystems use are factorisation of large numbers and discrete logarithms. In 1994, Peter Shor demonstrated a quantum algorithm which can solve these two problems quickly [2]. This threatens the security of the current most popular public key cryptosystems which, as we previously mentioned, are reliant on the difficulty of these two problems.

The idea of quantum computers date back to the 1960's and even Shor's algorithm is over thirty years old. Considering this timeline, there has been considerable progress more recently. In the year 2025, quantum computers remain unavailable to the general population and are confined to research facilities. There remains some significant challenges in quantum computing such as addressing error rates however, the area has also seen some major growth. For example there has been progress in quantum software, though this remains a new area [3]. Due to the signs of progress in quantum computing as well as the severity of potential impacts to cryptography, we have seen a push in the cryptography community to transition to post-quantum cryptography. That is, cryptographic algorithms that are thought to be secure against attacks that utilise quantum computers. Currently, the National Institute of Standards and Technology (NIST) [4] are leading an effort to standardise post-quantum cryptography algorithms [1]. These algorithms aim to utilise mathematical concepts thought to be safe against quantum attacks such as lattice or code based cryptography [3].

Having first established a basic understanding of quantum computing and why it is impactful to cryptography, the remainder of this thesis will aim to review the impact

of quantum computing on some cryptographic protocols. Firstly, we will motivate the importance of post quantum cryptography by exploring Shor's Algorithm. In particular, we will look at how the algorithm works, what it achieves, and how it impacts modern cryptosystems such as RSA and Diffie-Hellman.

Then we will consider the impacts of this realisation in e-voting, using the e-voting protocol used in Estonia as a case study. We aim to explore the current protocol and how quantum computing may threaten the security of it. We will also explore what a potential transition to a quantum safe protocol may entail.

# 2 Preliminaries

We will now establish some preliminaries relevant for understanding the later discussions. In particular, we will discuss Dirac notation, qubits, Fourier transforms, Shor's algorithm and some security concepts.

## 2.1 Dirac Notation

The reader needs to understand the Dirac notation used for descriptions in quantum mechanics to be able to follow the discussion. We will briefly discuss it here but Rieffel et al. [5] provide a more detailed explanation.

Dirac notation uses two main notions: the bra and the ket.

**Definition (Ket):** The ket, written $|v\rangle$, refers to a vector representing the state of the quantum system. A vector $|v\rangle$ is said to be a linear combination of $|s_1\rangle, |s_2\rangle, ..., |s_n\rangle$ if there exists coefficients $a_i \in \mathbb{C}$ such that $|v\rangle = a_1|s_1\rangle + a_2|s_2\rangle + ... + a_n|s_n\rangle$. If a basis is fixed, e.g. consisting of two states $|\beta_1\rangle, |\beta_2\rangle$, then the ket state can alternatively be expressed by the coefficients, e.g. $|v\rangle = a|\beta_1\rangle + b|\beta_2\rangle$ can be written as $\begin{pmatrix} .a & b \end{pmatrix}^T$ [5].

**Definition (Bra):** The bra is written as $\langle v|$ and it is the conjugate transpose of the ket. Taking the transpose means to switch the rows and columns of a matrix. The conjugate of a complex number $a + ib$ (where $i$ is $\sqrt{-1}$) is $a - ib$. Thus we have $|v\rangle = \begin{pmatrix} a_1 & . & . & . & a_n \end{pmatrix}^T$ and $\langle v| = \begin{pmatrix} \bar{a_1} & . & . & . & \bar{a_n} \end{pmatrix}$ [5].

**Definition (Bracket/Inner Product):** The inner product is written as $\langle a \,|\, b\rangle = \begin{pmatrix} \bar{a_1} & . & . & . & \bar{a_n} \end{pmatrix} \cdot \begin{pmatrix} b_1 & . & . & . & b_n \end{pmatrix}^T$ and is often referred to as the bracket [5].

Thus we have seen that a ket represents a quantum state, a bra represents an operation applied to the quantum state and the result of such an operation is represented by the bracket.

## 2.2 Qubit

The state space refers to the set of all possible states of a physical system and qubits are used to model a quantum system. In quantum computing there are infinitely many possible states, all of which can be represented as a linear combination of two chosen basis states [5].

Classical computers function using bits. In the context of quantum computing, we encode the classical bits of 0 and 1 as the basis states $|0\rangle$ and $|1\rangle$ which are orthonormal. In addition to the basis states, there are infinitely many possible states represented as a linear combination of $|0\rangle$ and $|1\rangle$. That is, there are states written as $|v\rangle = a|0\rangle + b|1\rangle$ where $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$. This state is called a superposition [5].

Consider $|v\rangle = a|0\rangle + b|1\rangle$ where $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$. Upon measurement, the state $|v\rangle$ collapses into one of the basis states $|0\rangle$ and $|1\rangle$. The probability that $|0\rangle$ is measured is $|a|^2$ and the probability that $|1\rangle$ is measured is $|b|^2$. Thus we see that quantum measurement limits the information we can obtain from a qubit to the equivalent of just one classical bit. However prior to measuring, since computation can be done on a superposition, counterparts to multiple classical computations can be carried out simultaneously. Thus, we can speed up computation [5].

Another key feature of a qubit involves the concept of entanglement. Entanglement describes a correlation between quantum particles. Quantum particles can be described by a wave function. We say that two quantum particles or quantum systems are entangled if they cannot be separated into individual wave functions. This means that given one wave function that describes their correlation, this wave function cannot be expressed as the product of individual wave functions where each individual wave function represents one of the quantum particles. The result of entanglement means that, regardless of the physical distance between the two particles, a measurement on one particle affects the other [6].

In terms of the role of entanglement in quantum computing, it enables fast computation. This is because it reduces the number of measurements required since a measurement on one qubit has the same impact on others that it is entangled with. This notion of computing multiple calculations at the same time is known as quantum parallelism [6].

## 2.3 Quantum Fourier Transformation

Before discussing Shor's Algorithm, first we should explore the quantum Fourier transformation which is utilised in quantum computing to speed up classical computing algorithms.

The Fourier transform takes as an input a function of time whilst the output is a function of frequency. We will define the discrete Fourier transform.

**Definition (Discrete Fourier Transform):** The discrete Fourier transform is a mapping from a function $a : [0, ..., N-1] \mapsto \mathbb{C}$ to another function $A : [0, ..., N-1] \mapsto \mathbb{C}$ using the rule

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{(2\pi \mathbf{i} \frac{kx}{N})} \quad \text{[5].}$$

**Example (Discrete Fourier Transform):** Suppose we choose $N = 3$. We also choose $a(0) = i$, $a(1) = 1 - i$ and $a(2) = 2$ (where $i = \sqrt{-1}$). Then using the above equation, we have:

$$A(x) = \frac{1}{\sqrt{3}} \sum_{k=0}^{2} a(k) e^{(2\pi \mathbf{i} \frac{kx}{3})}$$

$$= \frac{1}{\sqrt{3}} (a(0) + a(1) e^{(2\pi \mathbf{i} \frac{x}{3})} + a(2) e^{(4\pi \mathbf{i} \frac{x}{3})})$$

.

Then using our chosen values of $a(0), a(1)$ and $a(2)$, we obtain:

$$A(0) = \frac{1}{\sqrt{3}} (i + (1 - i) + 2)$$

$$= \sqrt{3},$$

$$A(1) = \frac{1}{\sqrt{3}} (i + (1 - i) e^{(\frac{2\pi \mathbf{i}}{3})} + 2 e^{(\frac{4\pi \mathbf{i}}{3})})$$

$$= (\frac{1}{2} - \frac{\sqrt{3}}{2}) + (-\frac{1}{2} + \frac{\sqrt{3}}{2}) i,$$

$$A(2) = \frac{1}{\sqrt{3}} (i + (1 - i) e^{(\frac{4\pi \mathbf{i}}{3})} + 2 e^{(\frac{8\pi \mathbf{i}}{3})})$$

$$= (-\frac{1}{2} - \frac{\sqrt{3}}{2}) + (\frac{1}{2} + \frac{\sqrt{3}}{2}) i,$$

where we have used the identity $e^{i\theta} = cos(\theta) + isin(\theta)$ [7].

**Definition (Quantum Fourier Transform):** The quantum Fourier transform, denoted $QFT$, is defined in the following manner. Note that in the following, it assumes that N can be written as $N = 2^n$:

$$QFT : |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{(2\pi \mathbf{i} \frac{yx}{N})} |y\rangle \quad [8].$$

In contrast to the classical discrete Fourier transform whose computation has exponential complexity $O(n2^n)$, the quantum Fourier transform can be done in $O(n^2)$ time units, thus we see an exponential speed up [5].

## 2.4 Shor's Algorithm

Having established the relevant background, we can now look at how Shor's algorithm works. Shor's algorithm aims to utilise the exponentially faster quantum Fourier transformation as well as the idea of reducing a problem to another problem. We will now discuss how the algorithm works for factorisation of an integer.

**Algorithm (Shor's Algorithm for Factoring):** Suppose we wish to factor an integer $N$:

1) Choose randomly a guess $m$ such that $0 < m < N$. Apply Euclid's algorithm to determine $\gcd(N, m)$. If $\gcd(N, m) \neq 1$ then we have found a factor of $N$ and we are done [5]. Otherwise, move to step 2. (Note that it is highly unlikely that $\gcd(m, N) \neq 1$, especially for large $N$ [5]).

2) We transform $m$ into a better guess. Note that number theory tells us that, $m^p = 1 \mod N$ for some integer $p$, though $p$ is unknown at this stage. Thus we can write, $m^p - 1 = 0 \mod N$. If $p$ happens to be even, then we have, $(m^{\frac{p}{2}} - 1)(m^{\frac{p}{2}} + 1) = 0 \mod N$. Now we could choose $(m^{\frac{p}{2}} - 1)$ and $(m^{\frac{p}{2}} + 1)$ as the new guesses. If neither of the new guesses is a factor of $N$, then they must be multiples of factors of N. In this case the factors can easily be found using Euclid's algorithm. Hence, providing that neither of our new guesses are a multiple of $N$ and that $p$ is not odd, we can factorise $N$. Go to step 3.

3) We note that for an integer $x$, we may compute $r = m^x \mod N$. Then we have $m^{x+np} = r \mod N$ where $n \in \mathbb{Z}$ since $m^p = 1 \mod N$. Hence, we see that when considering the inputs $x, x+1, x+2, ...$of the function, the output of the function $f(x) = m^x \mod N$ repeats with the order of $p$ being the period. Thus, we have successfully reduced the factorisation problem into the problem of finding the period of a function. By utilising the capabilities of a quantum computer, we can take as an input a superposition and use quantum parallelism (the ability to perform computations simultaneously) to compute the possible values of $f(x) = m^x \mod N$ for each $x \in \{0, ..., 2^n - 1\}$ where $n$ is such that $N^2 \leq 2^n < 2N^2$ (Rieffel et al. explain why these possible values of $x$ are sufficient [5]). Then if $r$ is a result, it repeats with period $p$. Now we can apply the quantum Fourier transform [5]. Go to step 4.

4) A measurement can now be taken and the output can be processed by a classical computer in such a way that we can obtain the desired period $p$. Rieffel et al. provide more details of how this quantum core and processing of the output occurs [5]. Go to step 5.

5) As long as $p$ is even we have as guesses $m^{\frac{p}{2}} + 1$ and $m^{\frac{p}{2}} - 1$. As long as these guesses are not a multiple of $N$, then they share factors with $N$ and $N$'s factors can be found using Euclid's Algorithm. If these conditions do not hold, repeat from step 1 [5].

**Example (Shor's Algorithm):** Suppose we wish to factorise 77.

Choose $m = 9$ and note that $\gcd(9, 77) = 1$. Note $77^2 = 5929$ and $2 * 77^2 = 11858$ so $5929 \leq 8192 = 2^{13} < 11858$, thus we choose$n = 13$.

Compute $9^x \mod 77$ for $x \in \{0, ..., 8191\}$. In doing so we get the output 1, 9, 4, 36, 16, 67, 64, 37, 25, 71, 23, 53, 15, 58, 60, 1, 9, 4,... and we see that the series has already began to repeat itself. In a quantum implementation this would not be measured directly but rather a quantum Fourier transform would be applied and then processing would be carried out to obtain the period of the function. In this

simple example it is easy to see that $p = 15$ which is odd so we shall repeat from step 1.

Choose $m = 3$ and note $\gcd(3, 77) = 1$.

Compute $3^x \mod 77$ for $x \in \{0, ..., 8191\}$. On a classical computer this yields: 1, 3, 9, 27, 4, 12, 36, 31, 16, 48, 67, 47, 64, 38, 37, 34, 25, 75, 71, 59, 23, 69, 53, 5, 15, 45, 58, 20, 60, 26, 1, ... Again, it is easy to see in this example that $p = 30$ which is even thus we have guesses $(3^{\frac{30}{2}} - 1) = 33$ and $(3^{\frac{30}{2}} + 1) = 35$.

Applying Euclid's algorithm, $\gcd(33, 77) = 11$ and thus we have 11 is a factor of 77. Similarly, $\gcd(35, 77) = 7$ and thus we have 7 is a factor of 77.

