Key Distribution

Kathrin Sayk

Geboren am 23. Oktober 1988 in Oldenburg

30th September 2011

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Nitin Saxena

Hausdorff Center for Mathematics

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

Chapter 1

Introduction

1.1 Motivation

The overall goal of cryptography is to secure communication. Since this is a very general statement, let us try to break it down in order to get closer to a solution. There are two main ingredients here: secrecy and authenticity.
Secrecy means, that it should not be possible for anybody but the intended receiver to read our message. This is where encryption schemes come into play. Instead of sending the original message or the plaintext, we encrypt it to a ciphertext, in a way that only someone with the right key can decrypt it again.
Authenticity is just as important, but easily overlooked. One might assume, that the possession of the valid decryption key, is enough to confirm the identity of the receiver. As we will see later it turns out that it is not. There are different ways to achieve authenticity, namely authentication codes and digital signatures.
The tasks of providing secrecy and authenticity are very different from each other, which is why we will analyze them separately. For secure communication it will be necessary to use both, encryption and authentication schemes.
At first glance it seems to be sufficient to require secrecy and authenticity, but since the goal is to actually communicate, i.e. transfer information from one party to another, we also need to keep an eye on practicality. Usually we will assume that any party involved can run polynomial time and space algorithms, no matter whether we are talking about the legitimate parties or an adversary.

We also just mentioned keys. We need keys to encrypt, decrypt, sign and verify. There are schemes, where the key is private, i.e. known only to the legitimate parties, and there are schemes that involve public keys as well.
The problem we are trying to solve here is, how to distribute such secret keys. That is how two parties A and B come into possession of a shared secret key and how can they be sure who they share this key with. So again the goal is achieving secrecy and authenticity.

Overview. In the following sections of this chapter we will introduce the basics of cryptography. We will define what private and public key encryption schemes are and take a closer look on security requirements.
In the second chapter we will start by introducing the Diffie-Hellmann key distribution protocol and then discuss its security and possible improvements. Section 2.2 gives some examples on how to use the idea behind the DH key exchange to tackle various cryptographic primitives, including public key cryptography and conference key sharing. We then turn to the issue of authenticity. We define message authentication codes and digital signature schemes and then use those in section 2.4 to describe key distribution protocols that, based on a set of reasonable assumptions give us both secrecy and authenticity in the setting of adversaries with polynomial time and space resources. Section 2.5 concludes the chapter on classical key distribution.
In chapter 3 we deal with the opportunities that quantum computation offers for key distribution. The first section contains the basic mathematical concepts and definitions in quantum computation. Section 3.2 is dedicated to modeling errors and defining measures to judge how well a quantum state is preserved during quantum computations. In section 3.3 we look at ways to detect and correct errors. We will do that by starting out with classical error correction and then use similar concepts to apply in quantum computation. Once we have all that, we are ready to introduce the BB84 protocol, a quantum key distribution protocol, in section 3.4. We will prove the security of the BB84 protocol, without making any assumption on the computational capabilities of the adversary.

1.2 Basics of Cryptography

The very basic setting where cryptographic schemes become necessary is this: Two (or sometime more) parties A and B want to communicate some private information, but they only have access to public or insecure communication channels, like the internet or a telephone line. Depending on the properties of the channel an eavesdropper E may be able to listen in on the conversation of A and B or even inject messages on to the channel.
So any cryptographic scheme should achieve two things: Privacy and Authenticity. We will now take a look on different approaches to achieve privacy and then turn to authenticity later in chapter two. This section is based on chapters six and seven of [1].

1.2.1 Private Key Cryptography

Definition 1.2.1 (Private Key Encryption Scheme)**.** *Let P denote the set of possible plaintexts and C be the set of possible ciphertexts, both finite. A private key encryption scheme is a triple $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ of polynomial time algorithms with the following properties:*

1. *$\mathcal{K}$ generates random keys of a certain length l. The set K of all possible outputs of $\mathcal{K}$ is finite and called the keyspace.*

2. *On input of a key $k_0 \in K$ and a plaintext $p_0 \in P$ the encryption algorithm $\mathcal{E}$ computes a ciphertext $c_0 \in C$. c_0 is not necessarily unique, i.e. encryption algorithm may be probabilistic.+*

3. *On input of a key $k_0 \in K$ and a ciphertext $c_0 \in C$ the decryption algorithm $\mathcal{D}$ recovers the plaintext $p_0 \in P$, i.e.*

$$\mathcal{D}(k, \mathcal{E}(k, p)) = p \quad \forall\ k \in K,\ p \in P.$$

In private key cryptography the parties involved all need to be in possession of the same secret key in order to be able to successfully communicate. But how do we distribute such a secret key? We cannot just send it over an insecure channel and encrypting it also does not work, since then the receiving party will not be able to decrypt it. There are quite a few variations of this problem. We will start out assuming that there is no previously shared information between the participating parties. Diffie and Hellman proposed a solution to that setting and also gave rise to public key cryptography in their truly revolutionary paper [2].

Security Requirements. The objective of cryptography is to exchange secret messages that are safe from any eavesdropper. We assume that the encryption and decryption algorithms are publicly known. So we definitely want it to be impossible to recover the plaintext or the key from any exchanged ciphertexts, even partial information on both the key and the plaintext should not be accessible to the eavesdropper. The notion of perfect secrecy implies just that:

Definition 1.2.2 (Perfect Secrecy)**.** *For $p \in P$, $c \in C$ let $\Pr[p]$ denote the probability that the plaintext p was encrypted and $\Pr[p|c]$ denote the probability of p being encrypted, knowing the ciphertext c was communicated. Then a private key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ achieves perfect secrecy if and only if*

$$\Pr[p] = \Pr[p|c] \quad \forall\ p \in P,\ c \in C$$

The idea behind perfect secrecy is, that the knowledge of the ciphertext bears no advantage in recovering the plaintext or the key.

Example (One Time Pad)**.** The one time pad was first introduced by Frank Miller in 1882. The idea behind it is rather simple: K, P and C are all given by the same set $\{1, 0\}^n$ of all binary n-bit strings.
To encrypt a plaintext $p \in P$ choose a random key $k \in K$. For a $p \in P$ and $k \in K$ the corresponding ciphertext is given by $c = \mathcal{E}(k, p) := p \oplus k$, where $\oplus$ denotes bit-wise addition modulo 2. To decrypt c again simply calculate $p = \mathcal{D}(k, c) := c \oplus k$.

The one time pad is proven to achieve perfect secrecy. The only information an adversary can derive from the the ciphertext, is the length of the key, which is not really an advantage, since all keys have the same length. The downside

here however is that the one time pad is not really practical: The key has to be as long as the message and for every message a new key is necessary.

1.2.2 Public Key Cryptography

The first published public key encryption scheme was developed by Diffie and Hellman in their paper [2]. In public key schemes it is not necessary for the participants to be in posession of a shared secret key. Here each party holds pair of keys (e, d), that is a public key e for encryption and a private key d for decryption. So encryption can be performed by anyone, but only the designated recipient to whom the used encryption key belongs, knows the corresponding decryption key and can decrypt the message.
Public key schemes are especially practical when more than two parties are involved, since all it takes to receive secret messages is a pair of keys and sending is not a problem either, since all the encryption keys and the algorithms are public anyway.
Formally a public key encryption scheme looks a lot like a private key scheme:

Definition 1.2.3 (Public Key Encryption Scheme)**.** *Let P denote the set of possible plaintexts and C be the set of possible ciphertexts, both finite. A public key encryption scheme is a triple $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ of probabilistic polynomial time algorithms with the following properties:*

1. *$\mathcal{K}$ generates random key pairs (e_0, d_0), each key of a certain length l_e and l_d respectively. e_0 and d_0 are corresponding encryption and decryption keys with respect to $\mathcal{E}$ and $\mathcal{D}$. e_0 is called the public key and d_0 the private key. The set K of all possible outputs of $\mathcal{K}$ is finite and called the keyspace.*

2. *On input of a public key e_0 and a plaintext $p_0 \in P$ the encryption algorithm $\mathcal{E}$ computes a ciphertext $c_0 \in C$. c_0 is not necessarily unique.*

3. *On input of a private key $d_0 \in K$ and a ciphertext $c_0 \in C$ the decryption algorithm $\mathcal{D}$ recovers the plaintext $p_0 \in P$, i.e.*

$$\mathcal{D}(d, \mathcal{E}(e, p)) = p \quad \forall\ (e, d) \in K,\ p \in P.$$

Since both the encryption and the decryption algorithm can be probabilistic here, the equality in the third condition only holds with high probability for certain schemes.

Security Requirements. What we require of a public key encryption scheme is basically the same as what we require in private key cryptography, namely secrecy and authenticity. The main differences are that here the eavesdropper cannot only read the ciphertexts sent on the channel, but also knows the public

key. So it should not be possible to retrieve the private key from the public key or get any information on how to decrypt from encryption, since the adversary can encrypt any plaintext he likes now.

A "secure" public key scheme is often compared to sending messages in non-transparent envelopes, that can only be opened by a designated recipient. From the outside every envelope looks the same, i.e. it is impossible to distinguish between the envelopes of two different plaintexts. This property is called indistinguishability. For a formal definition check [1, p. 125]

In chapter two we will introduce a public key encryption scheme based on the Diffie-Hellmann secret key exchange.

Classical Cryptography today. Most private and public key encryption schemes in use today rely on the hardness of solving some mathematical problem, like integer factoring or computing discrete logarithms. Their security relies on the assumption that the underlying problem is hard, which has usually not been proven and there are known ways for computationally unbounded adversaries to break those schemes.
That does not mean, that classical cryptography is insecure in general. Luckily there are no computationally unbounded adversaries so far. Scheme are considered to be secure, if adversaries with polynomial resources in time and space can only break them with a small probability.
The rise of quantum computers makes things worse for classical cryptography though: There are already polynomial time algorithms to solve integer factoring and discrete logarithms on quantum computers. What seems to still be safe in the sense, that it is hard to break even with the help of quantum computation, is lattice based cryptography.

Chapter 2

Classical Key Distribution

As we have seen in chapter 1, private key encryption schemes require a private key. So how do we get one? The Diffie-Hellman (DH) secret key exchange is one way to do it. The problem with that protocol is, that it does not achieve authenticity. In a symmetric setting, where the legitimate parties do not have any kind of information advantage over the adversary, it is simply not possible to perform an authenticated and secret key exchange. The DH protocol achieves secrecy, but not authenticity. The DH protocol also gives rise to public key encryption scheme, in fact Diffie and Hellman were among the first to introduce public key cryptography in [2]. We will also see how to distribute the same key to a number of different parties using ideas of the DH protocol.
Authenticated key exchange however requires a preexisting secret between A and B, usually a secret key. We will see some protocols to share secret keys in an authenticated manner in that setting.

2.1 The DH Secret Key Exchange

The DH Secret Key Exchange Protocol enables two parties A and B to obtain a shared secret key. It does not require any previously shared information and is generally assumed to be secure, since the only known way to break it so far is to compute discrete logarithms in $\mathbb{Z}_p^*$ for some large prime p.

2.1.1 The DH-Protocol

In this protocol A and B communicate over a public channel. For now we assume that the adversary E only has passive access, i.e. E is able to listen, but cannot inject any messages onto the channel. In that setting, the DH-Protocol also achieves authenticity, since the adversary has no way of communicating with the legitimate parties. This restriction is not always realistic however, and therefore it does not solve the key distribution problem entirely. For now we focus on achieving secrecy.

The protocol:

1. A and B agree on some prime p and a generator g of $\mathbb{Z}_p^*$. Both g and p are publicly known.
2. A chooses $a \in_R \mathbb{Z}_{p-1}$ and sends $g^a \mod p$ to B.
3. B chooses $b \in_R \mathbb{Z}_{p-1}$ and sends $g^b \mod p$ to A.
4. A and B compute $g^{ab} \mod p = (g^a)^b \mod p = (g^b)^a \mod p$.

Most private key encryption schemes require a key of a certain length l. Since a and b are chosen at random from $\mathbb{Z}_p^*$ the length of $g^{ab} \mod p$ varies. Hence it will be necessary to agree on a way to extract an l-bit key from the resulting DH-key g^{ab}, for example one might use the l most significant bits. But while assuming, that retrieving the whole DH-key from the values $g^a \mod p$ or $g^b \mod p$ is hard, making that assumption for subsets or even single bits of $g^{ab} \mod p$ is a different story.

2.1.2 Security of the DH-Protocol

Any adversary E knows $g^a \mod p$, $g^b \mod p$, g and p. Therefore the security of the secret key depends on the hardness of computing $g^{ab} \mod p$ on those inputs.
The obvious and so far only known way to approach this problem is to compute either a or b from $g^a \mod p$ or $g^b \mod p$ respectively, but that requires solving the discrete log problem in $\mathbb{Z}_p^*$, which is believed to be hard. It is still an open question, whether breaking the DH-protocol is equivalent to computing discrete logarithms in $mathbbZ_p^*$. For a more detailed examination of this question, see [4].

Definition 2.1.1 (The DH-Problem)**.** *Computing the function DH*$(g^a \mod p, g^b \mod p) := g^{ab} \mod p$ *on inputs* $g^a \mod p$ *and* $g^b \mod p$ *for a prime* p *and a generator* g *of* $\mathbb{Z}_p^*$ *is called the DH-Problem (DHP) in* $\mathbb{Z}_p^*$.

Definition 2.1.2 (The Discrete Log Problem)**.** *Computing the function DLOG*$(g^a \mod p) := a$ *on input* $g^a \mod p$ *for a prime* p *and a generator* g *of* $\mathbb{Z}_p^*$ *is called the discrete log problem (DLP) in* $\mathbb{Z}_p^*$.

Both of the above problems can be generalized to arbitrary finite groups, for the purpose of this paper however it suffices to consider the DHP and the DLP in $\mathbb{Z}_p^*$.
The DHP and the DLP are assumed to be hard for suitable choices of the prime p, i.e. it is believed that there is no algorithm, which can solve the DLP or the DHP in polynomial time in the size of p. The most efficient algorithms to solve the DLP so far have a runtime of $O(\sqrt{p})$.
Ideally p should be at least a 1024 bit prime, s.t. $p = 2q + 1$ for a prime q. For a lower bound on computing discrete logarithms using generic algorithms, check [5].

Equivalence of the DHP and the DLP In his paper [4] Maurer showed that the DHP is equivalent to the DLP, if some additional string of information S is given. S has length $2\log(p)$ and depends solely on p. The equivalence even holds for arbitrary finite groups G, with S depending on the size of G.

Authentication poses a problem. For the DH-Protocol, we assumed, that E only has passive access to A and B's channel. If we allow E to also inject messages onto the channel authentication becomes a problem. E could simply take A's or B's place in the protocol by intercepting all of B's messages and sending her own instead. Therefore the legitimate parties can never be sure, who they are communicating with.
A possible solution to this problem would be using digital signatures in combination with time stamps. There are also several protocols that provide secure authentication and key distribution, then the parties involved already share a secret key. We will take a closer look on how to authenticate messages in section 2.3.

