CS191 – Fall 2014
Lectures 12 & 13: Introduction to quantum key distribution

Mohan Sarovar
(Dated: October 11, 2014)

These two lectures are an introduction to quantum key distribution (QKD).

I. CLASSICAL CRYPTOGRAPHY

We begin with a brief review of classical cryptography, especially the parts that are most relevant for QKD. Firstly, it is imperative to introduce the key players in most cryptographic scenarios: Alice and Bob are the parties that want to securely communicate and Eve is the malicious eavesdropper. Communication is achieved by the encoding of a secret message into what is called cypher text by Alice and then transmission of the this cypher text over the air (where it is readable by anyone) to Bob, who them decodes the cypher text into the original message. This communication is called "secure" if despite having full access to the cypher text, Eve cannot decode the message. Whether Eve can decode the message or not depends on what resources we give her, and this point raises the first important definitions: a communication protocol is called *computationally secure* if Eve requires exponential computational resources [1] to break the cypher text. In contrast, an *information-theoretic secure* protocol does not make any assumptions on Eve's computational power. It relies on being able to prove that the amount of information the cypher text contains about the message is not enough to decode the message, *regardless* of Eve's computing power. Obviously information-theoretic security is a stronger notion and is therefore harder to achieve.

A final note about computationally secure schemes: they do not usually guarantee *future security*, meaning that an eavesdropper could simply store the cypher text until he/she has enough computational power to crack it. This is a concern in certain application where long-term security desired.

A. Symmetric key cryptography

The simplest way to securely communicate is to use *symmetric keys*, where Alice and Bob have the same key, with which they encode and decode the message. For example, this key could be a large bit string that they hash the message with:

$$\begin{aligned} \text{Encode}: &\quad Cypher = Key \oplus Message \\ \text{Decode}: &\quad Message = Key \oplus Cypher \end{aligned}$$

These relationships will hold as long as the same Key is used to encode and decode. However, this scheme is perfect only when Alice and Bob to share a secret Key that is: (i) truly random, (ii) as long as the message they want to communicate (in number of bits), (iii) not re-used to encode multiple message (if it is, Eve can use frequency analysis to infer the key). If all three of these conditions are met, then the above symmetric key scheme is information-theoretic secure (if all three conditions are satisfied the Key is what is called a *one-time pad*). However, the catch is that sharing such keys is obviously a very resource intensive task. There have however been many imperfect implementations of this scheme in history, for example, such a scheme was implemented by trading codebooks during World War 2.

In modern times, the most prominent symmetric key scheme is the *Advanced encryption standard* (AES). This scheme is not as simple as the one-time pad shown above, but the lack of encoding/decoding simplicity is balanced by ease in distribution of the shared key. It should be noted that these practical symmetric key implementations that are not ideal one-time pads have not been shown to be information-theoretic secure, but only computationally secure (for example, there are no known "easy" cryptanalysis attacks on AES).

---

[1] Exponential in something that is controllable by Alice and Bob, typically the size of the key used to encode and decode the message.

B. Public key cryptography

Due to the difficulty of distributing symmetric keys, much of modern secure communication is carried out through public-key cryptography. In this case, each user has two keys, a public key and a private key. As the names suggest, the public key is advertised to everyone and the private key is kept secret. When Alice wants to securely communicate with Bob, she encodes her message using his public key, and Bob decodes it using his private key:

$$\begin{aligned} \text{Encode}: &\quad Cypher = PublicKey_{\text{Bob}}(Message) \\ \text{Decode}: &\quad Message = PrivateKey_{\text{Bob}}(Cypher) \end{aligned}$$

Implicit in this scheme is that the private key is the inverse of the public key, and this is in fact the case. But these two keys are related to each other by *one way functions*, which are hard to invert. That is, given the public key, it is computationally difficult to figure out what the private key is. This means that anyone can encode a message for Bob, but only Bob can decode such messages. Thus the security of such public key schemes rely on the computational difficulty of inverting one way functions.