Note that when working with practically important implementations numbers would be much larger. As a result at the step where we compute $9^x \mod 77$ for $x \in \{0, ..., 8191\}$, the number of possible values of $x$ would be significant. The ability of a quantum computer to carry these computations out simultaneously in a single step allows for efficient implementation. However, since we are working with small numbers in this instance, running all computations on a classical computer is feasible.

Shor also demonstrated the ability of a quantum computer to solve the discrete logarithm problem. Though we omit the details of it here, we will provide an intuitive idea as to how it works. A more rigorous explanation can be found in Shor's paper [2].

The aim of the discrete logarithm problem is to find $r$ such that $g^r = x \mod p$ where $p$ is prime. The algorithm uses the expression $g^a x^{-b} \mod p$. We note that $g^a x^{-b} \mod p = g^a g^{-rb} = g^{a-rb} \mod p$. Then we note that, similar to in the factoring algorithm, this expression is periodic. Using this fact, we know there must exist some $g^{a+w_1} g^{r(-b+w_2)}$ such that $g^a g^{-rb} = g^{a+w_1} g^{r(-b+w_2)}$. Then we have that $a - rb = a + w_1 - rb + rw_2 \mod p - 1$. (Note we use $\mod p - 1$ since $a^{p-1} = 1 \mod p$). Simplifying this gives $-w_1 = rw_2$ which implies that $r = -w_1 w_2^{-1}$. Thus we see how we can find $r$ if we know $w_1$ and $w_2$ and if an inverse exists for $w_2$ [2]. The values $w_1$ and $w_2$ can be found by using the quantum Fourier transform to extract the period. This idea is similar to that of the factoring algorithm which we described previously.

Thus, we see that cryptographic algorithms based on factorisation and discrete logarithms may be compromised in the future if the progress in increasing the number of qubits in a quantum computer continues. This is significant given that RSA, DH and elliptic curve cryptography are widespread in current security and based on these problems. Thus we see the motivation for a transition to post quantum cryptography.

## 2.5 Security Concepts

We will now describe the terms used to describe the security level of both encryption systems and signature schemes.

An encryption scheme takes some unencrypted information (plaintext) as an input and encodes it to output some encrypted information (ciphertext). Transforming the plaintext to ciphertext is called encryption and transforming the ciphertext to plaintext is called decryption. We will now discuss encryption schemes more formally in the context of public key cryptography [9].

A public key encryption scheme consists of a psuedorandom key generation algorithm with generates a public key, $k_{pub}$, and a private key, $k_{priv}$. Also, there is an encryption algorithm which takes as an input some plaintext message, $m$ and the public key, $k_{pub}$. It then outputs some ciphertext, $y$. The encryption algorithm can be denoted by $Enc$. Thus, $y = Enc(m, k_{pub})$. The decryption algorithm takes as an input some ciphertext, $y$ and the private key, $k_{priv}$, and outputs the plaintext message, $m$. We denote the decryption algorithm by $Dec$. Thus, we have $Dec(Enc(m, k_{pub}), k_{priv}) = m$ [9].

Having defined a public key encryption scheme we can now describe some security terms. One important note is that the encryption system in the following discussion must be non-deterministic in nature meaning that encrypting the same message twice will output different ciphertexts. Otherwise the adversary could use the public key to encrypt both messages and compare them to the challengers ciphertext to determine which message was encrypted [9].

We first describe the term IND-CPA which describes the security of some encryption systems. This concept can be defined in terms of a 'game'. The game involves an adversary and a challenger. The challenger has full knowledge of the encryption system whilst the adversary aims to break the cryptosystem with only access to the public key and ciphertext [9]. The 'game' can be described as follows.

**Definition (IND-CPA):** The challenger generates a public and private key pair, $(k_{pub}, k_{priv})$, where $k_{priv}$ is the private key and $k_{pub}$ is the public key. The public key is sent to the adversary. The adversary may then choose two plaintexts of the same length that they wish to encrypt. We will call these plaintexts $m_0$ and $m_1$. These plaintexts are sent to the challenger. Then the challenger picks at random a bit $b \in \{0, 1\}$ to determine which plaintext message they will encrypt. They then encrypt $m_b$ to get $y = Enc(m_b, k_{pub})$ and send $y$ to the adversary. The adversary then tries to guess which message was encrypted based only on the ciphertext and public key [9]. We denote this guess by $b'$. If the adversary can not guess, with probability greater than $\frac{1}{2}$, which message was encrypted we say the messages are indistinguishable, denoting this IND [9].

Also, in this game, the adversary may only pick the messages to be encrypted. This feature of the game reflects a chosen plain text attack, denoted CPA. Thus, using

the above 'game' where the adversary is not able to determine which plaintext has been encrypted, the encryption system is described to be IND-CPA secure [9].

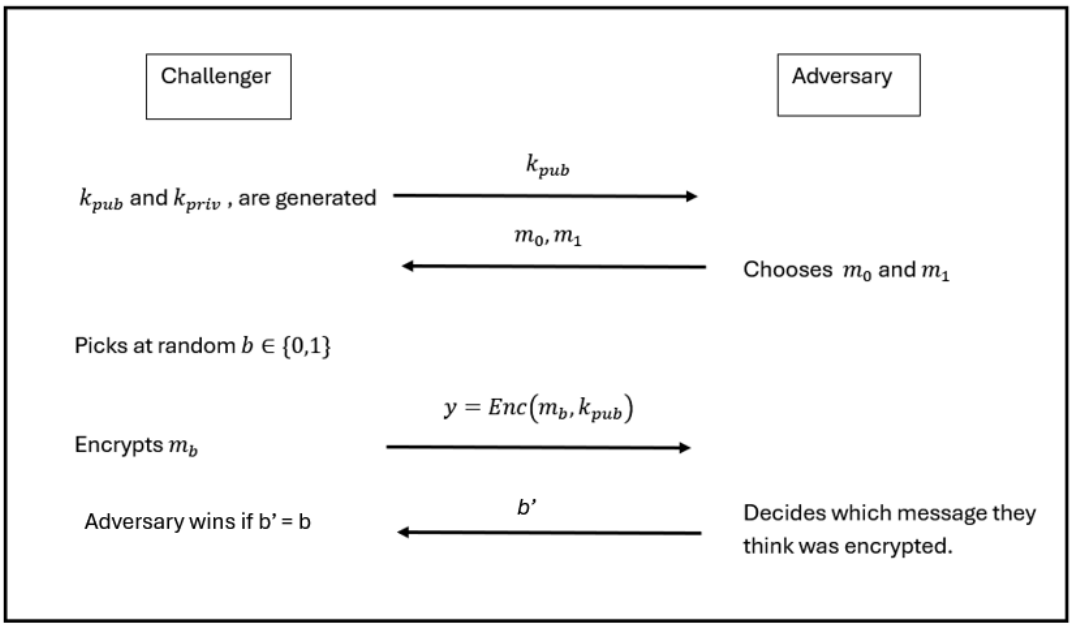


Figure 1: IND-CPA 'game'. This figure is based on a figure in [9].

**Definition (IND-CCA1):** Now we will describe the term IND-CCA1. Suppose we have the following 'game'. The challenger generates public and private keys, $k_{pub}, k_{priv}$. These keys are sent to the adversary. The adversary is then able to use a decryption machine as they like. Then, they must suspend access to the decryption machine, Following this, the adversary chooses two plaintexts to be decrypted. We call these $m_0$ and $m_1$. These plaintexts are sent to the challenger. Then, the challenger chooses at random a bit, $b \in \{0, 1\}$, and encrypts the corresponding plaintext, $m_b$, to generate the ciphertext, $y = Enc(m_b, k_{pub})$. This is sent to the adversary who must now guess which of the plaintexts was encrypted. Based on this 'game', if the adversary is unable to guess which plaintext was encrypted with probability greater than $\frac{1}{2}$, then the encryption system is said to be IND-CCA1 secure (where CCA refers to a chosen ciphertext attack) [9].

**Definition (IND-CCA2):** Now we can described the term IND-CCA2. If we consider the game described in the definition of IND-CCA1, IND-CCA2 can be described with a similar game however there is one significant change. Unlike with IND-CCA1, the adversary may now maintain access to the decryption machine even upon receiving the challengers ciphertext $y$, however, the adversary may not use the decryption machine to decrypt $y$. Otherwise, they may use the machine freely. IND-CCA1 is stronger than IND-CPA as the adversary has more options. By the same reasoning, IND-CCA2 is stronger than IND-CCA1 [9].

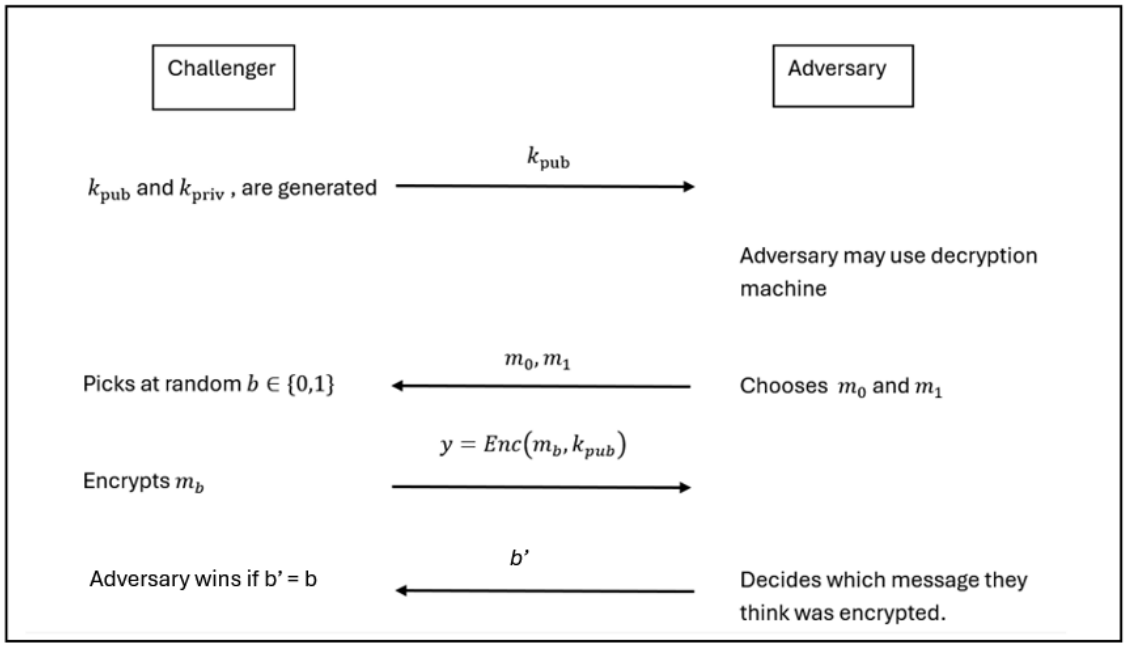


Figure 2: IND-CCA1 'game'. This figure is based on a figure in [9].

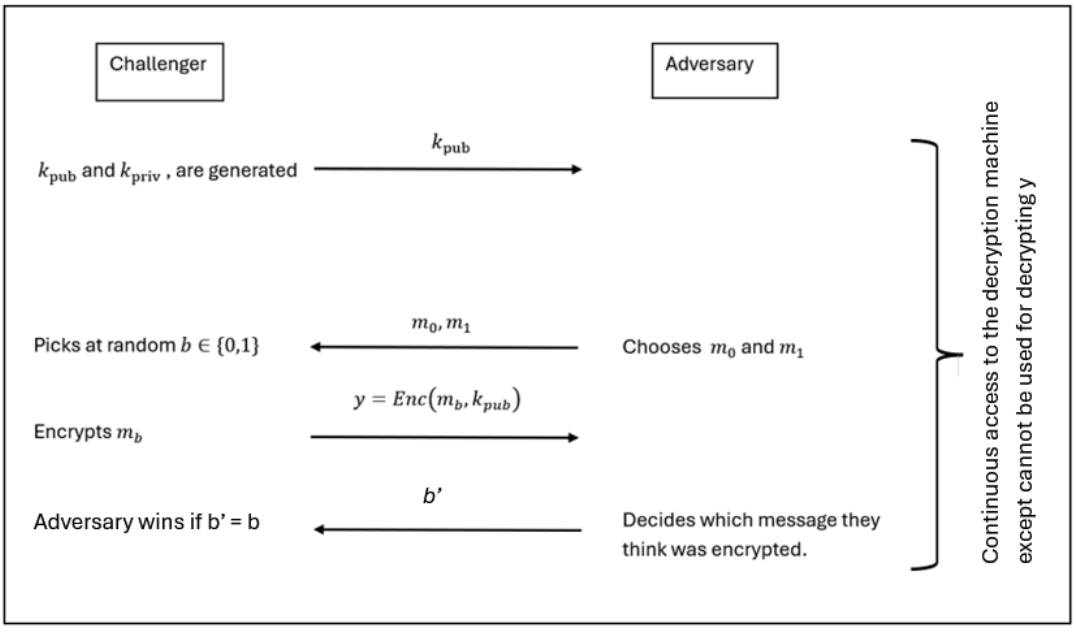


Figure 3: IND-CCA2 'game'. This figure is based on information in [9].