2.1.3 Computing the most significant bits of the DH-function

As mentioned above not every bit of $\mathrm{DH}(g^a \mod p, g^b \mod p) = g^{ab} \mod p$ will necessarily be used to serve as the secret key of A and B. In the following we are going to examine the bit security of the DH-function to check, under which conditions using only subsets of bits of $g^{ab} \mod p$ is secure under the DH-assumption. Indeed it will turn out, that recovering the k most significant bits of $g^{ab} \mod p$ is as hard as computing the whole thing, where k is an integer depending on the length of p. To prove that, we first need some further definitions. Sections 2.1. and 2.2., including definitions, theorems and proofs, is based on the paper [6] by Boneh and Venkatesan.

Definition 2.1.3 (The Most-Significant-Bit function)**.** *Let p be a prime, $n := \lceil \log(p) \rceil$ and $k \in \mathbb{N}$. Define $MSB_k(x) := t$ to be the integer t, such that $t\frac{p}{2^k} \leq x < (t+1)\frac{p}{2^k}$.*
This also gives us an integer t' such that $\mid x - t' \mid < \frac{p}{2^k}$ by setting $t' := \lceil t\frac{p}{2^k} \rceil$. In later calculations $MSB_k(b)$ will denote t' for simplicity.

Definition 2.1.4 (The Hidden Number Problem)**.** *Define $O_{\alpha,g}(x) := MSB_k(\alpha g^x \mod p)$ to be an oracle that computes the k most significant bits of $MSB_k(\alpha g^x \mod p)$ on input x. Then computing the hidden number α given access to $O_{\alpha,g}(x)$ in polynomial time in $\log(p)$ is called the hidden number problem in $\mathbb{Z}_p^*$.*

There are two versions of the HNP:

1. The sampling version: $\mathrm{O}_{\alpha,g}(x)$ can only be queried on random inputs $x \in \mathbb{Z}_p^*$.
2. The chosen-value version: $\mathrm{O}_{\alpha,g}(x)$ can only be queried on any chosen value $x \in \mathbb{Z}_p^*$.

How is this connected to the DHP? An efficient algorithm computing $\mathrm{MSB}_k(g^{ab})$ from $g^a \mod p$ and $g^b \mod p$ can be used to evaluate the function $\mathrm{O}_{g^{ab},g^b}(x) = \mathrm{MSB}_k(g^{ab}(g^b)^x \mod p)$ for arbitrary x.
Set $\alpha := g^{ab}$ and $h := g^b$. Once we can compute $\mathrm{O}(x) = \mathrm{MSB}_k(\alpha h^x \mod p)$, we can also compute $\mathrm{O}'(t) = \mathrm{MSB}_k(\alpha t \mod p)$ for a number of values of t , which we will use to compute $\alpha = g^{ab}$ itself.

Lemma 2.1.5. *Let $k = \lceil\sqrt{n}\rceil + \lceil\log n\rceil$, given an efficient algorithm to compute $MSB_k(g^{ab} \mod p)$ on inputs $g^a \mod p$, $g^b \mod p$, we can evaluate the function $O(x) = MSB_k(\alpha h^x \mod p)$, where $\alpha = g^{ab}$ and $h = g^b$ for any $x \in \mathbb{Z}_{p-1}$.*

Proof. Clear. □

Theorem 2.1.6. *Let $\alpha \in \mathbb{Z}_{p-1}$, $O'(t) = MSB_k(\alpha t \mod p)$ as above with $k = \lceil\sqrt{n}\rceil + \lceil\log n\rceil$ and $d = 2\sqrt{n}$. Then there exists a deterministic polynomial time algorithm A such that*

$$\Pr_{t_1,...,t_d}[\ A(t_1,...,t_d, O(t_1),...,O(t_d)) = \alpha] \geq \frac{1}{2},$$

where the probability is taken over random values $t_1,...,t_d \in \mathbb{Z}_{p-1}$.

To prove the theorem, we first need some further definitions and lemmata:

Definition 2.1.7 (Lattice)**.** *A d-dimensional lattice with base $\{b_1,...,b_d\}$ is a set of points $\mathcal{L} = \{\sum_{i=1}^{d} t_i b_i \mid t_i \in \mathbb{Z} \ \forall\ i\}$, where $b_1,...,b_d \in \mathbb{R}^m$ are linearly independent vectors. $\mathcal{L}$ is called a full rank lattice if $m = d$.*

The following lemma was discovered by Babai in [7] using the lattice basis reduction algorithm of Lenstra, Lenstra and Lovasz [8]. It allows us to find a lattice point which is approximately the closest to some vector v, given a lattice $\mathcal{L}$.

Lemma 2.1.8. *Given a full rank lattice $\mathcal{L} \subseteq \mathbb{R}^d$ and a point $v \in \mathbb{R}^d$ there is an algorithm with polynomial runtime in d to find a lattice point $w \in \mathcal{L}$ such that*

$$||v - w|| \leq\ 2^{\frac{d}{4}} \min\{||v - b|| \mid b \in \mathcal{L}\}.$$

We will construct a lattice $\mathcal{L}$ and a vector u, such that we can recover the hidden number α. We do this by showing that any vector sufficiently close to u is of a certain form with probability $\leq \frac{1}{2}$, this form makes it easy to retrieve α.

Let $t_1,...,t_d$ be chosen uniformly and independently at random from $[1, p-1]$ and $a_1 = \mathrm{O}(t_1),...,a_d = \mathrm{O}(t_d)$ the corresponding numbers such that
$\forall\ i\ \ |(\alpha t_i \mod p) - a_i|$.
Let $\mathcal{L}$ be the (d+1)-dimensional lattice with basis

$$B = \{(p,0,...,0,0)^T, (0,p,...,0,0)^T, ..., (0,0,...,p,0)^T, (t_1,t_2,...,t_d,\tfrac{1}{p})^T\}$$

It can easily be seen, that the vector $v_\alpha = (\alpha t_1 \mod p,\ \alpha t_2 \mod p,\ ...,\ \frac{\alpha}{p})^T$ lies in $\mathcal{L}$ by multiplying the last basis vector with α and subtracting the right multiples of the other vectors .
Notice that $\forall\ i \leq d \mid \alpha t_i \mod p - a_i \mid < \frac{p}{2^k}$. Define $u := (a_1,\ ...,\ a_d,\ 0)^T$, then $||v_\alpha - u|| \leq \sqrt{d+1}\frac{p}{2^k}$ and therefore $\min\{||u-w|| \mid w \in \mathcal{L}\} \leq \sqrt{d+1}\frac{p}{2^k}$.

We need one last lemma for the proof of the theorem 2.1.6.:

Lemma 2.1.9. *Let $d := 2\lceil\sqrt{n}\rceil$, $\lambda := \frac{1}{2}\sqrt{n}+3$ and $\alpha \in [1, p-1]$ fix. Let $\mathcal{L}$ be the lattice constructed above, $v_\alpha = (\alpha t_1 \mod p,\ \alpha t_2 \mod p,\ ...,\ \frac{\alpha}{p})^T$ with integers $t_1, ..., t_d \in [1, ..., p-1]$ and $u = (a_1, ..., a_d, 0)^T$. Let v be a vector satisfying $||u-v|| < \frac{p}{2^\lambda}$, then v is of the form*

$$v = (t_1\beta \mod p,\ ...,\ t_d\beta \mod p,\ \frac{\beta}{p}) \text{ and } \alpha \equiv \beta \mod p.$$

with probability at least $\frac{1}{2}$.

Proof. Define the modular distance between two integers β and γ as

$$\text{dist}_p(\beta,\gamma) = \min_{b\in\mathbb{Z}} |(\beta-\gamma)-bp| = \min_{b\in\mathbb{Z}}[\min\{(\beta-\gamma) \mod p,\ p-(\beta-\gamma) \mod p\}]$$

e.g. $\text{dist}_p(1,p) = \min_{b\in\mathbb{Z}} |1-p-bp| = 1$.

For $\beta, \gamma \in \{1, ..., p-1\}$, $\beta \neq \gamma \mod p$ and integers t chosen uniformly at random from $[1, p-1]$ consider

$$\begin{aligned}\Pr_t[\text{dist}_p(\beta t,\ \gamma t) > \frac{2p}{2^\lambda}] &= \Pr_t[\frac{2p}{2^\lambda} < t(\beta-\gamma) \mod p < p - \frac{2p}{2^\lambda}] \\ &= \frac{\lfloor p - \frac{2p}{2^\lambda}\rfloor - \lceil\frac{2p}{2^\lambda}\rceil}{p-1} \geq \frac{p - \frac{2p}{2^\lambda} - \frac{2p}{2^\lambda} - 2}{p-1} \\ &= \frac{p}{p-1}(1 - \frac{4}{2^\lambda} - \frac{2}{p}) > 1 - \frac{5}{2^\lambda}.\end{aligned}$$

This follows, since $(\beta-\gamma)t \mod p \neq 0\ \forall\ t \in \{1, ..., p-1\}$ and therefore for every $x \in \{\lceil p - \frac{2p}{2^\lambda}\rceil,\ \lceil p - \frac{2p}{2^\lambda}\rceil + 1, ..., \lfloor\frac{2p}{2^\lambda}\rfloor\} \subseteq \{1, ..., p-1\}$ there is a $t \in \{1, ..., p-1\}$ with $(\beta-\gamma)t \mod p = x$.

Recall that every lattice point has the form

$$v = (\beta t_1 - b_1 p \ \ ... \ \ \beta t_d - b_d p \ \ \frac{\beta}{p})$$

for some $\beta, b_1, ..., b_d \in \mathbb{Z}$.

Consider $v \in \mathcal{L}$ with $||u-v|| < \frac{p}{2^\lambda}$ and $\beta \equiv \alpha \pmod p$. Then $\alpha t_i \equiv \beta t_i - b_i p \pmod p$ and $\alpha t_i - (\beta t_i - b_i p) \in p\mathbb{Z}$.

Since $|\alpha t_i \mod p - a_i| < \frac{p}{2^\lambda}$ and $|\beta t_i - b_i p - a_i| < ||v-u|| < \frac{p}{2^\lambda}$, it follows that $|\beta t_i - b_i p| \in \{0, ..., p-1\}\ \forall\ i$.

Now examine the probability, that there exists a lattice point contradicting the theorem:
For $\alpha \not\equiv \beta \pmod p$, we have

$$\begin{aligned}\Pr[\ ||u-v|| > \frac{p}{2^\lambda}\ ] &\geq \Pr[\ \exists\ i:\ \mathrm{dist}_p(t_i\beta, t_i\alpha) > \frac{2p}{2^\lambda}\ ] \\ &= 1 - (1 - \Pr_t[\ \mathrm{dist}_p(\beta t,\ \gamma t)\ >\ \frac{2p}{2^\lambda}\ ])^d \\ &\geq 1 - (\frac{5}{2^\lambda})^d,\end{aligned}$$

hence $\Pr[\ ||v-u|| < \frac{p}{2^\lambda}\ ] \leq (\frac{5}{2^\lambda})^d$.

Since there are $p-1$ values for $\beta \mod p$ such that $\alpha \not\equiv \beta (\mod p)$, we obtain

$$\begin{aligned}\Pr[\ \exists\ v \in \mathcal{L}:\ ||v-u|| < \frac{p}{2^\lambda} \text{ and } \beta \not\equiv \alpha \mod p\ ] \leq \\ \leq (p-1)(\frac{5}{2^\lambda})^d = \frac{p-1}{2^{d(\lambda - \log(5))}} < \frac{p-1}{2^{\log(p)+1}} < \frac{1}{2}.\end{aligned}$$

□

Proof of Theorem 2.1.6. Recall $\lambda = \frac{1}{2}\sqrt{n} + 3$, $d = 2\lceil\sqrt{n}\rceil$ and $k = \lceil\sqrt{n}\rceil + \lceil\log n\rceil$. With the lattice construction from above we can now find a vector $v \in \mathcal{L}$ such that

$$||v-u|| \leq 2^{\frac{d}{4}} \min\{||b-u||\ :\ b \in \mathcal{L}\}$$

in polynomial time.

Since

$$\begin{aligned}\min\{||b-u|| \mid b \in \mathcal{L}\}\ &\leq 2^{\frac{d}{4}}\sqrt{d+1}\frac{p}{2^k} = 2^{\frac{d}{4}+\frac{1}{2}\log(d+1)-k}p \\ &= 2^{\frac{1}{2}\log(2\lceil\sqrt{n}\rceil+1)-\frac{1}{2}\lceil\sqrt{n}\rceil-\lceil\log n\rceil}p \\ &< 2^{-(\frac{1}{2}\sqrt{n}+3)}p = \frac{p}{2^\lambda},\end{aligned}$$

we have $||v-u|| < \frac{p}{2^\lambda}$ and therefore

$$\Pr[\ v = v_\alpha = (r_1, ..., r_d, \frac{\alpha}{p})^T\ ] > \frac{1}{2},$$

so that we can recover the hidden number α in polynomial time. □

2.1.4 The small Generator Case

We just showed, that the k most significant bits of the DH-key are as hard to compute as the entire secret, given an adversary who is able to query an MSB-oracle on random inputs.
Now we allow the adversary to query a slightly modified MSB-oracle on chosen values. For this setting, the length of the fixed generator g of $\mathbb{Z}_p^*$ makes all the difference. The result will give rise to a modified version of the DH-protocol, which uses small generators in order to increase bit-security.
For a generator g of $\mathbb{Z}_p^*$ let the significant bit function $\mathrm{SB}_g(x \mod p) := t$, where t is an integer satisfying

$$t\frac{p}{g} \leq x \mod p \leq (t+1)\frac{p}{g}.$$

Then $t \in \{0, ..., g-1\}$ and SB_g returns at most $\log_2(g)$ bits. So for $g = 2$ we have $\mathrm{SB}_g(x) = \mathrm{MSB}_1(x)$.

Theorem 2.1.10. *Let $\alpha \in \mathbb{Z}_{p-1}$ and O be the function defined by $O(x) := SB_g(\alpha g^x \mod p)$, where g is a generator of $\mathbb{Z}_p^*$. Then there exists an algorithm A, which finds α in polynomial time in the size of p, given oracle access to O.*

Proof. The algorithm to recover α, given oracle access to the function $\mathrm{O}(x)$, is quite simple. The idea is to define a lower bound L and an upper bound U, so that $L \leq \alpha < U$ and $U - L = \frac{p}{g^{i+1}}$.
In every iteration i increases by one, the algorithm stops, when $U - L < 1$, so we get polynomial runtime in $\log p$. The algorithm:

1. Query $\mathrm{O}(0) = t_0$ and set $L_0 := t_0\frac{p}{g}$, $U_0 := (t_0+1)\frac{p}{g}$, $i := 0$.
2. If $U_i - L_i < 1$ return $\alpha = \lceil L_i \rceil$, else $i := i + 1$.
3. Query $\mathrm{O}(i) =: t_i$ and set $L_i := L_{i-1} + t_i\frac{p}{g^{i+1}}$, $U_i := L_{i-1} + t_i\frac{p}{g^{i+1}}$, go to 2.

