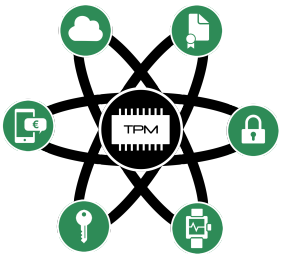

FutureTPM

D1.3

Security Risks in QR Deployments

Project number:	779391
Project acronym:	**FutureTPM**
Project title:	Future Proofing the Connected World: A Quantum-Resistant Trusted Platform Module
Project Start Date:	1st January, 2018
Duration:	36 months
Programme:	H2020-DS-LEIT-2017
Deliverable Type:	Report
Reference Number:	DS-LEIT-779391 / D1.3 / v1.0
Workpackage:	WP 1
Due Date:	30th September, 2018
Actual Submission Date:	28th September, 2018
Responsible Organisation:	UL
Editor:	Alfredo Rial
Dissemination Level:	PU
Revision:	v1.0
Abstract:	We analyze how the security of current TPM deployments would be affected if sufficiently large quantum computers were available. First, we recall the quantum algorithms that are relevant to cryptography. Then we analyze how the security of many currently deployed cryptographic schemes would be affected if we had sufficiently large quantum computers. Finally, we describe in more detail the security risks that would arise in the event of quantum adversaries in the three-use cases of the FutureTPM project: e-payment, activity tracking and device management.
Keywords:	quantum computing, quantum algorithms, security analysis

The project FutureTPM has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779391.

Editor

Alfredo Rial (UL)

Contributors (ordered according to beneficiary numbers)

SURREY, UL, UPRC

Executive Summary

The goal of FutureTPM is to design a Quantum-Resistant (QR) Trusted Platform Module (TPM) by designing and developing QR algorithms suitable for inclusion in a TPM. The algorithm design will be accompanied with implementation and performance evaluation, as well as formal security analysis in the full range of TPM environments: i.e. hardware, software and virtualization. Use cases in e-payment, activity tracking and device management will provide environments and applications to validate the FutureTPM framework.

In this deliverable, we analyze how the security of current TPM deployments would be affected if sufficiently large quantum computers were available. In order to perform this analysis, first we describe basic concepts of quantum computing. Then we show how to design quantum circuits and we describe some quantum algorithms, with particular focus on quantum algorithms that affect the security of many currently deployed cryptographic schemes. After that, we discuss how the security of those cryptographic schemes is affected. Finally, we describe more in detail the security risks that quantum computers pose in the three use cases of the FutureTPM project: e-payment, activity tracking and device management.

As result, this deliverable shows why the design of quantum-resistant TPMs is necessary. We demonstrate that the most basic security properties of currently deployed TPMs do not hold against an adversary equipped with a quantum computer.

Contents

List of Figures

Chapter 1

Introduction

Quantum computation is computing using quantum-mechanical phenomena, such as superposition and entanglement [5]. A quantum computer is a device that performs quantum computation. While a classical computer works with bits that are either 0 or 1, a quantum computer works on quantum bits (qubits), which are physical objects in a quantum superposition of two states and therefore can be both of these two states (interpreted as 0 and 1) simultaneously. More precisely, qubits are equal to 0 and 1 at the same time until their value is measured. When measured, qbits take the value 0 or 1 with some probability (not necessarily 50%) [1].
Qubits are at the hardware level of a quantum computer. The software is built on quantum algorithms. A quantum algorithm consists of transformations over quantum states (groups of qubits), or more precisely transformations of the probabilities characterising the state of superposition of the qubits. Instead of transforming zeroes and ones as a classical algorithm does, a quantum algorithm transforms quantum probabilities. Quantum probabilities are the probabilities that characterise the state of superposition of the qubits.

1.1 Quantum Supremacy and Cryptography

Quantum supremacy or “quantum advantage” is the potential ability of quantum computers to solve problems that classical computers practically cannot [13]. In computational complexity-theoretic terms, this generally means providing a superpolynomial speed-up over the best known or possible classical algorithm [12].
The security of many cryptographic schemes is based on the hardness of problems such as integer factorization and discrete logarithm computation. Although it is yet to be proved, these problems are generally believed to be hard using classical resources. For example, factoring a 232-digit number (RSA-768) utilizing hundreds of machines took two years, and it is estimated that a 1024-bit RSA modulus would take a thousand times more [8].
The most interesting quantum algorithm as far as cryptography is concerned is Shor’s algorithm. Shor’s algorithm brings an exponential speed-up for solving the factoring and discrete logarithm problems. The most widely used secure communication protocols, such as TLS, SSH or IPSec, use e.g. RSA digital signatures or Diffie-Hellman key agreements, whose security is based on the hardness of those problems. Shor’s algorithm potentially breaks the security of these algorithms and the security of cryptographic protocols widely deployed on the Internet.

1.2 Quantum-Resistant Cryptography

Quantum-resistant cryptography, also called quantum-safe or post-quantum cryptography, refers to cryptographic algorithms that are thought to be secure against an attack by a quantum computer. Currently, post-quantum cryptography research is mostly focused on the following different approaches: lattice-based cryptography, multivariate cryptography, hash-based cryptography, code-based cryptography, and supersingular elliptic curve isogeny cryptography. The security of the cryptographic algorithms developed following those approaches is based on the hardness of number-theoretic problems that are thought to be intractable by quantum computers, i.e. no efficient quantum algorithm to solve those problems is known.
One common characteristic of many post-quantum cryptography algorithms is that they require larger key sizes than cryptographic algorithms whose security is based on the hardness of integer factorization or discrete logarithm computation. There are often tradeoffs to be made in key size, computational efficiency and ciphertext or signature size.

1.3 Scope and Purpose

The purpose of this deliverable is to analyze the security risks that quantum computers pose to current TPM deployments, which use cryptographic primitives and protocols that are not quantum-resistant. In particular, we analyze in detail what security properties are affected in cryptographic protocols for the three use cases of the project: e-payment, activity tracking and device management. As a result, we show that, if the current TPM standard is used to deploy those protocols, then many basic security properties provided by current TPMs are not effective against an adversary equipped with a quantum computer. This justifies the need of designing and implementing quantum-resistant TPMs.

1.4 Deliverable Structure

This deliverable is structured as follows. Chapter 2 describes some basic concepts of quantum computing. The purpose of Chapter 2 is to provide the notation and background needed to describe how quantum algorithms work. In Chapter 3, we describe some quantum algorithms. We focus on quantum algorithms that affect the security of cryptographic primitives: Shor's algorithms for integer factorization and discrete logarithm computation and Grover's algorithm for quantum search. In Chapter 4, we describe how the security of many cryptographic primitives and protocols is affected by those quantum algorithms. We discuss in more detail the security risks that those quantum algorithms pose to cryptographic protocols for the three use cases of the FutureTPM project in Chapter 5. We conclude in Chapter 6.

1.5 Relation to Other WPs and Deliverables

This deliverable is related to Deliverable 1.1 "FutureTPM Use Cases and System Requirements", which describes the three use cases of the TPM project (e-payment, activity tracking and device management) and lists their required security properties. This deliverable uses the description of the use cases provided in D1.1 to analyze how the security of protocols for e-payment, activity tracking and device management is affected by quantum computers.

This deliverable is also related to Deliverable 2.1 “First Report on New QR Cryptographic Primitives”, which describes and compares quantum-resistant cryptographic schemes and also provides a classification of quantum security models. This deliverable refers to D2.1 when analyzing how quantum computers affect the security of non-quantum-resistant schemes and how security against quantum adversaries should be defined and analyzed.
In addition, this deliverable provides an analysis of how the security of cryptographic protocols for the three use cases of the FutureTPM project would be affected by sufficiently large quantum computers. This analysis will provide a basis for better identifying the security requirements that such systems should exhibit. This will be needed in WP3 “QR TPM Integration and Provable Security Modelling and Analysis” for security modelling and in WP4 “Run-time Risk Assessment and Vulnerability Analysis” for risk assessment.

Chapter 2

Quantum Computing

This chapter describes some basic concepts of quantum computing. First, we define qubits and systems of qubits. Then we describe quantum gates that act on single qubits and quantum gates that act on multiple qubits. Finally, we describe how to build quantum circuits by applying quantum gates. The purpose of this chapter is to introduce the notation and background needed to describe quantum algorithms in Chapter 3. The following sources have been used to write this chapter: Nielsen and Chuang [10], Aumasson [1], De Wolf [4] and Oskin [11].

