

DARMSTADT UNIVERSITY OF TECHNOLOGY

CRYPTOGRAPHY AND COMPUTERALGEBRA
DEPARTMENT OF COMPUTER SCIENCE



BACHELOR THESIS

Overview of Bit Commitment Schemes

Author:
Hristo LULEV

Supervisors:
Prof. Johannes
BUCHMANN and Erik
DAHMEN

December 16, 2007

Abstract

Commitment schemes have become cornerstones for building secure communication protocol. But in the past few years it has been shown that quantum computers could break most of the now existing secure communication structures. Problems began in 1995 when Mayers provided a theorem stating that secure commitment schemes are impossible in the quantum computer era. Major steps have been made since 2001 when the Shor's algorithm for first time was implemented on a 7-qubit quantum computer and according to some physicians in a few years this technology will be powerful enough to break the in practice using cryptographic protocols. Here I am considering the question weather there are some commitment protocols that could remain secure when quantum computers are used in the practice.

Contents

1	Introduction	3
2	Applications	3
3	Construction of a commitment scheme and its properties	5
4	Bit-commitment from Collision-Free Hashing	7
4.1	Definitions and Assumptions	8
4.2	The Commitment Scheme	9
4.3	Complexity and Analysis	10
5	Bit-commitment from a pseudo-random generator	11
5.1	Definitions and Assumptions	12
5.2	The Commitment Scheme	13
5.3	Complexity and Analysis	16
6	Quantum Computing	17
6.1	Some definitions and observations	17
6.2	Representation of qubit	17
6.3	Quantum Computing and Commitment schemes	18
6.4	Quantum Bit Commitment	20
7	Conclusions	22

1 Introduction

Commitment scheme is a protocol that allows one party (Alice - Sender) to commit to other party (Bob - Receiver) to a value. At a later stage Alice can reveal that value, but until this moment she is keeping it hidden. After the revealing Bob can verify that this is indeed the value to which Alice has committed. It's an electronic way to temporary hide a value that can not be changed later. One can commit to either a single bit, called Bit Commitment schemes (but often only Commitment schemes), to commit to a collection of bits $b_1, b_2 \dots b_m$ and have to revealed at the same time, strings etc. Such schemes are important to a variety of cryptographic protocols, especially zero-knowledge proof and secure computation, a simple example of using Commitment schemes is the coin flipping over the phone and zero-knowledge protocols such as the one described by Impagliazzo and Yung [1].

They can be used in scenarios like bidding for a contract, where committing to a bit rather than sending it in the clear can eliminate the risk of it being "leaked" to the competitors. The interactions in a commitment scheme take place in two phases: the *commit phase* during which a value is chosen and specified. At the end of the stage Bob has some information that represents the value sent. The *reveal phase* during which the value is revealed and checked. At the end of which Bob knows the real value. In the simplest commitment schemes, the commit phase consists of a simple message from the sender to the receiver, while the reveal phase consists of a single message from the sender to the receiver, followed by a check performed by the receiver.

It is therefore very important to construct good and reliable schemes, such that the Sender cannot alter the value to which he commits.

A "bit commitment" scheme is simply a commitment scheme where the value chosen is a bit. The terms "bit commitment" and "commitment" are sometimes used interchangeably.

2 Applications

The concept of bit commitment schemes was first formalized in a peer-reviewed paper by Giles Brassard, David Chaum, and Claude Crepeau in 1988 [2], but the concept was used without being treated formally prior to that, [3] most notably in the 1987 paper of Oded Goldreich, Silvio Micali,

and Avi Wigderson showing that all languages in NP have zero-knowledge proofs.

Commitments are used in zero-knowledge proofs for two main purposes: first, to allow the Prover to participate in "cut and choose" proofs where the verifier will be presented with a choice of what to learn, and the Prover will reveal only what corresponds to the verifier's choice. Commitment schemes allow the Prover to specify all the information in advance in a commitment, and only reveal what should be revealed later in the proof. Commitments are also used in zero-knowledge proofs by the verifier, who will often specify their choices ahead of time in a commitment. This allows zero-knowledge proofs to be composed in parallel without revealing additional information [4].

Commitment schemes are used the so called "coin flipping" (or tossing) over the phone. Suppose Alice and Bob need to flip a coin to settle up an argue. If they are in the same room everything is ok, but if not and Bob does not trust Alice, then protocol of the kind where the one flips the coin and imparts the result to the other will not work. How might we solve this problem such that it is impossible for Alice or Bob to have some kind of advantage?

What we really want is to make Alice and Bob commit to bits, in such a way that they cannot change their answer later. M. Blum proposed an explicit protocol [10] for Alice and Bob to follow, such that the outcome is a "random" coin toss that both Alice and Bob can agree is random. Moreover, Alice and Bob can verify to each other that they did not cheat: i.e. they followed the protocol exactly.

This protocol is based on the Discrete Logarithm Problem and Bob cannot "win" the coin toss with probability greater than $\frac{1}{2} + \frac{1}{p(n)}$, for all polynomials $p(n)$.

Another important application of commitments is in verifiable secret sharing - each of several parties receives "shares" of a value that is meant to be hidden from everyone. This is a critical building block of secure *multi-party computation* [5]. This problem was initially suggested by Andrew C. Yao in a 1982. In these kinds of schemes each of several parties receives "shares" of a value that is meant to be hidden from everyone. If enough parties get together, their shares can be used to reconstruct the secret, but even malicious parties of insufficient size should learn nothing. That idea is a cornerstone of the so called *secure computation*: in order to securely compute a function of some shared input, the secret shares are manipulated instead. However,

if shares are to be generated by malicious parties, it may be important that those shares can be checked for correctness. In a verifiable secret sharing scheme, the distribution of a secret is accompanied by commitments to the individual shares. The commitments reveal nothing that can help dishonest parties, but the shares allow each individual party to check to see if their shares are correct.

3 Construction of a commitment scheme and its properties

The two cryptographic properties of a commitment scheme are the *hiding* property - the value chosen during the commit phase cannot be discovered and the *binding* property - the value chosen during the commit phase is the only one that can be revealed during the reveal phase.