**Definition (EUF-CMA):** The final security term we will define is EUF-CMA. Unlike the above terms which refer to the security of an encryption system, this

refers to the security of a signature scheme.

We will begin by defining a public key signature scheme. This consists of a psuedorandom key generation algorithm which generates the signing key, $k_{sig}$, and the verification key, $k_{ver}$. There is a signature algorithm that takes as an input the signature key, $k_{sig}$, and the message we wish to sign, $m$, and outputs a signature, $u$. That is, the signature algorithm, denoted $Sig$, is defined as $Sig(m, k_{sig}) = u$. Also, there is a verification algorithm which takes as an input the message, $m$, the signature, $u$, and the verification key, $k_{ver}$ and verifies that the signature is correct. That is, the verification algorithm, denoted $Ver$, can be defined in terms of accepting or rejecting the signature with $Ver(m, Sig(m, k_{sig}), k_{ver}) = accept$ and $Ver(m, u', k_{ver}) = reject$ if $u'$ is not equal to $Sig(m, k_{sig})$ [9].

Having now defined a public key signature scheme, we can proceeed with our discussion on signature scheme security.

If a signature is subject to existential forgery then this means it is possible for the adversary to forge the signature on a single message of the adversary's choosing. In an adaptive active attack, the adversary has access to the signature machine, verification machine and the public key. If the adversary is unable to create an existential forgery on a message that they have not passed through the signature scheme then we describe the signature scheme to be EUF-CMA secure (where EUF refers to existential unforgeability and CMA refers to chosen message attack) [10].

In terms of a 'game', suppose we have a challenger and an adversary. The challenger generates the signing key, $k_{sig}$, and the verification key, $k_{ver}$. They send the verification key to the adversary. The adversary is then able to use a signing oracle to produce a signature $u$ for a chosen message $m$. They are able to repeat this process for an arbitrary number of messages. Each message, $m$, becomes an element of the set $A$ where initially $A = \emptyset$. Then, the adversary chooses a message $m^*$ such that $m^* \notin A$ and tries to generate a signature $u^*$ for $m^*$ without using the signing oracle. The message signature pair, $(m^*, u^*)$ is sent to the challenger. The challenger checks the validity of the signature and also checks that $m^* \notin A$. The second check is the verify that the signature was not produced using the signing oracle. If both of these conditions are met, a successful forgery has been produced. If no adversary is able to produce a correct signature with probaility greater than $\frac{1}{2}$, then we say that the signature scheme is EUF-CMA secure [10].

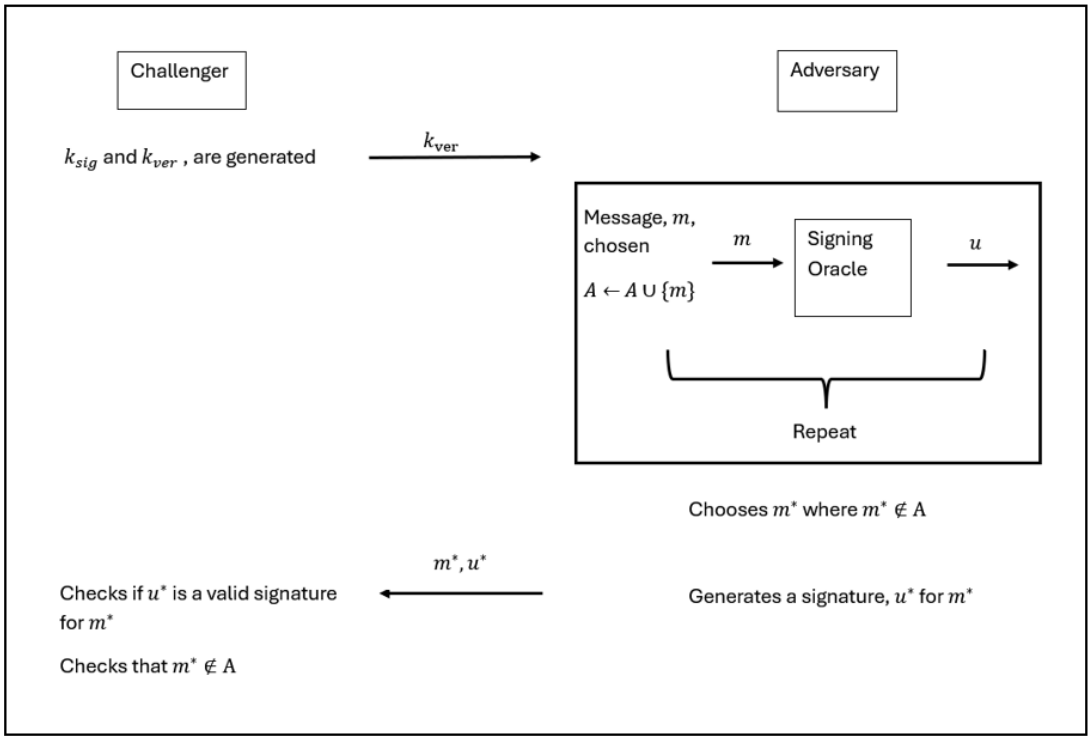


Figure 4: EUF-CMA 'game'. This figure is based on a figure in [10].

# 3 E-Voting

Having seen how quantum computing could impact the future of cryptography in a general sense, we will now explore the impacts to specific cryptographic protocols. We will begin with a focus on the e-voting protocol used in elections in Estonia. Estonia began to offer e-voting in their elections in 2005 [11] and we will use their protocol [12] as a case study due to the fact it is well documented.

## 3.1 Requirements of an E-Voting Protocol

To begin with, we shall note the security requirements of internet voting. The Council of Europe released a document regarding the requirements of e-voting. In particular, in the section relating to security, they note that the e-voting system should be as secure as a traditional, non electronic voting system [13]. As a means to maintain security they outline some requirements.

In the pre voting stage they note the importance of authenticity, integrity and availability of the voter registration as well as the importance of being able to check that candidate nominations, acceptance of nominations and voter registration all happened in the allowed time window [13].

In the voting stage they note the importance of ensuring the integrity of the pre-voting data, that ballots can be authenticated as coming from the official server, and that voting systems are protected. In addition they require that you can check that voters are eligible, votes are made in the allowed time frame, the voters choice is accurate, accounted for and that residual information regarding the vote is destroyed or deleted from the device following the vote [13].

Post voting, the integrity of data from the voting stage should be maintained, the vote counting should be accurate and reproducible, and availability and integrity of ballot boxes and results should be maintained for as long as deemed necessary [13].

## 3.2 IVXV protocol

Having established an overview of the requirements of the protocol, we will outline Estonia's e-voting protocol called the IVXV protocol and identify the cryptographic primitives being used in it.

The IVXV protocol can be separated into three stages: establishing what is needed for the vote, the voting period, and the end of the voting period.

1. Prior to voting lists of relevant information need to be established such as electoral districts and polling stations and possible choices. This is done by the election organiser. Most notably, a list of eligible voters is required. This list contains the voters names and personal ID codes as well as the relevant district number. The list is hashed using SHA256 and then signed using a

2048-bit RSA key [12].

2. During the voting period the voter's make their choice. This is done in the voter application. They use a 'double envelope' method whereby the vote is made then encrypted using the ElGamal cryptosystem. This ensures the confidentiality of the vote. Following this, it is then digitally signed. Before signing, the files to be signed are hashed using SHA256. They are then signed using either an ID card, digital ID or mobile ID. ID cards and digital ID use RSA keys for signing whilst mobile ID uses ECC keys [12].

   Then the voter application sends the encrypted ballot with a signature and a certificate for the signature to the collector application where they use a protocol called OCSP (Online Certificate Status Protocol) [14], to confirm the validity of the certificate. As well as checking the validity of the certificate the collector service checks that the voter is eligible, the vote is in the correct form and the digital signature is right. It can then register the vote with the registration service and store it [12].

   The voter is then able to check that the vote has been successful through the verification application. To avoid voter coercion voters are able to vote repeatedly with the most recent vote being counted. The exception to this is if a vote is done at a polling station. In this instance the in person vote overrides any online votes [12].

   In the processing application, it checks that the registration and collector services are consistent. It also makes the same checks as in the collector service. Furthermore, it determines the most recent vote which will be the vote that counts [12].

3. The processing application outputs an anonymised ballot box with signatures removed to ensure the privacy of the voters [12]. The mixing application can then mix the anonymised votes. The mix net takes the list of ciphertexts as input and performs shuffling, passing from mix server to mix server until the final output is given. This output is a shuffled list of votes. The purpose of the mix net is to shuffle the votes in such a way that votes can't be traced back to the voter [11].

   Then the ballots are sent to the decryption application which has the private keys necessary for decryption. The decryption application decrypts and counts the votes omitting votes that are invalid. For example, a vote may be deemed invalid if the vote does not match any of the eligible candidates on the list. The votes are sorted by electoral district and polling stations. The final result contains, for each polling station, the number of spoiled ballots as well as the

number of votes for each candidate [12].

The auditers may now check the correctness of the process. They check the shuffling using Verificatum [15] (meaning they have zero knowledge proof that the input of the mixing and output of the mixing decrypt to the same thing) as well a checking the decryption proof [12].

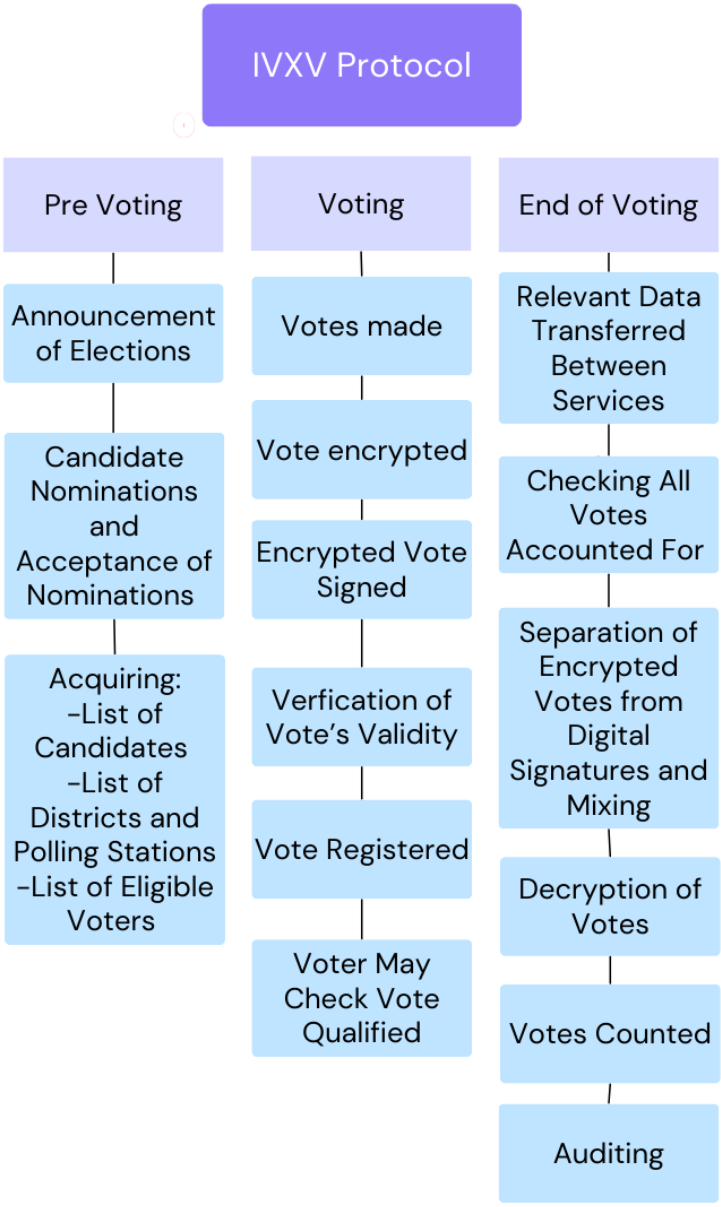


Figure 5: Election Process. This figure is based on information in [12].

## 3.3 Impact of Quantum Computing on IVXV

Further information of the protocol can be found in the official documentation [12] but here we note the cryptographic primitives in use. Most notably is the use of ElGamal for encryption. The security of this algorithm is based on the difficulty of the discrete logarithm problem. We have previously seen that Shor's algorithm threatens such security. This is an immediate and obvious flaw to the current IVXV protocol.

Another aspect of the IVXV protocol that could be vulnerable to a quantum attack is the signing of the votes. As noted previously, the votes are signed by either digital ID or ID cards which use RSA keys, or by mobile ID which uses ECC keys [12]. Both RSA and ECC are not quantum safe. Thus, they are a potential flaw.

# 4 Post Quantum Transition

In this section we will consider the transition to post quantum cryptography. This includes the urgency of the transition, how this transition may occur and challenges facing such as transition.

## 4.1 Urgency of Transition

First we consider the urgency of a transition to post quantum cryptography in the general sense and then we will explore it in the specific context of e-voting.