2.1 Qubit

A *quantum bit* or *qubit* is the basic unit of quantum information. A qubit is a two-state quantum-mechanical system. In a classical system, a binary digit, characterized as 0 and 1, is used to represent information in classical computers. Quantum mechanics allows a qubit to be in a coherent superposition of both states at the same time.
There are two possible outcomes for the measurement of a qubit, represented by the value “0” and “1”, like a bit or binary digit. However, whereas the state of a bit can only be either 0 or 1, the general state of a qubit according to quantum mechanics can be a coherent superposition of both.
The Dirac notation $|\ \rangle$ is the standard notation for states in quantum mechanics. $|0\rangle$ and $|1\rangle$ represent the classical “0” and “1” states. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form linear combinations of states, i.e., to have a qubit in the state $|\varphi\rangle$, where $|\varphi\rangle$ is the superposition

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

where α and β are complex numbers. In other words, the state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis for this vector space.
When we measure a qubit we get either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. Naturally, $|\alpha|^2 + |\beta|^2 = 1$, since the probabilities must sum to one. Thus, in general the state of a qubit is a unit vector in a two-dimensional complex vector space.
Therefore, a qubit can exist in a continuum of states between $|0\rangle$ and $|1\rangle$, until it is measured. We emphasize again that when a qubit is measured, it only ever gives “0” or “1” as the measurement result, probabilistically. For example, a qubit can be in the state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.2}$$

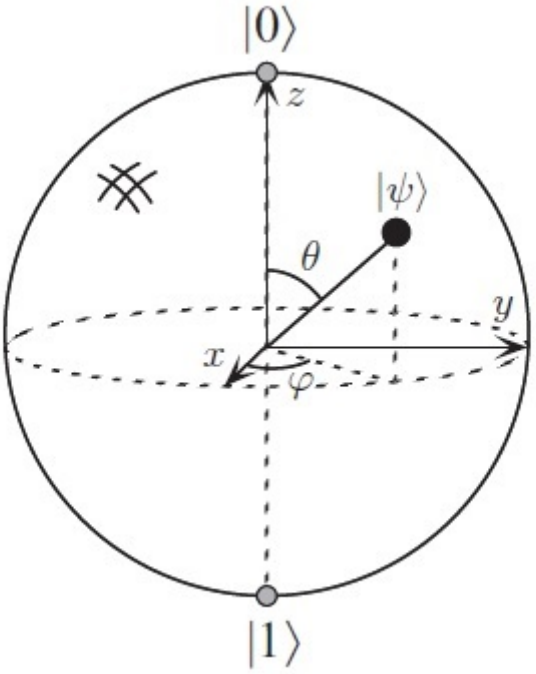

Figure 2.1: Bloch sphere representation of a qubit

which, when measured, gives the result 0 fifty percent of the times, and the result 1 fifty percent of the times. This state is sometimes denoted $|+\rangle$.
Because $|\alpha|^2 + |\beta|^2 = 1$, we may rewrite Equation 2.1 as

$$|\varphi\rangle = e^{i\lambda}\left(\cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle\right) \tag{2.3}$$

where θ, ϕ and λ are real numbers. The factor $e^{i\lambda}$ can be ommitted because it has no observable effects.
The numbers θ and ϕ define a point on the unit three-dimensional sphere. This sphere is often called the Bloch sphere and it provides a useful means of visualizing the state of a single qubit, as depicted in Figure 2.1.
A measurement of a qubit would destroy its coherence and irrevocably disturb the superposition state, i.e., measurement changes the state of a qubit, collapsing it from its superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result. For example, if measurement of $|+\rangle$ gives 0, then the post-measurement state of the qubit will be $|0\rangle$.

2.2 Multiple Qubits

Suppose that we have two qubits. A two qubit system has four computational basis states denoted $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ indicating all the possible states of a system when measured. A pair of qubits can also exist in superpositions of these four states, so the quantum state of two qubits involves associating a complex coefficient, sometimes called amplitude, with each computational basis state. Therefore, the state vector describing the two qubits is

$$|\varphi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.4}$$

The measurement result $x \in \{00, 01, 10, 11\}$ occurs with probability $|\alpha_x|^2$, with the state of the qubits after the measurement being $|x\rangle$. The condition that probabilities sum to one is therefore expressed by the normalization condition that $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$.
In a two qubit system, we could measure just the first qubit. The result would be 0 with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, and the post-measurement state would be

$$|\varphi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.5}$$

Four important two qubit states are the Bell states or EPR pairs

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}, \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.6}$$

Consider for instance the state $(|00\rangle + |11\rangle)/\sqrt{2}$. This state has the property that upon measuring the first qubit, one obtains two possible results: 0 with probability 1/2, leaving the post-measurement state $|\varphi'\rangle = |00\rangle$, and 1 with probability 1/2, leaving $|\varphi'\rangle = |11\rangle$. As a result, a measurement of the second qubit always gives the same result as the measurement of the first qubit, i.e., the measurement outcomes are correlated.
More generally, we may consider a system of n qubits. The computational basis states of this system are of the form $|x_1x_2 \ldots x_n\rangle$, where each x_i is in $\{0, 1\}$, and so a quantum state of such a system is specified by 2^n amplitudes. Therefore, a system of several hundred qubits involves a huge number of amplitudes, which cannot be stored on a classical computer.

2.3 Single Qubit Gates

Classical computer circuits consist of wires and logic gates. The wires are used to carry information around the circuit, while the logic gates perform manipulations of the information, converting it from one form to another. Consider, for example, classical single bit logic gates. The only non-trivial member of this class is the NOT gate.
A quantum computer is built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate the quantum information. A quantum NOT gate takes as input the state

$$\alpha|0\rangle + \beta|1\rangle \tag{2.7}$$

and ouptputs the corresponding state in which the role of $|0\rangle$ and $|1\rangle$ have been interchanged,

$$\alpha|1\rangle + \beta|0\rangle \tag{2.8}$$

Therefore, the quantum NOT gate acts linearly. The quantum NOT gate is represented by the matrix

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.9}$$

If the quantum state $\alpha|0\rangle + \beta|1\rangle$ is written in vector notation as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.10}$$

then the output of the quantum NOT gate can be calculated as

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{2.11}$$

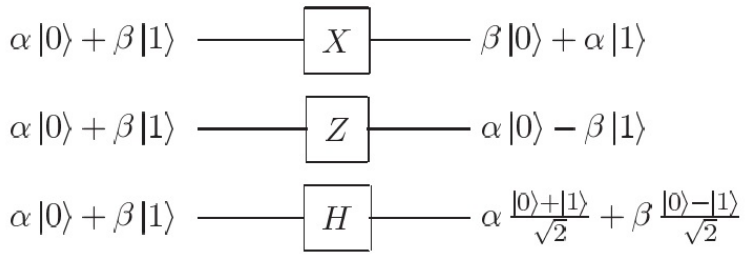

Figure 2.2: NOT, Z and Hadamard Gates

Quantum gates on a single qubit can be described by 2×2 matrices. A matrix U representing the the single qubit gate must be unitary, i.e., the conjugate transpose U^* of U is also the inverse of U. This condition is necessary to fulfill the condition that, after the gate has acted, the quantum state $|\varphi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ given as output fulfills $|\alpha'|^2 + |\beta'|^2 = 1$. This is the only condition that a matrix representing a gate must fulfill, i.e., any unitary matrix represents a valid quantum gate. Therefore, there are many non-trivial single qubit gates. Two important gates are the Z gate, which flips the sign of $|1\rangle$,

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.12}$$

and the Hadamard gate

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.13}$$

Given the Bloch sphere, it can be seen that single qubit gates correspond to rotations and reflections of the sphere. The Hadamard operation is just a rotation of the sphere about the y axis by $90°$, followed by a rotation about the x axis by $180°$. Figure 2.2 represents the circuits of the NOT, Z and Hadamard gates.
In general, it is proven that an arbitrary 2×2 unitary matrix can be decomposed as

$$U = e^{i\alpha} \begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix} \begin{bmatrix} \cos(\frac{\lambda}{2}) & -\sin(\frac{\lambda}{2}) \\ \sin(\frac{\lambda}{2}) & \cos(\frac{\lambda}{2}) \end{bmatrix} \begin{bmatrix} e^{-i\delta/2} & 0 \\ 0 & e^{i\delta/2} \end{bmatrix} \tag{2.14}$$

where α, β, λ and δ are real numbers. The second matrix is a rotation, while the first and third matrices can also be understood as rotations in a different plane. The constant multiplier $e^{i\alpha}$ is a phase shift. This decomposition can be used to describe any single qubit quantum gate.