A commitment scheme is *binding* if Alice cannot change the value of b and it is *concealing* if Bob cannot obtain any information about b without the help of Alice. It is *secure* if it is simultaneously binding and concealing, and it is *unconditionally* secure if it is secure against a cheater, either Alice or Bob, equipped with unlimited technology and computational power. A commitment scheme can either be *perfectly binding* or *perfectly concealing* but not both.

There are a few ways of constructing reliable commitment scheme: from One-Way Function, from Pseudo-Random Generator or for example we can create it based on the Discrete Logarithm Problem.

In the next section I will introduce the basic commitment schemes explained in the papers of Moni Naor using cryptographically secure pseudo-random generator [3] and the scheme described by Shai Haveli and Silvio Micali based on a hash function [6]. In the third section I will present the basis of quantum computing, the hazards that it contains for the classical cryptographic protocols. A couple of algorithms that could break them and which protocols could be considered safe from that point of view. I will describe what Quantum cryptography is, what are the differences between Public Key Cryptography and Quantum Cryptography and what Secure Communication based on Quantum Cryptography is.

It is easily seen that if both parties have unlimited computational power, they cannot emulate the process by just exchanging messages back and forth.

Thus, at least one of the two parties must be computationally bounded, so that cryptographic technology can be applied. Indeed, many commitment schemes have been suggested in the literature. A particularly important case of string commitment is when the Sender is computationally bounded, but the Receiver may have unlimited computational resources. This is so for a few reasons:

- Bounded-to-unbounded commitment schemes allow one to use suitable short security parameters even if the Receiver has a lot of computational power.
- Bounded-to-unbounded commitment schemes protect the Sender even if the underlying cryptographic assumption happens to be wrong, say, if the Receiver has an algorithm for factoring.
- There are theoretical applications in which one must use bounded-to-unbounded commitment schemes to yield the desired result; for instance, to obtain constant-round computational zero-knowledge proofs for NP (shown in [7]), or to obtain statistical zero-knowledge arguments for NP (shown in [8],[9])

Many commitment schemes in the unbounded-receiver model are known based on number-theoretic constructions. The first such scheme was suggested by Blum [10] in the context of flipping coin over the phone. He described a commitment scheme for one bit. Blum's scheme calls for one or two modular multiplications and a k -bit commitment string for every bit which is being committed to, where k is the size of the composite modulus. A more efficient construction, which is also based on the hardness of factoring, was introduced by Goldwasser, Micali and Rivest [11]. Their collision-free permutation-pairs enables one to commit to long messages using about the same amount of local computations as in Blum's scheme, but to send only k -bit commitment string, regardless of the length of the message being committed to. Since then, this construction was used in a many other works. One common problem of all those is that they rely on composite numbers of a special form.

Several other constructions are based on the difficulty of extracting discrete logarithms. In particular, Pedersen [12] and Chaum, van Heijst and Pfitzmann [13], described a scheme in which the Sender can commit to a string of length k , where k is the size of the prime modulus, by performing two

modular exponentiations, and sending a k -bit commitment string.

There are also a few implementations using more generic complexity assumptions. Naor [3] presented a commitment scheme in the unbounded-to-bounded model, which can be implemented using any pseudo random generator. This scheme I am going to present later.

A bit-commitment scheme can also be based on the discrete logarithm problem. Suppose that Alice wants to commit to a value x , which is between 0 and $p - 1$, where p is the prime order of a group and g is its generator. She calculates $c = g^x$ and publishes c . On a non-quantum computer this scheme, based on the discrete logarithm problem should be binding: it is computationally infeasible to compute x , so under this assumption, Bob cannot compute x . On the other hand, Alice cannot compute an $x' \neq x$, such that $c = g^{x'}$.

This scheme isn't perfectly concealing as someone could find the commitment if he manages to solve the discrete logarithm problem, what I'll actually show later.

In addition to the above work, several researchers showed that a commitment scheme for a single bit can be implemented using "quantum computing devices". The first such scheme was scheme by Bennet and Brassard [14], unfortunately a considerable flaw was found. Better schemes were later suggested by Brassard and Crépeau [15] and Brassard, Crépeau, Jozsa and Langois [16], which also were doomed.

4 Bit-commitment from Collision-Free Hashing

One can create a bit-commitment scheme from any one-way permutation. A one-way permutation is an on one-way function that is also a permutation, that is, a one-way function that is both injective and surjective i.e. bijective. One-way permutations are an important cryptographic primitive, and it is not known that their existence is implied by the existence of one-way functions.

Here I present the scheme from the paper of Shai Halevi and Silvio Micali - "Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing". That scheme is provably secure under a standard assumption that the Sender is computationally bounded and the Receiver is all-powerful and

more efficient than many others. We assume as well the existence of collision-free hash function. These are functions that map strings of arbitrary length to fixed-length ones, so that it is infeasible to find two different pre-images of one output string. In fact such functions are believed to exist and are widely used in many cryptographic applications.

4.1 Definitions and Assumptions

To point out the efficiency of that scheme let's consider the three important resources for a protocol:

Interaction: protocols are typically interactive i.e. communication by exchanging messages back and forth. That scheme is non-interactive, i.e. in each stage the Sender sends a single message to the receiver who needs not to reply at all.

Communication: another important resource in a protocol is the number of bits sent by its parties. In a commitment scheme, this is measured against the length of the message being committed to (denoted by n), and the security parameter (denoted by k). (1) The number of bits exchanged during the Commit stage must be at least $n + k$. The presented one requires that the Sender sends $O(k)$ bits in the commit stage and $n + O(k)$ bits in the de-commit stage (2). Thus the communication complexity is optimal. *Computation:* the amount of local computation is crucial resource. The described scheme calls for (1) a single collision-free hashing of the message; (2) one collision-free hashing of a random $O(k)$ -bit string; and (3) one evaluation of a universal hash function on a $O(k)$ -bit string.