In step 1 we have

$$\begin{aligned} & t_0\frac{p}{g} \leq \alpha g^0 \mod p < (t_0+1)\frac{p}{g} \\ \Leftrightarrow\ & 0 \leq \alpha g - t_0 p < p \\ \Leftrightarrow\ & \alpha g \mod p = \alpha g - t_0 p = \alpha g - g L_0. \end{aligned}$$

For $i = 1$ we have $\mathrm{O}(1) = t_1$, such that

$$\begin{aligned} & t_1\frac{p}{g} \leq \alpha g \mod p < (t_1+1)\frac{p}{g} \\ \Leftrightarrow\ & t_1\frac{p}{g} \leq \alpha g - t_0 p < (t_1+1)\frac{p}{g} \\ \Leftrightarrow\ & 0 \leq \alpha g^2 - t_0 p g - t_1 p < p \\ \Leftrightarrow\ & \alpha g^2 \mod p = \alpha g^2 - t_0 p g - t_1 p = \alpha g^2 - g^2 L_1. \end{aligned}$$

Iteratively we get $\alpha g^i \mod p = \alpha g^i - g^i L_{i-1}$. In step 3 the oracle function O returns t_i, such that

$$\begin{aligned} & t_i \frac{p}{g} \leq \alpha g^i \mod p < (t_i + 1)\frac{p}{g} \\ \Leftrightarrow\ & t_i \frac{p}{g} \leq \alpha g^i - g^i L_{i-1} < (t_i + 1)\frac{p}{g} \\ \Leftrightarrow\ & L_{i-1} + t_i \frac{p}{g^{i+1}} \leq \alpha < Li - 1 + (t_i + 1)\frac{p}{g^{i+1}} \\ \Leftrightarrow\ & L_i \leq \alpha < U_i. \end{aligned}$$

The fact, that $U_i - L_i = \frac{p}{g^{i+1}}$ completes the proof of the theorem.

□

It is important to notice, that we deal with the chosen-value version of the HNP here. The algorithm only works, if the oracle O can be queried on any chosen value $x \in \mathbb{Z}_{p-1}$.

2.1.5 A modified DH-Protocol

In the algorithm above an output of at most $\log_2(g)$ already suffices to efficiently recover the hidden number α, i.e. the $\log_2(g)$ most significant bits of $g^{ab} \mod p$ are already as hard to compute, as $g^{ab} \mod p$ in whole.
It directly follows from Artin's conjecture [12] that there are infinitely many primes p, for which 2 is a primitive root mod p, that is for which 2 is a generator of $\mathbb{Z}_p^*$. This gives rise to a modified version of the DH-Protocol, first suggested by Boneh and Venkatesan in [6]. Here computing the most significant bit of the hidden number is as hard as computing the entire secret.
In the following protocol 2 will be fixed as the generator of $\mathbb{Z}_p^*$. A will choose a second generator g, such that she knows an $x \in \mathbb{Z}_p^*$ with $2^x \mod p = g$ and $\gcd(x, p-1) = 1$. The gcd-condition ensures that there exists an $x^{-1} \in \mathbb{Z}_p^*$ such that $xx^{-1} \mod p - 1 = 1$, hence A can compute $2^y \mod p$ on input g^y.

The protocol:
As in the original version of the DH-Protocol A and B first fix a prime p, which is public knowledge.

1. A chooses $x \in_R \mathbb{Z}_{p-1}$ with $\gcd(x, p-1) = 1$ and sends $g := 2^x \mod p$ to B.

2. B chooses $y \in_R \mathbb{Z}_{p-1}$ and sends $g^y \mod p$ to A.

3. A and B compute $\alpha = \mathrm{DH}(g^y, 2) = \mathrm{DH}(g^y, g^{x^{-1}}) = 2^y \mod p$.

The shared secret key α is safe in the way, that it s not efficiently computable from the messages sent by A and B, even the most significant bit of $\mathrm{DH}_g(g^y, 2)$ is as hard to recover, as the whole DH-key. The following corollary states this:

Corollary 2.1.11. *If there exists a polynomial time algorithm A to compute* $MSB_1(DH_g(g^y, 2))$ *from* g *and* g^y*, then there is an algorithm to compute* $DH_g(g^y, 2)$ *itself in polynomial time and thereby break the modified DH-Protocol.*

Proof. The corollary follows directly from Theorem 2.1.10. □

2.2 Applications to various cryptographic Primitives

We will now take a look at a number of different cryptographic primitives. First there is a public key encryption scheme, ElGamal's public key cryptosystem. Then we introduce Okamoto's conference key sharing system, which enables one party A to distribute the same secret key to any number of parties, who want to participate in the conference. The last primitive we will be looking at is a way to directly pass a message, without sharing a private key, or publishing a public key beforehand.
All of the following protocols are based on the DH-Protocol. Therefore their security also relies on the assumption, that the DH-function is hard to compute. In conclusion they can also be reduced to the HNP for special choices of the hidden multiplier α. Thus we can show that computing the k most significant bits of any of the following functions is as hard as computing the function in its entirety by applying Theorem 2.1.6.
In this section the prime p and the generator g of $\mathbb{Z}_p^*$ will be fixed and publicly known. All calculations are done in $\mathbb{Z}_p^*$.

2.2.1 ElGamal's Public Key Cryptosystem

This public key encryption scheme provides a way for two parties A and B to directly and securely communicate. A shared secret key is not necessary, but in order for A to send a message to B, B needs to publish a public key first.

The protocol:

1. B chooses $b \in_R \mathbb{Z}_{p-1}$ as his secret key and publishes (g, g^b, p) as his public key.
2. A chooses $a \in_R \mathbb{Z}_{p-1}$ and sends $(g^a, m(g^b)^a)$ to B.
3. B computes the original message $m = \frac{m(g^b)^a}{(g^a)^b}$.

We define the ElGamal-function $\mathrm{EL}_g(g^b, g^a, mg^{ab}) := m$. Breaking this scheme is equivalent to evaluating the ElGamal-function at arbitrary values.
Any eavesdropper can observe g^a, g^b and mg^{ab}, so computing $\mathrm{DH}(g^a, g^b)$ suffices to recover the secret message m. It is also possible to recover m using an algorithm, which computes $\mathrm{MSB}_k(\mathrm{EL}_g(.))$:

Corollary 2.2.1 (Boneh, Venkatesan)**.** *Let* $k = \lceil \sqrt{n} \rceil + \lceil \log n \rceil$*. If there exists a polynomial time algorithm A to compute* $MSB_k(EL_g(g^b, g^a, mg^{ab}))$*, there is a polynomial time algorithm to compute the function in its entirety.*

Proof. The idea is, to use the algorithm A to compute the function $\mathrm{O}(x) = \mathrm{MSB}_k(\alpha h^x \mod p)$. That reduces the problem to an HNP, which can be solved in polynomial time by applying Theorem 1.1.6.

Set $h := g^b$ and $\alpha := m$, then

$$\begin{aligned} O(-x) &= \mathrm{MSB}_k(\alpha h^{-x} \mod p) = \mathrm{MSB}_k(m(g^{-xb} \mod p)) \\ &= \mathrm{MSB}_k(\mathrm{EL}_g(g^{a+x}, g^b, (mg^{-xb})g^{(a+x)b})) = \mathrm{MSB}_k(\mathrm{EL}_g(g^{a+x}, g^b, mg^{ab})) \\ &= \mathrm{A}(g^{a+x}, g^b, mg^{ab}) \end{aligned}$$

Using $\mathrm{O}(x)$ as an oracle, we can find the hidden multiplier $\alpha = m$.

□

2.2.2 Okamoto's Conference-Key Sharing System

Here one of the parties, e.g. A, would like to share the same secret key with multiple parties in order to have a conference. This cannot be achieved by the DH-protocol, since random choices of both parties influence the outcome of the DH-key and therefore the outcome of the shared secret key α. In the setting of more than two parties wanting to communicate, it is a lot easier if every party uses the same key for encryption. Otherwise it would be necessary for every pair of participants to share a secret key.

The protocol:

1. Any party B that wants to partake in the conference chooses $b \in_R \mathbb{Z}_{p-1}$ and sends g^b to A.
2. A chooses $a \in_R \mathbb{Z}_{p-1}$ and sends $(g^b)^a$ to B.
3. B computes $g^a = (g^{ba})^{b^{-1}}$, which will serve as the conference key.

Because A can fully control the outcome of the secret key, she can distribute the same key to every party. As above in the DH-protocol, the conference key will only be a subset of the bits of g^a. In that case, the participants also need to agree on a procedure to extract the conference key from g^a beforehand.

We define the Okamoto-function $\mathrm{OK}_g(g^{ba}, g^b) := g$. Breaking this scheme is equivalent to evaluating the Okamoto-function at arbitrary values.

Corollary 2.2.2. *Let $k = \lceil \sqrt{n} \rceil + \lceil \log n \rceil$. If there exists a polynomial time algorithm A to compute $MSB_k(OK_g(g^{ba}, g^b))$, there is a polynomial time algorithm to compute the function in its entirety.*

Proof. In analogy to the proof above, we use an algorithm A to compute the oracle-function $\mathrm{O}(x)$ in the following way:

Let $\alpha := g^a$ and $h := g$, then

$$\begin{aligned} O(x) &= \mathrm{MSB}_k(\alpha h^x \mod p) = \mathrm{MSB}_k(g^a g^x \mod p) \\ &= \mathrm{MSB}_k(\mathrm{OK}_g(g^{b(a+x)}, g^b)) = \mathrm{A}(g^{b(a+x)}, g^b) \end{aligned}$$

$g^{b(a+x)} = g^{ba}(g^b)^x$ is easily computable, since g^{ba}, g^b and x are known. As above we can recover $\alpha = g^a$ by applying Theorem 1.1.6.

□

2.2.3 Shamir's Message Passing Scheme

Like ElGamal's scheme the Shamir Message Passing Scheme enables two parties A and B to safely communicate without a previously existing shared secret. This protocol even works without publishing a public key first. We assume A wants to send a message m to B.

The protocol:

1. A chooses $a \in_R \mathbb{Z}_{p-1}$ and sends m^a to B.
2. B chooses $b \in_R \mathbb{Z}_{p-1}$ and sends $(m^a)^b)$ to A.
3. A computes $((m^a)^b)^{a^{-1}}$ and sends it to B.
4. B computes $m = (((m^a)^b)^{a^{-1}})^{b^{-1}}$.

We define the Shamir-function $\mathrm{SH}(m^{ab}, m^a, m^b) := m$. Breaking this scheme is equivalent to evaluating the Shamir-function at arbitrary values.

Corollary 2.2.3 (Boneh, Venkatesan)**.** *Let $k := \lceil\sqrt{n}\rceil + \lceil\log n\rceil$. If there exists a polynomial time algorithm A to compute $MSB_k(SH(m^{ab}, m^a, m^b))$, there is a polynomial time algorithm to compute the function in its entirety.*

Proof. Again we use the same idea as above.

Define $\alpha := m$ and $h := m^b$, then

$$\begin{aligned} \mathrm{O}(x) &= \mathrm{MSB}_k(\alpha h^x \mod p) \\ &= \mathrm{MSB}_k(m(m^b)^x \mod p) \\ &= \mathrm{MSB}_k(m^{1+bx} \mod p) \\ &= \mathrm{MSB}_k(SH((m^{1+bx})^{\frac{ab}{1+xb}}, (m^{1+bx})^{\frac{a(1+xb)}{1+xb}}, (m^{1+bx})^{\frac{b}{1+xb}})) \\ &= \mathrm{MSB}_k(\mathrm{SH}(m^{ab}, m^{a(1+bx)}, m^b)) \\ &= \mathrm{A}(\mathrm{SH}(m^{ab}, m^{a+xab}, m^b)) \end{aligned}$$

$m^{a+xab} = m^a(m^{ab})^x$ is easily computable, since m^a, m^{ab} and x are known. Again we recover $\alpha = m$ by applying Theorem 2.1.6.

□

Remark. While all of these protocols provide the secrecy of either the message or the key, none of them achieves authentication in the process. In the next section we examine in which settings authentication is possible and how to do it.

2.3 Authentication

As mentioned in the first chapter the main goals of cryptographic schemes are providing secrecy and authenticity. Up to now we have seen how to obtain a joint secret key and even how to communicate using public key encryption. Now we still need to figure out how to authenticate.
Consider a setting with two legitimate parties A and B who communicate over an insecure channel. Now we have an adversary E, who is in full control of the channel. She can intercept anything sent on the channel and also send messages to A and B or just block the channel. Suppose A and B are not in possession of any joint secrets. So all in all we have three parties with symmetrical information and therefore it is not possible for any of the parties to identify any of the other parties. No matter what A might do to identify herself to B, E can simply intercept A's message and use it to pass herself off as A.
In this section we will briefly introduce two approaches to authenticate messages, namely message authentication codes and digital signatures. This section is based on chapters 9 and 10 of [1].

2.3.1 Message Authentication Codes

Let us describe the setting, in which message authentication is needed. A wants to send a message m to B in a way, that B can be sure that m originated with A and that the message B received is exactly the same as A intended to send. Message authentication codes make that possible. The basic idea is to send a certificate or tag with each message, which depends on the sender as well as the message and can only be produced by the legitimate sender.
As discussed before in order to be able to use message authentication codes, A and B need to already share some secret key k.

Definition 2.3.1 (Message Authentication Code)**.** *Let M denote the set of possible messages and T be the set of possible tags, both finite. A message authentication code is a triple $(\mathcal{K}, \mathcal{MAC}, \mathcal{V})$ of polynomial time algorithms with the following properties:*

1. *$\mathcal{K}$ generates random keys of a certain length l. The set K of all possible outputs of $\mathcal{K}$ is finite and called the keyspace.*

2. *On input of a key $k_0 \in K$ and a message $m_0 \in M$ the tag-generation algorithm $\mathcal{MAC}$ computes a tag $\in T$. tag is not necessarily unique, i.e. the encryption algorithm may be probabilistic.*

3. *On input of a key $k_0 \in K$, a tag $\in T$ and a message m the verification algorithm $\mathcal{V}$ returns either 1, if the tag is valid for the received message*

and the legitimate sender and 0 otherwise, i.e.

$$\forall\ k \in K,\ m \in M: \quad \mathcal{V}(k, tag, m) = 1,\ \textit{if } tag = \mathcal{MAC}(k, m),$$
$$\mathcal{V}(k, tag, m) = 0,\ \textit{if } tag \neq \mathcal{MAC}(k, m).$$

Security Requirements. To understand what properties we want for message authentication codes, let us look at possible attacks first.
The obvious and easiest one is for E to wait, until A initiates a conversation by sending some authenticated message. Then E could simply copy the message with the tag and later use it to make B believe, he was talking to A, when he is really talking to E. This is called a replay attack. Anticipating replay attacks can easily be achieved by adding a timestamp to each message, so that the receiver can immediately detect it, when E tries to replay a message and if A really wants to send some message twice, she will use different timestamps. It may be argued, that adding timestamps is not safe, since E might also know the exact time, a message is sent and therefor have some extra information she could use to break the code. So instead of timestamps A might use a random integer instead. B will accept the message as authentic, only if the integer has not been used in any previous message.
Since E controls the channel, she can observe valid pairs of messages and tags. In some settings she might even be able to have tags for her own messages computed and it may also be possible for her to query the verification algorithm on any message-tag-pair she created. Of course those tags then also identify her as the origin of the message, but access to this kind of information may be very useful in forging a tag and thereby breaking the message authentication code. In fact it should not be possible for E to find any valid message-tag-pair, that identifies A as the sender and passes verification.
Another issue is message integrity: Even small changes to the original message by E should be detectable.