2.4 Multiple Qubit Gates

In classical computing, an important theoretical result is that any function on bits can be computed from the composition of NAND gates alone, which is thus known as a universal gate. In quantum computing, there is a similar result: any multiple qubit logic gate may be composed from single qubit gates and a multi-qubit gate called CNOT.
Therefore, the prototypical multi-qubit quantum logic gate is the controlled-NOT or CNOT gate. This gate has two input qubits, known as the control qubit and the target qubit, respectively. The action of the gate may be described as follows. If the control qubit is set to 0, then the target qubit is left alone. If the control qubit is set to 1, then the target qubit is flipped. Figure 2.3 represents

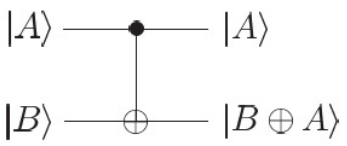

Figure 2.3: CNOT gate

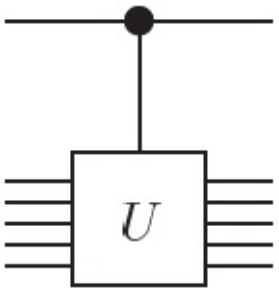

Figure 2.4: Controlled-U gate

the circuit of the CNOT gate. Its matrix representation is the following:

$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.15}$$

As for the single qubit case, the requirement that probability be conserved is expressed in the fact that U_{CN} is a unitary matrix.
The CNOT gate can be generalized as follows. Suppose U is any unitary matrix acting on some number n of qubits, so U can be regarded as a quantum gate on those qubits. Then we can define a controlled-U gate, depicted in Figure 2.4. Such a gate has a single control qubit, indicated by the line with the black dot, and n target qubits, indicated by the boxed U. If the control qubit is set to 0 then nothing happens to the target qubits. If the control qubit is set to 1 then the gate U is applied to the target qubits.

2.5 Quantum Circuits

A quantum circuit is a model for quantum computation in which a computation is a sequence of quantum gates, which are reversible transformations on an n-qubit register. As an example, let's consider the circuit in Figure 2.5, which contains three CNOT gates. The circuit is read from left to right. Each line represents a wire in the quantum circuit. (This wire does not necessarily correspond to a physical wire.) It is conventional to assume that the state input to the circuit is a

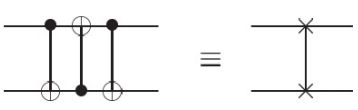

Figure 2.5: Circuit swapping two qubits, and an equivalent symbol for this circuit

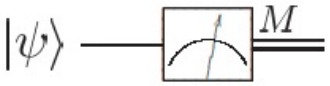

Figure 2.6: Measurement symbol

computational basis state, usually the state consisting of all $|0\rangle$s. This circuit swaps the states of the two qubits.

$$\begin{aligned} |a,b\rangle &\rightarrow |a, a \oplus b\rangle \\ &\rightarrow |a \oplus (a \oplus b), a \oplus b\rangle = |b, a \oplus b\rangle \\ &\rightarrow |b, (a \oplus b) \oplus b\rangle = |b, a\rangle \end{aligned}$$

Quantum circuits are acyclic, i.e., loops are not allowed. The FANIN operation of classical circuits, where wires are joined together with the resulting single wire containing the bitwise OR of the inputs, is not possible, since this operation is not reversible. The FANOUT operation of classical circuits, where several copies of bits are produced, is not possible either, since quantum mechanics forbids the copying of a qubit.
The measurement operation is represented by the meter symbol shown in Figure 2.6. It converts a single qubit state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ into a classical bit M, which is distinguished from a qubit by drawing it as a double wire.

Chapter 3

Quantum Algorithms

A quantum algorithm is a step-by-step procedure to calculate a function, where each of the steps can be performed on a quantum computer. In this chapter, we describe some quantum algorithms, with particular focus on those quantum algorithms that have an impact on the security of cryptographic primitives and protocols: Shor's algorithms for integer factorization and discrete logarithm computation [14] and Grover's algorithm for quantum search [6]. First, we explain that any "classical circuit" can be simulated using a quantum circuit. Second, we describe key features used in quantum algorithms, parallelism and interference, which allow some quantum algorithms to outperform their classical counterparts. Third, we describe some quantum algorithms. We describe the quantum Fourier transform and its application to quantum phase estimation, which is used in Shor's algorithms for integer factorization and discrete logarithm computation. Then we describe the quantum search algorithm by Grover. Finally, we give a brief overview on quantum complexity theory. The following sources have been used to write this chapter: Nielsen and Chuang [10], De Wolf [4] and Oskin [11].

3.1 Simulation of Classical Logic Circuits

In Chapter 2, we have described quantum circuits as sequences of quantum gates. A natural question is what class of computations can be performed by using quantum circuits.
First, we explain that any classical logic circuit can be simulated by using a quantum circuit. Quantum circuits cannot be used to directly simulate classical circuits because unitary quantum logic gates are inherently reversible, whereas many classical logic gates, such as the NAND gate, are inherently irreversible. Nevertheless, any classical circuit can be replaced by an equivalent circuit containing only reversible elements, by making use of a reversible gate known as the Toffoli gate. The Toffoli gate has three input bits and three output bits, as illustrated in Figure 3.1. Two of the bits are control bits. The third bit is a target bit that is flipped if both control bits are set to 1, and otherwise is left alone.
The Toffoli gate can be used to simulate NAND gates by setting the bits a and b to the input of the NAND gate and $c \leftarrow 1$. The third output bit is the output of the NAND gate. The Toffoli gate can also be used to simulate FANOUT by setting $a \leftarrow 1$, $c \leftarrow 0$ and b to the input bit of the FANOUT. Then the second and third output bits of the Toffoli gate are the output of the FANOUT. By simulating NAND and FANOUT, it is possible to simulate any classical logic circuit.
The inverse of a Toffoli gate is the Toffoli gate itself. Because the Toffoli gate is reversible, it can also be implemented as a quantum gate. The Toffoli gate can be described as an 8×8 unitary matrix. In conclusion, the quantum Toffoli gate can be used to simulate any classical logic circuit.

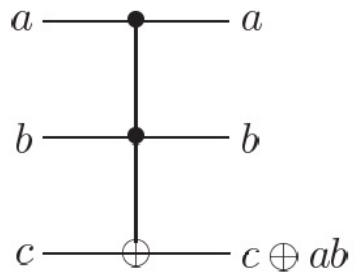

Figure 3.1: Toffoli Gate

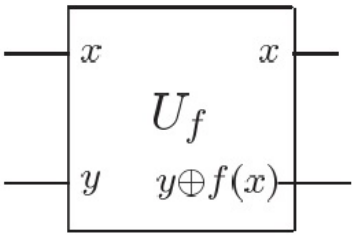

Figure 3.2: Circuit U_f

A quantum computer can also simulate a classical computer that is non-deterministic, i.e., that has the ability to generate random bits. For this purpose, it can prepare a qubit in the state $|0\rangle$, send it through a Hadamard gate to produce $(|0\rangle + |1\rangle)/\sqrt{2}$, and measure the result, which will be $|0\rangle$ or $|1\rangle$ with probability $1/2$.

3.2 Quantum Parallelism

Roughly speaking, quantum parallelism allows quantum computers to evaluate a function $f(x)$ for many different values of x simultaneously. As an example, consider a function $f(x) : \{0,1\} \rightarrow \{0,1\}$ with a one-bit domain and range. Consider the quantum circuit U_f depicted in Figure 3.2, which performs the transformation $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$. When $x \leftarrow (|0\rangle + |1\rangle)/\sqrt{2}$ and $y \leftarrow |0\rangle$, the output is the state

$$\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}} \tag{3.1}$$

As can be seen, the output state contains information about both $f(0)$ and $f(1)$, while a single circuit for $f(x)$ was evaluated once. Quantum parallelism exploits the ability of a quantum computer to be in superpositions of different states.

The idea described above can be generalized for an arbitrary number n of qubits. Consider a function $f(x)$ with n bits as input and 1 bit as output, and define the quantum circuit U_f similarly as above, with x representing n qubits. First, prepare $n + 1$ qubits in the $|0\rangle$ state. Then perform a Hadamard transform over the first n qubits. In the Hadamard transform, n Hadamard gates operate in parallel on the n bits. The result of the Hadamard transform is an equal superposition of all computational basis states for n qubits, i.e. $(1/\sqrt{2^n}) \sum_x |x\rangle$, where the sum is taken over all possible values of x. Finally, evaluate the corresponding quantum circuit U_f on input the $n + 1$

qubits, where the first n qubits underwent the Hadamard transform. The result is the state

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle \tag{3.2}$$

As can be seen, this state contains information of f(x) evaluated on all the possible values of x. Nevertheless, after measurement, only one evaluation of $f(x)$ for a single value of x can be obtained. Therefore, in order to outperform a classical algorithm, a quantum algorithm needs not only parallelism, but also a way to extract information about more than one value of $f(x)$ from superpositions of states like $\sum_x |x\rangle|f(x)\rangle$.