First of all I will point out why simpler constructions based on collision-free hashing do *not* work. Let MD (for Message-Digest) be a collision-free hash function. One example of a false solution is provided by the "minimal" strategy the Sender commits to a message M simply by sending $C = MD(M)$ to the Receiver, and de-commit by simply sending M . Trying to fix the flaw, one could try to have the Sender first pad the Message M with some sufficiently long random string R , and then sends $C = MD(M \circ R)$, where \circ is a concatenation. Unfortunately this construction may not work either, even when the Receiver is bounded. Indeed, it may be that MD , though collision-free, leaks some of the bits of M . In addition, in our more difficult model, the unbounded Receiver may get a good probabilistic advantage in guessing which of two messages M and M' is more likely to be the committed one. For instance, he can compute the size of the pre-image of C when the

message is M , compare it to a size of the pre-image of C when the message is M' , and guess accordingly.

Of course, the latter attack can be prevented if MD has additional properties besides being collision-free. However, we do not want to assume additional properties in our construction, since the more assumptions we make, the less likely it is that these assumptions are true, yet we wish to have an efficient scheme whose security against unbounded Receiver is provable.

The scheme is close to the second idea above, but "adds random bits" to the message in a more sophisticated way, thus enabling a proof of correctness against an unbounded Receiver without any additional assumptions.

Suppose that we want to commit to a string of length $O(n + k)$, where n is the length of the message being committed to a k is the security parameter and a little bit later I'll show how to modify it so as to get an $O(k)$ -bit commitment.

For the next construction we will need definition what universal hash function is. Let S and T be two sets, and let H be a family of functions from S to T . We say that H is a universal family of hash functions if for any two different elements $s_1 \neq s_2$ in S and for any two elements t_1, t_2 in T we have

$$Pr [h(s_1) = t_1 \text{ and } h(s_2) = t_2] = \frac{1}{|T|^2}$$

$h \in H$

Informal Universal hash function is a theoretical construct primarily used to show that an algorithm based on a hash function cannot be forced to have bad performance by an adversary. Bad performance in hashing comes from collisions, and a universal hash function guarantees that these cannot be forced to occur too often.

4.2 The Commitment Scheme

Fix the message length n and the security parameter k and set $L = 4k + 2n + 4$. Let $MD : \{0, 1\}^L \rightarrow \{0, 1\}^k$ be a collision-free hash function. That is we assume that the Sender cannot find $x \neq y \in \{0, 1\}^L$ so that $MD(x) = MD(y)$. Also, let H be a universal family of functions from $\{0, 1\}^L$ to $\{0, 1\}^k$. To commit to a message $m \in \{0, 1\}^n$, the Sender first picks a random $r \in \{0, 1\}^L$ and computes $y = MD(r)$ and then picks a random function h from H such that $h(r) = m$.

The commit-string is $c = \langle h, y \rangle$, and the de-commit string is $d = r$. To de-commit m the Sender sends r to the Receiver, who verifies that $y = MD(r)$ and computes $m = h(r)$.

The scheme is indeed non-interactive and requires very little local computation. If we use the construction of universal hashing function then the size of the commitment string is

$$|h| + |y| = (L + 2k) + k = 7k + 2n = O(k + n).$$

To get an $O(k)$ -bit commitment scheme we need to do the following: on a message m , the Sender first computes the k -bit string $s = MD(m)$, and then apply the above commitment string to the string s . To de-commit m the Sender sends both the message m and the de-commit message of first scheme. The Receiver checks that s is the string being committed to in the first message and that $MD(m) = s$. Since we execute the first scheme on a message of length k , then the commitment string is of length $7k + 2k = 9k$, regardless of the message length.

4.3 Complexity and Analysis

For any string $m \in \{0, 1\}^n$, let $C_k(m)$ is the distribution on the first coordinate of a pair which is obtained by running the algorithm $SEND(1^k, m)$. We require that

$$\forall m_1, m_2 \in \{0, 1\}^n, \| C_k(m_1) - C_k(m_2) \| = O(2^{-k})$$

i.e. the Receiver gets almost no statistical advantage about m from the committed string. (any two distributions : $C_k(m_1), C_k(m_2)$ are statistically close up to 2^{-k})

The commitment string is of length $9k \in O(k)$, regardless of message length. *Open problems:* One open problem is to design efficient commitment schemes which have nice homomorphism properties. In particular, in some scenarios it is desirable to be able to compute a commitment for $a + b$ (or $a.b$) from the commitments to a and to b .

Another interesting one is to reduce the assumption that universal one-way hash functions are sufficient for commitment in the unbounded receiver model.

5 Bit-commitment from a pseudo-random generator

Here I will show how in [3] is created a bit commitment protocol from pseudo-random generator and the same method is used to commit to many bits very efficiently. It is interactive, and requires 2 rounds of communication to commit to a string. The Sender in this scheme generates an $O(n)$ -bit pseudorandom string and sends an $O(n)$ -bits commitment string in order to commit to an n -bit message. The following scheme is based on the paper of Mani Noar, in which he, in 1991, showed how to create a bit-commitment scheme from a cryptographically secure pseudo-random generator. In this paper he shows that a bit-commitment scheme can be constructed with any pseudo-random generator. In fact this is weaker condition than the one-way functions, since Yao [17] has shown that pseudo-random generators can be based on one-way permutations. In fact he also analyzed the communication costs and showed that the assumption of the existence of a pseudo-random generators suffices to assure amortized $O(1)$ bits of communication per bit commitment. A pseudo-random generator is a function that maps a string (the seed) to a longer one, such that if the seed is chosen at random, the output is indistinguishable from a truly random distribution for all polynomial time machines. Later Impagliazzo, Levin and Luby [18] have shown that given any one-way function, a pseudo-random (not necessary a permutation), a pseudo-random generator can be constructed, under non-uniform assumptions, and Hasted [19] has shown the same under uniform assumption. On the other hand, Impagliazzo and Luby [20] have argued that the existence of one-way functions is required for any protocol that must rely on computational complexity. Thus we can conclude that if any computational complexity based cryptography is possible, then bit commitment protocol exist, and so do the protocols that rely on bit commitment, such as zero-knowledge proofs, identification schemes, ect.