Joeseph et al. [16] advocate for a pro-active approach to a post quantum transition. They advocate for organisations to, at minimum, begin planning for a post quantum transition or even implementing it. They acknowledge that harvest now decrypt later (whereby data is collected now and when advances are made in quantum technology the data is decrypted) may not be a threat in all circumstances. For example, in contexts where the secrecy of data is only necessary for a relatively short time. However, they argue that certain data is already threatened by harvest now decrypt later. They note medical data and national security data requires long term security and thus they argue against a delayed transition [16].

On the other hand, Joseph et al. acknowledge the importance of a high quality standardisation process. They note that the focus of standardisation agencies such as NIST should be on security and ensuring clarity for organisations adopting the standards rather than speed [16].

Another point provided by [16] regarding the urgency of transition is with regard new long term projects. They note that some products which use cryptographic protocols have a long life span, e.g. vehicles. As a result such products that are being developed now should be designed with quantum threats in mind. They note that some products require specific hardware and software for compatibility purposes and thus transitioning an existing product may not be as simple as replacing the cryptographic module. Thus planning new products with this in mind is important [16].

In terms of the urgency of a transition to post quantum e-voting, Rodriguez-Perez et al. [17] aim to explore different stakeholders' perceptions of the threat of quantum computing to e-voting through a series of interviews with experts such as cyber security professionals and election officials.

The article notes the threat of harvest now, decrypt later. They claim that even violating the privacy of votes cast years ago could have both political and personal implications and thus this threat should be considered when determining the transition urgency. However, during the interviews there was disagreement with regard to the issue of voter privacy. Some experts argued that the expectation of voter privacy is limited to during the election process whilst others argued for the impor-

tance of long-term voter privacy. Even among those arguing the latter, their view on how long privacy should be maintained varied. This varied from several elections to beyond a voter's death [17]. Thus, based on this we see there is no clear consensus among experts on the security requirements of a e-voting system. Additionally, most experts did not see decrypt later as a significant threat citing that this threat isn't limited to the development of quantum technologies but also remains a threat in the instance that a classical algorithm is found to be vulnerable [17].

In terms of a potential mitigation, the experts were in favour of data deletion. Whilst the experts recognised the inability to confirm all data was deleted, especially when using cloud services, they argue that it not only provides a means of mitigation but also increases trust in the system [17].

Overall, they conclude that transitioning towards quantum safe e-voting is a long-term ideal but they do not feel that there needs to be an imminent practical implementation. Thus, they claim that post quantum research should be the current focus [17].

## 4.2 Hybridisation

Hybridisation is a suggested approach for transitioning to quantum safe protocols. It involves using both quantum safe algorithms and classical algorithms together to enhance security. (Note in this context, we use the term classical to refer to modern cryptography widely used in classical computing). This allows for the benefit that classical algorithms are time-tested and have been extensively studied. However, it also provides security against quantum computers given the use of quantum safe algorithms [16].

In 2024, Cybernetica published a paper noting their efforts in transitioning to a quantum safe e-infrastructure in Estonia. Whilst it was largely written from an engineering perspective, they note that their efforts thus far have been only on implementing quantum safe measures. However, they also state their intention to hybridise by adding back in time tested classical cryptography at a later stage [18].

From the aforementioned paper interviewing experts [17], when questioned on the importance of hybridisation, there was a division between stakeholders. On one hand, the standardisation agencies and post quantum cryptography experts argued for a hybrid approach. On the other hand, other experts argued against it in the context of e-voting owing to the complexity of such systems and citing that hybridisation is not easily supported by zero-knowledge proofs and mix nets which are key features of e-voting protocols [17].

In terms of how the transition should look, experts supported the idea of taking note of the current cryptographic primitives in use and identifying those that need to be upgraded. Most experts argued for a lattice based transition [17]. We will explore in Section 5 potential post quantum lattice based alternatives to current

cryptography.

## 4.3 Approach to a Post Quantum Transition

In a 2024 paper published by Cybernetica [19], they consider a quantum safe transition. They note three key phases to the process. This involves firstly identifying the vulnerable cryptographic primitives in use, then developing a plan for migrating to quantum safe alternatives and finally carrying out this plan. With regard to this process, they refer to an ETSI document [20] which provides a check list for migration. We note that this checklist is made for a general transition to post quantum cryptography. We will later explore the challenges posed in the specific context of Estonia's e-voting landscape.

The first stage of migration involves determining the risks, requirements and current cryptography in use. For example, with regards to determining risk, possible questions that need to be asked include the severity of the impacts in the event of exposed data. This could be financial or legal impacts, or it could be impacts to the public image of the organisation. The requirements could include how long the data needs to be protected and whether or not data is deleted or stored after use. Finally, with regard to the current cryptography, possible questions include what cryptographic libraries are being used, what cryptographic hardware is being used, and what is the lifetime of the cryptographic keys that are used [20].

In terms of planning the migration, considerations must be made. This includes, determining whether suppliers have migrated, or are planning to migrate to post quantum cryptography. Also, in the planning stage it is important that product testing is carried out and that compliance with regulations is met [20].

For the execution of migration, considerations include ensuring reponsibilities are clear. This includes determining who is responsible for the transition and who is part of the team enabling the transition. Also, when executing the migration process, the budget should be considered [20]. For a more detailed checklist of questions to be asked in the migration process, the reader can refer to the ETSI document [20].

## 4.4 Challenges of a Post Quantum Transition

We will now consider the work of Vakarjuk et al. [19] which provides insight into the challenges posed by a post quantum transition in the context of Estonia's e-voting system.

One notable challenge is in relation to the ID cards used. An ID card is a constrained device meaning that it is limited in its resources. This poses challenges to a post quantum transition. For example, Vakarjuk et al. [19] note that ID cards use two key pairs, each with a certificate associated with them. One key pair is used for signatures and the other key pair is used for authentication and decryption. Vakarjuk et al. compare the key sizes for various algorithms that provide approximately

128 bit security. Notably, RSA3072 has a public key size of 400 bytes, a private key size of 384 bytes and a signature size of 384 bytes. In contrast, Dilithium2 has a public key size of 1312 bytes, a private key size of 2528 bytes and a signature size of 2420 bytes [19]. From this we see that the key size is significantly larger for the post quantum signature. This poses a challenge when migrating to post quantum cryptography on a constrained device whose computational power and memory is limited [19].

Similarly, another issue caused by the constrained nature of ID cards, is that ID cards use two RSA key pairs. One is used for signatures and the other can be used for both authentication and encryption. As of 2025, there is no comparable post quantum primitive which can be used for both authentication and encryption. This means that a post quantum transition would require an extra key pair. As previously discussed, this is difficult for ID cards which already have limited resources [19].

The potential for hybridisation poses similar issues for ID cards. The hybrid approach would require the ID card to store all the keys and certificates for both the classical cryptography and the post quantum cryptography. The memory limits of ID cards makes this difficult. Vakarjuk et al. note several possible means to achieve a hybrid approach for public key certificates, each with certain advantages and disadvantages [19].

For example, one approach is to simply store multiple certificates separately. This avoids having to change the current infrastructure. However, it creates challenges if the current architecture only supports one certificate [19].

Another approach is to use an extension called 'AltPublicKey' for X.509 certificates. This is the certificate used in the IVXV protocol [14]. This extension adds a post quantum key and signature to the existing certificate. Systems supporting only classical cryptography can use the classical signatures as normal whilst those systems supporting post quantum cryptography can also verify the post quantum signatures. Vakarjuk et al. note that whilst this has advantages such as the compatibility with legacy systems, it also poses challenges similar to those described previously. That is, it requires the transfer of post quantum keys which are large in size, even when they are not used [19].

There are other possible appproaches for hybridisation with regard to certificates which are omitted here but can be found in [19].

Another challenge facing a post quantum transition in the context of e-voting can be seen with regards to mix nets. Current mix nets are insecure against quantum threats and thus post quantum mix nets are required. As of 2024, some post quantum mix nets have been created but either their performance is insufficient for use in e-voting or they are designed in such a way that the entire voting protocol would have to be changed [19].

Another challenge facing post quantum mix nets is that of hybridisation. As of 2024, according to [19], there is no known mix net that can support hybrid encryption. If only one layer of encryption can be mixed then the hybridisation becomes redundant [19].

The final challenge we note is that, as of 2025, whilst there is a increase in post quantum research, there remains a lack of research on specific use cases of post quantum cryptography. In particular, their is a lack of research in the context of e-voting [19].

Having established the urgency of the transition, the suggested approach towards a transition and the obstacles facing a transition, we will now consider one possible approach for a transition. This is a lattice based approach.

# 5 Lattices

Cybernetica [21] give a zero knowledge proof for homomorphic tallying. An attacker could take encrypted votes and even if signatures have already been removed to anonymise the ballot, it is possible for the attacker to link votes to voters based on the order they were recieved in or by side channel attacks, thus violating privacy. There are many possible solutions and Farzaliyev et al. [21] discuss two of them. The first is mix nets where votes are shuffled using mix servers to sever ties between votes and voter identities. The second way is through homomorphic tallying where votes are not individually decrypted but instead use a homomorphic property to tally them while still encrypted, and then output the final tally. This tallying process reveals nothing on individual votes [21]. As of 2025, Estonia uses mix nets to shuffle votes but relies on the El Gamal algorithm which is not quantum safe.

As an alternative researchers are looking at lattice based mix nets. Research is currently being done on both homomorphic tallying and mix nets, though according to Farzaliyev et al. [21] more has been done on mix nets thus far. In particular, it is difficult to establish a zero knowledge proof of correct shuffling. That is, decryption before and after shuffling should give the same result [21]. The mathematics involved in [21] is beyond the scope of this thesis, however, it demonstrates a key issue in the transition to a quantum safe e-voting protocol. That is, the ability to develop a verifiable means of obscuring voter identity from voter choice through zero knowledge proofs for homomorphic tallying or zero knowledge proofs for mix nets. We now aim to explore security based on lattices.

## 5.1 Introduction to Lattices

We will begin our discussion of lattices by defining a lattice and exploring some lattice based problems.

**Definition (Lattice):** A lattice can be defined by some basis of $n$ linearly independent vectors $b_1$,...,$b_n$ where each vector is in $\mathbb{R}^n$. All linear combinations of basis vectors with integer coefficients form the lattice $L$. That is, $L(b_1,...,b_n) = \{\sum_{i=1}^{n} x_i\ b_i : x_i \in \mathbb{Z}\}$. Thus, we see that a lattice $L$ is as a discrete, infinite set. The points contained in the set are in n-dimensional Euclidean space. The structure of $L$ is periodic [22].

The relevance of lattices to cryptography, in particular post quantum cryptography, comes from the difficulty of some lattice based problems [22]. We will proceed by exploring some of these hard problems.

Firstly, there is the shortest vector problem. Whilst there are several variants of the problem, the premise remains somewhat similar. That is, to find the shortest non-zero vector in the lattice. The variations include finding the shortest vector, finding the length of the shortest vector and deciding whether the shortest vector is

shorter than a given number, $r \in \mathbb{R}$. Another type of problem based on lattices is the closest vector problem. This problem aims to find the the vector which is closest to a target point, $t$ [22].

We will now discuss the learning with errors problem and its relevance to quantum safe cryptography.

## 5.2 Learning With Errors

The learning with errors problem was first established by Regev in 2005. He wrote a paper in which he demonstrated its difficulty and established a cryptosystem based on the problem. We will begin by defining the problem.

**Problem (Learning With Errors):** The learning with errors problem (LWE) is as follows: Let $a_i, i = 1, ..., m$ be polynomials whose coefficients are chosen uniformly and at random from $\mathbb{Z}_q^n$ to form the matrix $A \in \mathbb{Z}_q^{m \times n}$. Note that $n$ is the degree of the lattice and $q$ is a prime number and serves as the modulus. Let $s$ be the secret which is also chosen uniformly and at random from $\mathbb{Z}_q^n$. Let $e$ be the error chosen from $\chi$ which is the error distribution. Given $A$ and $b = A \cdot s + e \mod q$, the learning with errors problem asks to find $s$. We note that $e$ must remain secret and if $e$ is the zero vector, then the problem is easy to solve [22].

**Example (Learning With Errors):** To provide further understanding of the LWE problem we will explore an example. Suppose $n = 4$, $m = 6$ and $q = 11$. Let us choose the following:

$$A = \begin{pmatrix} 2 & 4 & 9 & 8 \\ 6 & 2 & 7 & 1 \\ 10 & 3 & 2 & 5 \\ 8 & 6 & 9 & 1 \\ 7 & 2 & 7 & 3 \\ 3 & 1 & 9 & 8 \end{pmatrix}, s = \begin{pmatrix} 2 \\ 6 \\ 10 \\ 4 \end{pmatrix}, e = \begin{pmatrix} 1 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \end{pmatrix}.$$

Then we have $b = A \cdot s + e \mod 11$ which is,

$$\begin{pmatrix} 2 & 4 & 9 & 8 \\ 6 & 2 & 7 & 1 \\ 10 & 3 & 2 & 5 \\ 8 & 6 & 9 & 1 \\ 7 & 2 & 7 & 3 \\ 3 & 1 & 9 & 8 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 6 \\ 10 \\ 4 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \\ 2 \\ 2 \\ 10 \\ 2 \end{pmatrix} \mod 11.$$

The LWE problem gives matrix $A$ and vector $b$ and asks what is $s$ (whilst keeping $e$ secret).