An example of public key cryptography is the RSA protocol that is widely used on the Internet. A sketch of the key components of this protocol is as so:

1. To generate his public and private keys, Bob first chooses two large prime numbers p and q, and computes their product: $n = pq$.
2. Bob then chooses a number $3 \leq e < (p-1)(q-1)$ such that e and $(p-1)(q-1)$ have no common factor; *i.e.*, $\nexists k \in \mathbb{Z}, \text{s.t. } \frac{e}{k} \in \mathbb{Z}$ and $\frac{n}{k} \in \mathbb{Z}$.
3. Bob then calculate d such that $ed = 1(\mod (p-1)(q-1))$.
4. Finally Bob publishes his public key pair (e, n) and keeps his private key pair (d, n) secret.
5. When Alice wants to communicate with Bob, she encodes her message m as: $c = m^e \mod n$, and sends this message over a channel.
6. When Bob receives Alice's cypher text, he decodes by performing: $c^d \mod n = m^{ed} \mod n = m$, where to get the last equality we have used a property of modular exponentiation that it's not important to explain.

The security of the RSA scheme relies on the difficulty of finding d even if one knows the public key pair (e, n). And this in turn, relies on the difficulty of factoring n (which is known) into its prime factors p and q (since if Eve were able to do this, then she could compute d in exactly the same way Bob did). Therefore, RSA is obviously a computationally-secure scheme. Quantum computers can in fact factor numbers much more efficiently than classical computers, and therefore RSA is not secure against quantum computer attacks. But more on this in later lectures.

Exercise: Prove that the decoded string in RSA is indeed the original message. That is, prove:

$$c^d \mod n = m \tag{1}$$

See Appendix 4 and Appendix 5 of Nielsen & Chuang for hints on how to do this.

II. QUANTUM KEY DISTRIBUTION

Quantum key distribution (QKD) is a way to establish a symmetric key that can be used as a one-time pad for secure communication. In fact, maybe a better name for QKD is symmetric key expansion since as you will see it relies on Alice and Bob having a small secret key to begin with for authentication purposes. Then it grows or expands this secret key.

QKD relies on two fundamental properties of quantum systems: (i) no cloning: the notion that a general quantum state cannot be copied deterministically, (ii) complementarity of observables: the notion that not all observables of a quantum system can be simultaneously determined without error.

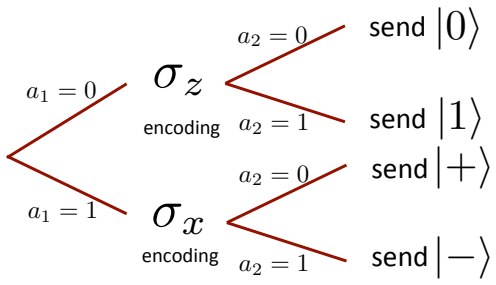


FIG. 1: Decision tree that describes the states Alice sends depending on her random bits a_1 and a_2.

A. BB84 protocol

The first complete QKD protocol was formulated in a paper by Bennett and Brassard in 1984, and thus this protocol goes by the name BB84. It assumes that Alice can send single photons polarized along horizontal/vertical and diagonal/antidiagonal axes and that Bob can detect these single photons with high efficiency. In the following we will sketch out the steps of the protocol. But since you have dealt more with spin systems in this class than with photons I will present the protocol in terms of Alice and Bob exchanging spins. But remember that in practice QKD is done with photons as quantum information carriers.

1. *Quantum phase of protocol*

The first phase of the BB84 protocol proceeds by many rounds of qubit transmission from Alice to Bob. Each round has 4 steps:

1. Alice generates two independent uniformly random bits (a_1, a_2) and Bob generates one uniformly random bit b.
2. Alice sends a qubit prepared in the eigenbasis of σ_z or σ_x according to the value of a_1 and in a state according to the value of a_2. See figure 1.
3. Bob, depending on his random bit (b), measures in either the σ_z basis (projects onto $|0\rangle\langle 0|, |1\rangle\langle 1|$) or the σ_x basis (projects onto $|+\rangle\langle +|, |-\rangle\langle -|$). The result of the measurement is a bit value.
4. *Sifting:* Over an authenticated public channel, Bob discloses the value of b (the basis he measured in). Alice instructs Bob over the same channel if a1 and b agree, and if so Alice keeps a_2 and Bob keeps his measured value. If $a_1 \neq b$ then they discards all the bits generated in this round.