3.3 Deutsch's Algorithm and Quantum Interference

As before, consider a function $f(x) : \{0,1\} \rightarrow \{0,1\}$ with a one-bit domain and range. The goal of Deutsch's algorithm is to check the condition $f(0) = f(1)$. This is equivalent to check $f(0) \oplus f(1)$, i.e., if $f(0) \oplus f(1) = 0$ then $f(0) = f(1)$, and if $f(0) \oplus f(1) = 1$ then $f(0) \neq f(1)$. Deutsch's algorithm combines quantum parallelism with a property of quantum mechanics known as interference. A quantum circuit for Deutsch's algorithm is depicted in Figure 3.3. First, the Hadamard gate is applied to prepare the first qubit, on input $|0\rangle$, as the superposition of $(|0\rangle + |1\rangle)/\sqrt{2}$, and the second qubit, on input $|1\rangle$, as the superposition of $(|0\rangle - |1\rangle)/\sqrt{2}$. After that, the gate U_f is applied, which gives as result

$$\begin{aligned}
&\frac{1}{2}[|0\rangle(|f(0)\oplus 0\rangle - |f(0)\oplus 1\rangle) + |1\rangle(|f(1)\oplus 0\rangle - |f(1)\oplus 1\rangle)] \\
&\quad = \frac{1}{2}[(-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle)] \\
&\quad = (-1)^{f(0)}\frac{1}{2}[|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle](|0\rangle - |1\rangle)
\end{aligned}$$

We can ignore the global phase and therefore we have the state

$$\frac{1}{\sqrt{2}}[|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle]$$

Then, by applying the last Hadamard gate to the first qubit, we obtain

$$\begin{aligned}
&\frac{1}{2}[|0\rangle + |1\rangle + (-1)^{f(0)\oplus f(1)}|0\rangle - (-1)^{f(0)\oplus f(1)}|1\rangle] \\
&\quad = \frac{1}{2}[(1 + (-1)^{f(0)\oplus f(1)})|0\rangle + (1 - (-1)^{f(0)\oplus f(1)})|1\rangle]
\end{aligned}$$

Therefore, if we measure a 0 then $f(0) \oplus f(1) = 0$, and if we measure a 1 then $f(0) \oplus f(1) = 1$. This happens because, if $f(0) = f(1)$, the probability of measuring 1 is 0, and, if $f(0) \neq f(1)$, the probability of measuring 1 is 1. Consequently, Deutsch's algorithm allows to determine $f(0) \oplus f(1)$ by evaluating $f(x)$ only once. In contrast, classical algorithms would require two evaluations because at any given time we evaluate either $f(0)$ or $f(1)$. In a quantum computer, it is possible that these two alternatives interfere with one another to yield some global property of the function $f(x)$. The essence of the design of many quantum algorithms is that a clever choice of function and final transformation allows efficient determination of useful global information about the function.

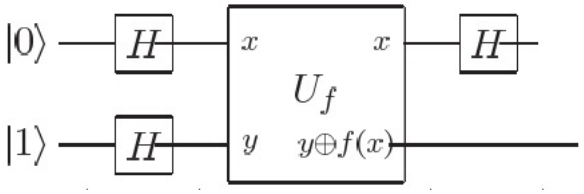

Figure 3.3: Quantum circuit for Deutsch's algorithm

3.4 An overview of Quantum Algorithms

There are three classes of quantum algorithms which provide an advantage over known classical algorithms. First, there is the class of algorithms based upon quantum versions of the Fourier transform. Shor's algorithms for factoring and discrete logarithm are an example of this type of algorithm. The second class of algorithms is quantum search algorithms. The third class of algorithms is quantum simulation, whereby a quantum computer is used to simulate a quantum system. In the next sections, we focus on the first two classes because they are relevant to cryptography.

3.5 Quantum Fourier Transform and its Applications

The quantum Fourier transform, described in Section 3.5.1, is an efficient quantum algorithm for performing a Fourier transform of quantum mechanical amplitudes. It does not speed up the classical task of computing Fourier transforms of classical data, but it enables phase estimation, i.e., the approximation of the eigenvalues of a unitary operator under certain circumstances, as described in Section 3.5.2. This allows us to solve problems such as integer factorization, as described in Section 3.5.3.

3.5.1 Quantum Fourier Transform

The discrete Fourier transform takes as input a vector of complex numbers, $x_0, \ldots, x_{N-1}$, where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers $y_0, \ldots, y_{N-1}$, defined by

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N} \tag{3.3}$$

for $k \in [0, N-1]$. We usually consider vectors of length $N = 2^n$. Then we can write

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w_n^{jk} \tag{3.4}$$

where $w_n = e^{2\pi i/N}$ is an N^{th} root of unity.
The quantum Fourier transform acts on a quantum state $\sum_{i=0}^{N-1} x_i|i\rangle$ to map it to a quantum state $\sum_{i=0}^{N-1} y_i|i\rangle$ according to the formula

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w_n^{jk} \tag{3.5}$$

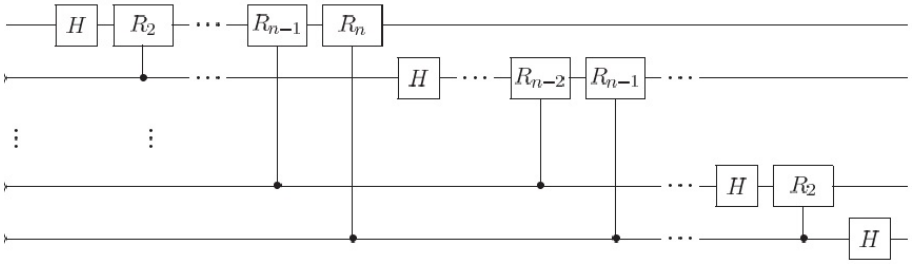

Figure 3.4: Quantum circuit for quantum Fourier transform

for $k \in [0, N-1]$ and w_n defined as above. The amplitudes y_k are the discrete Fourier transform of the amplitudes x_j.
The quantum Fourier transform can be viewed as the following unitary matrix

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & w_n & w_n^2 & w_n^3 & \dots & w_n^{N-1} \\ 1 & w_n^2 & w_n^4 & w_n^6 & \dots & w_n^{2(N-1)} \\ 1 & w_n^3 & w_n^6 & w_n^9 & \dots & w_n^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & w_n^{N-1} & w_n^{2(N-1)} & w_n^{3(N-1)} & \dots & w_n^{(N-1)(N-1)} \end{bmatrix} \tag{3.6}$$

Since the quantum Fourier transform can be represented by a unitary matrix, there exists a quantum circuit that implements it. To describe this circuit, it is useful to use the following formula to describe the quantum Fourier transform:

$$QFT(|x_1 x_2 \dots x_n\rangle) = \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle)(|0\rangle + e^{2\pi i[0.x_{n-1}x_n]}|1\rangle) \cdots (|0\rangle + e^{2\pi i[0.x_1 x_2 \dots x_n]}|1\rangle) \tag{3.7}$$

where $[0.x_1 \dots x_n] = \sum_{k=1}^{n} x_k 2^{-k}$.
A circuit to implement the quantum Fourier transform is depicted in Figure 3.4. This circuit uses the Hadamard gate H and the controlled phase gate R_n, which is described by the following matrix:

$$R_n \equiv \begin{bmatrix} 1 & 0 \\ 0 & w_n \end{bmatrix}$$

To explain why this circuit implements the quantum Fourier transform, let us consider the case of three qubits, given in Figure 3.5. Applying the Hadamard gate to the first qubit gives the state

$$\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i[0.x_1]}|1\rangle)|x_2 x_3\rangle$$

because $e^{2\pi i[0.x_1]} = -1$ when $x_1 = 1$ and $+1$ otherwise. Applying the controlled-R_2 gate produces the state:

$$\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i[0.x_1 x_2]}|1\rangle)|x_2 x_3\rangle$$

Finally, applying the controlled-R_3 produces the state

$$\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i[0.x_1 x_2 x_3]}|1\rangle)|x_2 x_3\rangle$$

Basically, each controlled-R_n gate adds an extra bit to the phase of the coefficient of the first $|1\rangle$. This procedure is similar for the remaining qubits.

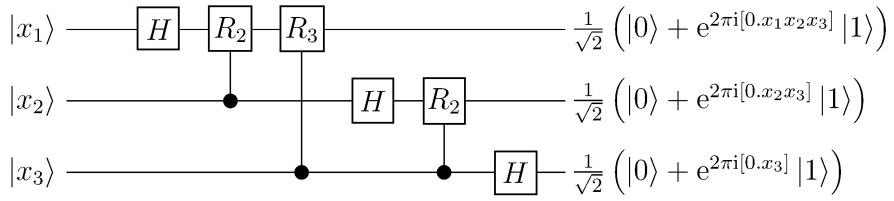

Figure 3.5: Quantum circuit for three qubit quantum Fourier transform

3.5.2 Quantum Phase Estimation

The quantum phase estimation algorithm, also called quantum eigenvalue estimation algorithm, allows one to estimate the phase θ of an eigenvalue of a unitary operator U, given the corresponding eigenvector $|\psi\rangle$. More precisely, given a unitary matrix U and a quantum state $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, the algorithm estimates the value of θ with high probability within additive error ε.