If we have a computationally unbounded adversary, she will eventually be able to forge tags, after seeing some valid message-tag-pairs. In modern cryptography security against computationally unbounded adversaries can usually not be provided. Provably secure schemes, such as the one time pad, are the exception.
We are content with the security of a scheme, if an adversary has a low chance of breaking it, using only polynomial time and space.

Informally a message authentication code is said to be secure, if the probability of an adversary succeeding to create a valid message-tag-pair is small. The probability is taken over any probabilistic choices $\mathcal{K}$, $\mathcal{MAC}$ and the adversary might make. The adversary is allowed black-box-access to $\mathcal{MAC}$ and $\mathcal{V}$, i.e. she can compute signatures for messages of her choice, that identify her as the sender, and she can check the validity of any attempted forgery. For a formal definition of security and examples, check chapter 9 of [1].

2.3.2 Digital Signatures

The setting in which digital signatures are used is slightly different. Here A and B do not need to have a joint secret key. It sort of works like public key cryptography, only that everything works the other way around:
Digital signatures allow for the sender to sign her messages, in such a way that anyone can verify the signature, but only the sender can create it. Just like in public key cryptography there are two keys involved here. The sender will use her private key s to compute a signature for her message and the receiver, or anyone else for that matter, can use the sender's public key v to verify the signature.

Definition 2.3.2 (Digital Signature Scheme)**.** *Let M denote the set of possible messages and S be the set of possible signatures, both finite. A digital signature scheme is a triple $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ of polynomial time algorithms with the following properties:*

1. *$\mathcal{K}$ generates random key pairs (sk_0, vk_0). sk_0 and vk_0 are corresponding signing and verification keys with respect to $\mathcal{S}$ and $\mathcal{V}$. vk_0 is called the public key and sk_0 the private key The set K of all possible outputs of $\mathcal{K}$ is finite and called the keyspace.*

2. *On input of a private signing key sk_0 and a message $m_0 \in M$ the signing algorithm $\mathcal{S}$ computes a signature $s_0 \in S$. s_0 is not necessarily unique, i.e. the signing algorithm may be randomized.*

3. *On input of a public key vk_0 and a message m_0 with signature $s_0 \in S$ the verification algorithm $\mathcal{V}$ returns either 1, if the signature is valid for the received message and the legitimate sender and 0 otherwise, i.e.*

$$\begin{aligned} \forall\ (sk, vk) \in K,\ m \in M: \quad & \mathcal{V}(vk, s, m) = 1,\ \text{if } s = \mathcal{S}(sk, m), \\ & \mathcal{V}(vk, s, m) = 0,\ \text{if } s \neq \mathcal{S}(sk, m). \end{aligned}$$

Security Requirements. Since digital signature schemes serve the same purpose as message authentication codes, the security requirements are almost the same. Only now the adversary has one more piece of information - the public verification key of the sender. As before it should not be possible for anyone but the legitimate sender to compute any valid message-signature-pair. Again we do not ask that a digital signature scheme be secure against computationally unbounded adversaries. It suffices, that the probability of an adversary breaking it in polynomial time and space is small. The probability is taken over any probabilistic choices of $\mathcal{K}$, $\mathcal{S}$ and the adversary. The adversary is allowed black-box-access to $\mathcal{S}$ and $\mathcal{V}$ and has knowledge of the public key vk_0, i.e. she can compute signatures for messages of her choice, that identify her as the sender, and she can check the validity of any attempted forgery. For a formal definition of security and examples, check chapter 10 of [1].

2.4 Session Key Distribution

As mentioned above there is no way to achieve an authenticated key exchange in a setting, where there is no information advantage of some kind between the legitimate parties. But if there already exists a shared secret, why do we need another one? There are a couple of good reasons:
Most of the encryption and authentication schemes in use are only secure with a certain probability, i.e. there is a small probability, that an adversary succeeds in breaking them; the more messages are encrypted or authenticated with the same key, the higher is the chance of an adversary to recover that key. It would be a lot safer to use a new key for every new communication session.
One could also imagine a setting, where a user wants to use a number of different applications, which all require secret keys. Now it is possible, that some applications reveal the key, when it is not needed anymore. So if we only have one shared secret key, we will not get very far.
This is where session keys distribution comes into play. It provides us with methods to obtain new secret keys, given that there is already some joint secret. We will take a look at two different approaches: The first approach is to assume that there is an information advantage between the legitimate parties, who wish to communicate. The second approach involves a third party, a trusted server, that shares a secret key with each of the legitimate parties.

Security Requirements. Encryption and authentication schemes already provide us with secrecy and authenticity. The only thing needed to use them is one secret key. In the case of private key encryption and message authentication there needs to be a shared secret key, in public key encryption and digital signature schemes only public keys need to be shared. Depending on the nature of the information, that is being communicated however, it may be possible that a supposedly secret key is compromised. Maybe the secret information only remains secret for a while and then gets published for example. If it was encrypted using a one time pad, that compromises the key, since then the ciphertext and the plaintext are available to the adversary.
Either way it cannot hurt, to have a few back up keys, or in our case, to not use the long lived key for encryption or authentication at all. Only use it in key distribution protocols and then use a fresh key for every new communication session, hence the term session key distribution. Of course there are some properties, we desire for those keys. Again the most important ones are secrecy and authenticity. The key should not be retrievable from any messages sent during the execution of the distribution protocol and it should be impossible for an active adversary to trick one of the legitimate parties into sharing a key. In addition to that, we also require, that the session keys, even if compromised do not leak any information on the long lived key or any other key outputted by the key distribution protocol, i.e. the session keys need to be independent of each other and the long lived key.

Now we introduce protocols for two-party and three-party models. Both of these work with public or private key encryption schemes, but we only consider

both cases in the two-party-setting.
The three-party-model is especially interesting since it can easily be generalized to a multi-party-model with any number of participants. The protocols in this section are all taken from chapter 11 of [1].

2.4.1 Two-Party-Protocols

We introduce two settings here, symmetric and asymmetric. The symmetric setting applies private key encryption and message authentication codes, i.e. requires that there is a preexisting shared secret key between A and B. The asymmetric setting uses public key encryption and digital signatures instead. No private secret key is needed, but A and B both possess key pairs of a private and a public key, one for encryption and one digital signatures.

The symmetric setting. Symmetric two-party-models are used, when the two legitimate parties already share a long-lived secret key. The goal here is to securely exchange an authenticated secret session key. Since we need both secrecy and authenticity, we are going to use a private key encryption scheme as well as a message authentication code.

Example. In this protocol A and B first need to agree on a private key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a message authentication code $(\mathcal{K}', \mathcal{MAC}, \mathcal{V})$. A and B are already in possession of a key pair (k_e, k_a), where k_e is a key for the encryption scheme and k_a is a key for the message authentication code. We assume that every message, sent over the channel, is accompanied by the identity of its sender. However any adversary has the ability to forge that identity, so that it cannot be used for authentication.

The protocol:

1. A chooses a random bit string R_A and sends it to B.

2. B chooses a bit string R_B and a session key α at random, encrypts α to $\mathcal{E}(k_e, \alpha)$ and computes the corresponding $tag = \mathcal{MAC}(k_a, m)$ for the message $m = (\mathcal{E}(k_e, \alpha), R_A, R_B, A, B)$. He sends R_B, $\mathcal{E}(k_e, \alpha)$ together with the tag to A.

3. A checks $\mathcal{V}(k_a, tag, m) \stackrel{!}{=} 1$, if so A computes a tag $tag' = \mathcal{MAC}(k_a, m')$ for the message $m' = (A, R_B)$ and sends it to B.

4. B checks $\mathcal{V}(k_a, tag', m) \stackrel{!}{=} 1$, and if so accepts to enter in a conversation with A using α as the session key.

The asymmetric Setting. In the asymmetric setting we revert to public key cryptography, namely public key encryption schemes to provide secrecy and digital signature schemes to provide authenticity. Accordingly each of the legal parties holds two key pairs: a pair (ek,dk), where ek is a public encryption key and dk is the corresponding private decryption key, and a pair (sk,vk) where

sk is a private signing key and vk is the corresponding public verification key. As in the example above, every sent message is linked to an unsecured identity.

Example. Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ denote a public key encryption scheme and $(\mathcal{K}', \mathcal{S}, \mathcal{V})$ denote a digital signature scheme. Let (ek_A, dk_A), (sk_A, vk_A), (ek_B, dk_B) and (sk_B, vk_B) be the keys for the encryption and the signature scheme of A and B respectively.

The protocol:

1. A chooses a random bit string R_A and sends it to B.
2. B chooses a bit string R_B and a session key α at random, encrypts α to $\mathcal{E}(ek_A, \alpha)$ and computes the corresponding signature $s = \mathcal{S}(sk_B, m)$ for the message $m = (\mathcal{E}(ek_A, \alpha), R_A, R_B, A, B)$. He sends R_B, $\mathcal{E}(ek_A, \alpha)$ together with the signature s to A.
3. A checks $\mathcal{V}(vk_B, s, m) \stackrel{!}{=} 1$, if so she computes a signature $s' = \mathcal{S}(sk_A, m')$ for the message $m' = (A, R_B)$ and sends both to B. A retrieves the secret session key by decrypting $\mathcal{D}(dk_A, \mathcal{E}(ek_A, \alpha)) = \alpha$.
4. B checks $\mathcal{V}(vk_A, s, m) \stackrel{!}{=} 1$, and if so accepts to enter in a conversation with A using α as the session key.

In both examples the random bit strings R_A and R_B make sure, that a replay attack becomes highly unlikely to succeed. Although the adversary, who is imitating one of the legitimate parties, gets to choose one of the random bit-strings, she has no influence on the choice of the party she is trying to trick into sharing a session key. Since we do not require the random strings to have a certain length or structure, the chance of the same bit string being chosen twice is negligibly small.
Choosing the session key α at random ensures, that different session keys are independent of each other and also of the long-lived keys. The encryption schemes then takes care of the secrecy and the message authentication code or the digital signature scheme allows for the legitimate parties to be sure of who they are exchanging keys with.
Notice, that in step two not only B and R_B are authenticated, but also A, R_A and the encrypted session key. Therefore any party, that the adversary E tried to imitate will be informed about the attempt and any tampering with the encrypted session key will be detected, unless E is able to break both the encryption and the authentication scheme.

For a more extensive and thorough discussion of two-party-protocols and a formal definition of security, check [9].

2.4.2 Three-Party-Protocols

Three-party-models are often referred to as trust-models. Besides the legitimate users A and B, there is a third trusted party S involved here. The problem A

and B have is still the same: they need to come in possession of a joint secret session key. So any protocol they might use has to offer authenticity and secrecy. Instead of sharing some information advantage with each other, A and B now share secret keys with the trusted server S. If A wants to initiate a conversation with B, she will contact B, who then request a session key from S. In response to the received request S generates a random session key and distributes it to the parties, named in the request.
It is easy to extend this model to any number of users or applications. All it takes is a shared key with the trusted party to obtain a session key with any other legitimate party. Some protocols, like the widely used Kerberos-protocol, are a lot more complicated. They involve requesting so-called tickets at different servers.
Multi-party-session key distribution can be realized in both the private and the public key setting. We now take a look at a simple example, that uses a private key encryption scheme and a message authentication code.

Example. Fix a private key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ and a message authentication code $(\mathcal{K}', \mathcal{MAC}, \mathcal{V})$. Each party I $\in$ {A,B} shares a pair of long-lived keys (k_e^I, k_a^I) with S. k_e^I denotes a key for the encryption scheme and k_a^I denotes a key for the message authentication code.

The protocol:

1. A chooses a random bit string R_A and sends it to B.

2. B chooses a bit string R_B and sends (A, R_A, R_B) to S.

3. S generate a random session key α and computes $\mathcal{E}(k_e^I, \alpha)$ with the corresponding $tag^I = \mathcal{MAC}(k_a, m)$ for the message $m^I = (\mathcal{E}(k_e^I, \alpha), R_I, A, B)$ for $I \in \{A, B\}$.
   S distributes α by sending $\mathcal{E}(k_e^I, \alpha)$ and tag^I to $I \in \{A, B\}$.

4. Every $I \in \{A, B\}$ checks $\mathcal{V}(k_a^I, tag^I, m^I) \stackrel{!}{=} 1$, and if so accepts to enter in the conversation using α as the session key.

As remarked earlier this protocol can be extended to a conference key sharing protocol. S has full control over the choice of the session key, so S can easily distribute it to more than two parties, given those parties share a secret key with S.
A more detailed discussion of three-party-models can be found in [10]. All of the examples in this section are taken from chapter 11 of [1].

2.5 Conclusions

So up to this point we have seen different ways to safely exchange keys, that is if the DH-assumption holds. The security of the DH-protocol and all related schemes, is based on that assumption. In today's world almost all of the cryptography schemes in use rely on the hardness of some mathematical problem,

usually either integer factoring or computing discrete logarithms. Quantum algorithm, that efficiently solve integer factoring and discrete logarithms, have already been developed. The only thing preventing application is the absence of quantum computers, that are up for the task.
So far, no one has succeeded in constructing a quantum computer, that can be used to break today's encryption schemes, but quantum computers certainly pose a big threat to complexity-based cryptography. The one time pad makes an exception of course, but it is also rather impractical. There is another exception: lattice-based cryptography. It is also complexity-based, i.e. relies on the hardness of a mathematical problem. The big difference is, that security, at least in regard to classical computers, is actually provable. Also there are no known quantum algorithms that can break lattice-based schemes so far. In fact, lattice-based cryptography is believed to be secure, even with quantum computers around. For more information on lattice-based cryptography, see [11].
So up to now classical cryptography is still reasonably safe, but with quantum computers on the rise, it is wise to look for provably secure methods to distribute keys.
Luckily quantum computers not only bear threats, but also provide possibilities in key distribution. In the next chapter we will see, that quantum key distribution is not only possible, but also secure without any assumption on complexity. The security of the protocol we are going to study is solely based on the laws of physics.

Chapter 3

Quantum Key Distribution

Although the concept of quantum computers has been around for more than 30 years now, there still is no quantum computer powerful enough to break the cryptography in use today. There are however a great number of papers on the subject of quantum computation
and the theoretical results certainly pose a threat to most fields of classical cryptography.
Even though it is an open problem, whether quantum computers are strictly more powerful than classical computers, the fact that there are algorithms to efficiently solve integer factoring and the DLP on quantum computers strongly suggests that quantum computation has some real advantages. We will show that even having unlimited computational power does not help an adversary to break the key distribution protocol, we will be looking at. The security is based on physics laws, not on assumed complexity of some mathematical problem.
Nonetheless quantum computers still have a long way to go. The technology today is far too expensive and requires dedicated hardware. Due to the instability of quantum states distances also pose a problem. Here we concentrate on the chances of quantum computation for key distribution.

Overview. In this chapter we will first introduce the basics of quantum computation. We start by introducing qubits, short for quantum bits, and examine their behaviour. We give formal definitions of what it means to measure or transform qubits.
Then we move on to systems with more than one qubit. Unlike in classical computation, the step from one qubit to many qubits is a rather big one, since qubits can interact with each other in very different ways than classical bits.
Once we obtained a mathematical model to describe quantum systems, we try to get a little closer to reality by also modeling errors in the quantum world and see how we can correct those errors in sections 3.2 and 3.3. That provides us with all the tools to introduce a quantum key distribution protocol, the BB84 protocol, and prove its security. In terms of security we omit talking about authenticity and focus on secrecy solely.