After performing N rounds of this phase of the protocol Alice and Bob share a partially correlated string of bits. The length of this string is $N_{\text{sift}} = N/2$ on average because half of the rounds are discarded by the sifting stage.

> *Example 1* We will see an example execution of phase one of BB84 in class on October 8th. You can also see an illustrative example on slide 16 of this set of slides by Richard Hughes:
> `http://cnls.lanl.gov/~chertkov/EC_Talks/Hughes.pdf`

2. *Classical (post-processing) phase of protocol*

The second phase of any QKD protocol is to take the partially correlated string established in phase one and turn it into a fully correlated *secret* string of bits if possible. The first step in doing this is to estimate how much error (disagreement) there is between Alice's and Bob's strings. To do this, they announce a small fraction of their strings and compare the values over the public channel. Let us denote the number of bits revealed for this error estimation as N_{est}. If the error rate is above a certain threshold (usually around 11%) then they abort (and maybe redo the quantum transmission steps) because it is unlikely that in this case they can *distill* a secret shared key if there are so

many errors. In other words, we assume all the errors are due to Eve's intervention and given this large rate of error, we conclude Eve has too much information about the strings Alice and Bob share.

We mentioned above that the effect of Eve's interception of the qubits being communicated between Alice and Bob is to introduce disagreement between the bit strings shared by Alice and Bob after sifting. To see why this is, consider what happens if Alice chooses to encode in the σ_z basis and sends the state $|0\rangle$ and Bob also chooses to measure in the σ_z basis. But during the transmission Eve captures the photon/qubit and measures in the σ_x basis to get $|+\rangle$ (this happens with probability 1/2). She then replaces the qubit she measured with another one prepared in the $|+\rangle$ state and sends this on to Bob. This is called an *intercept-resend attack* and is one of many attacks Eve could perform. When Bob measures the $|+\rangle$ state in the σ_z basis he gets $|0\rangle$ with probability 1/2 and $|1\rangle$ with probability 1/2. In the sifting stage Alice and Bob keep this round since the encoding and measurement bases are the same. But if this bit in included in the portion that is used for estimating errors, with probability 1/2 Alice and Bob will notice that there is an error. They are obliged to attribute that error to Eve's intervention.

Let us see what kind of error rate Alice and Bob will see on average if Eve intercepted every qubit and measured it before resending it to Bob. Let's denote the basis Alice prepares in, the basis Bob measures in and the basis Even measures in by a three-tuple (a, b, e). For example, (Z, Z, X) is the case where Alice prepares in the σ_z basis, Bob measures in the σ_z basis and Eve measures in the σ_x basis. There are 8 possible tuples of this kind each corresponding to an experimental configuration. But only half of these 8 will proceed past the sifting stage, and these are: $(Z, Z, Z), (Z, Z, X), (X, X, Z), (X, X, X)$. Out of these four, (Z, Z, Z) and (X, X, X) will result in no error between the bit value Alice and Bob get because Eve measures in the same basis and Alice and Bob (and assuming that the only source of error is Eve's intervention). For the other two configurations, with probability 1/2 the value Bob measures after Eves disturbance of the state will lead to an error. Therefore, if Eve intercepts and resends *every* qubit Alice and Bob will notice a $0.5 \times 0.5 = 0.25$ rate of errors. If the error rate is this high, Alice and Bob should not proceed further since they must assume that Eve has too much information about their bit strings.

In practice, the threshold beyond which QKD cannot proceed is lower, around 11% because there are other modes of attack that we haven't analyzed. We will mention the broad classes of possible attacks below.