Now we would like to analyze what the communication complexity of a bit commitment is, i.e. how many bits must be transferred during the execution of the protocol? We exclude the case that only a fix number of bits will be exchanged during the execution of the protocol, because if that was the case after the commit stage Bob can guess with high probability what Alice would send in the revealing stage, and can verify that the guess is consistent with what she send so far and deduce the value if the bit. However

in many applications Alice wants to commit to collection of bits $b_1, b_2 \dots b_m$ and they are to be revealed at the same time. Such applications are coin flipping over the phone and zero-knowledge protocol like Impagliazzo and Yung [1]. Furthermore, Kilian, Micali and Ostrovsky [21] have shown that many of the known protocols for zero-knowledge can be converted to ones that have that property. Therefore it is desirable to amortize the communication complexity of bit commitment. It has been shown that if m is big enough, at least linear in the security parameter n , then Alice can commit to $b_1, b_2 \dots b_m$ while exchanging only $O(1)$ bits per bit commitment. The total computational complexity of the protocol is the same as the complexity of the protocol for committing to one bit.

5.1 Definitions and Assumptions

After the two stages (commit and revealing) the protocol must obey the following: for all probabilistic polynomial time Receivers, for all polynomials p and for large enough security parameter n

1. After the commit stage Bob can not guess what he has with probability more then $\frac{1}{2} + \frac{1}{p(n)}$
2. Alice can reveal only one value and if she tries to cheat she is going to be caught with probability at least $1 - \frac{1}{p(n)}$

A commitment to many bits protocol consists of the same two stages:

- The commit stage: Alice has a sequence of bits $D = b_1, b_2 \dots b_m$ to which she wants to commit to Bob. She and Bob exchange messages. At the end of the stage Bob has some information that represents D .
- The revealing stage: at the end of which Bob knows D .

The protocol must obey the following: for all probabilistic polynomial time Bobs (Receivers), for all polynomials p and for large enough security parameter n

1. For any two sequences $D = b_1, b_2 \dots b_m$ and $D' = b'_1, b'_2 \dots b'_m$ selected by Bob, following the commit stage Bob cannot guess whether D or D' was committed with probability greater then $\frac{1}{2} + \frac{1}{p(n)}$

2. Alice can reveal only one sequence of bits. If she tries to reveal a different one, then she is going to be caught with probability at least $1 - \frac{1}{p(n)}$

Pseudo-Random Generators: Let $m(n)$ be some function such that $m(n) > n$.

$$G : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$$

is a pseudo-random generator if for all polynomials p and all probabilistic polynomial time machines A that attempt to distinguish between outputs of the generator and truly random sequence, except for finitely many n 's:

$$|Pr[A(y) = 1] - Pr[A(G(s)) = 1]| < \frac{1}{p(n)}$$

where the probabilities are taken over $y \in \{0, 1\}^{m(n)}$ and $s \in \{0, 1\}^n$ chosen uniformly at random.

Pseudo-random sequence have the unpredictability of the next bit property: given the first m bits of a pseudo-random sequence, any polynomial time algorithm that tries to predict the next bit in the sequence has success probability smaller than $\frac{1}{2} + \frac{1}{p(n)}$ for any polynomial $p(n)$. It has been shown by Yao [22] that both definitions are equivalent.

In the rest of the section we assume some pseudo-random generator G . Let n be a security parameter which is assumed to have been chosen so that no feasible machine can break the pseudo-random generator for seeds of length n . We will use $G_l(s)$ to denote the first l bits of the pseudo-random sequence on seed $s \in \{0, 1\}^n$. $B_i(s)$ will be used to denote the i -th bit of the pseudo-random sequence on seed s .

5.2 The Commitment Scheme

To construct the commitment scheme on top of the above equivalent definitions for pseudo-random generator we construct the protocol:

- Commit stage - Alice selects seed $s \in \{0, 1\}^n$ and sends $G_m(s)$ and $B_{(m+1)}(s) \oplus b$, where b is the bit Alice wants to commit to.
- Reveal stage - Alice sends s , Bob verifies that $G_m(s)$ is what Alice sent him before and computes $b = B_{m+1}(s) \oplus (B_{m+1}(s) \oplus b)$

Because of the unpredictability of the pseudo-random sequence Bob cannot guess the bit Alice commits to before the reveal stage with a probability bigger than $\frac{1}{2} \frac{1}{\text{poly}(n)}$. Alice could be able to cheat if she finds two seeds s_1 and s_2 such that $G_m(s_1)G_m(s_2)$, but $B_{m+1}(s_1) \neq B_{m+1}(s_2)$, then she would be able to cheat sending any of the two bits she wants, but there is nothing in the definition of the pseudo-random generator that forbids the existence of such pairs. Furthermore, given any pseudo-random generator, one can construct another one G' that has such pairs.

Indeed there is nothing that could force Alice to stick to one seed, since there may be two seeds that could yield the same sequence, what the following protocol does is to make Alice to stick to the same sequence, or she will be caught with high probability.

- Commit stage

1. Bob selects a random vector $\vec{R} = (r_1, r_2, \dots, r_{3n})$ where $r_i \in \{0, 1\}$ for $1 \leq i \leq 3n$ and sends it to Alice.
2. Alice selects a seed $s \in \{0, 1\}^n$ and sends to Bob the vector $\vec{D} = (d_1, d_2, \dots, d_{3n})$ where:
$$d_i = \begin{cases} B_i(s), & \text{if } r_i = 0 \\ 3n + 1, & \text{if } r_i = 1 \end{cases}$$

- Reveal stage: Alice sends s and Bob verifies that for all $1 \leq i \leq 3n$, if $r_i = 0$ then $d_i = B_i(s)$, and if $r_i = 1$ then $c_i = B_i(s) \oplus b$.

This protocol maintains the property that Bob learns nothing about the bit b , otherwise we claim that Bob has the power to distinguish between outputs of the pseudo-random generator and truly random strings. If there exists some probabilistic polynomial time Bob' that can learn something about b when Alice uses a pseudo-random sequence, then Bob' can be used to construct a distinguisher between outputs of G and truly random sequences. On the other hand Alice still could cheat and her only chance to do this is if there are two seeds s_1, s_2 such that $G_{3n}(s_1)$ and $G_{3n}(s_2)$ agree in all positions i where $r_i = 0$ and totally disagree in all positions where $r_i = 1$. We say that such a pair fools \vec{R} .