3.1 Concepts and Definitions

While in classical computation the state of a bit can only be one of the values 0 or 1, quantum computers have more options: They operate on qubits which can be regarded as two dimensional complex vectors. The state of a qubit can be any linear superposition of the base vectors. So really a qubit can be any point in a two-dimensional complex vector space, that is infinitely many different possible states for just one qubit. One might jump to the conclusion that this allows us to store an infinite amount of information in just one qubit as well. But the problem with that is, that we can just read or measure the value of a qubit like we read the value of a classical bit. Measuring a qubit affects its state and depending on the measurement we choose to perform, there will usually be more than one possible outcome. A measurement cannot be reversed, so if we are given a qubit in an unknown state, there is no way to find out what state it is in, if we do not have any previous information.
The section is based on chapters 2, 4 and 5 of [14] and [15]. The no-cloning theorem, including the proof, is taken from [13, p. 532]. A more extensive treatment of this topic can be found in chapters 1 and 2 of [13].

3.1.1 Single Qubits

Definition 3.1.1 (Qubit)**.** *A representation* $|\Psi\rangle$ *of a qubit is a vector in* $\mathbb{C}^2$*. For an arbitrary base* $\{|0\rangle, |1\rangle\}$ *the state of* $|\Psi\rangle$ *can be written as* $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

For simplicity we only allow normalized states, i.e. $||\,|\Psi\rangle\,|| = 1$ where $||.||$ denotes the standard vector norm in $\mathbb{C}^2$. So for $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ we get $|\,\alpha\,|^2 + |\,\beta\,|^2 = 1$.

Quantum computers can perform two different types of operations on a qubit: Unitary transformations and measurements. In general only the unitary transformations are reversible.

There are a number of particles, such as photons, ions or electrons, that can serve as qubits in quantum computation. To get a better intuition for the behavior of a qubit, we can take a look at photons for example.

The state of a photon would be its polarization. Let $|0\rangle$ denote vertical polarization and $|1\rangle$ horizontal polarization.
A unitary transformation on $|\Psi\rangle$ is nothing else than a rotation of the polarization. After a rotation by 45 degrees a horizontally polarized photon $|\Psi\rangle = |1\rangle$ has diagonal polarization $|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
A measurement can be compared to letting the photon through a filter. A filter that lets vertically polarized light through, will always let $|0\rangle$ through, never $|1\rangle$ and $|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ with probability $\frac{1}{2}$. Any photon that has passed the

filter will then be in state $|0\rangle$, regardless of the state it was in beforehand.
If we take a different filter, the probabilities will also change.

Definition 3.1.2 (Unitary Transformation)**.** *A unitary transformation on a qubit $|\Psi\rangle$ is a unitary matrix $U \in \mathbb{C}^{2\times 2}$, i.e. $UU^* = U^*U = I$ where U^* denotes the conjugate transpose of U and I the identity matrix.*
Performing U on $|\Psi\rangle$ yields the state $U|\Psi\rangle$.

Unitary matrices preserve the norm of a vector, so the condition that all quantum states be normalized is not a problem.

Example. A frequently used unitary transformation in quantum key distribution is the *Hadamard transformation*

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Applying H to the orthonormal basis $B := \{|0\rangle, |1\rangle\}$ with $|0\rangle := (1 \quad 0)^T$, $|1\rangle := (0 \quad 1)^T$ yields

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) =: |0\rangle_\times \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =: |1\rangle_\times \end{aligned}$$

We call a pair of bases B and B' conjugate if a measurement in B necessarily randomizes the outcome of a measurement in B', i.e. if measuring one of the basis states $|b\rangle \in B$ in B' yields any result $|b'\rangle \in B'$ with equal probability.

Measuring a qubit is a little more complicated. For a qubit in an arbitrary unknown state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ it is impossible to find out in which state $|\Psi\rangle$ is in exactly, i.e. recover the coefficients α and β
Moreover any measurement destroys the original state of a qubit and forces it to collapse onto one of state in a set of states, defined by the measurement.

Definition 3.1.3 (Orthogonal Measurement)**.** *An orthogonal measurement on a single qubit is defined by its possible outcomes $|m_1\rangle, |m_2\rangle \in \mathbb{C}^2$, which are orthonormal states . The probability to obtain the result $|\mathrm{m_i}\rangle$ when measuring a qubit in state $|\Psi\rangle$is*
$\Pr[m_i] = \langle\Psi| \ |m_i\rangle\langle m_i| \ |\Psi\rangle = |\langle m_i|\Psi\rangle|^2$,
with $\Pr[m_1] + \Pr[m_2] = 1$. If the result of the measurement is $|\mathrm{m_i}\rangle$, the qubit then is in state $|\mathrm{m_i}\rangle$ and we say $|\Psi\rangle$ collapsed onto the vector $|\mathrm{m_i}\rangle$.

Example. Let $B := \{|0\rangle, |1\rangle\}$ and $B_\times := \{|0\rangle_\times, |1\rangle_\times\}$,then those two orthonormal bases of $\mathbb{C}^2$ each define an orthogonal measurement.
Measuring the state $|0\rangle$ in B yields $|0\rangle$ with certainty, since $|\langle 0|0\rangle|^2 = 1$.
When measuring $|0\rangle$ in $B_\times$ both possible outcomes $|0\rangle_\times$ and $|1\rangle_\times$ occur with probability $\frac{1}{2}$, since $|\langle 0_\times|0\rangle|^2 = |\langle 1_\times|0\rangle|^2 = \frac{1}{2}$

A qubit $|\Psi\rangle$ is said to be in a *pure state* with respect to a fixed basis B, when measuring the state in B has a certain outcome with probability 1, i.e. $|\Psi\rangle$ is one of the base vectors.
$|\Psi\rangle$ is said to be in a *mixed state*, if both base vectors of B occur with strictly positive probabilities.

3.1.2 Higher Dimensional Quantum States

In classical computation the step from a single bit to an n-bit string is rather unspectacular. The bit string is already defined by the values of each single bit, increasing the number of bits by 1 simply increases the number of possible different bit strings by the factor 2.
With qubits we the situation gets a lot more complicated, as soon as more than one qubit is involved: In certain cases the outcomes of individual measurements of a number of qubits become correlated. This phenomenon is called entanglement.

Definition 3.1.4 (Quantum State)**.** *A representation $|\Psi\rangle$ of an n-dimensional quantum state is a vector in $\mathbb{C}^n$, where $\mathbb{C}^n = \mathbb{C}^{2^m} \equiv (\mathbb{C}^2)^{\otimes m}$ for some integer m.*
Again we require, that $|| \, |\Psi\rangle \, || = 1$.

Example. Let $|\Psi\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$ and $|\Phi\rangle = \beta_1 |0\rangle + \beta_2 |1\rangle$ be two qubits, each living in a Hilbert space $\mathcal{H} = \mathbb{C}^2$. Then the composite system $\mathbb{C}^2 \otimes \mathbb{C}^2$ is in the 4-dimensional quantum state

$$\begin{aligned} |\Psi\rangle \otimes |\Phi\rangle &= \alpha_1\beta_1 |0\rangle |0\rangle + \alpha_1\beta_2 |0\rangle |1\rangle + \alpha_2\beta_1 |1\rangle |0\rangle + \alpha_2\beta_2 |1\rangle |1\rangle \\ &= \alpha_1\beta_1 |00\rangle + \alpha_1\beta_2 |01\rangle + \alpha_2\beta_1 |10\rangle + \alpha_2\beta_2 |11\rangle \end{aligned}$$

$\mathbb{C}^2 \otimes \mathbb{C}^2$ has the orthonormal base $B \times B = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Any superposition of the base states, like $|\Psi\rangle \otimes |\Phi\rangle$ is a mixed state, the only pure states are the base states.

Of course this construction can be extended to higher dimensional quantum systems.

Definition 3.1.5 (Composite Systems)**.** *Consider two quantum systems $\mathcal{H}_1 = \mathbb{C}^n$ and $\mathcal{H}_2 = \mathbb{C}^m$. Then the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ is called the composite system of $\mathcal{H}_1$ and $\mathcal{H}_2$.*
For bases $B_1 = \{|b_i\rangle\}_{i=1,\ldots,n}$ and $B_2 = \{|b'_j\rangle\}_{j=1,\ldots,m}$ of $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, $B := B_1 \times B_2 = \{|b_i b'_j\rangle\}_{i=1,\ldots,n;\ j=1,\ldots,m}$ is a base of $\mathcal{H}_1 \otimes \mathcal{H}_2$.

The composite state of two quantum states $|\Psi_1\rangle \in \mathcal{H}_1$ *and* $|\Psi_2\rangle \in \mathcal{H}_2$ *is defined as*

$$|\Psi_1\rangle \otimes |\Psi_2\rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j \, |b_i b'_j\rangle \,,$$

if $|\Psi_1\rangle = \sum_{i=1}^{n} \alpha_i \, |b_i\rangle$ *and* $|\Psi_2\rangle = \sum_{j=1}^{m} \beta_j \, |b'_j\rangle$.

Analogous to unitary transformations on qubits, a unitary transformation on a quantum state is described by a unitary matrix $U \in \mathbb{C}^{n \times n}$ and for two unitary transformations U_1 and U_2 of Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively and quantum states $|\Psi_1\rangle \in \mathcal{H}_1$, $|\Psi_2\rangle \in \mathcal{H}_2$ we obtain a unitary transformation $U_1 \otimes U_2$ on the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ by

$$(U_1 \otimes U_2)(|\Psi_1\rangle \otimes |\Psi_2\rangle) := U_1 \, |\Psi_1\rangle \otimes U_2 \, |\Psi_2\rangle \,.$$

Any unitary transformation on $\mathcal{H}_1$ can be extended to a unitary transformation on $\mathcal{H}_1 \otimes \mathcal{H}_2$ by tensoring it with the identity on $\mathcal{H}_2$.

In the same way we can extend measurements of single qubits to measurements on the composite system. Interpreting the identity matrix I as a projector allows us to perform a measurement M on the subsystem $\mathcal{H}_1$ by performing $M \otimes I$ on the composite state of $\mathcal{H}_1 \otimes \mathcal{H}_2$.

Definition 3.1.6 (Positive Operator valued Measurement)**.** *A positive operator valued measurement (POVM) is a family of positive-semidefinite, hermitian operators* $\mathcal{M} := \{M_x\}_{x \in \mathcal{X}}$ *on the vector space of a quantum state such that*

$$\sum_{x \in \mathcal{X}} M_x = I,$$

where $\mathcal{X}$ *is the set of possible outcomes of* $\mathcal{M}$*. Performing* $\mathcal{M}$ *on a quantum state* $|\Psi\rangle$ *results in* $x \in \mathcal{X}$ *with probability* $\Pr[x] = \langle\Psi| \, M_x \, |\Psi\rangle$ *and again we have* $\sum_{x \in \mathcal{X}} \Pr[x] = 1$.

For an orthonormal basis $B := \{|b_1\rangle \, , ..., |b_n\rangle\}$ of $\mathbb{C}^{\mathrm{n}}$ we can construct a POVM $\mathcal{M}_B := \{M_i\}_{i=1,...,n}$ where $M_i := |b_i\rangle \langle b_i| \;\; \forall \; i$ are orthogonal projectors onto the basis states. Measuring $|\Psi\rangle$ in a basis B denotes performing the POVM $\mathcal{M}_B$. A POVM $\mathcal{M}_B$ where B is an orthonormal basis is called *complete measurement.*

Pure and mixed quantum states with respect to a basis B are defined analog to the one qubit case:
For a *pure quantum state* we have $\Pr[b_i] = 1$ for exactly one $i \in \{1, .., n\}$. $|\Psi\rangle$ is in a *mixed quantum state* if at least two of the probabilities $\{\Pr[b_i]\}_{i=1,...,n}$ are strictly positive.

The No-Cloning Theorem. In classical computation it is a fairly common practice to copy bits. It can easily be done, there is nothing special about it. Quantum computation in contrast does not allow for that in general. While it is possible to prepare multiple qubits in the same known state, copying an unknown state just does not work. The tools we have in quantum computation are measurements and unitary transformations. So if one tries to copy a qubit, it would have to be done by applying a unitary transformation. We now show, why this does not work.

Theorem 3.1.7. *Given a quantum system $\mathcal{H}$ and quantum system $\mathcal{H}'$ with the same state spaces. Let $|h'\rangle$ denote the state $\mathcal{H}'$ is in. Then there is no unitary transformation U on the composite system $\mathcal{H} \otimes \mathcal{H}'$, such that $U(|\Psi\rangle \otimes |h'\rangle) = |\Psi\rangle \otimes |\Psi\rangle$ for arbitrary unknown states $|\Psi\rangle$ of $\mathcal{H}$.*

Proof. Let $|\Psi\rangle$, $|\Phi\rangle \in \mathcal{H}$ be two arbitrary unknown states. Assume that U is the unitary transformation we are looking for, i.e.

$$\begin{aligned} U(|\Psi\rangle \otimes |h'\rangle) &= |\Psi\rangle \otimes |\Psi\rangle \quad \text{and} \\ U(|\Phi\rangle \otimes |h'\rangle) &= |\Phi\rangle \otimes |\Phi\rangle \,. \end{aligned}$$

Then

$$\begin{aligned} \langle U(\Psi \otimes h')|U(\Phi \otimes h')\rangle &= \langle \Psi \otimes \Psi | \Phi \otimes \Phi \rangle \\ \langle \Psi \otimes h' | \Phi \otimes h' \rangle &= (\langle \Psi | \Phi \rangle)^2 \\ \langle \Psi | \Phi \rangle \langle h' | h' \rangle &= (\langle \Psi | \Phi \rangle)^2 \\ \langle \Psi | \Phi \rangle &= (\langle \Psi | \Phi \rangle)^2, \end{aligned}$$

so either $\langle \Psi | \Phi \rangle = 0$, which means that $|\Psi\rangle$ and $|\Phi\rangle$ have to be orthogonal, or $\langle \Psi | \Phi \rangle = 1$. which means $|\Psi\rangle = |\Phi\rangle$. It follows, that cloning arbitrary states is not possible. □

Quantum states can be interpreted as statistical ensembles $\{|b_i\rangle, \Pr[b_i]\}_{i=1,\dots n}$ where the $|b_i\rangle$ are orthonormal vectors and $\sum_{i=1}^{n} \Pr[b_i] = 1$.
We call $\{|b_i\rangle, \Pr[b_i]\}_{i=1,\dots n}$ a quantum ensemble.
Extending a quantum system $\mathcal{H}$ in state $\{|b_i\rangle, \Pr[b_i]\}_{i=1,\dots n}$ by a system $\mathcal{H}'$ in state $|\Psi\rangle$ results in the ensemble $\{|b_i\rangle \otimes |\Psi\rangle, \Pr[b_i]\}_{i=1,\dots n}$ over $\mathcal{H} \otimes \mathcal{H}'$.

Although at first glance the notation of a quantum sate $|\Psi\rangle$ as an ensemble contains less information than the notation as a linear superposition of the basis states, it does not really, since there is no way to distinguish two states with the same statistical ensemble by performing any number of unitary transformations, measurements and extensions.