The error estimate is also used by Alice and Bob to estimate the information Eve has about their strings (in number of bits), I_{Eve}. The exact relation between the error rate and the I_{Eve} is complicated and depends strongly on type of attacks Eve is assumed to make. Let's consider the simplest example, where Eve performs the intercept-resend attack and Alice and Bob estimate the error rate as 25%. As we saw above this error rate is consistent with Eve intercepting every qubit, and if this is the case she knows on average 75% of Bob's bits. Hence $I_{\text{Eve}} = 0.75(N_{\text{sift}} - N_{\text{est}})$ bits. In more general situations (general attacks by Eve and lower error rate) we can calculate I_{Eve} as a function of the error rate, even though the relationship between the two is more complicated.

Exercise: Confirm that $I_{\text{Eve}} = 0.75(N_{\text{sift}} - N_{\text{est}})$ bits in the above example.

Let us now complete the description of BB84 if the error rate is estimated to be less than the threshold. In this case, Alice and Bob perform interactive error correction on the remaining bits. This step is often called *reconciliation* since Alice and Bob are attempting to reconcile their partially correlated bit strings. This reconciliation can be performed by a variety of error correction codes. One of the original schemes is called the *Cascade* protocol and it proceeds by exchanging parities of subblocks of the shared bit string. These days, the are more efficient schemes that rely on LDPC (low density parity check) codes. The efficiency of a reconciliation protocol is calculated in terms of the number of bits that must be revealed (communicated between Alice and Bob) compared to the length of key that is reconciled. Each bit that is revealed can be used by Eve to reconstruct the key that Alice and Bob are trying to establish. So we want reconciliation protocols that are as efficient as possible. Fortunately Cascade, and especially the newer LDPC protocols, are very efficient (they reveal $\sim \log(n)$ bits where n is the reconciled key size). Let us denote the number of bits revealed during reconciliation as I_{EC}.

Example 2 We will demonstrate the Cascade protocol with an example in class on October 10th. It will be based on the example on figure 1 of this paper:
`http://dl.acm.org/citation.cfm?id=2179363`

After error correction Alice and Bob share a perfectly correlated bit string. But it is not secret because (i) Eve could have intercepted some of the communication in phase one thereby learning I_{Eve} bits, and (ii) information about some of the bits is revealed during reconciliation. Therefore the final step in establishing a secret shared key is to perform *privacy amplification*, which is essentially hashing portions of the bit string in order to make knowledge of any small potion of the bit string insufficient to reconstruct the hashed values.

Let us account for the number of bits Eve may have access to. We begin with $N_{\text{sift}} = \frac{N}{2}$ bits after sifting. Then we reveal N_{est} bits for error estimation, and learn that Eve could know I_{Eve} bits of the sifted key. Finally, we reveal I_{EC} during the reconciliation process. Therefore out of the $N_{\text{sift}} + N_{\text{est}}$ bit we have remaining after sifting and error

estimation, Eve may know $I_{\rm Eve} + I_{\rm EC}$ bits. The privacy amplification must compress the $N_{\rm sift} + N_{\rm est}$ string of bits such that knowledge of any $I_{\rm Eve} + I_{\rm EC}$ bits in the string does not reveal the compressed string. This is usually done through *universal hash functions*.

Example 3 Suppose Alice and Bob share the bits $X_1, X_2, ...X_K$, and they estimate that Eve knows 1 bit. Then a way to hash these bits to get a obtain a secret key is to compute: $X_1 \oplus X_2, X_3 \oplus X_4, ...X_{K-1} \oplus X_K$. Thus, no matter which bit Eve knows there is no way for her to know these hashed values.

Note that we sacrificed a lot to perform privacy amplification in this example. We compressed the bit string to half its length even though Eve only knew 1 bit. Fortunately universal hash functions are much more efficient at privacy amplification.

Example 4 You will apply universal hashing to illustrate privacy amplification in next week's homework.

So we see that the classical post processing (reconciliation and privacy amplification) can reduce the length of the shared string, but after these steps Alice and Bob can be sure that the bit values they share are truly random and also secret. This is the symmetric key that they can use as a one-time pad for future secure communication.