**Problem (Module Learning With Errors):** The Module Learning With Errors problem (MLWE) is a generalisation of the above LWE problem with a key difference

being that instead of working with $\mathbb{Z}_q^n$ we work with $R_q^k$. $R_q^k$ contains vectors who have $k$ components, each of which is a polynomial in $\mathbb{Z}_q[X]/(X^n+1)$ [22].

## 5.3 Regev's Cryptosystem

In 2005, Regev both demonstrated the LWE problem for the first time and also provided the first lattice based encryption scheme using LWE. We will briefly discuss this cryptosystem as it will be helpful in understanding the ML-KEM scheme which we will discuss next. Also it will be intuitive for understanding the security of a post-quantum mix net which will be discussed later. Regev's cryptosystem can be described in the following way.

*Choosing parameters:*
Let $n$ be a security parameter (meaning that our choice of $n$ will determine the security level of the cryptosystem). In particular, our choice of $n$ determines the public key size. Then we choose $p \geq 2$ such that $n^2 < p < 2n^2$. Then, let $m = (1+\varepsilon)(n+1)\log p$ where $\varepsilon > 0$ is chosen at random. We use a probability distribution $\chi$ on $\mathbb{Z}_p$. More detail on these parameters can be found in Regev's paper [23].

*Establishing public and private keys:*
The private key, $k_{priv}$, is chosen uniformly at random from $\mathbb{Z}_p^n$. Then, choose $a_1, ..., a_m \in \mathbb{Z}_p^n$ for $i = 1, ..., m$ independently and uniformly. We also choose $e_1, ..., e_m \in \mathbb{Z}_p$ independenlty from $\chi$. Then we compute $b_i = a_i \cdot s + e_i \mod p$. The public key, $p_{pub}$, is equal to $(a_i, b_i)$ for $i = 1, ..., m$ [23].

*Encryption:*
We begin the encryption process by choosing randomly and uniformly a subset of $m$. Let us call this $S$. Then we encrypt a bit, 0, as $(\sum_{i\in S} a_i, \sum_{i\in S} b_i)$. We encrypt a bit, 1, as $(\sum_{i\in S} a_i, \sum_{i\in S} b_i + \lfloor \frac{p}{2} \rfloor)$ [23].

*Decryption:*
In order to decrypt, we consider the encrypted pair $(a, b)$. We compute $b - a \cdot s$. If the result is closer to 0, we decrypt it to the bit 0. If the result is closer to $\lfloor \frac{p}{2} \rfloor$ we decrypt it to 1 [23].

Thus, we have seen how the learning with errors can be applied to create a cryptosystem. In Section 6 we will describe a newer and more detailed cryptographic algorithm that utilises the learning with errors problem.

# 6 ML-KEM

In 2016, NIST began the process of designing and standardising public key algorithms believed to be quantum safe. The process resulted in NIST choosing four post quantum cryptography algorithms to be standardised in 2024. We will now explore one of the algorithms in more detail, namely ML-KEM. This is a lattice based quantum safe key encapsulation mechanism [24].

A key encapsulation mechanism is a way to share between two parties a secret key. Person A runs a key generation algorithm which provides an encapsulation key and decapsulation key. They send the decapsulation key to person B. Person B can use this encapsulation key as the input to an encapsulation algorithm. The output is a copy of the secret key and some ciphertexrt. The ciphertext is sent to person A and is used alongside the decapsulation key as the input into the decapsulation algorithm. The output is a copy of the secret key [24].

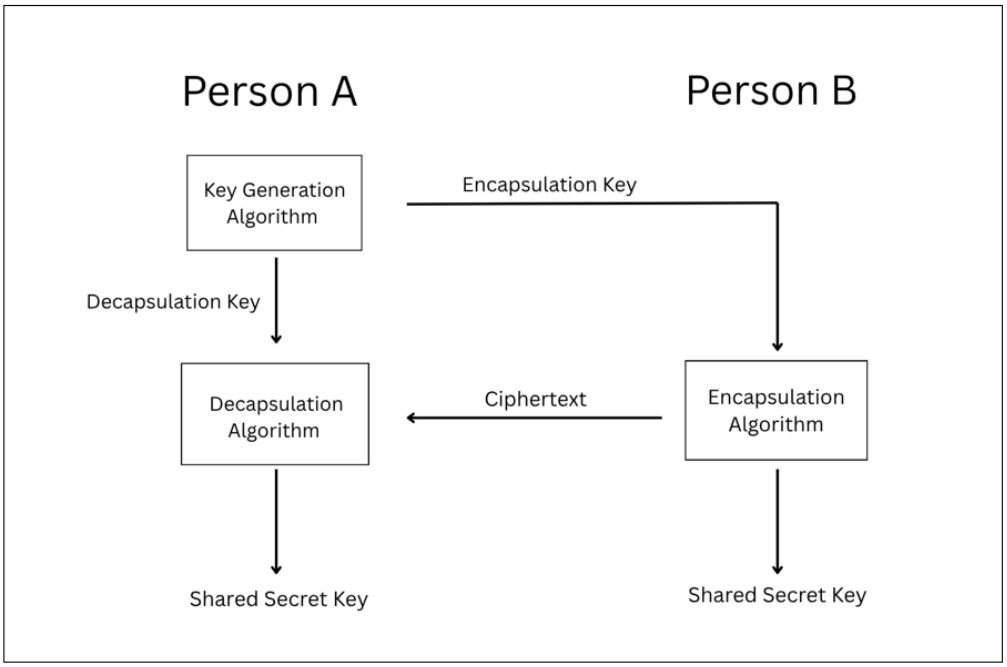


Figure 6: Key Encapsulation Mechanism. This figure is based on a figure in in [24].

ML-KEM is a key encapsulation algorithm whose security is based on the aforementioned MLWE problem. The algorithm is comprised of three parts. That is, a key generation algorithm, a key encapsulation algorithm and a key decapsulation algorithm. In terms of the construction of ML-KEM, this involves first constructing a public key encryption (PKE) algorithm and then utilising the Fujisaki Okamoto Transform [25] to turn the PKE algorithm into a key encapsulation mechanism (KEM).

## 6.1 Subroutines

ML-KEM uses subroutines in the algorithm. We will begin by exploring these subroutines. The first two subroutines that we should familiarise ourselves with is compression and decompression.

**Definition (Compression):** Compression can be defined by the following function: $\mathsf{Comp_d} : \mathbb{Z}_q \to \mathbb{Z}_{2^d},\ x \mapsto \lceil (2^d/q) \cdot x \rfloor \mod 2^d$, where $d < 12$ and $q = 3329$. (Note that, $\lceil \rfloor$, denotes rounding to the nearest integer) [24].

**Example (Compression):** Let $q = 3329$, $d = 10$ and $x = 21$. Then the compression function computes $\lceil (2^d/q) \cdot x \rfloor \mod 2^d$. That is, $\lceil (2^{10}/3329) \cdot 21 \rfloor \mod 2^{10} = 6$.

**Definition (Decompression):** Similarly, decompression can be defined by the following function: $\mathsf{Decomp_d} : \mathbb{Z}_{2^d} \to \mathbb{Z}_q,\ y \mapsto \lceil (q/2^d) \cdot y \rfloor \mod q$, where $d < 12$ and $q = 3329$ [24].

**Example (Decompression):** Let $q = 3329$, $d = 10$ and $y = 6$. Then the compression function computes $\lceil (q/2^d) \cdot y \rfloor \mod q$. That is, $\lceil (3329/2^{10}) \cdot 6 \rfloor \mod 3329 = 20$.

We note that compression followed by decompression does not guarantee that the initial value will be returned. This can be seen in the above example where we compress $x = 21$ but after decompressing we get 20. This is because multiple values of $x$ can compress to the same value of $y$.

Another subroutine that is used in ML-KEM is the Number-Theoretic Transform (NTT). NTT is an isomorphism between two rings, $\mathbb{Z}_q[X]/(X^n + 1)$ and $T_q$ where $T_q$ is a direct sum of 128 quadratic extensions of $\mathbb{Z}_q$. (This will not be important for later discussion thus we omit further explanation but details on quadratic extensions can be found here [26]). Then, given that this is an isomorphism, this means there also exists an inverse transform $\text{NTT}^{-1}$. It is a transform that allows for fast multiplication of polynomials in the ring $\mathbb{Z}_q[X]/(X^n + 1)$. The algorithm takes as an input the integer modulo $q$ coefficients of the polynomial $f \in \mathbb{Z}_q[X]/(X^n + 1)$ and outputs the coefficients of the NTT of the polynomial $f$ (which we will call $\hat{f}$). In the implementation of NTT used in ML-KEM, $q = 3329$ and $n = 256$ [24]. For further details on NTT, see NIST's standardisation of ML-KEM [24].

When choosing the variables in the ML-KEM scheme, often the polynomials, whose coefficients are small, are sampled by sampling their coefficients from a centred binomial distribution on $\mathbb{Z}_q$ [24]. In the following discussion we denote this sampling by CBD.

## 6.2 PKE Scheme

We can now explain the public key encryption scheme which acts as a building block for ML-KEM. This has three parts. These are key generation, encryption and de-

cryption.

*PKE Key Generation:*
Firstly, the key generation algorithm establishes an encryption and decryption key from a random input seed. This works by using the random seed to obtain the matrix $\hat{A} \in (\mathbb{Z}_q^n)^{k \times k}$. We then generate $s \in (\mathbb{Z}_q^n)^k$ and $e \in (\mathbb{Z}_q^n)^k$ where elements of $s$ and $e$ are sampled from the CBD. Then we apply the NTT to obtain $\hat{s} = \text{NTT}(s)$ and $\hat{e} = \text{NTT}(e)$. Computing $\hat{t} = \hat{A} \cdot \hat{s} + \hat{e}$ gives the final output. $(\hat{A}, \hat{t})$ is the public/encryption key $e_{PKE}$ and $\hat{s}$ is the private/decryption key $d_{PKE}$. In the NIST standardisation, $n = 256$ [24]. Note that the difficulty of obtaining $\hat{s}$ relies on the difficulty of the previously discussed MLWE problem.

**Example (PKE Key Generation):** Note, in this example we omit the NTT for the sake of simplicity. Let $q = 127$, $n = 4$ and $k = 2$. This means computation will be done $\mod X^4 + 1$ and polynomial coefficients will be reduced $\mod 127$. Since we are working $\mod X^4 + 1$, this means that $X^4 + 1 = 0$. Hence, we know that $X^4 = -1$. This then allows us to reduce the degree of our polynomials since $x^5 = x \cdot x^4 = -x$ and $x^6 = x \cdot x^5 = -x^2$. This logic can be extended further for higher degree polynomials but for this example this will be sufficient. Now let us choose the following values:

$$A = \begin{pmatrix} 12 + 36x + 71x^2 + 19x^3 & 16 + 41x + 53x^2 + 8x^3 \\ 9 + 10x + 83x^2 + 17x^3 & 120 + 84x + 61x^2 + 92x^3 \end{pmatrix},$$

$$s = \begin{pmatrix} 1 + 2x^2 - x^3 \\ 2x + x^3 \end{pmatrix}, e = \begin{pmatrix} 1 + x^2 - x^3 \\ 1 - x^2 \end{pmatrix}.$$

Then we compute the following:

$$t = A \cdot s + e$$

$$= \begin{pmatrix} 12 + 36x + 71x^2 + 19x^3 & 16 + 41x + 53x^2 + 8x^3 \\ 9 + 10x + 83x^2 + 17x^3 & 120 + 84x + 61x^2 + 92x^3 \end{pmatrix} \cdot \begin{pmatrix} 1 + 2x^2 - x^3 \\ 2x + x^3 \end{pmatrix}$$

$$+ \begin{pmatrix} 1 + x^2 - x^3 \\ 1 - x^2 \end{pmatrix}$$

$$= \begin{pmatrix} 12 + 68x + 177x^2 + 201x^3 + 163x^4 + 20x^5 - 11x^6 \\ 9 + 250x + 269x^2 + 270x^3 + 424x^4 + 12x^5 + 75x^6 \end{pmatrix} + \begin{pmatrix} 1 + x^2 - x^3 \\ 1 - x^2 \end{pmatrix}$$

$$= \begin{pmatrix} 13 + 68x + 178x^2 + 200x^3 + 163x^4 + 20x^5 - 11x^6 \\ 10 + 250x + 268x^2 + 270x^3 + 424x^4 + 12x^5 + 75x^6 \end{pmatrix}$$

Then we can apply the reductions $\mod X^4 + 1$ that we described previously to obtain the following:

$$\begin{pmatrix} -150 + 48x + 189x^2 + 200x^3 \\ -414 + 238x + 193x^2 + 270x^3 \end{pmatrix}$$

Then we can reduce the polynomial coefficients using $\mod 127$ to obtain:

$$t = \begin{pmatrix} 104 + 48x + 62x^2 + 73x^3 \\ 94 + 111x + 66x^2 + 16x^3 \end{pmatrix}.$$

Thus we obtain the encryption key $(A, t)$ and the decryption key $s$. Next we describe the encryption algorithm.