To perform the estimation, we assume that we have available black boxes (sometimes known as oracles) capable of preparing the state $|\psi\rangle$ and performing the controlled-U^{2^j} operation, for suitable non-negative integers j. The use of black boxes indicates that the phase estimation procedure is not a complete quantum algorithm in its own right. Rather, it is a subroutine to be combined with other subroutines.

The quantum phase estimation procedure uses two registers. The first register contains n qubits initially in the state $|0\rangle$. How we choose n depends on two things: the number of digits of accuracy we wish to have in our estimate for θ, and with what probability we wish the phase estimation procedure to be successful. The second register begins in the state $|\psi\rangle$, and contains as many qubits as is necessary to store $|\psi\rangle$.

The circuit for phase estimation is depicted in Figure 3.6. The circuit begins by applying a Hadamard transform to the first register. Then, the circuit applies controlled-U operations on the second register, with U raised to successive powers of two. More precisely, given a unitary operator U with eigenvector $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, we have that

$$U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}e^{2\pi i\theta}|\psi\rangle = e^{2\pi i2^j\theta}|\psi\rangle$$

$C-U$ is a controlled U gate that applies the operator U to the second register only if the control bit of the first register is $|1\rangle$. After applying all the n controlled operations $C-U^{2^j}$ with $0 \leq j \leq n-1$, and kicking back phases to the control bits in the first register, the state of the first register can be described as

$$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i2^{n-1}\theta}|1\rangle)(|0\rangle + e^{2\pi i2^{n-2}\theta}|1\rangle)\cdots(|0\rangle + e^{2\pi i2^0\theta}|1\rangle) = \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle$$

The next stage is to apply the inverse quantum Fourier transform on the first register. This is obtained by reversing the circuit for the quantum Fourier transform in Section 3.5.2. By comparing the state of the first register with Equation 3.7, we can see that, if the phase can be expressed exactly with n bits, then the result of the inverse quantum Fourier transform is $|\theta_1 \ldots \theta_n\rangle$. If the phase cannot be described with n bits, then the procedure will give a good approximation of θ with high probability.

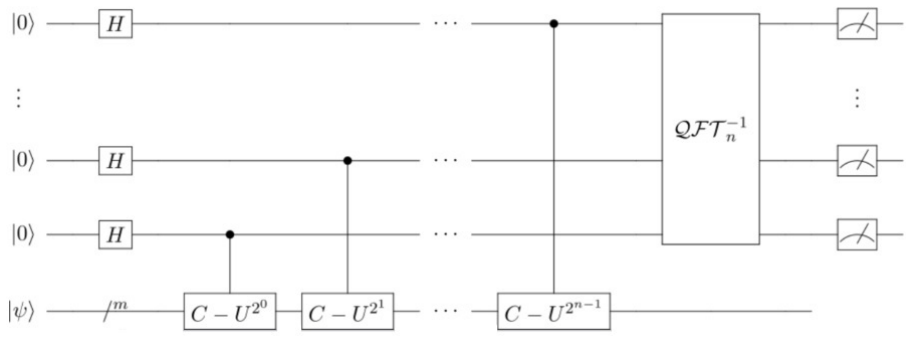

Figure 3.6: Quantum circuit for phase estimation

3.5.3 Integer Factorization and Discrete Logarithm Computation

We describe how quantum phase estimation can be used as subroutine in algorithms that solve problems of interest in cryptography, such as integer factorization and discrete logarithm computation. First, we describe an algorithm to solve the order-finding problem. Then we show how this algorithm can be used as subroutine in an algorithm for integer factorization. Finally, we show how similar ideas can be used to compute discrete logarithms and solve similar problems.

Quantum Order-Finding Algorithm

For positive integers x and N, $x < N$, with no common factors, the order of x modulo N is defined to be the least positive integer, r, such that $x^r = 1(\mathrm{mod}\ N)$. The order-finding problem is to determine the order for some specified x and N. Order-finding is believed to be a hard problem on a classical computer.
The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator

$$U|y\rangle = |xy\ (\mathrm{mod}\ N)\rangle$$

with $y \in \{0,1\}^L$.
There are two requirements to allow us to use the phase estimation procedure. First, we must have efficient procedures to implement a controlled-U^{2^j} operation for any integer j. This is satisfied by using a procedure known as modular exponentiation. In this procedure, we wish to compute the transformation:

$$\begin{aligned}|z\rangle|y\rangle &\rightarrow |z\rangle U^{z_t 2^{t-1}} \cdots U^{z_1 2^0}|y\rangle \\ &= |z\rangle|x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} y\ (\mathrm{mod}\ N)\rangle \\ &= |z\rangle|x^z y\ (\mathrm{mod}\ N)\rangle\end{aligned}$$

where t is the number of bits of the first register of the phase estimation procedure.
The modular exponentiation procedure consists of two stages. First, we use modular multiplication to compute x^{2^j} for all j up to $t-1$. The second stage consists in computing

$$x^z\ (\mathrm{mod}\ N) = (x^{z_t 2^{t-1}}\ (\mathrm{mod}\ N))(x^{z_{t-1} 2^{t-2}}\ (\mathrm{mod}\ N)) \cdots (x^{z_1 2^0}\ (\mathrm{mod}\ N))$$

which can be accomplished by performing $t-1$ modular multiplications. In total, the cost of this modular exponentiation procedure is $O(L^3)$.
The second requirement consists in preparing an eigenstate $|u_s\rangle$ with a non-trivial eigenvalue, or at least a superposition of such eigenstates. There is a way to circumvent the problem of

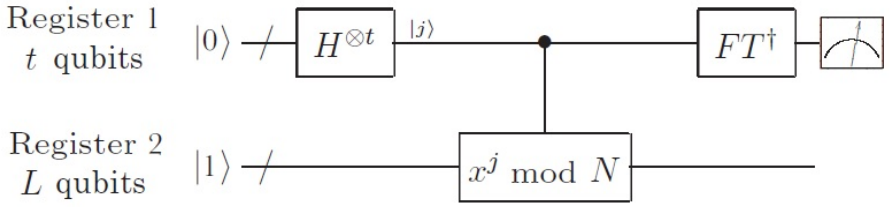

Figure 3.7: Quantum circuit for order finding

preparing $|u_s\rangle$, which consists in initializing it to $|1\rangle$, which is trivial to construct. This is based on the observation that

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

As a result, for each s in $[0, r-1]$, we wil obtain an estimate of the phase $\phi \approx s/r$. Finally, the desired value r can be obtained from ϕ by using the continued fractions algorithm.
In summary, the phase estimation procedure can be used to solve the order finding problem as shown in Figure 3.7. First, initialize the first t qubits register to $|0\rangle$ and the second L qubits register to $|1\rangle$. Second, create a superposition of states in the first register by using a Hadamard transform. Third, use the black box $U_{x,N}$ to apply the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, which gives the result

$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle \approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u_s\rangle$$

Fourth, apply the inverse Fourier transform, which gives the result

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$$

Finally, measure the first register to obtain $\widetilde{s/r}$, and apply the continued fractions algorithm to get the order r. The runtime of the algorithm is $O(L^3)$.

Integer Factorization

Given an odd composite number N, integer factorization consists in finding an integer $d \in (1, N)$ that divides N. We describe Shor's algorithm, an algorithm for integer factorization that uses the algorithm for order-finding as building block.
For the algorithm to work, we need that N is not the power of a prime, which can be tested by checking that $\sqrt[k]{N}$ is not an integer for $k \leq \log_2(N)$. When N is not the power of a prime, then it has to be the product of coprime numbers greater than 1. By the Chinese remainder theorem, the number 1 has at least 4 square roots modulo N, two of which are 1 and -1. The aim of the algorithm is to find a square root b different from 1 and -1, which will lead to a factorization of N, as explained below.
Finding b consists in finding a number a of even order r such that $a^{r/2} \not\equiv -1 \pmod N$. The quantum order-finding algorithm is used to compute the order of randomly chosen numbers a. Therefore, the algorithm for integer factorization consists of two parts. First, a reduction to the order-finding problem, which can be performed in a classical computer, and second, the quantum order-finding algorithm. The algorithm works as follows:

- Pick a random number $a < N$.
- Compute $\gcd(a, N)$. If $\gcd(a, N) \neq 1$, then return a.
- Use the quantum order finding algorithm to compute the order r of a modulo N.
- If r is odd, go back to step 1.
- If $a^{r/2} \equiv -1 \pmod{N}$, go back to step 1.
- At least one of $\gcd(a^{r/2} + 1, N)$ or $\gcd(a^{r/2} - 1, N)$ is a non-trivial factor of N.