III. TYPES OF ATTACKS

In section II A 2 we analyzed the effect of Eve intercepting each qubit, measuring it in a random basis and resending the result. This is only one type of attack that Eve could implement. The major difficulty in proving the security of any QKD protocol is showing that the protocol is secure against *any* possible attack.

In considering Eve's attacks it is traditional to divide them into three classes: individual, collective and coherent attacks. It is useful to think about Eve's attacks as consisting of preparation of a set of ancilla, which then interact with the qubits in the channel before Eve measures the ancilla. We have seen that any measurement can be modeled this way, and this model makes it clear that the qubits continue onto Bob after Eve's intervention. In terms of this model the three classes of attacks can be described as:

1. *Individual attacks.* These are attacks where Eve interacts with each qubit in the channel separately and independently. Each of Eve's ancilla systems are prepared independently and each interact with only one qubit in the channel before being measured independently. The intercept-resend attack considered above is an example of this class of attack. See figure 2(a).

2. *Collective attacks.* These are attacks where Eve prepares independent ancilla, interacts each qubit independently, but can perform a joint measurement on all the ancilla. Such joint measurements can yield more information than individual measurements. See figure 2(b).

3. *Coherent attacks.* This is the most powerful class of attacks and includes the most general thing that Eve could do. In collective attacks Eve can prepare an arbitrary joint (entangled) state of the ancillas, which then interact with the qubits in the channel before being measured jointly. See figure 2(c).

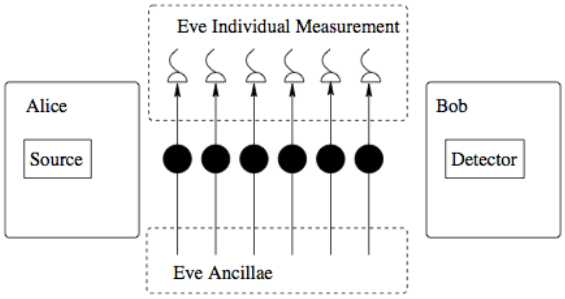
It should be said that individual attacks are really the only ones that are feasible with current technology. To perform collective and coherent attacks Eve would require long-lived quantum memories and the ability to perform coherent quantum operations with high fidelity; *i.e.*, something like a quantum computer. However, a QKD scheme is generally not considered information-theoretic secure until it is proven to be secure against coherent attacks. The BB84 protocol is one of the protocols that has been proven to be secure against this most general class of attacks.

We will not say much about how to construct security proofs for QKD protocols. This is a vast and complex subject since there are in fact many ways to demonstrate security. However, most proofs focus on the first phase of QKD since if we know that this stage can be made secure, and the error rate is below threshold, then we know the conditions under which the remaining post processing stages maintain security.

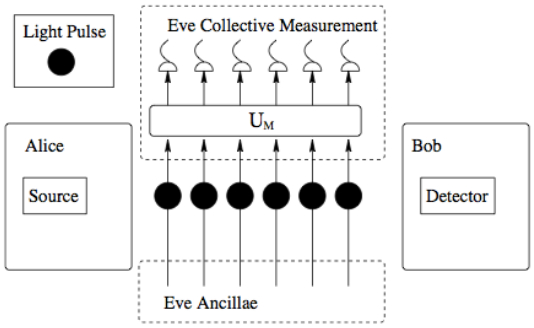
IV. ENTANGLEMENT-BASED QKD PROTOCOLS

The BB84 protocol proceeds by one party sending quantum states to the other. There is another way to implement QKD that is completely equivalent but practically very different. These implementations are referred to variously as entanglement-based protocols, Ekert protocols, or EPR protocols. In each round of such protocols a trusted central source sends one qubit in an two-qubit maximally entangled state (*e.g.*, $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$) to both Alice and Bob.