(Claim) The probability that there exists a pair of seeds s_1, s_2 that fools \vec{R} is at most 2^{-n} , where the probability is taken over the choices of \vec{R} .

(Theorem) We claim that if G is a pseudo-random generator, for all polynomials p and for large enough security parameter n

- Following the commit stage, no probabilistic polynomial time Bob can guess b with a probability greater than $\frac{1}{2} + \frac{1}{p(n)}$
- Alice can reveal only one possible bit, except with probability less than 2^{-n}

We are now interested to extend this protocol and make possible Efficient Commit to Many Bits. Up to now the protocol has communication cost of $O(n)$ bits. If Alice wants to commit to many bits $b_1, b_2 \dots b_m$, which she is to reveal simultaneously we could use a better idea: to use many bits to force Alice to stick to one pseudo-random sequence and use that sequence to commit to many bits.

We could prevent the situation where Bob would like to see part of the pseudo-random sequence Xor-ed with b and he asks to see its bit-wise Xor with $b_1, b_2 \dots b_m$ and Alice might be able to change one of the b_i 's, since it is enough that there is a pair of seeds that agree on all the bits but one, by using error correction code C with large distance between code words. Let $C \subset \{0, 1\}^n$ be a code of 2^m words such that the distance between any $c_1, c_2 \in C$ is at least $\epsilon \cdot q$. Also let E be an efficiently computing function $E : \{0, 1\}^m \rightarrow \{0, 1\}^q$ for mapping words in $\{0, 1\}^m$ to C . The code requires that $q \cdot \log \frac{2}{2-\epsilon}$ must be at least $3n$, and $\frac{q}{m}$ be a fixed constant. Such codes and example could be the Justesen code [22]. For the amortization to work it suffices that m be linear in n .

For a vector $\vec{R} = r_1, r_2 \dots r_k$ with $r_i \in \{0, 1\}$ and with exactly q indices i such that $r_i = 1$ let $G_{\vec{R}}(s)$ denote the vector $\vec{A} = a_1, a_2 \dots a_{1k}$ where $a_i = B_{j(i)}(s)$ and $j(i)$ is the index of the i -th 1 in \vec{R} . If $e_1, e_2 \in \{0, 1\}^q$, then $e_1 \oplus e_2$ denotes the bitwise Xor of e_1 and e_2 .

- Commit stage
 1. Bob selects a random vector $\vec{R} = r_1, r_2 \dots r_{2q}$ where $r_i \in \{0, 1\}$ for $1 \leq i \leq 2q$ and exactly q of the r_i 's are 1 and sends it to Alice
 2. Alice computes $c = E(b_1, b_2 \dots b_m)$. Alice selects a seed $s \in \{0, 1\}^n$ and sends to Bob $e = c \oplus G_{\vec{R}}(s)$ (the bitwise Xor of $G_{\vec{R}}(s)$ and c), and for each $1 \leq i \leq 2q$ such that $r_i = 0$ she sends $B_i(s)$.

- **Reveal stage:** Alice sends s and $b_1, b_2 \dots b_m$. Bob verifies that for all $1 \leq i \leq 2q$ such that $r_i = 0$ Alice had sent the correct $B_i(s)$, computes $c = E(b_1, b_2 \dots b_m)$ and $G_{\vec{R}}(s)$ and verifies that $e = c \oplus G_{\vec{R}}(s)$

As in the commitment with one bit, Bob can learn nothing about any of the b_i 's:

(Claim) for any two different sequences $D = b_1, b_2 \dots b_m$ and $D' = b'_1, b'_2 \dots b'_m$ that Bob selects, for any polynomial p , following the commit stage Bob cannot decide with probability greater than $\frac{1}{2} + \frac{1}{p(n)}$ to which sequence Alice has committed.

5.3 Complexity and Analysis

Still Alice could cheat and her only chance is if there exists a pair of seeds s_1 and s_2 that agree on all the indices that \vec{R} has a 0, and there exist two different sequences $b_1, b_2 \dots b_m$ and $b'_1, b'_2 \dots b'_m$ such that $G_{\vec{R}(s_1)} \oplus E(b_1, b_2 \dots b_m) = G_{\vec{R}(s_1)} \oplus E(b'_1, b'_2 \dots b'_m)$. We say in this case that s_1 and s_2 fool \vec{R} .

- Alice can reveal only one possible sequence of bits, except with probability less than 2^{-n}

(Claim) for any pair of seeds s_1 and s_2 , the probability that it fools \vec{R} is at most $(1 - \frac{\epsilon}{2})^q$, where the probability is taken over the choices of \vec{R} . The number of bits exchanged in the protocol is $O(q)$, and when amortized over m bits it is $O(\frac{q}{m})$ which is $O(1)$, since C is a good code.

- For $m > n$, the communication cost is $O(m)$.

We saw how a bit commitment scheme is constructed from pseudo-random generator and how bit commitment scheme to many bits can be implemented quite efficiently. Thus, many Zero-Knowledge protocols can be implemented under the assumption that one-way functions exist. In both protocols, Bob selects a random \vec{R} , where it has been argued that almost all \vec{R} 's are good. Therefore if there is a trusted party at some point in time, it can choose \vec{R} and the same \vec{R} will be used in all executions of the protocol. This scheme is statistically binding, meaning that even if Alice is computationally unbounded she cannot cheat with probability greater than 2^{-n} .

The concealing property follows from a standard reduction, if Bob can tell whether Alice committed to a zero or one, he can also distinguish the output of the pseudo-random generator G from true-random, which contradicts the cryptographic security of G .

6 Quantum Computing

6.1 Some definitions and observations

Quantum Computer is a device for computation that makes use of some quantum mechanical phenomena like superposition and entanglement, to perform operations on data. In contrast to a normal computer, whose memory is build up from bits, where each bit could be either 0 or 1, quantum computer stores the information in qubits. A single qubit can hold a one, a zero, or, crucially, a superposition of these, allowing for an infinite number of states. Qubit describe binary information in the form of two-state quantum systems, such as: two distinct polarization states of a photon; two energy levels of an atomic electron; or the two spin directions of an electron or atomic nucleus in a magnetic field. With two or more qubits, it becomes possible to consider quantum logical-”gate” operations in which a controlled interaction between qubits produces a (coherent) change in the state of one qubit that is contingent upon the state of another. These gate operations are the building blocks of a quantum computer.