To understand the last step, let $b = a^{r/2} \pmod{N}$ and note that, because $b^2 \equiv 1 \pmod{N}$, then N divides $b^2 - 1 = (b + 1)(b - 1)$ and thus N must have a common factor with $b + 1$ or $b - 1$. Additionally, by assumption $1 < b < N - 1$, so the common factor cannot be N itself. Therefore computing $\gcd(b + 1, N)$ or $\gcd(b - 1, N)$ must give a non-trivial factor of N. Regarding steps 4 and 5, a theorem guarantees that with probability at least $1/2$, the order r will be even and $a^{r/2} \equiv -1 \pmod{N}$ will not hold.

Discrete Logarithm Computation and Other Algorithms

The ideas behind the algorithm for order-finding can be used to solve a more general problem referred to as the hidden subgroup problem. This problem can be thought of as a generalization of the task of finding the unknown period of a periodic function, in a context where the structure of the domain and range of the function may be very intricate.
The hidden subgroup problem can be stated as follows. Let f be a function from a finitely generated group G to a finite set X such that f is constant on the cosets of a subgroup K, and distinct on each coset. Given a quantum black box for performing the unitary transform $U|g\rangle|h\rangle = |g\rangle|h \oplus f(g)\rangle$, for $g \in G$, $h \in X$, and $\oplus$ an appropriately chosen binary operation on X, find a generating set for K.
We can view the order-finding algorithm as a special case of the hidden subgroup problem where G is $(\mathbb{Z}, +)$, X is $\{a^j\}$ such that $j \in \mathbb{Z}_r$ and $a^r = 1$, K is $\{0, r, 2r, \ldots\}$ with $r \in G$, and the function is $f(x) = a^x$ such that $f(x + r) = f(x)$. In other words, order-finding can be seen as computing the period of the function $f(x) = a^x$.
In fact, it is straightforward to generalize the quantum algorithm for order-finding into a quantum algorithm for period-finding for a periodic function $f(x)$ such that $f(x) = f(x+r)$ for an unknown r. In this algorithm, we would use a black box that performs the transformation $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. The algorithm then proceeds as in the case for order-finding. Period-finding is also a concrete case of the hidden subgroup problem.
Similarly, the discrete logarithm problem can also be seen as a specific case of the hidden subgroup problem. Consider the function $f(x_1, x_2) = a^{sx_1+x_2} \bmod N$, where the variables are integers and r is the smallest integer such that $a^r \bmod N = 1$. This function is periodic since $f(x_1+l, x_2-ls) = f(x_1, x_2)$ with a two-tuple period $(l, -ls)$. This function allows one to solve the discrete logarithm problem, i.e., given a and $b = a^s$, calculate s. To this end, we would instantiate the quantum algorithm for the hidden subgroup problem with the function $f(x_1, x_2) = b^{x_1}a^{x_2}$ and a black box that performs the operation $U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|y \oplus f(x_1, x_2)\rangle$. The main modification to the algorithm is that we need two t qubit registers because the period is a tuple of two values. From the period, we can compute the discrete logarithm s.
More concretely, the discrete logarithm problem is a special case of the hidden subgroup problem where the group G is $(\mathbb{Z}_r \times \mathbb{Z}_r, +(\bmod r))$, X is $\{a^j\}$ such that $j \in \mathbb{Z}_r$ and $a^r = 1$, and K is $(l, -ls)$, with $l, s \in \mathbb{Z}_r$. The function f is $f(x_1, x_2) = a^{sx_1+x_2}$ such that $f(x_1 + l, x_2 - ls) = f(x_1, x_2)$.

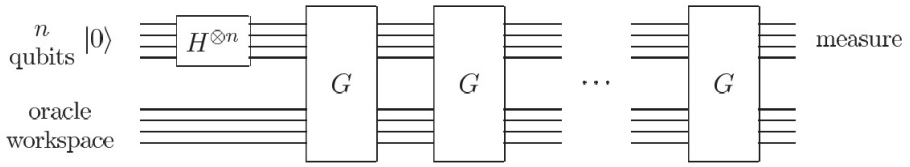

Figure 3.8: Quantum circuit for the search algorithm

The algorithm for discrete logarithm computation is the most relevant to cryptography. Nevertheless, other problems can also be seen as a special case of the hidden subgroup problem, such as finding the order of a permutation or even Deutsch's problem in Section 3.3.

3.6 Quantum Search Algorithms

A classical algorithm that searches a list of N elements by checking every element in the list until finding the right one needs $O(N)$ operations. Remarkably, there is a quantum search algorithm, known as Grover's algorithm [7], that needs $O(\sqrt{N})$ operations in order to perform this task. In this section, we describe Grover's algorithm. First, we define the search problem in terms of an oracle.

3.6.1 The Oracle

Suppose that we wish to search a specific element in a search space of $N = 2^n$ elements. A convenient way to represent the search problem is to consider a function f, whose domain is an integer in $[0, N-1]$, such that $f(x) = 1$ if x is a solution to the search problem and $f(x) = 0$ otherwise. Note that we search for the index of the element rather than the element itself.
Consider a quantum oracle defined by the unitary operator O as follows:

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle$$

where $|x\rangle$ is the index register and $|q\rangle$ is a single oracle qubit that is flipped if $f(x) = 1$. In the quantum search algorithm, it is useful to apply the oracle with the oracle qubit initially in the state $(|0\rangle - |1\rangle)/\sqrt{2}$. If x is not a solution to the search problem, applying the oracle to that state does not change the state, but, if x is a solution, then the state becomes $-|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$. Therefore, the action of the oracle is

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{O} (-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Because the state of q does not change, the action of the oracle can be described as

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}|x\rangle$$

The oracle thus marks the solution to the search problem by shifting the phase of the solution. The construction of the oracle depends on the concrete instance of the search problem.

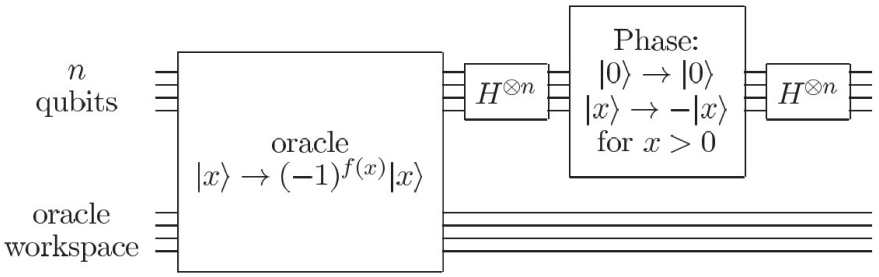

Figure 3.9: Quantum circuit for the Grover iteration

3.6.2 The Search Algorithm

The goal of the search algorithm is to find a solution to the search problem by applying the oracle as less as possible. The algorithm is depicted in Figure 3.8. It uses an n-qubit register initialized to $|0\rangle^{\otimes n}$. First, the Hadamard transform is applied to obtain an equal superposition state

$$|\psi\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle \tag{3.8}$$

Then the search algorithm repeatedly applies a subroutine G known as the Grover iteration, which is depicted in Figure 3.9. This subroutine works as follows. First the oracle O is applied. Second, a Hadamard transform is applied. Third, a conditional phase shift in which every computation basis except $|0\rangle$ receives a phase shift of -1 is applied. Fourth, a Hadamard transform is applied again. The combined effect of steps 2, 3 and 4 is

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I$$

where $|\psi\rangle$ is the superposition of states in Equation 3.8. I denotes the identity matrix and $\langle\psi|$ is the Hermitian conjugate, represented as a row vector, of $|\psi\rangle$. Consequently, the Grover iteration can be written as $(2|\psi\rangle\langle\psi| - I)O$.

3.6.3 Geometric Proof of Correctness

The Grover iteration can be regarded as a rotation in the two-dimensional space spanned by the starting vector $|\psi\rangle$ and the state consisting of a uniform superposition of solutions to the search problem. The initial state can be expressed as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$$

where, if $\sum_x^{'}$ denotes the sum over all x that are solutions to the search problem and $\sum_x^{''}$ over all x that are not, $|\alpha\rangle$ and $|\beta\rangle$ are the normalized states:

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_x^{''} |x\rangle,\ |\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_x^{'} |x\rangle$$

The oracle O performs a reflection about the vector $|\alpha\rangle$ in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$, i.e., $O(a|\alpha\rangle + b|\beta\rangle) = a|\alpha\rangle - b|\beta\rangle$. Similarly, $2|\psi\rangle\langle\psi| - I$ performs a reflection in the plane defined by

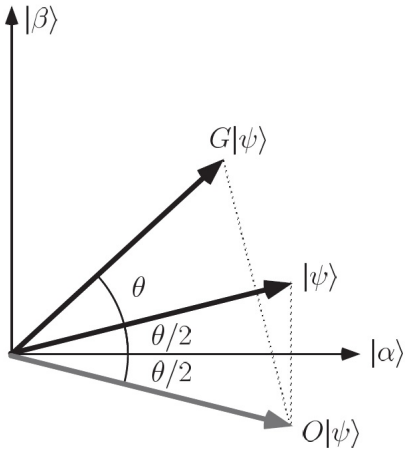

Figure 3.10: The action of a Grover iteration. First, the oracle O reflects the state vector $|\psi\rangle$ about the state $|\alpha\rangle$. Second, the operation $2|\psi\rangle\langle\psi| - I$ reflects the result about $|\psi\rangle$.