Say we have a quantum ensemble $\{|b_i\rangle, \Pr[b_i]\}_{i=1,\dots n}$ and POVM $\mathcal{M} := \{M_x\}_{x \in \mathcal{X}}$. Then the probability of an outcome x of $\mathcal{M}$ is

$$\Pr[x] = \sum_{i=1}^{n} \Pr[b_i] \langle b_i | M_x | b_i \rangle = \operatorname{tr}\left(M_x \sum_{i=1}^{n} \Pr[b_i] |bi\rangle \langle b_i|\right),$$

where $\operatorname{tr} A$ denotes the trace of a matrix A.

Definition 3.1.8 (Density Matrix)**.** *For an ensemble* $\{|b_i\rangle, \Pr[b_i]\}_{i=1,\dots n}$ *the density matrix* ρ *is defined as*

$$\rho := \sum_{i=1}^{n} \Pr[b_i] \, |b_i\rangle \langle b_i| \, .$$

ρ is also called a density operator.

Density matrices are a very convenient way to represent a quantum state, since they facilitate computing the outcome probabilities of arbitrary measurements. From now on, we will represent quantum states $|\Psi\rangle$ by their density matrix ρ acting on a Hilbert space $\mathcal{H}$, where $\mathcal{H}$ denotes the vector space $|\Psi\rangle$ lives in.
By extending a quantum system $\mathcal{H}$ in state ρ by a system $\mathcal{H}'$ in state described by the density matrix σ we obtain the density matrix $\rho \otimes \sigma$ over $\mathcal{H} \otimes \mathcal{H}'$.

In classical computation measuring the value of a bit simply means reading it. The procedure does not change the state of the bit being measured and has no effect whatsoever on other bits.
Given a quantum system with more than one qubit, we say it is in a separable state if measuring one qubit does not affect the states of the other qubits in the system.

Definition 3.1.9 (Entanglement and Separability)**.** *Let* $\mathcal{H}_1$ *be a Hilbert space with base* $B = \{|b_i\rangle\}_{i=1,\dots,n}$ *and* $\mathcal{H}_2$ *be Hilbert spaces with base* $B' = \{|b'_i\rangle\}_{j=1,\dots,m}$*. A state* $|\Psi\rangle = \sum_{i,j} \gamma_{i_j} |b_i b'_j\rangle$ *of* $\mathcal{H}_1 \otimes \mathcal{H}_2$ *is called separable if and if only there are quantum states* $|\Psi_1\rangle = \sum_{i=1}^{n} \alpha_i |b_i\rangle \in \mathcal{H}_1$ *and* $|\Psi_2\rangle = \sum_{j=1}^{m} \beta_j |b'_j\rangle \in \mathcal{H}_2$ *such that* $|\Psi\rangle = |\Psi_1\rangle \otimes |\Psi_2\rangle$*, i.e.* $\gamma_{i_j} = \alpha_i \beta_j \ \forall \ i, j$*.*
Let ρ *be a density matrix operating on* $\mathcal{H}_1 \otimes \mathcal{H}_2$*. Then* ρ *is separable if it is of the form*

$$\rho = \sum_{i=1}^{n} \sum_{j=1}^{m} \Pr[b_i b'_j] \, |b_i b'_j\rangle \langle b_i b'_j| = \sum_{i=1}^{n} \sum_{j=1}^{m} \Pr[b_i b'_j] \, |b_i\rangle \langle b_i| \otimes |b'_j\rangle \langle b'_j| \, ,$$

with $\sum_{i,j} \Pr[b_i b'_j] = 1$*, i.e. it can be written as a sum of projectors onto product states.*
A quantum state is said to be an entangled state, if it is not separable.

Example (The Bell States)**.** The four Bell states in $\mathbb{C}^2 \otimes \mathbb{C}^2$ are defined as

$$|\Phi^+\rangle := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \qquad |\Phi^-\rangle := \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$|\Psi^+\rangle := \frac{1}{\sqrt{2}}(|01\rangle + |01\rangle), \qquad |\Psi^-\rangle := \frac{1}{\sqrt{2}}(|01\rangle - |01\rangle).$$

To check whether the Bell states are entangled states, consider an arbitrary separable state $|\Gamma\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$. Then $|\Gamma\rangle$ is of the form

$$|\Gamma_1\rangle \otimes |\Gamma_2\rangle = \alpha_1\beta_1 |00\rangle + \alpha_1\beta_2 |01\rangle + \alpha_2\beta_1 |10\rangle + \alpha_2\beta_2 |11\rangle,$$

for some $|\Gamma_1\rangle, |\Gamma_2\rangle \in \mathbb{C}^2$.
For $|\Phi^+\rangle$ and $|\Phi^-\rangle$ we have $\gamma_{1_1} \neq 0$ and $\gamma_{2_2} \neq 0$. Separability then requires $\alpha_1\beta_1 \neq 0$, $\alpha_2\beta_2 \neq 0$ and therefore $\gamma_{1_2} = \alpha_1\beta_2 \neq 0$, which is not the case. Analogously we can show, that $|\Psi^+\rangle$ and $|\Psi^-\rangle$ are entangled states.

We have already seen, that given state spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ in a states given by density operators $\rho_1 = \sum_{i=1}^{n} \Pr[b_i] |\mathrm{b_i}\rangle \langle b_i|$ and $\rho_2 = \sum_{j=1}^{m} \Pr[b'_j] |\mathrm{b'_j}\rangle \langle b'_j|$, we obtain a higher dimensional separable quantum state $\rho_1 \otimes \rho_2 \in \mathcal{H}_1 \otimes \mathcal{H}_2$. Then given $\rho_1 \otimes \rho_2$ it is easy attribute states to the subsystems $\mathcal{H}_1$ and $\mathcal{H}_2$ of $\mathcal{H}_1 \otimes \mathcal{H}_2$, i.e. to reduce $\rho_1 \otimes \rho_2$ to ρ_1 or ρ_2.
However it is not obvious how to do that with a quantum system in an entangled state.

In the following let $\mathcal{S}(\mathcal{H})$ denote the set of density operators on a Hilbert space $\mathcal{H}$.

Definition 3.1.10 (Partial Trace)**.** *Let $\mathcal{H}_i$ and $\mathcal{H}_j$ be two Hilbert spaces. Then the partial trace tr_i over the subsystem $\mathcal{H}_i$ is the linear map of operators from $\mathcal{S}(\mathcal{H}_i \otimes \mathcal{H}_j)$ to $\mathcal{S}(\mathcal{H}_j)$ defined by*

$$\begin{aligned} tr_i(|v_1\rangle \langle v_2| \otimes |w_1\rangle \langle w_2|) &:= tr(|v_1\rangle \langle v_2|) |w_1\rangle \langle w_2| \\ &= \langle v_2|v_1\rangle |w_1\rangle \langle w_2|, \end{aligned}$$

for $|v_1\rangle, |v_2\rangle \in \mathcal{H}_i$ and $|w_1\rangle, |w_2\rangle \in \mathcal{H}_j$.

Definition 3.1.11 (Reduced Density Operator)**.** *Let $\mathcal{H} = \bigotimes_{i=1}^{n} \mathcal{H}_i$ be a composite quantum system in a state, described by ρ. Then the state of a subsystem $U = \bigotimes_{j \in \mathcal{J}} \mathcal{H}_j$ with $\mathcal{J} \subset \{1, ..., n\}$ is represented by the reduced density operator $\rho_U = tr_V(\rho)$, with $V := \mathcal{H} \setminus U$.*

Tracing out a subsystem $V \subset \mathcal{H}$ means calculating the reduced density operator $\rho_{\mathcal{H} \setminus V} = \mathrm{tr}_V(\rho)$.

Example. Let us take a look at the Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. As mentioned above $|\Phi^+\rangle$ is an entangled state in $\mathbb{C}^2 \otimes \mathbb{C}^2$. The density operator of $|\Phi^+\rangle$ is given by

$$\begin{aligned} \rho &= \big(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\big)\big(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\big) \\ &= \frac{1}{2}(|00\rangle \langle 00| + |00\rangle \langle 11| + |11\rangle \langle 00| + |11\rangle \langle 11|). \end{aligned}$$

Tracing out the first qubit yields

$$\begin{aligned}\rho_2 =& \operatorname{tr}_1(\frac{1}{2}(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|)) \\ =& \frac{1}{2}(\operatorname{tr}_1(|00\rangle\langle 00|) + \operatorname{tr}_1(|00\rangle\langle 11|) + \operatorname{tr}_1(|11\rangle\langle 00|) + \operatorname{tr}_1(|11\rangle\langle 11|)) \\ =& \frac{1}{2}(|0\rangle\langle 0|\langle 0|0\rangle + |0\rangle\langle 1|\langle 0|1\rangle + |1\rangle\langle 0|\langle 1|0\rangle + |1\rangle\langle 1|\langle 1|1\rangle) \\ =& \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|).\end{aligned}$$

We will now introduce purifications. They allow us to associate an entangled state ρ of a Hilbert space $\mathcal{H}$ with a pure state ρ' of a larger system $\mathcal{H}'$. $\mathcal{H}'$ is a composite system, consisting of $\mathcal{H}$ and a so-called reference system $\mathcal{R}$, i.e. $\mathcal{H}' = \mathcal{H} \otimes \mathcal{R}$. Purifications are not unique.

Definition 3.1.12 (Purification)**.** *Let $\mathcal{H}$ be a quantum system in an entangled state ρ. Then a system $\mathcal{H}' = \mathcal{H} \otimes \mathcal{R}$ in state ρ' is called a purification (system) of ρ, if ρ' is a pure state and $\rho = tr_R(\rho')$*

For a system $\mathcal{H}$ in an arbitrary quantum state ρ we can construct a purification in the following way:
Let $\{|h_i\rangle\}_{i\in I}$ be an orthonormal basis of $\mathcal{H}$, then we can write $\rho = \sum_{i\in I} \alpha_i |h_i\rangle\langle h_i|$.
Now we need a reference system. Let $\mathcal{R}$ be a quantum system with the same state space as $\mathcal{H}$ and let $\{|r_i\rangle\}_{i\in I}$ be an orthonormal basis of $\mathcal{R}$.
Then $|\Phi\rangle := \sum_{i\in I} \sqrt{\alpha_i}\, |h_i\rangle\, |r_i\rangle$ is a purification on ρ.

To check, whether this is actually true, let us trace out the reference system again:

$$\begin{aligned}\operatorname{tr}_R(|\Phi\rangle\langle\Phi|) =& \sum_{i,j} \sqrt{\alpha_i\alpha_j}\, |h_i\rangle\langle h_j| \operatorname{tr}(|r_i\rangle\langle r_j|) \\ =& \sum_{i,j} \sqrt{\alpha_i\alpha_j}\, |h_i\rangle\langle h_j|\, \delta_{ij} = \sum_i \alpha_i |h_i\rangle\langle h_i| \\ =& \rho.\end{aligned}$$

3.2 Introducing Errors

In the previous section we explained how quantum systems work in theory in an ideal world. In reality however, it is not that easy to build perfectly closed quantum system, i.e. systems that do not interact with their environment at all.
In this section we introduce methods to model such quantum noise.

3.2.1 Models in Classical Computation

In the classical world errors are usually modeled using Markov processes. Since classical bits only take values in $\{0, 1\}$ there is only one type of error to consider, the bit-flip error. Assume that p is the probability that a bit-flip error occurs in a certain period of time, assume a bit-flip error appears with equal probability in every period and that at most one error occurs per period. So depending on the value $i \in \{0, 1\}$ of a bit in one period, we can calculate the conditional probabilities $\Pr[j \mid i]$ of the bit taking the value $j \in \{0, 1\}$ in the following period. Those probabilities are called transition probabilities. All in all we have a stochastic multi-stage process. We can capture it in the following equation:

$$\begin{pmatrix} p_{i+1}^0 \\ p_{i+1}^1 \end{pmatrix} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \begin{pmatrix} p_i^0 \\ p_i^1 \end{pmatrix}$$

where p_i^j denotes the probability of the bit value j in period i.
This also allows us to calculate the outcome probabilities, if there is more than one period difference to the input probabilities, by multiplying with higher powers of the matrix of transition probabilities. So no matter how many periods pass between the starting state and the final state, we still have a linear relation between the two.

3.2.2 Models in Quantum Computation

We model an open quantum system as a principal system $\mathcal{H}$, our original basic quantum system, together with another quantum system $\mathcal{H}_{env}$, we call environment. Since we do not require the environment to be of a certain dimension, we can assume, that it starts in a pure state $|env\rangle$. Otherwise we could simply extend it to a purification system. Let ρ denote the state of the principal system.
Together the principal system and the environment are a closed system in state $\rho \otimes |env\rangle$. Interactions between the principal system and the environment can thus be described as unitary transformations on $\mathcal{H} \otimes \mathcal{H}_{env}$. While starting out in a separable state $\rho \otimes |env\rangle$ applying a unitary transformation U might result in an entangled state σ of $\mathcal{H} \otimes \mathcal{H}_{env}$. Tracing out the environment then results in a state $\rho' \neq \rho$. We model this process with quantum operations. Formally a quantum operation $\mathcal{E}$ can be captured in the following equation:

$$\mathcal{E}(\rho) = \mathrm{tr}_{env}\left[U(\rho \otimes |e\rangle)U^T\right] \tag{3.1}$$

There are different approaches to introducing quantum operations. While the one we just stated is a nice way to give an intuition of quantum noise, it is a little complicated to handle in calculations. Thus we now introduce the operator-sum representation of a quantum operation.
Set $n := dim\, \mathcal{H}_{env}$, then

$$\mathcal{E}(\rho) = \sum_{i=1}^{n} E_i \rho E_i^T.$$

The E_i are called operation elements. They are operators on the principal system, that satisfy $\sum_{i=1}^{n} E_i = I$, where I denotes the identity on $\mathcal{H}$. This

condition guarantees, that quantum operations preserve the trace, i.e. the $\mathcal{E}(\rho)$ is a density matrix on $\mathcal{H}$.

Notice, that in this representation the environment is not explicitly mentioned, which makes it a lot more practical then the representation in (1.1). For a more extensive examination of quantum operations, we refer to chapter 8 of [13].

In contrary to the classical world, quantum states are not only vulnerable to bit flip errors, but to a whole variety of errors, like bit flips, phase flips, rotations and of course arbitrary combinations of those.
At first glance this may look like a serious obstacle to error correction, but as it turns out, it can be overcome. If a quantum error-correcting-code is able to correct bit flips, phase flips and bit-phase flips, it can also deal with any other error, at least when we consider single qubits. For composite systems the situation is a bit more complicated, but also manageable. Further information can be found in chapters 8 and 10 of [13].

3.2.3 Error Correction

Besides the greater variety of possible errors, there are a few other facts, that prevent us from using classical error correction strategies. In the classical world, one strategy to detect and correct errors is repetition. Instead of just sending a bit once, we can send it multiple times. Then taking the "majority vote" of all the bits will give us the right value with high probability. In the quantum world however the no-cloning theorem prevents us from applying similar strategies and even if it were possible to copy quantum states, finding out the "majority vote" is a problem, since any measurement would destroy the original state. Luckily efficient quantum error correction is still possible though.
This section starts with introducing classical linear codes, that then give rise to quantum codes, namely Calderbank-Shor-Steane-codes.
It is based on chapter 10 of [13].