6.2 Representation of qubit

The states a qubit may be measured in are known as basis states (or vectors). As is the tradition with any sort of quantum states, Dirac, or bra-ket notation is used to represent them. This means that the two computational basis states are conventionally written as $|0\rangle$ and $|1\rangle$ (pronounced: 'ket 0' and 'ket 1').

A pure qubit state is a linear superposition of those two states. This means that the qubit can be represented as a linear combination of $|0\rangle$ and $|1\rangle$:

$$\psi = \alpha |0\rangle + \beta |1\rangle$$

where α and β are probability amplitudes and can in general both be complex numbers.

When we measure this qubit in the standard basis, the probability of outcome $|0\rangle$ is $|\alpha|^2$ and the probability of outcome $|1\rangle$ is $|\beta|^2$. Because the absolute squares of the amplitudes equate to probabilities, it follows that α and β must be constrained by the equation

$$|\alpha|^2 + |\beta|^2 = 1$$

6.3 Quantum Computing and Commitment schemes

In 1994 Peter Shor, discovered a remarkable algorithm. It allowed a quantum computer to factor large integers quickly, namely factoring an integer N in $O((\log N)^3)$ time and $O(\log N)$ space, It solved both the factoring problem and the discrete log problem, where with classical algorithms, factoring becomes increasingly time-consuming as N grows large. For this reason a quantum computer with sufficiently many quantum bits could "break" RSA and many of the cryptographic systems used nowadays, which like RSA are based on the difficulty of factoring integers. These are used to protect secure Web pages, encrypted email, and many other types of data. Breaking these would have significant ramifications for electronic privacy and security. Its invention sparked an enormous interest in quantum computers, even outside the physics community.

Like many quantum computer algorithms, Shor's [23] algorithm is probabilistic. Furthermore, it gives the correct answer with constant bounded probability. A proposed answer can be easily verified by dividing the RSA key by the suspected factor and looking for a remainder. By running the algorithm multiple times a correct answer can be obtained with exponentially small error. In fact, there have been made many modifications to Shor's algorithm. For example, whereas, an order of twenty to thirty runs are required on a quantum computer in the case of Shor's original algorithm, and with some of the other modifications, in the case of the modification done by David McAnally an order of only four to eight runs on the quantum computer is required. In 2001, it was demonstrated by a group at IBM, which factored 15 into 3 and 5, on a working 7-qubit NMR computer.

The only way to increase the security of an algorithm like RSA would be to increase the key size and hope that an adversary does not have the resources to build and use a powerful enough quantum computer. It is likely that it will always be possible to build classical computers that have more bits than the number of qubits in the largest quantum computer. The same is valid for

the hash functions as well. It has been shown that on a classical computer one needs about $2^{\frac{n}{2}}$ operations to find a collision, on a quantum computer the number of operations is highly reduced: about $2^{\frac{n}{3}}$, where n is the key size. Still with $n > 240$ it is computationally infeasible to find a collision in practice. Today's computationally secure classical bit commitment schemes are based on the existence of one-way permutations (functions). However, the existence of such permutations is an open question. In 2000 P. Dumas, D. Mayers and L. Salvail suggested a perfectly concealing Quantum Bit Commitment from any one-way function, although unconditionally secure quantum bit commitment is impossible. This protocol is non-interactive and has communication complexity $O(n)$ qubits for n a security parameter. Still it had the problem finding candidates for families of quantum one-way permutations or functions [35].

This definite advantage of quantum computers over normal has been investigated for a few problems up to now: factoring, discrete logarithm. However, there is no proof that the advantage is real: an equally fast classical algorithm may still be discovered. There is one other problem where quantum computers have significant advantage. Consider a problem that has properties like this: The only way to solve it is to guess answers repeatedly and check them; there are n possible answers to check; every possible answer takes the same amount of time to check, and there are no clues about which answers might be better: generating possibilities randomly is just as good as checking them in some special order. A good example is trying to crack a password for an encrypted file, assuming that the password has maximum possible length. Very efficient quantum database search algorithm invented Lov Grover in 1996[24]. The quadratic speedup isn't as dramatic as the speedup for factoring, discrete logs, or physics simulations. However, the algorithm can be applied to a much wider variety of problems. Any problem that had to be solved by random, brute-force search could now have a quadratic speedup. Such *quantum database search* problems efficiently can be solved by Grover's algorithm. In this case the advantage compared to a classical computer and algorithm is provable. The algorithm takes $O(\sqrt{N})$ time for searching in unsorted database with N entries, yielding a polynomial time algorithm for NP-complete problems. A classical one would take in average $O(N)$ time for a search like this one. Despite other quantum algorithms could exponentially speedup some processes and Grover's algorithm provides quadratic speedup it is still the fastest possible quantum algorithm for searching an unsorted database and this quite significant with a big N . This ensures that quantum

computers are superior to classical computers for at least one problem. This algorithm like many other quantum computer algorithms is probabilistic. It gives the answer with a high probability to be the right one, yet we can obtain much higher probability for a right answer by running it several times. There are of course other quantum computer algorithms that compute the right answer with probability one, namely the deterministic algorithms, one example the Deutsch-Jozsa algorithm [25].

By searching a database one could find really good application for the Grover's algorithm as inverting a function. A good example is the collision-free hashing or finding two seeds that s_1 and s_2 , such that they fool \vec{R} in the example with the Bit Commitment scheme using Pseudo-Random Generator. Namely for a function $f(x)$, which can be executed and evaluated on a quantum computer, one could search a database of estimated and possible solutions and find another x such that it produces the same y with the given function y , or search in a set of functions one that could produce a particular value of y if x is from a set of desirable values. This not only solves the collisions problem, but could help a lot with solving another important problem: the NP-complete by performing an exhaustive search over a set of possible solutions.

6.4 Quantum Bit Commitment

It is therefore very exciting question whether there are secure bit commitment protocols under no conditions and restrictions about the computational resources of the Sender and/or the Receiver and still are binding and concealing. We will actually see that such unconditionally secure bit commitment could not be build.