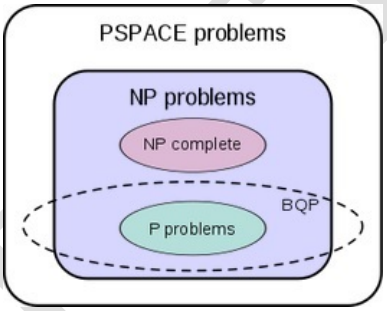

Figure 3.11: Relation between BQP and classical complexity classes.

$|\alpha\rangle$ and $|\beta\rangle$ about the vector $|\psi\rangle$. The product of this two reflections is a rotation. Both reflections are depicted in Figure 3.10. Let $\cos\theta/2 = \sqrt{(N-M)/N}$, so that $\psi = \cos(\theta/2)|\alpha\rangle + \sin(\theta/2)|\beta\rangle$. The rotation angle is θ. After k applications of the Grover iteration, the state is

$$G^k|\psi\rangle = \cos\left(\frac{2k+1}{k}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle$$

Therefore, repeated applications of G gets the state closer to $|\beta\rangle$. When G is repeated sufficient times, a measurement outputs with high probability one of the solutions to the search problem superposed in $|\beta\rangle$. A performance analysis indicates that $O(\sqrt{N/M})$ Grover iterations are sufficient, in contrast to $O(N/M)$ oracle calls required in a classical computer. It is proven that Grover's algorithm is asymptotically optimal [2].

3.7 Quantum Complexity Theory and Classes

It is still not proven that quantum computers are more powerful than classical computers, despite algorithms such as Shor's algorithm for integer factorization suggest that they are. In order to study what problems can be solved efficiently using a quantum computer, quantum complexity theory is developed.
In classical computers, two of the most important complexity classes are P and NP. P is the class of decision problems that can be solved in polynomial time by a deterministic Turing machine. NP

is the class of decision problems for which a solution can be verified by deterministic computations in polynomial time. P is a subset of NP, but it is not proven that $\mathrm{P} \neq \mathrm{NP}$. NP-complete problems are a subclass of NP problems. An algorithm to solve an NP-complete problem can be adapted to solve any problem in NP. It is not known whether a quantum computer can solve efficiently any problem in NP. Factoring is believed to be in NP and not in P, but factoring is not known to be NP-complete.
Another two classes of problems are PSPACE and BPP. PSPACE is the class of decision problems solvable by a Turing machine with a polynomial amount of space. PSPACE is believed to be larger than NP. BPP is the class of problems solvable by a probabilistic Turing machine in polynomial time with a bounded error probability for all instances. BPP contains P, but it is not known if they are equal.
In quantum complexity theory, BQP is the class of decision problems that can be solved by quantum computers in polynomial time, with a bounded error probability for all instances. The exact relation between BQP and P, NP and PSPACE is not known. It is known that BQP includes P, and it is known that BQP is included in PSPACE [10]. Figure 3.11 shows the suspected relation between those classes.

Chapter 4

Impact of Quantum Computers on Cryptography

Typically, the security of a cryptographic scheme is analyzed as follows. First, a security definition for a cryptographic task is provided. The security definition describes all the security properties that any cryptographic scheme for that task must provide. In doing so, the security definition also describes the type of adversary considered, i.e., it defines the computational resources of the adversary and the capabilities of the adversary (e.g., whether the adversary can control the network, whether he can corrupt parties, the level of corruption, etc.).
Second, one or more security assumptions are defined. Security assumptions specify anything upon which the security of a cryptographic scheme is based. There are several types of security assumptions. For example, it is possible to assume certain properties of the network, that certain parties are trusted or that they do not collude, etc. Particularly relevant in the analysis of the impact of quantum algorithms on cryptography are assumptions related to the hardness of a problem, i.e. assumptions that state that solving a problem by using some given resources is intractable.
Third, a security analysis that shows that a cryptographic scheme for a certain task fulfills the security definition for that task is provided. Normally, this security analysis consists of theorems that state that a certain security property holds if one or more assumptions hold, and the corresponding security proofs that demonstrate those theorems. Security proofs frequently are proofs by contraction, i.e., a proof shows that, if the type of adversary considered in the definition was able to break a security property, then such an adversary could be used to show that a security assumption does not hold anymore.
For example, to analyze the security of a signature scheme, first one chooses a security definition for the unforgeability property, which specifies the type of forgeries that are not allowed and the type of adversaries against which the unforgeability property must hold. Second, one defines a security assumption, which is typically a problem believed to be intractable given the resources of the adversary. Third, one enunciates a theorem that states that a signature scheme is unforgeable if a given problem is intractable, and one proves that, if an adversary with the defined resources produces a forgery that is not allowed by the definition, then this adversary can be used to solve the problem assumed to be intractable.
Many widely-used cryptographic schemes fulfill a security definition where the adversary is defined as having limited resources. For example, the adversary is defined as a probabilistic polynomial time Turing machine, which roughly speaking means that the resources of the adversary are limited to those of classical computers. Thanks to that, it is possible to prove that a cryptographic scheme fulfills the security definition by assuming the intractability of a problem for which

no efficient classical algorithm is known.
In Chapter 3, we have described some quantum algorithms that outperform their classical counterparts. In particular, Shor's algorithms for integer factorization and discrete logarithm computation present an exponential speedup in comparison to classical algorithms for those tasks. Although it is yet to be proven that the class BPP is a proper subset of BQP, algorithms such as Shor's algorithm are a strong suggestion that this is the case.
The security of several widely-used cryptographic schemes relies on the hardness of integer factorization, discrete logarithm computation, or problems that can be efficiently solved with access to oracles for the integer factorization or discrete logarithm computation problems (e.g. the RSA problem or the Diffie-Hellman problem). That is, the intractability of integer factorization or discrete logarithm computation is assumed in order to prove that a scheme fulfills the security definition. Obviously, such schemes are not secure anymore when large quantum computers are available.
A brief overview of non-quantum-resistant schemes can be found in [9]. The impact of Shor's algorithm on the security of cryptographic primitives whose security proof is based on the assumption that integer factorization or discrete logarithm computation (or related problems such as the RSA problem or the Diffie-Hellman problem) are intractable is clear: those cryptographic primitives are not secure anymore because the assumption does not hold anymore since those problems can be solved in polynomial time by a quantum computer that executes Shor's algorithm. On the other hand, Grover's algorithm mainly impacts the security of symmetric key cryptographic primitives such as hash functions or block ciphers because quantum search can be used to perform an exhaustive search for the secret key. Because Grover's algorithm provides a quadratic speedup over classical algorithms, to maintain their security level in the presence of quantum computers, the number of bits of the secret keys of symmetric-key primitives should be doubled.
Quantum-resistant cryptographic schemes are schemes whose security is believed to hold even if large quantum computers are available. At first glance, one could think that any scheme is quantum-safe if its security is proven by assuming the intractability of a problem for which no efficient quantum algorithm is known. Nevertheless, if the security definition that the scheme is proven to fulfill still considers a type of adversary with "classical" capabilities, this is not the case. In general, in addition to using security assumptions that are not threatened by quantum computers, it is necessary to provide a new security definition that considers "quantum adversaries".
Deliverable D2.1 "First Report on New QR Cryptographic Primitives" gives a detailed description of quantum-resistant cryptographic primitives. Additionally, Section 2.3 of D2.1 presents a classification of quantum security models. This classification explains the type of "quantum adversary" that is considered by a security definition for a cryptographic primitive. We refer to D2.1 for more detail.

Chapter 5

Security Risks of Quantum Algorithms on Non-Quantum-Resistant Cryptographic Protocols

As described in Sections 3 and 4, when a large scalable quantum computer becomes reality, quantum algorithms can break all the asymmetric cryptographic algorithms and reduce the security levels of all the symmetric cryptographic algorithms, that are used by TPM 2.0. More specifically, TPM 2.0 supports the following cryptographic primitives (the reference numbers are suggested in the D1.1 document of this project):

- *[SR.1.1.1]* Pseudorandom Number Generator (PNG).
- *[SR.1.1.2]* Key generation and storage functionalities;
- *[SR.1.1.3]* Hash functions;
- *[SR.1.1.4]* Message Authentication Code (MAC);
- *[SR.1.1.5]* Symmetric encryption;
- *[SR.1.1.6]* Digital signatures;
- *[SR.1.1.7]* Public key encryption and key exchange;
- *[SR.1.1.8]* Direct Anonymous Attestation (DAA).