*PKE Encryption:*
The encryption algorithm takes as an input, a plaintext message $m$ which we wish to encrypt and the previously generated encryption key $e_{PKE}$. Then, we select a $y \in (\mathbb{Z}_q^n)^k$, $e_1 \in (\mathbb{Z}_q^n)^k$ and $e_2 \in \mathbb{Z}_q^n$. They are all selected from the CBD. Then we compute $u = A^\top y + e_1$ and $v = t^\top y + e_2 + w$ where $w$ is some encoding of the plaintext message $m$. In particular $w = \lceil q/2 \rfloor m$ [27]. Then $u$ and $v$ are compressed and concatenated. The output is the ciphertext [24] .

Here we provide a simple example of encryption using the PKE in ML-KEM. Note that this PKE is not secure and should not be used outside of ML-KEM as a stand alone PKE.

**Example continued (PKE Encryption):** Suppose we wish to encrypt the message $m = 10$. In binary this is $m = 1010$. Then the polynomial representation would be $1 + x^2$. We now select $y$, $e_1$ and $e_2$:

$$y = \begin{pmatrix} x - 2x^2 + x^3 \\ 1 + x \end{pmatrix}, e_1 = \begin{pmatrix} 1 + x^2 + x^3 \\ -2x + x^2 \end{pmatrix}, e_2 = 1 + 2x + x^3.$$

We note that the coefficients of $e_1$ and $e_2$ are chosen such that they are close to 0. Also, note the value of $w$:

$$w = \lceil q/2 \rfloor m = \lceil 127/2 \rfloor \cdot (1 + x^2) = 64 + 64x^2.$$

Then we compute the following:

$$u = A^\top y + e_1$$

$$= \begin{pmatrix} 12 + 36x + 71x^2 + 19x^3 & 9 + 10x + 83x^2 + 17x^3 \\ 16 + 41x + 53x^2 + 8x^3 & 120 + 84x + 61x^2 + 92x^3 \end{pmatrix} \cdot \begin{pmatrix} x - 2x^2 + x^3 \\ 1 + x \end{pmatrix} + \begin{pmatrix} 1 + x^2 + x^3 \\ -2x + x^2 \end{pmatrix}$$

$$= \begin{pmatrix} 9 + 31x + 105x^2 + 111x^3 - 70x^4 + 33x^5 + 19x^6 \\ 120 + 220x + 154x^2 + 140x^3 + 35x^4 + 37x^5 + 8x^6 \end{pmatrix} + \begin{pmatrix} 1 + x^2 + x^3 \\ -2x + x^2 \end{pmatrix}$$

$$= \begin{pmatrix} 10 + 31x + 106x^2 + 112x^3 - 70x^4 + 33x^5 + 19x^6 \\ 120 + 218x + 155x^2 + 140x^3 + 35x^4 + 37x^5 + 8x^6 \end{pmatrix}.$$

Then applying reduction $\mod X^4 + 1$ we get,

$$\begin{pmatrix} 80 - 2x + 87x^2 + 112x^3 \\ 85 + 181x + 147x^2 + 140x^3 \end{pmatrix}.$$

Finally, reducing the coefficients mod 127 gives,

$$u = \begin{pmatrix} 80 + 125x + 87x^2 + 112x^3 \\ 85 + 54x + 20x^2 + 13x^3 \end{pmatrix}.$$

Also, we compute the following:

$$v = t^\top y + e_2 + w$$

$$= \begin{pmatrix} 104 + 48x + 62x^2 + 73x^3, & 94 + 111x + 66x^2 + 16x^3 \end{pmatrix} \begin{pmatrix} x - 2x^2 + x^3 \\ 1 + x \end{pmatrix}$$

$$+(1 + 2x + x^3) + (64 + 64x^2)$$

$$= (94 + 309x + 17x^2 + 152x^3 + 13x^4 - 84x^5 + 73x^6) + (1 + 2x + x^3) + (64 + 64x^2)$$

$$= 159 + 311x + 81x^2 + 153x^3 + 13x^4 - 84x^5 + 73x^6$$

Then reducing mod $X^4 + 1$ gives,

$$146 + 395x + 8x^2 + 153x^3$$

And finally applying reduction mod 127 we obtain,

$$v = 19 + 14x + 8x^2 + 26x^3.$$

Here we omit compression and concatenation, thus obtaining the ciphertext $c = (u, v)$. We can now describe the decryption aspect of the PKE algorithm.

*PKE Decryption:*
The decryption algorithm uses the decryption key $d_{PKE}$ obtained in the key generation algorithm alongside the ciphertext $c$ as the input and outputs the original plaintext message $m$. The ciphertext $c$ is split into two parts and each part decompressed. Then the resulting pair $(u', v')$ is used alongside the decryption key $d_{PKE}$ to obtain $v = s^\top u'$. Then by computing $v' - v$ it is possible to use this to recover $m$ [24].

**Example Continued (PKE Decryption):** Continuing our previous example, having omitted compression and concatenation, we can skip the splitting and decompressing of the ciphertext. Thus we compute the following:

$$v - s^\top u$$

$$= 19 + 14x + 8x^2 + 26x^3 - \begin{pmatrix} 1 + 2x^2 - x^3, & 2x + x^3 \end{pmatrix} \begin{pmatrix} 80 + 125x + 87x^2 + 112x^3 \\ 85 + 54x + 20x^2 + 13x^3 \end{pmatrix}$$

$$= 19 + 14x + 8x^2 + 26x^3 - (80 + 295x + 355x^2 + 407x^3 + 129x^4 + 157x^5 - 99x^6)$$

$$= -61 - 281x - 347x^2 - 381x^3 - 129x^4 - 157x^5 + 99x^6$$

Then reducing mod $X^4 + 1$ gives,

$$68 - 124x - 446x^2 - 381x^3$$

Now applying reduction mod 127 to the coefficients we obtain,

$$68 + 3x + 62x^2.$$

Now we round, recalling that we initially encoded $m$ using $\lceil q/2 \rfloor = \lceil 127/2 \rfloor = 64$. Thus, we can now look at the coefficients of $68 + 3x + 62x^2$ and determine whether they are closer to 0 or 64. If the coefficient is closer to 0, we round it to 0. If the coefficient is closer to 64, then we round it to 64. Thus $68 + 3x + 62x^2$ becomes $64 + 64x^2$. Then scaling back down by dividing by 64, we obtain $1 + x^2$ which represents the binary 1010 or the message $m = 10$ .

## 6.3 Key Encapsulation Mechanism

Now we will describe the algorithms used in ML-KEM. In the following, let $H$ represent the hash function SHA3-256, $J$ represent the hash function SHAKE256 and $G$ represent the hash function SHA3-512. See NIST's paper [24] for most details on these hash functions.

*KEM Key Generation:*
Firstly, we have the key generation algorithm which works by running the key generation algorithm for PKE which we previously described. The encapsulation key is simply the PKE public key. The decapsulation key is obtained by concatenating the PKE decryption key, the encapsulation key, a hash of the encapsulation key (using the hash SHA3-256) and some randomness $z$. That is, we obtain the encapsulation key, $e_{KEM} = e_{PKE}$. Also, we obtain the decapsulation key $d_{KEM} = (d_{PKE}||e_{KEM}||H(e_{KEM})||z)$ [24].

*KEM Encapsulation:*
Then the ML-KEM internal encapsulation algorithm takes as an input the encapsulation key and some randomness $r_1$. It concatenates the randomness with the hashed the encapsulation key (which is hashed using SHA3-256). Thus we have $(r_1||H(e_{KEM})$. This is then hashed using the hash function SHA3-512 to get $G((r_1||H(e_{KEM}))$.The output of this is $(K, r_2)$ where $K$ is the shared secret key and $r_2$ is the randomness. Then we encrypt $r_1$ using randomness $r_2$ and the encapsulation key $e_{KEM}$, resulting in the output $c$. Thus, the final output of the encapsulation algorithm is $(K, c)$ where K is the shared secret key and c is the ciphertext [24].

*KEM Decapsulation:*
The ML-KEM internal decapsulation algorithm takes the decapsulation key, $d_{KEM}$ and the ciphertext, $c$, as an input. The algorithm begins by the obtaining decryption key, $d_{PKE}$, from the PKE scheme. Then using the decryption algorithm from

the PKE with the decryption key $d_{PKE}$ and the ciphertext $c$ as inputs, we obtain a plaintext $m'$. We concatenate $m'$ and $H(E_{KEM})$. We use this as an input to the SHA3-512 hash which gives the output $(K', r') = G(m'||H(e_{KEM})$. Then, we use the randomness $z$ from the decapsulation key and concatenate it with the ciphertext $c$. The result is used as the input to a hash function SHAKE256. The output is $K'' = J(z||c)$. Then we use the PKE encryption algorithm with parameters $e_{PKE}$, $m'$ and $r'$ to get a ciphertext $c'$. If $c' \neq c$, we let $K'$ be $K''$. The final output is $K'$ [24].

# 7 Post Quantum Mix Net

We have discussed the e-voting scheme that is in place in Estonia as of 2025. Also we have explored the transition to post-quantum cryptography more generally. Thus, we will now explore the transition to post quantum cryptography in the specific context of e-voting.

Estonia uses mix nets in its e-voting protocol as discussed in Section 3 and Section 4. Mix nets work by taking the encrypted votes and mixing them in such a way that the output can not be tied back to specific voters. More specifically, there are two types of mix nets: decryption and re-encryption mix nets [28].

A decryption mix net involves mix servers generating public and private keys. Each vote is encrypted with each mix server's public key. For a mix net with $n$ mix servers, indexed by $1, ..., n$, the first layer of encryption of a vote uses the public key of mix server $n$. The next layer of encryption uses the public key of mix server $n-1$. This continues with the last layer of encryption using the public key of mix server 1. Thus, each vote has multiple layers of encryption. Encryption of each layer is carried out by the mix server which generates the corresponding public key. The first server decrypts the first layer of encryption, shuffles the votes and then outputs the shuffled votes to the next server. This process is repeated until the final server outputs shuffled plaintext [28].

On the other hand, a re-encryption mix net works by first establishing a public key (which is used by all of the mix servers) and secret key shares. Each server has one share of the secret key. The encrypted votes are taken as input into a mix server, the votes are encrypted again using the public key and then they are shuffled by the mix server. This output is then taken as the input of the next mix server. Again, the process is repeated. After the votes have been encrypted and shuffled by all of the mix servers, the final output is shuffled ciphertext. (Note this is different to the input ciphertext as it has been encrypted by each mix server). The secret shares can then be used to decrypt the mix servers encryption. Thus the final output is the shuffled version of the input ciphertext. How and if this is decrypted depends on the wider scheme outside of the mixing process [28].

An important security feature of e-voting is that the mix net must be verifiable. That is, we must be able to check that the input and output are a 1-to-1 mapping. Another desirable feature is accountability as a means to deter malicious actors [26]. Boyen et al. [28] develop a lattice based decryption mix net that is verifiable and accountable with the intention that it is also practically feasible.

The protocol of Boyen et al. is a proposed quantum safe mix net which also allows for verification and accountability. The protocol is separated into a set up phase, voting phase, mixing phase, auditing phase and finally a decryption phase.

We first identify the participants. There are voters $(V_1, ..., V_{n_v})$. There are also

mix servers $(M_1, ..., M_{n_m})$ and auditors $(A_1, ..., A_{n_a})$. Also, there is a public bulletin board $(B)$ which is append only (meaning data may be added but not modified or deleted) and $n_t$ tripwires inserted by each auditor with $t_q = n_t \cdot n_a$ tripwires in total. We assume the use of an authenticated channel from each $V_i$ to $B$ in order to ensure only those who are eligible may cast votes [28].

In terms of the cryptographic primitives, we use an IND-CCA2 secure encryption scheme $E$, an EUF-CMA secure signature scheme $S$ and a IND-CCA2 secure $(n_a, n_a)$ threshold public key encryption scheme $E_d$ [28].

The protocol also makes several assumptions. Firstly, anytime a party which has signing keys publishes information, that information is signed. Anytime there is dishonesty that is obvious such as refusal to participate, the protocol is stopped. In this instance, the dishonest party then faces repercussions which will vary depending on the wider scheme. If a ciphertext appears multiple times only the first instance is kept [28]. Having established the relevant parties and primitives we can now describe the protocol stages.

## 7.1 Set Up

The auditor, $A_j$, runs a key generation algorithm for both the signature scheme $S$ and the encryption scheme $E$. This generates a verification/signing key pair, $(aver_j, asig_j)$, and a public/private key pair, $(pk_j, sk_j)$, respectively. The verification key and the public key are posted to the bulletin board, $B$. $A_j$ runs a key generation algorithm for the distributed public key encryption scheme $E_d$ to get a public/private key pair $(pk_j^{dis}, sk_j^{dis})$. Then, $pk_j^{dis}$ is posted to the bulletin board, $B$. Using all of the $pk_j^{dis}$'s posted to the bulletin board, anyone can use a public key generation algorithm to get the joint public key $pk^{dis}$. Finally, each mix server $M_k$ runs a key generation algorithm for the signature scheme $S$ to get the verification/signing key pair, $(mver_k, msig_k)$, and a key generation algorithm for the encryption scheme $E$ to get a public/private key pair, $(mp_k, ms_k)$. The verification key and the public key are posted to the bulletin board [28]. This completes the set up phase.