3.2.4 Classical Linear Codes

Definition 3.2.1 (Linear Code)**.** *A linear [n,k]-code C consists of a set $\mathcal{C}$ of 2^k n-dimensional binary vectors, the codewords and an $(n \times k)$-dimensional generator matrix G, mapping a k-bit message x to the encoded n-bit message Gx. Hence the columns of G span the set of codewords as a vector space and we require them to be linearly independent.*

When we use linear codes to encrypt data, error correction can easily be achieved through a so-called parity check matrix H. H is an $((n-k) \times n)$ matrix, such that $Hc = 0 \quad \forall \quad c \in (C)$. If an error e occurs, i.e. a message x is encoded to c, but the receiver gets a codeword $c' = c + e$ instead, then $Hc' = Hc + He = He$. He is called the error syndrome. The error e can be corrected by using information provided by its syndrome. Naturally there is a limit to the amount of errors a liner code can correct. Single bit flip errors are not a problem, the corrupted codeword is still "close enough" to the original

one to correct it. Closeness or distance in this setting is usually measured by the Hamming weight. The Hamming weight of an n-dimensional binary vector $v = (v_1, ..., v_n)$ is defined as $H(v) := \sum_{i=1}^{n} v_i$. The Hamming distance of two vectors v and w is defined as the Hamming weight of $v + w := (v_1 + w_1 \mod 2, ..., v_n + w_n \mod 2)$. The easiest approach to correct errors now, is to find the codeword $c \in \mathcal{C}$ with the shortest Hamming distance to the corrupted codeword c'. The only problem is, that if too many errors occurred the closest valid codeword may not be the original codeword, we intended to send.
Let d be the minimal Hamming distance between two codewords of $\mathcal{C}$, then we can correct up to $t := \frac{d-1}{2}$ errors, just by replacing c' by the closest codeword in $\mathcal{C}$.

Any linear [n,k] code C with generator matrix G and parity check matrix H gives rise to a dual code $C^{\perp}$ with generator matrix $G^{\perp} := H^T$ and parity check matrix $H^{\perp} := G^T$. The set of codewords $\mathcal{C}^{\perp}$ of $C^{\perp}$ are all vectors, that are orthogonal to every codeword in $\mathcal{C}$.

Using classical linear codes and corresponding dual codes, we can now define Calderbank-Shor-Steane-Codes, or short CSS-Codes. CSS-Codes allow for error correction in the quantum world.

3.2.5 Calderbank-Shor-Steane-Codes

Given classical linear $[n_1, k_1]$ and $[n_2, k_2]$ codes C_1 and C_2, respectively, with $C_2 \subset C_1$, then for every $x \in C_1$ we can define a quantum state

$$|x + C_2\rangle := \frac{1}{\sqrt{|\, C_2 \,|}} \sum_{y \in C_2} |x + y\rangle$$

Here + denotes bitwise addition mod 2. This construction gives rise to an $[n, k_1 - k_2]$ quantum code, called the CSS-code of C_1 over C_2.

Definition 3.2.2 (CSS-Code)**.** *An $[n, k_1 - k_2]$ CSS-code $CSS(C_1, C_2)$ is the $|\, C_1 \,|/|\, C_2 \,|$-dimensional vector space spanned by the n-qubit quantum states $|x + C_2\rangle$ defined above.*

If C_1 and $C_2^{\perp}$ can both correct up to t errors, then $CSS(C_1, C_2)$ corrects maximally t bit and phase flip errors. Quantum error correction using CSS-Codes works like this:

In order to make error detection possible, without interfering with the quantum state, we extend the original codewords $|x + C_2\rangle$ by a qubit in state $|0\rangle$, which gives us a state

$$|x + C_2\rangle' := \frac{1}{\sqrt{|\, C_2 \,|}} \sum_{y \in C_2} |x + y\rangle\, |0\rangle\,.$$

Suppose instead of $|x + C_2\rangle'$ we obtain a state

$$\frac{1}{\sqrt{|\, C_2 \,|}} \sum_{y \in C_2} (-1)^{(x+y)\cdot e_2}\, |x + y + e_1\rangle\, |0\rangle,$$

where e_1 and e_2 denote binary n bit vectors. The entries of e_1 correspond to bit flip errors and the entries of e_2 to phase flip errors. If an error occurred on the i-th qubit of our quantum state, then the i-th entry of the corresponding error vector is 1, otherwise it is 0.
We start with correcting the bit flip errors by computing the syndromes for the linear code C_1 using its parity check matrix H_1. For a quantum state $|\Psi\rangle|0\rangle$ there is a transformation to compute $|\Phi\rangle|H\Phi\rangle$. For more details on how this works, check [13, p. 451]. Since $H_1|x+y+e_1\rangle = |H_1 e_1\rangle$, we can now compute

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y)\cdot e_2} |x+y+e_1\rangle |H_1 e_1\rangle .$$

Measuring $|H_1 e_1\rangle$ gives us all the information we need to correct the bit flip errors. The quantum state then is

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y)\cdot e_2} |x+y\rangle .$$

Phase flip errors can be dealt with in the same way. First we turn our phase flip errors into bit flip errors by applying the Hadamard transformation, then we correct the bit flip errors as described above, and apply the Hadamard transformation again to recover the original state $|x+C_2\rangle$. A more detailed description of the calculations can be found in [13, p. 451].

For two n-dimensional binary vectors $w \notin C_1$ and $v \in C_2$ we can define a CSS-code
$CSS_{v,w}(C_1, C_2)$. For $x \in C_1/C_2$ the codeword is given by

$$|x, v, w\rangle := \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{v\cdot y} |x+y+w\rangle .$$

So any CSS-code $CSS(C_1, C_2)$ gives rise to a family $\{CSS_{(v,w)}(C_1, C_2)\}_{v,w}$ of CSS-codes with the same error correcting properties. Notice, that the $\{|x, v, w\rangle\}$ form a basis of a 2^n dimensional vector space.

3.2.6 Distance Measures for Quantum Information

Now that we have established quantum noise, the question of how to measure the distance from a desired quantum state to the actual resulting state of quantum computation arises. This section is based on chapter 9 of [13], the focus lies on the trace distance and fidelity and the relationship between the two. The trace distance and the fidelity function are both distance measures for probability distributions in the classical world. So we will start with the definitions in the classical case.

Definition 3.2.3 (Classical Trace Distance)**.** *Let $\mathcal{X}$ denote some finite set and $p = \{p_x\}_{x\in\mathcal{X}}$, $q = \{q_x\}_{x\in\mathcal{X}}$ be two probability distributions on $\mathcal{X}$. The trace distance between p and q is defined as*

$$D(p, q) := \frac{1}{2} \sum_{x \in \mathcal{X}} |\, p_x - q_x \,| .$$

The trace distance defines a metric on the set of probability distributions on $\mathcal{X}$.

Definition 3.2.4 (Classical Fidelity)**.** *Let $\mathcal{X}$ denote some finite set and $p = \{p_x\}_{x\in\mathcal{X}}$, $q = \{q_x\}_{x\in\mathcal{X}}$ be two probability distributions on $\mathcal{X}$. The fidelity of p and q is defined as*

$$F(p,q) := \sum_{x\in\mathcal{X}} \sqrt{p_x q_x}$$

While the fidelity is not a metric, it still is a good measure for the distance of probability distributions. It is easy to see from the definition, that $0 \leq F(p_x, q_x) \leq 1$ and $0 = F(p_x, q_x)$, iff $p_x = q_x \ \forall\ x \in \mathcal{X}$.
Now we turn to the quantum world. The question we are trying to answer here is, how close two quantum states are. Here the notion of density matrices comes in especially handy and it will also become clear, what the trace distance has to do with the trace function. We will later use the trace distance for a notion of secrecy in quantum key distribution.

Definition 3.2.5 (Trace Distance)**.** *Let ρ and ρ' be two density operators of quantum states with the same state space. Then the trace distance of ρ and ρ' is defined as*

$$D(\rho,\rho') := \frac{1}{2}\, tr\,|\,\rho - \rho'\,|,$$

where $|\,A\,| := \sqrt{A^T A}$ denotes the positive square root of a matrix $A^T A$.

Unitary transformation leave the trace distance unchanged, i.e. $D(U\rho U^T, U\rho' U^T) = D(\rho,\rho')$

Definition 3.2.6 (Fidelity)**.** *Let ρ and ρ' be two density operators of quantum states with the same state space. Then the fidelity of ρ and ρ' is defined as*

$$F(\rho,\rho') := tr\sqrt{\rho^{\frac{1}{2}}\rho'\rho^{\frac{1}{2}}}.$$

Again the fidelity is not a metric, but it is symmetric in its inputs. A case we will be paying special attention to later, is the fidelity between a pure state $|\Psi\rangle$ and an arbitrary state ρ:

$$\begin{aligned} F(|\Psi\rangle,\rho) =& \operatorname{tr}\sqrt{\langle\Psi|\,\rho\,|\Psi\rangle\,|\Psi\rangle\,\langle\Psi|} \\ =& \sqrt{\langle\Psi|\,\rho\,|\Psi\rangle} \end{aligned}$$

The fidelity and the trace distance are closely related, which is reflected in the following equation

$$1 - F(\rho,\rho') \leq D(\rho,\rho') \leq \sqrt{1 - F(\rho,\rho')^2}$$

3.3 The BB84 Protocol

The BB84 protocol is named after Charles H. Bennett and Gilles Brassard, who introduced it in 1984. Just like the DH-protocol it allows two parties A and B to share a secret key, without sharing any secrets previously. It is probably the best known key distribution protocol in quantum cryptography.
The main advantage of the BB84 protocol is, that it is provably secure. We will define what security means in quantum cryptography later.

In this section we first introduce the prepare-and-measure-version of the BB84 protocol, followed by a notion of security. In order to prove, that the prepare-and-measure-version fulfills this notion, we then present an entanglement-based version of the BB84 protocol. So why not start with that in the first place? While proving security is a lot easier for the entanglement-based version, the advantage of the prepare-and-measure version is its realizability.
As you will see the prepare-and-measure version only requires the parties to perform measurements. A prepares the necessary qubits in some state at the beginning, after that no unitary transformation needs to be applied. The entanglement-based version in contrast requires both parties to perform the Hadamard transformation on all of their qubits at some point, which is rather impractical, considering how rare and expensive even small quantum computers are.

The bigger part of this section is based on [15]. The proof of security follows [15], but also includes additional material from chapter 12 of [13].

3.3.1 The Prepare-and-Measure Version

Two parties A and B need to obtain a joint secret key. They have access to a quantum channel and a classical channel. The eavesdropper E can only listen to the classical channel, but both inject or alter messages. Let $B_+ := \{|0\rangle, |1\rangle\}$ and $B_\times := \{|0\rangle_\times, |1\rangle_\times\}$ the conjugate bases from above, with

$$|0\rangle_\times := H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle_\times := H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The protocol:

1. A prepares 2n qubits, each in a random state $|x_i\rangle \in \{|0\rangle, |1\rangle, |0\rangle_\times, |1\rangle_\times\}$ and sends them to B.
2. For every received state $|x_i\rangle$ B chooses a bases $\mathcal{B} \in \{B_+, B_\times\}$ at random and measures $|x_i\rangle$ in the chosen basis.
3. A announces her base choices. A and B only keep those bits, where they used the same base. This is called the sifted key s.

4. To estimate the error-rate the bit-values of a random subset of bits of s are compared. The protocol is aborted, if a certain threshold of the error-rate is surpassed, that is if too many values differ.

5. A and B perform error correction and privacy amplification on those bits of s whose bit values have not been announced and take the resulting bit string as their joint secret key.

In step 2 the result of B's measurement is perfectly correlated with the state A prepared if B chooses the bases the state was prepared in. If the base of preparation does not coincide with the base the state is measured in, B's result is perfectly random.
Step 3 reduces the number of qubits roughly by half and then error-estimation, -correction and privacy amplification shorten it further. Assuming that each of these operations also expends about half of the qubits left, the joint key of A and B will be of expected length $\frac{n}{8}$.

A's state	$\|0\rangle$	$\|0\rangle_\times$	$\|0\rangle_\times$	$\|0\rangle_\times$	$\|1\rangle_\times$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle_\times$
B's base choice	B_+	$B_\times$	B_+	$B_\times$	$B_\times$	$B_\times$	B_+	$B_\times$	B_+	$B_\times$
B's result	$\|0\rangle$	$\|0\rangle_\times$	?	$\|0\rangle_\times$	$\|1\rangle_\times$	?	$\|0\rangle$	?	$\|1\rangle$	$\|0\rangle_\times$
sifted key	0	0		0	1		0		1	0

3.3.2 The intercept-and-resend attack

In order to get knowledge of the joint key E would need to somehow achieve listening to the quantum channel. That is a tricky task though, since it is not possible to copy qubits and measuring them alters the state.
A very simple attack on the BB84 protocol is to try just that. The adversary intercepts all of the qubits sent by A in step 1 and guesses bases to measure them. For about half of the bits E's guess will coincide with the base A prepared the qubit in. In those case the state of the qubit remains unchanged.
Whenever E guesses the wrong bases, the state of the qubit will be changed. So if E now sends the measured qubits to B, who is expecting to receive 2n qubits from A, only about half of the qubits are still in the original state.
Now A and B continue with steps 2 and 3. They will still choose the same base roughly half of the time, but because of E's interference B will get random results for half of the bases where he and A used the same bases.
The error-estimation in step for should therefore detect an error-rate around 25%, which will lead to aborting the protocol.

A's state	$\|0\rangle$	$\|0\rangle_\times$	$\|0\rangle_\times$	$\|0\rangle_\times$	$\|1\rangle_\times$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle_\times$
E's base choice	B_+	B_+	$B_\times$	B_+	$B_\times$	B_+	$B_\times$	$B_\times$	B_+	$B_\times$
E's result	$\|0\rangle$	?	$\|0\rangle_\times$	?	$\|1\rangle_\times$	$\|1\rangle$	?	?	$\|1\rangle$	$\|0\rangle_\times$
B's base choice	B_+	$B_\times$	B_+	$B_\times$	$B_\times$	$B_\times$	B_+	$B_\times$	B_+	$B_\times$
B's result	$\|0\rangle$	?	?	?	$\|1\rangle_\times$	?	?	?	$\|1\rangle$	$\|0\rangle_\times$
sifted key	0	?		?	1		?		1	0

3.3.3 Notion of security

The basic setting we assume for a quantum key distribution protocol is usually this: Two parties A and B communicate over an authenticated classical channel and a quantum channel. The quantum channel is noisy and possibly under the control of an eavesdropper.

Let us see what qualities a quantum key distribution protocol should have ideally. The first thing, that comes to mind here obviously is provable security. While classical key distribution relies on the assumed hardness of some mathematical problem, the laws of quantum mechanics guarantee the security of quantum key distribution. The downside however is that quantum channels are noisy, which allows for an eavesdropper to pass himself off as white noise and makes it necessary, that the protocol still works with a certain amount of error. Since the noise on a quantum channel increases with the distance, communication over arbitrary distances still poses a problem.
Another problem is, that quantum computers are still being developed and very expensive, so a protocol should work without requiring the parties to perform unitary transformations.
To sum it up we need a quantum key distribution protocol to be provably secure, error tolerant, executable over arbitrary distances and of course realizable in the way that it should only require transmitting and measuring qubits.