(a) Individual attacks



(b) Collective attacks



(c) Coherent attacks

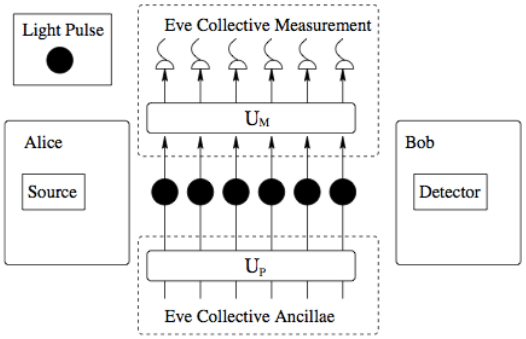


FIG. 2: The classes of attacks possible by Eve. These figures are from the Ph.D. thesis of Raul Garcia-Patron (Université Libre de Bruxelles, 2008).

Then Alice and Bob each choose a basis to measure in (σ_z or σ_x) and record their measurement result. At the end of N such rounds they perform sifting as before by announcing their measurement bases and only keeping the subset of rounds where these bases agree. One can think of this protocol as both Alice and Bob just choosing one random bit each, a and b (as opposed to Alice choosing two random bits), on which they base their measurement basis on. Alice's second random bit (which was a_2 above) is generated as the measurement result on her qubit.

Such entanglement-based protocols are logically the same as point-to-point transmission protocols like BB84 but have some benefits for constructing security proofs. Any security proof constructed for one is valid for the other type of protocol.

See Chapter 4 of Benenti, Casati, & Strini for more details on the Ekert91 entanglement-based protocol.

V. PRACTICAL ISSUES

QKD is an informationally theoretic secure way to distribute keys in the ideal case. That is, if the implementation of BB84 is exactly as analyzed in theory then Alice and Bob have information theoretic security on their shared key. However, in reality there are always practical imperfections and possible loopholes. For example, if Alice's basis selection is associated with a loud noise then Eve can just listen in to get information on the basis choice and hence siphon the key (by measuring in the correct basis each time) without the knowledge of Alice and Bob. The field of

finding practical weaknesses in QKD implementations is called *quantum hacking* (see `http://www.vad1.com/lab/`). One must be extremely careful when implementing QKD to avoid such implementation errors or side-channels through which the security can be compromised.

Another class of practical difficulties in implementing BB84 is the technological challenges associated with creating and detecting single photons. Both of these tasks are difficult, however, there is a lot of research in developing single photon sources and single photon detectors in order to fill this need. The lack of true single photon detectors does not compromise QKD security, but can reduce the secure key rate (especially if the detectors have dark current, inefficiencies and other non-idealities). The absence of true single photon sources does impact QKD security because if Alice actually sends two photons in the same state instead of one, Eve can syphon off one of them (this is called the photon number splitting attack). Fortunately there is an easy fix to this (at the expense of a hit in efficiency), which is called the *decoy state protocol*. Under this modification to BB84, Alice randomly replaces one of her data photons with a multi-photon "decoy" state. Bob measures this state as usual, and after measurement by Bob (perhaps in the sifting stage) Alice reveals which of her pulses were the decoy states. If Eve attempts a photon number splitting attack on any of the decoy pulses Alice and Bob can infer this by looking at Bob's measurement results for the decoy pulses. And Eve cannot try to avoid just these pulses because their identity is only revealed after the quantum phase when all quantum signals have been transmitted and detected. This simple modification can be shown to provide security to BB84 even when Alice's source is not a true single photon source.

VI. REFERENCES AND FURTHER READING

1. Chapter 12.6 of Nielsen & Chuang.
2. Chapter 4 of Benenti, Casati, & Strini, Volume 1.
3. The book *Quantum cryptography and secret key distillation* by Gilles Van Assche (Cambridge University Press, 2012) is an in-depth treatment of many of the important aspects of QKD.
4. There are a number of review articles on quantum cryptography:
   - N. Gisin et al., *Quantum Cryptography*, Reviews of Modern Physics 74, 145 (2002).
   - M. Dusek et al., *Quantum Cryptography*, Progress in Optics 49, 381 (2006).
   - H-K. Lo, Y. Zhao, *Quantum Cryptography*, `http://arxiv.org/pdf/0803.2507.pdf`.