If any of these primitives makes use of an asymmetric key algorithm and the security of the algorithm is based on either the factorisation problem or the discrete logarithm problem, if it makes use of a hash function with a small block length, or if it makes use of a MAC or a block cipher with a small key size or block length, this primitive is non-quantum-resistant. In this section, we discuss the security risks for the three use cases under the assumption that some of these primitives may be non-quantum-resistant, in the quantum computer age, following the detailed investigation and description of the corresponding security risks of the current TPM 2.0 commands, APIs and interfaces and the appropriate mitigation measures that need to be explored in the context of FutureTPM [3].

5.1 Security Risks for ePayment Protocols

Following the discussion in D1.1, in the use case of secure mobile wallet and payments, a TPM is embedded in a mobile platform that is owned by a user of the FreePOS service. This case makes use of key generation and storage functionalities, hash functions, MAC, symmetric encryption, and digital signatures. If some of these primitives is non-quantum-resistant, the following security risks can be predicted:

1. A user stores his login information (referred to as an access token) in the TPM. If the key generation and storage functionalities in the TPM are non-quantum-resistant, the user cannot securely log in to the FreePOS service.

2. A user also stores other sensitive information in the TPM, such as the information that allows the user to authenticate against the secure PCI compliant infrastructure (referred to as an OAuth bearer token). Therefore, if the key generation and storage functionalities in the TPM are non-quantum-resistant, the user cannot tokenise their credit card securely.

3. A user encrypts his financial transaction data (referred to as the local SQLite Database) by using a symmetric encryption algorithm. The encryption key is generated and stored by the TPM. If either the key generation and storage functionalities or the symmetric encryption algorithm is non-quantum-resistant, the financial transactions of the user will not be private.

4. A user not only lets the TPM protect their financial transactions with data confidentiality, but also with data integrity that is achieved using a signature, either using a MAC or an asymmetric key digital signature. As any signature scheme, a hash function is also used. So, if any of the hash function, MAC or digital signature scheme is non-quantum-resistant, the user cannot be sure that their financial transactions are not tampered with.

5. A user may also want to avoid revealing their credit card data to the server. If the access token and OAuth bearer token are securely stored in the TPM, this can be achieved; otherwise, if the key generation and storage functionalities of the TPM is non-quantum-resistant, this is impossible.

5.2 Security Risks for Activity Tracking Protocols

Following the discussion in D1.1, in the use case of personal activity and health tracking, a TPM is embedded in the computing platform of a S5PersonalTracker, a S5Tracker Analytics Engine and a S5DataAnalysis. Each of these three entities have security requirements that rely on the cryptographic functionalities of the TPM. This use case makes use of all the cryptographic primitives that are required for FutureTPM, from *[SR.1.1.1]* to *[SR.1.1.8]*. If some of these primitives is non-quantum-resistant, the following security risks can be predicted:

1. A user connects to the S5Tracker Analytics Engine via his S5PersonalTracker, he then uploads his data to the engine and lets the engine analyse it. In this case, the user wants to make sure that the S5Tracker Analytics Engine is trusted. This is achieved by using the TPM remote attestation service. The user also wants to maintain his anonymity from the engine in order to meet his privacy requirements, the TPM DAA functionality will be utilized to achieve that. In addition, the user wants his data to be kept in the engine with confidentiality and integrity, which can be realized with the use of the TPM authenticated encryption

functionality. These comprehensive user requirements are reflected on all of the TPM cryptographic primitives that FutureTPM aims to improve in order to attain quantum resistance. Obviously, if the encryption and signature algorithms of the TPM are non-quantum-resistant, the user data cannot be handled by the S5Tracker Analytics Engine safely and if the TPM DAA scheme is also non-quantum-resistant, the user privacy will not be preserved.

2. An S5 data analyst connects to the S5Tracker Analytics Engine, loads data belonging to a user from the engine and analyses the data on behalf of the user. In this case, the data analyst wants to be sure that the engine is trusted and that he is also trusted by the engine. This is done via the TPM remote attestation service. Furthermore, when the analyst wants to get access to the original data even if the data is encrypted in the engine, the TPM key management, data encryption and integrity check functionalities are used to carry out this task. Clearly, if the TPM cryptographic algorithms are non-quantum-resistant, neither of these security requirements can be met.

3. An S5Tracker Analytics Engine connects to both the S5PersonalTracker of each user and the devices of each S5 data analyst. The engine needs to have a two-way trust relationship with both the S5PersonalTracker and with the S5 data analyst; as mentioned before, this is achieved using the TPM remote attestation service. Since the user data can be stored in the engine either anonymously or non-anonymously, the engine needs to handle the DAA signature verification. As we discussed before, if the TPM cryptographic algorithms are non-quantum-resistant, the security requirements of the S5Tracker Analytics Engine are impossible to attain.

5.3 Security Risks for Device Management Protocols

Following the discussion in D1.1, in the use case of device management, embedding a TPM in the computing platform of each network router allows the network management system (NMS) to precisely identify the controlled devices and determine whether those devices are able to correctly perform management commands. Both authentication and remote attestation rely on the cryptographic primitives of the TPM (from *[SR.1.1.1]* to *[SR.1.1.7]*). If some of these primitives are non-quantum-resistant, the following security risks can be predicted:

1. An attacker retrieves the private key of a legitimate device and replaces that device with a rogue one that is able to successfully authenticate with the NMS.

2. An attacker forges integrity reports, hiding the true state of a device. The NMS would continue to trust the device and allow it to process user data.

3. An attacker that retrieved the private key of a device is also able to intercept and modify the data protected by a trusted channel between the device and the NMS.

In summary, the security and privacy of these three use cases rely on the security of the cryptographic functionalities of the TPM. If the TPM does not support the quantum-resistant cryptographic algorithms, in the post-quantum age, neither of these use cases can hold the required security and privacy.

Chapter 6

Conclusion

The goal of FutureTPM is to design a Quantum-Resistant (QR) Trusted Platform Module (TPM) by designing and developing QR algorithms suitable for inclusion in a TPM. In this deliverable, we have analyzed how the security of current TPM deployments would be affected if sufficiently large quantum computers were available. In particular, we have described the security risks that quantum computers pose in the three use cases of the FutureTPM project: e-payment, activity tracking and device management. As result, this deliverable shows why the design of quantum-resistant TPMs is necessary. We demonstrate that the most basic security properties of currently deployed TPMs do not hold against an adversary equipped with a quantum computer.
Although the main goal of FutureTPM is the specification and design of a quantum-resistant TPM in line with the same functionalities as the current TPM, the identified QR algorithms of FutureTPM will be valid beyond that and, therefore, can be exploited by other similar hardware and software platforms that have a similar goal as the TPM. These include the IBM Powerchips, Hardware Security Modules (HSM), Trusted Execution Environments as defined by the GlobalPlatform, the ARM TrustZone, Intel's SGX, etc. In this regard, we highlight that all the security risks related to the development of sufficiently large quantum computers, which we have identified in this deliverable, affect not only protocols that use the current TPM standard but also protocols that use those aforementioned secure hardware solutions. In fact, as a cryptographic engine, a TPM is similar to those security devices. While these devices do not necessarily support trusted computing functions and may have their own special applications, they typically make use of the same or similar cryptographic primitives and algorithms, and consequently the identified security risks also apply to them.

References

[1] Jean-Philippe Aumasson. The impact of quantum computing on cryptography. *Computer Fraud & Security*, 2017(6):8–11, 2017.

[2] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[3] The FutureTPM Consortium. Futuretpm reference architecture (d1.2). 2018.

[4] Ronald De Wolf. Quantum computing: Lecture notes, 2013.

[5] Neil Gershenfeld and Isaac L Chuang. Quantum computing with molecules. *Scientific American*, 278(6):66–71, 1998.

[6] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.

[7] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[8] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K Lenstra, Emmanuel Thomé, Joppe W Bos, Pierrick Gaudry, Alexander Kruppa, Peter L Montgomery, Dag Arne Osvik, et al. Factorization of a 768-bit rsa modulus. In *Annual Cryptology Conference*, pages 333–350. Springer, 2010.

[9] Vasileios Mavroeidis, Kamer Vishi, Mateusz D Zych, and Audun Jøsang. The impact of quantum computing on present cryptography. *arXiv preprint arXiv:1804.00200*, 2018.

[10] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[11] Mark Oskin. Quantum computing-lecture notes. *Notes to CSE590mo, University of Washington*, 2004.

[12] Anargyros Papageorgiou and Joseph F Traub. Measures of quantum computing speedup. *Physical Review A*, 88(2):022316, 2013.

[13] John Preskill. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.

[14] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134, 1994.