## 7.2 Voting

Each voter, $V_i$, chooses their vote $v_i$ and encrypts it. To encrypt it, they first encrypt using the shared auditors' key $pk^{dis}$ then they encrypt it using all of the mix servers' public keys $(mp_1, ..., mp_{n_m})$ in reverse order, followed by the auditors' public keys $(pk_1, ..., pk_{n_a})$ in reverse order. Thus resulting in the ciphertext, $c_i = Enc(pk_1, (..., Enc(pk_{n_a}, Enc(mk_1, (..., Enc(mk_{n_m}, Enc(pk^{dis}, v_i)))))))$. This ciphertext is sent to the mix net to be taken as the input [28].

Also, each $A_j$ generates $n_t$ messages $m$ and does the same encryption process described above to obtain the $n_t \cdot n_{n_a}$ many tripwires, $t_q$. The auditors locally store

the random coins which provide the randomness used to generate the tripwires. The tripwires, $t_q$, are also used as inputs in the mixing phase alongside the ciphertexts $c_i$ [28].

Overall, the ciphertexts, $c_i$, alongside the tripwires, $t_q$, together form the input of the mixing phase, denoted $c_r^{tot}$ [28].

## 7.3 Mixing

The mixing phase has two parts. That is the auditors' part and the mix servers' part.

In the auditors' part, they take the input $c_r^{tot}$ and use their private keys $(sk_1, ..., sk_{n_a})$. Each auditor $(A_j)$, starting with $A_1$, takes the input and decrypts each input with their private key (effectively removing a layer of encryption). Following the decryption, they perform some permutation $\pi_j$ which is chosen uniformly and at random to shuffle their decrypted input. The output is posted to the bulletin board where the next auditor takes this as its input and repeats the process. The final output is the shuffled input ,$c_r^{tot}$, but now with all layers of auditor encryption removed. We will denote this as $c_r^{mix}$ [28].

The mix servers then take this $c_r^{mix}$ as the input of the first mix server and following an analogous process they too shuffle and remove a layer of encryption until the final output is the shuffled ciphertexts and tripwires with only the final layer of encryption remaining. Note this layer of encryption is the encryption performed using the shared auditors' key [28].

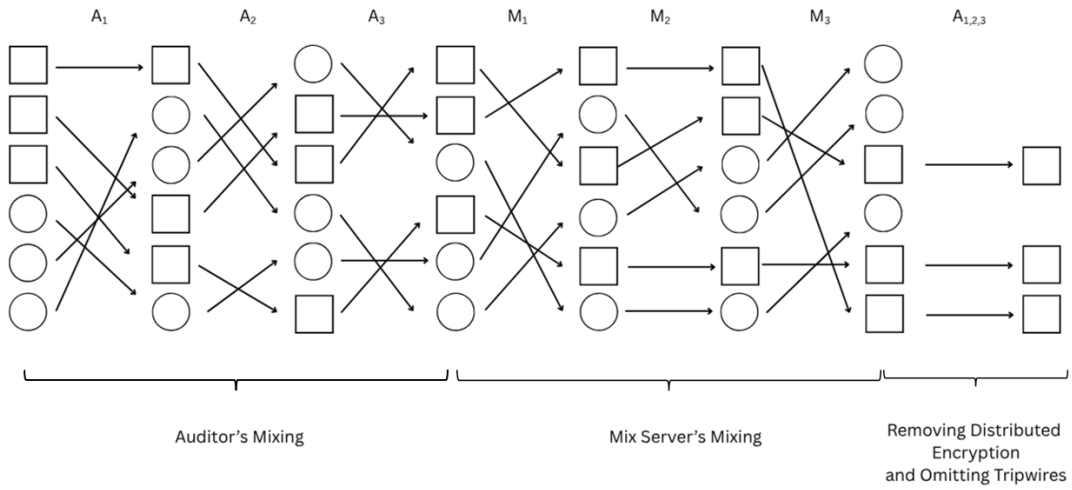


Figure 7: The mixing process with 3 auditors, 3 mix servers, 3 votes represented with squares and 3 tripwires (one generated per auditor) represented with circles. This figure is based on a figure in [28].

## 7.4 Auditing

In the auditing phase each auditor, $A_j$, can now publish their decryption key $sk_j$ and thus anyone can now check that the mixing performed by the auditors was done correctly. If this is found to not be the case meaning there is a malicious auditor, then the protocol must be aborted. Additionally, the random coins used by the auditors to generate the tripwires are published. This allows anyone to check the integrity of the tripwires through the mixing done by the mix servers and auditors. If they reveal how their tripwires are generated then anyone can encrypt the tripwires using the public keys, thus determining which of the inputs are tripwires. Also since the auditors decryption keys, $sk_j$ are published, anyone can follow their path through the auditor's mixing process. Then we know which are encrypted votes and which are encrypted tripwires when entered into the mix servers. Now given that the random coins are publicly available and the public key shares of the distributed encryption scheme are available, we should be able to determine which are the tripwires in the output of the mix servers. This is because we can use the random coins to generate the tripwires and then we can apply the distributed encryption layer to the tripwires since the public key shares are available on the bulletin board. The deterministic nature of the encryption scheme means the result of this process will be the same as the output of the mix servers. If the mix servers are to tamper with their input, then they risk tampering with a tripwire, thus changing the output of the mix server which would be detected at this point. If the integrity can not be verified then the protocol is aborted. In both instances, if it is found that there is a malicious actor, they can be identified through their digital signature [28].

## 7.5 Decryption

In the decryption phase, auditors, $A_j$, publish their secret share key, $sk_j^{dis}$, to the bulletin board. Then, Boyen et al. [28] notes that there are two possibilities. One possibility is that each ciphertext output by the mixing server, alongside the share key, are used to get the decryption key share. Then all of the decryption key shares can be used together to decrypt and get the shuffled plaintext $\tilde{v}_i$. Alternatively and more efficiently, if the threshold encryption scheme allows for this, the secret share keys are used to get the joint secret key, $sk^{dis}$. This is then used to decrypt to get the shuffled plaintext votes $\tilde{v}_i$ [28].

## 7.6 Verifiability

Having demonstrated a mix-net we wish to be able to verify its correctness of shuffle. This means that the input and output of the mix net should be the same only the output is in a different order to the input [28].

**Definition (Verifiability):** We say that the mix net is $(\delta, n)$ verifiable if and only if the mix net is accepted as correct with probability at most $\delta$ where more than $n$

inputs were either manipulated or illegitimate votes entered [28].

Using the above definition, Boyen et al. make the following assumptions in order to demonstrate verifiability. We assume that the PKE, $E$, and the distributed PKE, $E_d$, that are used are IND-CCA 2 secure. We also assume that the signature scheme, $S$ is EUF-CMA secure. We assume that the mixing authority, the bulletin board, $B$, and a minimum of one of the auditors are honest. We also assume that all honest voters and auditors input plaintext messages that are the same size for each implementation. Finally, we assume that if the PKE and distributed PKE have plaintexts that are of the same length, then so are the ciphertexts as well [28].

Having established a definition as well as the assumptions being made, Boyen et al. then make the following verifiability claim. (Note that in the following $n_v^{hon}$ refers to the number of honest voters and $t_q$ is the number of tripwires). Based on the above assumptions, the mix net is $(\delta_n(n_v^{hon}, t_q), n)$ verifiable where $\delta_n(n_v^{hon}, t_q) = \frac{\binom{n_v^{hon}}{k+1}}{\binom{n_v^{hon}+t_q}{k+1}}$ [28].

Boyen et al. justify the above equation by first noting that the auditors and the distributed encryption are both verifiable since the relevant information is published at the end. Then we note that the encryption schemes used are IND-CCA2 and thus it must be that the auditors tripwires, $t_q$ and the honest voters $n_v^{hon}$ are indistinguishable. Thus manipulating a vote cast by an honest voter must be done blindly in the sense that it can not be distinguished from the tripwires. Suppose the attacker manipulates $k+1$ many votes from the $n_v^{hon} + t_q$ many votes. We assume combinatorically that the attacker takes a vote and doesn't replace it since this is the equivalent of manipulating $k+1$ *different* votes. Then the probability that no tripwires were touched is given by

$$\frac{\text{number of ways to choose k+1 votes from the honest voters without replacement}}{\text{number of ways to choose k+1 votes from all the votes without replacement}}$$

$$= \frac{\binom{n_v^{hon}}{k+1}}{\binom{n_v^{hon}+t_q}{k+1}} \quad \text{[28].}$$

## 7.7 Quantum Safeness

In order to make the mix net quantum safe, the encryption used for both the PKE, $E$, and the distributed PKE, $E_d$ must be quantum safe. Boyen et al. note that they use a hybridisation of an AES256 based DEM/MAC and a lattice based CCA2 secure KEM. The KEM is very similar to Regev's cryptosystem which we described in Subsection 5.3. The result of their hybridisation is a IND-CCA2 secure encryption scheme. Boyen et al. provide more details on the quantum safe encryption scheme used [28].

The mix net is discussed by Farzaliyev et al. [29], where they note both some advantages and disadvantages. For example, they note that by avoiding using non-interactive zero knowledge proofs, the mix net gets very fast results. However, they do criticise the reliance of the mix net on the honesty of auditors, noting the restrictiveness of this.

# 8 Post Quantum Signing Keys

As noted in 3.3, the signing process in the IVXV protocol has vulnerabilities due to the use of ECC and RSA, each of which are not quantum safe. Estonia use Splitkey technology developed by Cybernetica for digital signatures. In particular, it is used in Estonia's e-voting protocol. We will begin this section by discussing Splitkey based on the white paper provided by Cybernetica [30]. Then we will discuss the potential post quantum transition.

## 8.1 Splitkey

Splitkey is a technology designed for signing and authentication using smart devices. The process involves two parties: the server and the client. The client is responsible for initiating the process using a smart device and the server is responsible for helping the client achieve auditability, trust and assurance [30].

In the key generation process, the final key $d$ consists of two separate keys $d_1$ and $d_2$. The key $d_1$ is also split further into $d_1'$ and $d_1''$ [30]. We will now describe the key generation process.

The process begins with the client initiating it. RSA key generation is carried out in the usual sense to obtain keys $n_1$ and $d_1$, where $n_1$ is the public key and $d_1$ is the private key. The client sends the public key, $n_1$, to the server. Also, the server carries out the same RSA key generation process to obtain the public key $n_2$ and the private key $d_2$. In the server, the relevant keys are stored in the HSM. Also, the the large, random exponent $e$ is chosen beforehand and is not sent between the two parties during this process [30].

Then, the client will split the key $d_1$ into separate parts, $d_1'$ and $d_1''$. This is done by first using a random number generator to produce a random bit string of a predefined size. This is $d_1''$. The role of $d_1''$ is twofold. Firstly, it is used to split $d_1$. Secondly, it works to provide randomness to $d_1'$. In order to generate $d_1'$, the following operation is performed; $d_1' = d_1 - d_1'' \mod (p_1 - 1)(q_1 - 1)$ where $p_1$ and $q_1$ are the primes used to generate $n_1$. That is, $n_1 = p_1 \cdot q_1$. (Note the value of the modulo is Euler's totient function). The part $d_1'$ is stored by the client. In order to store it securely, it is encrypted using AES and stored in its encrypted form. In order to aceess it, a pin is required. The part $d_1''$ is sent to the server using a secure channel and it is stored in the HSM. By having random values $d_1'$ and $d_1''$ be stored in two separate locations, it requires an attacker to exploit both in order to obtain $d_1$, thus making a successful attack less likely. Additionally, $d_1$ and $d_1''$ are both marked to indicate that they need to later be erased [30].

We will now describe the process of certifying the new key pair which the white paper describes as a continutation of the process of key generation. The key pair must be associated with the user's identity and in order to achieve this a certificate is generated in the service provider's system [30].

The certificate request is carried out by the server. Using the identity of the authenticated person a certificate signing request (CSR) is sent. This CSR includes the identity of the person which can be provided by an external registration authority. It also includes the compound public key, $n$. Prior to the CSR, the server takes the public key $n_1$ from the client and their own public key $n_2$ and multiplies them to get the compound key $n = n_1 \cdot n_2$. Also included in the CSR is a timestamp, a definition of the policy of the certification and a URL which links to the certificate revocation list (CRL). Assuming all of the above is successfully provided and the requirements are met, the server contacts the certificate authority and sends the CSR, applying for a certificate. Then the certificate authority provides a new certificate which can be stored by the server. The server can produce logs for the purpose of auditing [30].

Having completed the key generation and acquiring the certificate, then the primes $p_1$ and $q_1$, the private key $d_1$ and the copy of $d_1''$ that the client has are all erased by the client. The key $d_1''$ remains stored by the server. We note also that in the case where a certificate can not be provided, then all the cryptographic information becomes useless and is destroyed securely [30].