Here I base some of the statements on the paper of Gilles Brassard, Claude Crepeau, Dominic Mayers, Louis Salvail - "The Security of Quantum Bit Commitment Schemes" [26], where they presented some of the basic ideas of quantum bit commitment and illustrate with the help of the Mayers's impossibility proof and some examples that all recent attempts to bypass it have been broken so far.

It has long been known that unconditionally secure bit commitment schemes cannot exist in the classical world. After the success of quantum key distribution, it was natural to hope that quantum mechanics might provide such an unconditionally secure scheme. A protocol for quantum bit commitment was proposed in 1993 and claimed to be provably secure [27], which

would also have allowed secure quantum oblivious transfer [28], another fundamental primitive in classical cryptography. Trouble began in October 1995 when Mayers found a subtle flaw in this protocol. He generalized this results showing that unconditionally secure commitment scheme is impossible and providing a theorem called the general impossibility theorem. The proof that Bob cannot cheat the protocol and learn information on the committed bit was correct, and so was the proof that it is not possible for Alice to simultaneously know how to open the commitment to show a 0 and how to open it to show a 1. However, it was possible for Alice, through the magic of quantum entanglement, to learn at her choice either how to open the commitment to show a 0 or how to open it to show a 1. Because of the destructive nature of quantum measurements, choosing either one of these alternatives destroyed her chances for learning how to open the commitment the other way, a situation that has no classical analogue. Though Mayers explained his discovery to several researchers interested in quantum information processing [30], his result was not made public until after Lo and Chau discovered independently a similar technique [29]. This prompted several attempts at fixing the protocol, but every time ad hoc successful attacks soon followed. This cat-and-mouse game went on until Mayers proved that all these attempts had been in vain: unconditionally secure quantum bit commitment schemes are impossible [31]. Despite all odds, various kinds of ideas continued to be proposed in the hope they would not fall prey to Mayers' result [32, 33]. Perhaps the most interesting approach consisted in using various types of short-lived classical bit commitment schemes to force the cheater to perform measurements at specified points in the execution of the protocol, which indeed would fix the problem. Unfortunately, this strategy was doomed as measurements can always be postponed until cheating becomes possible. This is not surprising because it has been realized that these apparently promising ideas were also ruled out by Mayers' general impossibility theorem [34]. Assume that you are given an unknown quantum bit and you are expected to measure it either in the computational basis, $|0\rangle$ versus $|1\rangle$, or the diagonal basis, $(|0\rangle + |1\rangle) = \frac{p}{\sqrt{2}}$ versus $(|0\rangle - |1\rangle) = \frac{p}{\sqrt{2}}$. One could cheat the protocol if only you could postpone the choice of basis (and therefore the measurement) until after subsequent classical information becomes available. To prevent this, you are asked to use a classical (perhaps multi-party) commitment scheme to commit to your choice of basis (computational or diagonal) as well as to the result of your measurement. Then, either you

are immediately challenged to open those commitments to "prove" that you had measured, in which case this quantum bit is not used any further, or you are allowed to keep secret your choice of basis and measurement result for later use. The hope was that this approach could turn a classical bit commitment scheme that is reasonably secure for a very short time into a quantum bit commitment scheme that is permanently secure. Unfortunately, it turns out that you can always postpone measuring the quantum bit and offer fake commitments even if you are unable to break the underlying commitment scheme in the time that elapses between the commitment and the challenge: if a challenge comes, you can open your "commitments" in a way consistent with what you would have committed to had you measured the quantum bit, and if the challenge does not come, you can keep the quantum bit intact for later measurement in the basis of your choice.

In particular, secure schemes would follow if we could force participants to perform measurements at specified points in the execution of the protocol. It has been suggested to use short-lived classical bit commitment schemes for this purpose. Unfortunately, this strategy was shattered as measurements can always be postponed in an undetectable way until cheating becomes possible.

Nevertheless, these attempts contributed to enhance our understanding of what is going on with quantum bit commitment schemes and other quantum cryptographic protocols.

7 Conclusions

Actually we saw that quantum algorithms endanger classical cryptographic protocols based on the hardness of factoring numbers, the discrete logarithm problem and the collision problem all of these used as building blocks for constructing up to now reliable commitment schemes. I showed some examples for protocols which with no doubt are vulnerable from quantum computer algorithms like Grover's and Shor's. There are possible ways discussed for breaking classical schemes using quantum computer and claims and proofs that no unconditionally secure classical bit commitment scheme exists [36]. Other protocols using hash functions, between classical participants and a quantum adversary could still be practically secure with high enough key size.

Unfortunately we cannot gain this advantage over the Quantum Bit Commit-

ment schemes. Mayers had presented a proof of the Impossibility Theorem, which claims that, no quantum bit commitment scheme can be unconditionally secure. The proof itself is divided into several cases, the first is when given a quantum transcript, there is an equal chance commits to 0 or to 1. The other one is when the quantum transcript is not equally commits to 0 and 1. Although it is widely believed that computationally secure commitment schemes exist, thus a computationally secure cryptographic protocols could be achieved.

References

- [1] **R. Impagliazzo and M. Yung**, Direct Zero-knowledge Protocols *Crypto 87*, pp. 40-51
- [2] **Giles Brassard, David Chaum, and Claude Crepeau**, Minimum Disclosure Proofs of Knowledge *Journal of Computer and System Sciences*, vol. 37, pp. 156-189, 1988
- [3] **Moni Naor**, Commitment Using Pseudo-randomness *Journal of Cryptology* 4: 2 pp. 151-158, 1991
- [4] **Oded Goldreich and Hugo Krawczyk**, On the Composition of Zero-Knowledge Proof Systems *SIAM Journal on Computing*, 25: 1, pp. 169-192, 1996
- [5] **Andrew Chi-Chih Yao**, Protocols for Secure Computations (Extended Abstract) *FOCS 1982*: 160-164
- [6] **S. Halevi and S. Micali**, Practical and provably-secure commitment schemes from collision-free hashing *In N. Koblitz, editor, Advances in Cryptology - CRYPTO '96, volume 1109 of Lecture Notes in Computer Science, pages 201-215. Springer-Verlag*, 18-22 Aug. 1996
- [7] **O. Goldreich and A.Kahan**, How to Construct Constant-Round Zero-Knowledge Proofs for NP *Journal of Cryptology*, 9(3):167-190, 1996
- [8] **Moti Yung and Russell Impagliazzo**, Direct minimum-knowledge computations *In C. Pomerance, editor, Proc. Crypto '87 Lecture Notes in computer science vol. 293, Pages 84-96.*, 1988