Definition 3.3.1 (Perfect Secrecy)**.** *Let $\mathcal{S}_\mathcal{P}$ denote the set of all possible key-outputs of a key distribution protocol $\mathcal{P}$. Then $\mathcal{P}$ is said to achieve perfect secrecy if and if only the following two conditions hold:*

1. *Any key $s \in \mathcal{S}_\mathcal{P}$ is equally likely to be the outcome of $\mathcal{P}$.*
2. *Any adversary has no information on the outcome s of $\mathcal{P}$, i.e. the state of any quantum system controlled by the adversary is independent of s.*

At the end of the entanglement-based version of the BB84 protocol, which we will use to proof the security of the prepare-and-measure version, A and B share some quantum state ρ_{AB}, that they will then measure. Each party measures its half of the qubits in order to obtain a classical bit string - the shared secret key. Ideally the quantum state A and B share is pure, i.e. not entangled with any quantum system an adversary E might hold. As mentioned in section 3.2. it is unrealistic to assume, that A and B possess a perfectly closed quantum system. So what is the worst case scenario here? - E being in full control of

the quantum channel translates into her holding a reference system ρ_E, that is part of a purification $|\Psi\rangle_{ABE}$, i.e. $\rho_E = \mathrm{tr}_{AB}(|\Psi\rangle_{ABE})$ and $\rho_{AB} = \mathrm{tr}_E(|\Psi\rangle_{ABE})$.

The ideal outcome of ρ_{AB} would be the pure state $|\Phi^+\rangle^{\otimes n}$. In that case the purifying system is of the form $|\Psi\rangle_{ABE} = \rho_{AB} \otimes \rho_E$, i.e. E cannot gain any information on the secret key through her reference system ρ_e.
If A and B each measure their part of $|\Psi\rangle_{ABE}$, the outcome will be a pair of keys $|s_A\rangle \otimes |s_B\rangle \otimes \rho_E$ together with a reference system depending on the outcomes s_A and s_B. Notice that the output-keys s_A and s_B do not necessarily have to be equal to one another.
Let $\mathcal{S}_\mathcal{P}$ be the set of all possible key-outputs of a protocol $\mathcal{P}$. Then ρ_{ABE} can be expressed in the following way:

$$\rho_{ABE} = \sum_{s_A, s_B \in \mathcal{S}} \Pr[s_A, s_B] \, |s_A\rangle \langle s_A| \otimes |s_B\rangle \langle s_B| \otimes \rho_E^{s_a, s_B}$$

If $\mathcal{P}$ were to achieve perfect secrecy, the output state that A and B share, should be independent of any system E holds, only key pairs with $s_A = s_B$ should have a positive probability, moreover every key pair with that property should be the outcome of a measurement with equal probability. The desired output state of the purification system looks like this:

$$\rho_{ideal} = \sum_{s \in \mathcal{S}} \frac{1}{|\,\mathcal{S}\,|} \, |s\rangle \langle s| \otimes |s\rangle \langle s| \otimes \rho_E$$

Achieving perfect secrecy will hardly be possible in the real world, but, as we shall see, we can get arbitrarily close to it. With the help of the fidelity function, we now introduce the notion of ϵ- security from [15].

Definition 3.3.2 (ϵ-Security)**.** *Let ρ_{ABE} be the outcome state of the key distribution protocol $\mathcal{P}$ shared by A, B and E defined above. The key pairs, that are output of $\mathcal{P}$ are said to be ϵ-secure with respect to, iff*

$$D(\rho_{ABE}, \rho_{ideal}) \leq \epsilon.$$

3.3.4 The Entanglement-based Version

In order to prove the perfect secrecy of the BB84 protocol, we now introduce the entanglement-based version of it. For the entanglement-based version, the proof of security will be much simpler, but the protocol itself is not practical, since it requires the involved parties to perform unitary transformations.
After proving the security, we will show that both versions of the BB84 protocol are equivalent in the sense, that they require the same operations.

Here A and B use the one of the Bell states, namely $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, to transmit the information via entanglement.
Notice, that $|\Phi^+\rangle$ has the same coefficients with respect to the bases B_+ and $\mathrm{B}_\times$, that is

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle\,|0\rangle + |1\rangle\,|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle_\times\,|0\rangle_\times + |1\rangle_\times\,|1\rangle_\times).$$

Therefore when the first and the second qubit are measured in the same base, the results will be perfectly correlated, measuring in different bases yields completely uncorrelated random results. For both bases the outcomes 0 and 1 have the same probability when only one of the qubits is measured.

The protocol:

1. A prepares 2n qubit pairs in the state $|\Phi^+\rangle^{\otimes 2n} = |\Phi^+\rangle \otimes ... \otimes |\Phi^+\rangle$ and selects random 2n-bit strings $c = (c_1, ..., c_{2n})$ with $\sum_{i=1}^{2n} n$ and $b = (b_1, ..., b_{2n})$.
2. A performs the Hadamard transformation H on her half of the i-th qubit pair, if $b_i = 1$ and sends the other half of each pair to B.
3. Once B received all 2n qubits, A announces b and c.
4. B applies H to his i-th qubit if $b_i = 1$.
5. A and B both measure their i-th qubit in B_+, if $c_i = 1$ and compare the values to estimate the error rate. They abort, if it surpasses a certain threshold.
6. A and B perform error correction on the remaining qubits, i.e. those with $c_i = 0$ and measure them in B_+ to obtain the joint secret key.

The Hadamard transformation in steps 2 and 4 is necessary make the intercept-and-resend attack detectable. Otherwise an eavesdropper could just intercept and measure all of the qubits A sends to B with respect to B_+ without causing any disturbance.
For the error correction A and B use a CSS-code, thus in step 6 A and B first measure the error syndrome for each of their qubits and then correct errors, where necessary.

3.3.5 Proof of Security

In this section we will proof the entanglement-based version of the BB84 protocol to be ϵ-secure.
In the last step of the entanglement-based version A and B share a state ρ_{AB}, may or may not be entangled with some reference system ρ_E E is holding.
It is clear, that, if ρ_{AB} is not entangled with E's system, any key pair resulting from measuring ρ_{AB} is by definition ϵ-secure for every $\epsilon > 0$. Unfortunately we have to assume, that ρ_{AB} is in fact entangled with some other state ρ_E, be that because of an adversary, or simply because of unwanted interaction with the environment. The following lemma will give us a way to link the fidelity ρ_{AB} and the ideal state $|\Phi^+\rangle^{\otimes n}$ to the security of the key pairs, generated by the BB84 protocol. We will state it here without proof. A proof can be found in [16].

Lemma 3.3.3. *Let ρ_{AB} be a quantum state shared by two parties A and B. Any 2n-bit string resulting from locally measuring ρ_{AB} in B_+ is an ϵ-secure key with respect to an adversary holding a reference system, that is part of a purification of ρ_{AB}, if $\epsilon > 0$ and*

$$F(AB, |\Phi^+\rangle^{\otimes n}) \geq \sqrt{1-\epsilon^2}.$$

Now we only need to make sure, that we can estimate the error rate from determining the error rate on the check qubits. For that purpose we will fall back on classical probability theory. Again we will state the lemma without proof. A sketch of the proof can be found in chapter 12 of [15].

Lemma 3.3.4. *Let s denote a random n-bit string and c be a random subset of n bits of s. Assume that s may contain some errors and that those errors occur with equal probability on each bit. Then for any $\epsilon, \delta > 0$ the probability of finding fewer than δn errors in c and more than $(\delta + \epsilon)n$ errors in $s \setminus c$ is upper-bounded by $e^{-\mathcal{O}(\epsilon^2 n)}$, for sufficiently large n.*

So by increasing the number of check qubits, we can estimate the error rate exponentially well. This allows us to estimate the fidelity of ρ_{AB} with an exponentially small rate of error, which concludes the proof of security.

There is still a downside however. While we can determine rather well how safe a key is, making it arbitrarily safe is a different story. As previously mentioned, an adversary is not the only source of quantum noise. Therefore A and B will need to establish a threshold on the error rate, which then gives the adversary the chance to attack undetected.

3.3.6 Equivalence of the two versions

To prove the security of the prepare-and-measure version, we now show the equivalence of the two versions. Equivalence here means, that it does not make a difference to an adversary which version of the protocol is executed, i.e. the information accessible is the same for both versions.
This section is based on chapter 12 of [13] and [15].
Starting with the entanglement-based version, we will simplify the protocol step by step to get to the prepare-and-measure version.
A major difference between the two versions is when the measurements take place. The first step is to justify, why A can perform all her measurements in the first step, i.e. that it is not necessary for A and B to share pairs of entangled qubits.
Sharing a state $|\Phi^+\rangle$ and then measuring it later essentially provides B with a state $|0\rangle$ or $|1\rangle$ with equal probability. Hence A can just as well create 2n qubits, each at random in state $|0\rangle$ or $|1\rangle$ and send those to B.

To clarify why the measurement of the syndrome can also be done right in the beginning, let us take a closer look at how the CSS-code is used here.

Recall that for a family of CSS-codes $\{CSS_{v,w}(C_1, C_1)\}$, the set

$$\{|x, v, w\rangle\}_{x \in C_1/C_2, v, w}, \text{ with } |x, v, w\rangle = \frac{1}{\sqrt{|\, C_2 \,|}} \sum_{y \in C_2} (-1)^{v \cdot y} |x + y + w\rangle ,$$

forms a basis of the 2^n-dimensional vector space. This allows us to write our shared state $|\Phi^+\rangle$ as

$$|\Phi^+\rangle^{\otimes n} = \sum_{i=0}^{2^n - 1} |i\rangle\, i = \frac{1}{\sqrt{2}} \sum_{x,v,w} |x, v, w\rangle\, |x, v, w\rangle , \tag{3.2}$$

where i is in binary notation. So measuring the syndromes for bit and phase flip errors provides A with random results for v and w. A last measurement with respect to the basis $\{|0\rangle, |1\rangle\}$ then yields a random codeword x.
Of course all those measurements do not leave B's qubits untouched. Let x_0, v_0 and w_0 denote the results of A's measurement. Then after the last measurement of A, B holds a state $|x_0, v_0, w_0\rangle$, as we can deduce from (3.2). $|x_0, v_0, w_0\rangle$ is a random quantum state encoded in a random CSS-code.
So instead of preparing Bell states, A can choose random n-bit strings x, v and w to encode x with $CSS_{v,w}(C_1, C_1)$. This gives us a random n-bit string, which A can then extend to a random 2n bit string by inserting check bits in random positions. Executing the second step of the entanglement-based version then puts us in the same position, we are in after the first step of the prepare-and-measure version.

In the prepare-and-measure version we use the state $|\Phi^+\rangle$, because of its properties regarding measurements in the bases B_+ and $\mathrm{B}_\times$. Since the state $|\Phi^-\rangle = \frac{1}{2}(|00\rangle - |11\rangle)$ has exactly the same properties, phase flip errors make no difference for A and B and do not need to be corrected, thus A does not need to send the v to B.
So really A can just create the state $\rho = \frac{1}{2^n} \sum_v |x, v, w\rangle \langle x, v, w|$ and send that to B. This can be simplified even further to

$$\begin{aligned} \rho &= \frac{1}{2^n} \sum_v |x, v, w\rangle \langle x, v, w| \\ &= \frac{1}{|\, C_2 \,|} \sum_{z \in C_2} |x + z + w\rangle \langle x + z + w| \end{aligned}$$

We already chose x at random from C_1/C_2, choosing x at random from C_2 has the same effect as the summation over all $z \in C_1$, i.e. sending a state $|x + e\rangle$ with $x \in C_1$. Now $y := x + w$ is an entirely random n bit string.
At this point A sends $|y\rangle$ to B who receives a state $|y + e\rangle$ due to noise on the channel or disturbance by an adversary. Measuring that state yields the classical bit string $y + e$, A then announces $y - x$ so that B can calculate $x + e$ and use error correction to obtain x.

The last difference between the two versions now is the Hadamard transformation. If A, instead of choosing a classical random bit string, just goes through steps 1 to 3 of the prepare-and-measure version, the equivalence of the two versions becomes clear.

3.4 Conclusions

Quantum cryptography is a two-edged sword. While it makes provably secure key distribution possible, it also breaks most of the encryption schemes in use today, that is in theory.
Although quantum computation has been the subject to much investigation over the past 30 years, it still is in its infancy. To this day the opinions diverge on whether it is physically possible to build quantum computers, powerful enough to keep up with the classical computers we have today. Although there have been a number of experiments, proving that quantum communication works, even over long distances, it still is far too expensive for the private sector and while the important questions in classical computation are of a rather theoretical nature, the most pressing question of the quantum world remains, whether and when it will reach a stage, where quantum computers become affordable and up to the tasks they can theoretically handle today.

Bibliography

[1] S. Goldwasser and M. Bellare, Lecture Notes on Cryptography. Notes, July 2008.

[2] W. Diffie and M. E. Hellman, New directions in cryptography. IEEE Trans. Inform. Theory IT-22:644-654, November 1976.

[3] D. Bru, G. Erdlyi, T. Riege and J. Rothe, Quantum cryptography: A survey. ACM Computing Surveys, 39(2), 2007, pp.1-27.

[4] U. M. Maurer Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In Proc. CRYPTO '94, pp. 271-281, Springer 1994.

[5] V. Shoup, Lower bounds for discrete logarithms and related problems. In Proc. Eurocrypt '97, Lecture Notes in Comp. Sci., pp. 256-266, Springer 1997.

[6] D. Boneh and R. Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In Proc. CRYPTO '96, Lecture Notes in Comp. Sci., Springer 1996.

[7] L. Babai, On Lovasz' lattice reduction and the nearest lattice point problem. Combinatorica, Vol. 6,1986, pp. 1-13.

[8] A. Lenstra, H. Lenstra and L. Lovasz, Factoring polynomial with rational coefficients, Mathematische Annalen, Vol. 261, 1982, pp. 515-534.

[9] M. Bellare and P. Rogaway, Entity authentication and key distribution. In Proc. CRYPTO '93, pp. 232-249, Springer, 1994. Lecture Notes in Comp. Sci. No. 773.

[10] M. Bellare and P. Rogaway, Provably secure session key distribution - the three party case. In Proc. 27th ACM Symp. on Theory of Computing, pp. 57-66, Las Vegas, 1995. ACM.

[11] D. Micciancio and O. Regev, Lattice-based cryptography. In Proc. PQCrypto 2008, Lecture Notes in Comp. Sci., Springer 2008.

[12] R. Gupta and M. Ram Murty, A remark on Artin's conjecture, Invent. Math., 78(1984), pp. 127-130

[13] M. Nielsen and I. Chuang, Quantum Computation and Quantum Information. Cambrigde University Press, 2000.

[14] D. Unruh, Short Notes for Quantum Cryptography, 2008.

[15] D. Bru, G. Erdlyi, T. Riege and J. Rothe, Quantum cryptography: A survey. ACM Computing Surveys, 39(2), 2007, pp.1-27.

[16] R. Konig, R. Renner, A. Bariska and U. Maurer, Locking of accessible information and implications for the security of quantum cryptography. Computing Research Repository (CoRR), 2006.