Now we move on to the signing process. Firstly, we define the relying party (RP) to be the party which holds the documents which are to be signed by the client. In order to prepare the documents for signing, the RP will compute a hash, let us call it $h$, from the document. This is then sent to the server. The server then pads the hash, $h$, to obtain the message $m$ whose size is the same as that of the key. The message $m$ is what we wish to sign. Then, the message $m$ is sent from the server to the client [30].

The client will enter the pin required to decrypt the private key $d_1'$. Regardless of the pin input, decryption will occur; however, only the correct pin will give the correct value of $d_1'$. Decrypting always is a security measure that means an attacker is provided no feedback on the success of their attack. Having obtained the private key $d_1'$, the client then uses it to calculate their part of the signature. This is done by calculating $m^{d_1'} \mod n_1$ [30].

Then the server will get the private key $d_1''$ from the HSM where it has been stored. It calculates its part of the signature by computing $m^{d_1''} \mod n_1$. We note that the server is only able to access the private key $d_1''$ per the client's request and its use is logged and audited [30].

Then, we note that the final half signature can be computed by combining the two parts as follows: $m^{d_1'} \cdot m^{d_1''} \mod n_1 = m^{d_1'+d_1''} \mod n_1 = m^{d_1} \mod n_1$ [30].

It is important to note that until this point, we have no means to verify that the correct pin and thus the correct private key $d_1'$ was used. However, now the server has access to both parts of the signature, computing $(m^{d_1})^e \mod n_1$ and checking that it is equivalent to $m$. If it is then we know that we must have the correct value

for $d_1'$ and thus a correct pin [30].

If the server finds that the client's pin is correct, they can begin obtaining the second half of the signature. The second half of the signature is computed by the server using the equation $m^{d_2} \mod n_2$ [30].

Then, the final signature can be calculated by the server. This is done by taking the two parts of the signature, $\beta_1 \equiv m^{d_1} \mod n_1$ and $\beta_2 \equiv m^{d_2} \mod n_2$, and applying the Chinese Remainder Theorem. (For those that are unfamiliar, an explanation of the Chinese Remainder Theorem can be found here [31]). The final signature is $\beta$, such that $\beta \equiv \beta_i \mod n_i$ for $i \in \{1, 2\}$. The correctness of the signature can be verified by checking that $\beta^e \equiv m \mod n_1 n_2$ [32]. We note that at no point was the full private key $d$ obtained or stored and thus the server does not exercise control of the signature [30].

## 8.2 Post Quantum Splitkey

We note that the above Splitkey technology is not quantum safe given its reliance on key generation using RSA which, as discussed, is vulnerable to quantum attacks. In an article published on their website, Cybernetica acknowledge this and note the need to transition to a post quantum solution. They analysed possible post quantum options based on the quantum safe signature schemes standardised by NIST. That is, they considered the possibility of using Crystals-Dilithium, Sphincs+ and Falcon [33]. They note that Sphincs+ lacks efficiency and Falcon has a more complex design and thus their focus was on Crystals-Dilithium [33]. We note that since this paper was published, these have now been standardised by NIST [34].

The Splitkey technology described above works by distributing the key between the client and the server and they note that, out of the post quantum algorithms, Crystals-Dilithium seems to be the most suitable for this purpose. That being said, it is not without some issues. One characteristic of the Crystals-Dilithium algorithm is that it performs checks in order to determine whether any information about the private key was leaked. If it was, the process must restart. On average the algorithm requires restarting approximately three or four times. This poses challenges when distributing the key [33].

One possible issue in the Splitkey technology is that, during the process of signing, the client and server pass information to each other. The Cybernetica article questions whether or not this is secure if we have not yet confirmed if the private key has been leaked. As a means to mitigate this issue, they suggest that instead of sending plaintext values to each other, the client and server use bit commitments and only once the private key is confirmed to have not leaked would they then reveal the commitments for verification [33].

Another issue in implementing Crystals-Dilithium into Splitkey is the need to restart

the process if the private key was leaked. Thus, as a means to mitigate this, the article suggests they run multiple instances of the protocol at once and combine relevant values to find a combination that passes the checks performed. This of course results in an increased amount of data transferred [33].

They note that as a result of the use of commitments the signature varies from that of Crystals-Dilithium and thus it is not possible to use the Crystals-Dilithium verification process to verify the signature. Thus, as of 2024 (when the article was published), they noted that they were focusing their research on whether it is possible to use a different cryptographic method to arrive at a Crystals-Dilithium signature, thus allowing for verification [33].

# 9 Conclusion

We have established through this thesis that the emergence of quantum technologies is shaping, and will continue to shape the future landscape of cryptography.

We have outlined the progress in quantum computing thus far. In particular, we have established both a fundamental understanding of quantum computer and demonstrated how this will affect cryptography through exploring Shor's algorithm. We have seen how Shor's algorithm could be used on a quantum computing to factor large integers and to solve the discrete logarithm problem. Through a toy example we have demonstrated the impacts this algorithm will have on current security.

We have explored the current e-voting protocol used in Estonia (the IVXV protocol) and the vulnerabilities of this protocol with regards to the quantum threat. In particular, we have seen that the current protocol uses vulnerable cryptographic primitives such as RSA and ElGamal.

We have also considered the transition to post quantum e-voting. We note that currently, there is disagreement among experts on some aspects of this transition, such as the risk of the 'harvest now decrypt later' approach. However, there is a consensus that more research is needed before a quantum transition should occur. Also, we have seen that there are still many unaddressed obstacles when migrating to post quantum e-voting. For example, this includes the large key size of post quantum algorithms and the issue this poses to constrained devices such as ID cards.

We have demonstrated the current state of post quantum cryptography as of 2025. This includes the basis of much of the current post quantum research; that is lattice based cryptography. We have explored the learning with errors problem that some post quantum algorithms are based on. There is work being done on supporting the post quantum transition. For example, through standardisation by the NIST, ML-KEM is the first post quantum key encapsulation mechanism that has been standardised.

We have also considered quantum safe e-voting. For example, we explored a possible quantum safe mix net. There is difficulty in finding a zero knowledge proof of correct shuffle for mix nets in the context of post quantum cryptography. Thus, the proposed mix net avoids zero knowledge proofs entirely. It uses quantum safe primitives alongside tripwires to create a verifiable post quantum mix net. Also, we have seen the work being done on quantum safe signatures and authentication. We note that the research of post quantum cryptography specifically for e-voting remains somewhat limited.

Possible areas for future research include the establishment of a post quantum mix net which has sufficient performance for supporting e-voting but which does not require a complete overhaul of the current e-voting infrastructure. Another area of future research is developing a quantum safe digital signature scheme that is

compatible with Estonia's current e-infrastructure. This could include developing a means to obtain a Crystals-Dilithium signature whilst using commitments. Cybernetica notes this as a current area of research they are exploring in order to obtain quantum safe SplitKey technology.

# 10 References

[1] M. A. Horowitz, E. Grumbling, (2019). *Quantum computing: progress and prospects*, The National Academies Press.

[2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, 1994, pp. 124-134. Available: DOI: 10.1109/SFCS.1994.365700

[3] S.S. Gill, et al, (2022, Jan). "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience* [Online]. vol. 52, 1, pp. 66-114. Available: DOI 10.1002/spe.3039

[4] National Institute of Standards and Technology. Available: https://www.nist.gov/

[5] E. G . Rieffel, W. H. Polak, (2011). *Quantum computing: A Gentle Introduction*, MIT Press.

[6] Microsoft. *Entanglement* [Online], Web Page. Available: https://quantum.microsoft.com/en-us/insights/education/concepts/entanglement

[7] W. Chang, A. V. Vasilakos, *Fundamentals of Quantum Programming in IBM's Quantum Computers*, Springer, 2020. Available: https://doi.org/10.1007/978-3-030-63583-1

[8] P. Kaye, R. Laflamme, M. Mosca, *An Introduction to Quantum Computing*, Oxford University Press, 2007.

[9] S. Vaudenay, "Public Key Cryptography," in *A Classical Introduction to Cryptography: Applications for Communications Security*, USA: Springer, Sept 2005. Available: https://doi.org/10.1007/b136373

[10] N. P. Smart, "Defining Security," in *Cryptography Made Simple*, Springer, pp.217-218, 2015. Available: https://doi.org/10.1007/978-3-319-21936-3

[11] "General Framework of Electronic Voting and Implementation thereof at National Eections in Estonia," State Electoral Service of Estonia, Tallin, 2017 [Online], Technical Paper. Available: https://www.regeringen.ax/sites/default/files/attachments/page/estonia-e-voting-2017.pdf

[12] E Estonia. (2022, Dec. 01). *IVXV protocols* [Online], Technical Paper. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV-protocols.pdf

[13] *Recommendation Rec(2004)11 of the Committee of Ministers to member states*

*on legal, operational and technical standards for e-voting*, Europarat, Ed, Strasbourg: Council of Europe Publishing, 2008 [Online]. Available: https://www.coe.int/t/dgap/goodgovernance/Activities/Key-Texts/Recommendations/Rec(2004)11_Eng_Evoting_and_Expl_Memo_en.pdf

[14] *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, RFC 6960, Internet Engineering Task Force (IETF) [Online], 2013. Available: https://datatracker.ietf.org/doc/html/rfc6960

[15] Open Verificatum. *Verificatum Mix-Net* [Online], Web Page. Available: https://www.verificatum.org/html/product_vmn.html

[16] D. Joeseph, R. Misoczki, M. Manzano, et al., "Transitioning organizations to post-quantum cryptography," *Nature*, 605, pp. 237-243, May 2022. Available: https://doi.org/10.1038/s41586-022-04623-2

[17] A. Rodriguez-Perez, N. Costa, T. Finogina. (2024, Oct). "An electoral exception? Quantum computing-readiness and internet voting." *JeDEM - eJournal of eDemocracy and Open Government* [Online], vol. 16, 3. Available: DOI 10.29379/jedem.v16i3.928

[18] P. Muzikant, J. Willemson P. Laud, "Migrating Some Legacy e-Governance Applications to Post-Quantum Cryptography," in *Fifth PQC Standardization Conference*, Rockville, Maryland, USA, 2024. Available: https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/migrating-some-legacy-e-governance-apps.pdf

[19] J. Vakarjuk, N. Snetkov, P. Laud, "Identifying Obstacles of PQC Migration in E-Estonia," in *CyCon 2024: Over the Horizon 16th International Conference on Cyber Conflict*, 2024, pp. 63-81.

[20] European Telecommunications Standards Institute, "CYBER; Migration strategies and recommendations to Quantum Safe schemes," ETSI, France, 2020.

[21] V. Farzaliyev, C. Pärn, H. Saarse, J. Willemson, "Lattice-Based Zero-Knowledge Proofs in Action: Applications to Electronic Voting," *Journal of Cryptology*, vol. 38, November, 2024. Available: DOI 10.1007/s00145-024-09530-5

[22] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, R. Cammarota, "Post-Quantum Lattice-Based Cryptography Implementations: A Survey", *ACM Computing Surveys*, vol. 51, 6, pp.1-41, 2019. Available: DOI 10.1145/3292548

[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, issue 6, pp. 1-40, Sept 2008.

[24] National Institute of Standards and Technology (2024) Module-Lattice-Based

Key-Encapsulation Mechanism Standard. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 203. Available: https://doi.org/10.6028/NIST.FIPS.203

[25] Fujisaki E, Okamoto T (2013) Secure integration of asymmetric and symmetric encryption schemes, *Journal of Cryptology*, 26, pp. 80–101. Available: https://doi.org/10.1007/s00145-011-9114-1.

[26] S. Roman, *Field Theory*, 2nd ed. New York: Springer New York, 2007. Available: https://doi.org/10.1007/0-387-27678-5

[27] A. Menezes. *Cryptography 101* [Online], Lecture Notes. Available: https://cryptography101.ca/

[28] X. Boyen, T. Haines, J. Müller, "A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing", in *Computer Security - ESORICS 2020: 25th European Symposium on Research in Computer Security*, Guildford, UK, 2020, pp. 336-356. Available: DOI 10.1007/978-3-030-59013-0_17

[29] V. Farzaliyev, J. Willemson, J. K. Kaasik, "Improved lattice-based mix-nets for electronic voting, *IET Information Security*, vol.17, issue 1, pp. 1-158, Jan 2023.

[30] Cybernetica. (2020, June. 03). *Introductions to Splitkey Foundations* [Online], White Paper. Available: https://cyber.ee/uploads/Split_Key_White_Paper_3506f03c34.pdf

[31] G. A. Jones, J. M. Jones, *Elementary Number Theory*, London: Springer London, 1998.

[32] A. Buldas, A. Kalu, P. LAUD, M. Oruaas, "Sever-Supported RSA Signatures for Mobile Devices," *Computer Security- ESORICS 2017: 22nd European Symposium on Research in Computer Security*, vol 10492, pp. 315-333, Aug 2017. Available: https://doi.org/10.1007/978-3-319-66402-6_19

[33] J. Vakarjuk. (2024, Aug. 27). *Towards post-quantum Splitkey* [Online], Web Page. Available: https://cyber.ee/resources/stories/towards-post-quantum-splitkey/

[34] National Institute of Standards and Technology (2024) Module-Lattice-Based Digital Signature Standard. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 204. Available: https://doi.org/10.6028/NIST.FIPS.204