- [9] **M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung**, Perfect zero-knowledge arguments for NP can be based on general complexity assumptions *Proceedings of CRYPTO92, Santa-Barbara, CA*, August 17-20 1992
- [10] **Manuel Blum**, Coin Flipping by Telephone *Proceedings of CRYPTO 1981, pp. 11-15, 1981, reprinted in SIGACT News vol. 15, pp. 23-27, 1983*
- [11] **S. Goldwasser, S. Micali, R. Rivest**, digital signature scheme secure against adaptive chosen-message attacks *SIAM J. Computing, 17(2):281-308*, april 1988
- [12] **T. P. Pedersen**, Interactive and Information Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor *Proc. Crypto '91, Lecture Note in Computer Science, volume 576, Springer-Verlag, 1992. Pages 129-140*, 1991
- [13] **D. Chaum, E. van Heijst and B. Pfitzmann**, Cryptographically Strong Undeniable Signatures, Unconditionally Secure for Signer. In J. Feigenbaum, editor *Proc. Crypto '91, Lecture Note in Computer Science, volume 576, Springer-Verlag, pages 470-484*, 1992
- [14] **C.H. Bennett and G. Brassard** , Quantum Cryptography: Public key Distribution and Coin Tossing *In Proc. Of IEEE International Conf. on Computers, Systems, and Signal Processing IEEE, pages 175-179*, 1984
- [15] **G. Brassard and C. Crépeau**, Quantum bit commitment and coin tossing protocols. In A.J. Menezes and S.A Vanstone, editors *Proc. Crypto '90, Lecture Notes in Computer Science, vol 537. Springer-Verlag, pages 49-61*, 1991
- [16] **G. Brassard and C. Crépeau, R. Jozsa and D. Langois**, A Quantum Bit Commitment Scheme Provably Unbreakable by Both Parties *In Proc. 34th IEEE Symp. On Foundations of Computer Science, IEEE, 1993*
- [17] **A. C. Yao**, Theory and Applications of Trapdoor Functions *Proc. 23rd Symposium on Foundation of Computer Science, pages 80-91*, 1982

- [18] **I. Impagliazzo, L. Levin and M. Luby**, Pseudo-random generator from one-way functions *21st Symposium on Theory of Computing*, pp 12-24, 1989
- [19] **J. Hastad**, Pseudo-random generators under uniform assumptions *Proc. 22nd Symposium on Theory of Computing*, to appear, 1987
- [20] **I. Impagliazzo and M. Luby**, One-way functions are essential to computational based cryptography *Proc 21st Symposium on Theory of Computing*, pp. 230-235, 1989
- [21] **J. Kilian, S. Micali and R. Ostrovsky**, Simple non-interactive zero-knowledge proofs *Crypto '89*
- [22] **J. Justen**, A class of constructive asymptotically good algebraic codes *IEEE Transactions on Information Theory* 18, pp. 652-656, 1972
- [23] Shor's Factoring Algorithm *from Lecture 9 of Berkeley CS 294-2*, 7 page postscript document, 4 Oct 2004
- [24] **L.K. Grover**, A fast quantum mechanical algorithm for database search *Proceedings, 28th Annual ACM Symposium on the Theory of Computing* p. 212, May 1996
- [25] **David Deutsch and Richard Jozsa**, Rapid solutions of problems by quantum computation *Proceedings of the Royal Society of London A* 439: 553, 1992
- [26] **Gilles Brassard, Claude Crepeau, Dominic Mayers, Louis Salvail**, The Security of Quantum Bit Commitment Schemes
- [27] **Brassard, Gilles, Claude Crépeau, Richard Jozsa and Daniel Langlois**, A quantum bit commitment scheme provably unbreakable by both parties *Proceedings of 34th Annual IEEE Symposium on Foundations of Computer Science*, pp. 362 - 371, 1993
- [28] **Bennett, Charles H., Gilles Brassard, Claude Crépeau and Marie-Hélène Skubiszewska**, Practical quantum oblivious transfer in *Cryptology: Proceedings of Crypto '91*, pp. 351 - 366, 1991

- [29] **Lo, Hoi-Kwong and H. F. Chau**, Is quantum bit commitment really possible? *Physical Review Letters*, Vol. 78, no. 17, 28 April 1997, pp. 3410 - 3413; first appeared as *quant-ph/9603004* in the LANL e-Print archive, 1996
- [30] **Mayers, Dominic**, The trouble with quantum bit commitment available as *quant-ph/9603015* in the LANL e-Print first discussed at a Workshop on Quantum Information Theory held in Montreal, October 1995
- [31] **Mayers, Dominic**, Unconditionally secure quantum bit commitment is impossible *Physical Review Letters*, Vol. 78, no. 17, pp. 3414 - 3417, 28 April 1997
- [32] **Brassard, Gilles and Claude Crépeau**, Cryptology - 25 years of quantum cryptography *Sigact News*, Vol. 27, no. 3, pp. 13 - 24, 1996
- [33] **Crépeau, Claude**, What is going on with quantum bit commitment? *Proceedings of Pragocrypt '96, Prague, Czech Technical University Publishing House*, pp. 193 - 203, October 1996
- [34] **Brassard, Gilles, Claude Crépeau, Dominic Mayers and Louis Salvail**, Defeating classical bit commitments with a quantum computer Available as *quant-ph/9806031* in the LANL e-Print archive at URL <http://xxx.lanl.gov/archive/quant-ph>, June 1998
- [35] **Paul Dumais, Dominic Mayers and Louis Salvail**, Perfectly Concealing Quantum Bit Commitment from Any Quantum One-Way Permutation *Lecture Notes in Computer Science*, 2000
- [36] **Orr Dunkelman**, Unconditionally Secure Quantum Bit Commitment Schemes , 2000