Chapter 11

(Quantum) Key Distribution

Cryptography is a truly groundbreaking discovery of our time. Being able to generate a secret in plain sight is mind-boggling.

11.1 Key Exchange

Definition 11.1 (Key Exchange). *Key exchange is a technique to establish a common key between two or more parties (often Alice and Bob), without prior knowledge, so that no eavesdropper (Eve) can obtain a copy of the key.*

Remarks:

- Key exchange is also known as key generation or key establishment. If quantum physics is involved, the problem is often called (quantum) key distribution.
- When two parties want to communicate secretly, they previously need to exchange a secret key. Historically, this required meeting in private (which can be inconvenient or impossible) or sending a (hopefully trustworthy) courier between the communicating parties.
- Is it possible to generate a key in plain sight?

11.2 Correlated Randomness

Definition 11.2 (Correlated Randomness). *We consider the satellite scenario depicted in Figure 11.3. Alice and Bob have access to a private noiseless channel for communication. Also, they receive random bits $R = R_1, R_2, \ldots$ emitted by a satellite. However, the bits Alice and Bob receive from the satellite are erroneous. Each bit is flipped with probability α for Alice and β for Bob, respectively. Therefore, Alice receives string $A = A_1, A_2, \ldots$ and Bob $B = B_1, B_2, \ldots$. Alice and Bob face an attacker Eve that can eavesdrop all communication on their private channel. Eve additionally receives satellite bits $E = E_1, E_2, \ldots$ with a bit error rate of ϵ. We assume $0 < \alpha, \beta, \epsilon < \frac{1}{2}$.*

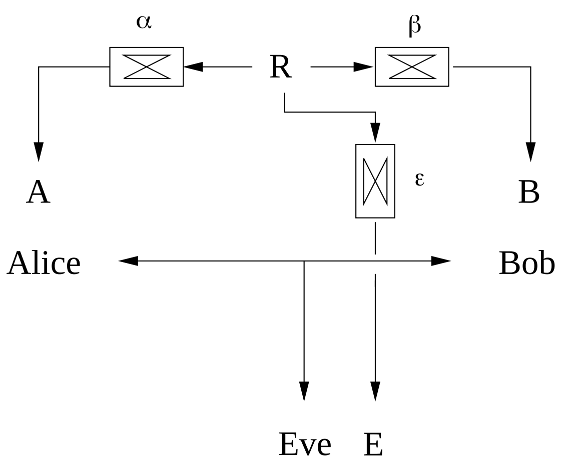

Figure 11.3: Scenario of communication channels for key exchange with correlated randomness.

Theorem 11.4. *In the satellite scenario, Alice and Bob can always generate a key, even if $\epsilon < \alpha, \beta$.*

Remarks:

- Lets briefly think about $\epsilon > \beta$ (or symmetrically $\epsilon > \alpha$). Then, Bob receives better satellite information than Eve, which allows Bob to have more information about Alice's bitstring A than Eve.
- Alice may be able to reveal A to Bob while only revealing parts of A to Eve. However, Eve will learn certain bits of the shared key.
- So first we want to get into this advantageous situation where Alice and Bob know more about each other's strings than Eve.

11.3 Advantage Distillation

Algorithm 11.5 Advantage Distillation

Input: Satellite string R, with errors: A, B.
Result: Strings $\hat{A}$, $\hat{B}$, with $\hat{A} \approx \hat{B}$.

```
1: repeat
2:   For every two consecutive bits of $R$, Alice and Bob compute and exchange
     the parities $p_A$ and $p_B$, respectively
3:   if $p_A \neq p_B$ then
4:     Alice and Bob discard the two consecutive bits
5:   else
6:     Alice and Bob both keep the first bit, hoping to have the same bit
7:   end if
8: until Alice and Bob have collected sufficiently many bits
```

Iteration	$\mathbf{P}[\hat{A}_i \neq \hat{B}_i]$	$\mathbf{P}[\hat{A}_i \neq \hat{E}_i]$
0	0.18	0.14
1	0.045	0.081
2	0.002	0.062
3	≈ 0	0.061

(a) $\alpha = \beta = 0.1, \epsilon = 0.05$

Iteration	$\mathbf{P}[\hat{A}_i \neq \hat{B}_i]$	$\mathbf{P}[\hat{A}_i \neq \hat{E}_i]$
0	0.48	0.4
1	0.46	0.396
2	0.421	0.388
3	0.346	0.375
4	0.217	0.350
5	0.072	0.320
6	0.006	0.305
7	≈ 0	0.3

(b) $\alpha = \beta = 0.4, \epsilon = 0.001$

Figure 11.7: Probabilities that the bits of Alice disagree with Bob's or Eve's after several iterations of advantage distillation. The error between Alice and Bob drops much faster, allowing them to beat Eve.

Lemma 11.6. *Algorithm 11.5 reduces the probability that Alice and Bob disagree on a bit.*

Proof. Instead of a formal proof, we argue along an example: Let $\alpha = \beta = 0.1$ and $\epsilon = 0.05$. Then each bit of Alice and Bob has a probability of $\alpha(1-\beta) + \beta(1-\alpha) = 18\%$ to be different, whereas only 14% of the bits between Alice and Eve differ. As a case analysis reveals, this already reverses after executing Algorithm 11.5. After the algorithm Alice and Eve disagree with about 8%, but Alice and Bob only disagree with about 4%. Alice and Bob can improve much more than Eve because they decide when to discard bits. An error between Alice and Bob is only kept if there is *another* error in the next bit. On the other hand, many errors between Alice and Eve stay.
Indeed, we can use these accepted bits as inputs for one or more iterations of Algorithm 11.5 (feeding the accepted bits back into the algorithm). $\square$

Remarks:

- Figure 11.7a shows the approximate error probabilities for several iterations. We see that after three iterations Alice and Bob pretty much agree on almost every bit.
- This algorithm also works in more extreme cases, e.g. $\alpha = \beta = 0.4$ and $\epsilon = 0.001$. After 7 iterations, Alice and Bob get down to an error rate of almost 0, whereas Eve has an error rate of 0.3 on Alice's bits, see Figure 11.7b.
- A formal proof would need a bit of Shannon's information theory, which is beyond the limits of this short lecture. However, information theory is a valuable tool, and a reader is encouraged to study it.
- After advantage distillation, $\hat{A}$ and $\hat{B}$ will likely not be equal. But now Alice and Bob know more about each other's bitstrings than Eve does.
- We need another algorithm to weed out the rest of the errors, and align Alice's and Bob's bitstrings.

11.4 Information Reconciliation

Algorithm 11.8 Information Reconciliation

Input: Bitstrings A for Alice and B for Bob with n bits each
Result: Alice and Bob generate $\tilde{A} = \tilde{B}$ (with high probability)

```
1: $k = 0$
2: repeat
3:     Alice sends Bob a random mask $M \in \{0,1\}^n$ using the private channel
4:     Let $m$ be the number of 1-bits in this mask $M$
5:     Alice computes $p_A = \sum_{i=1}^{n} M_i \cdot A_i \bmod 2$
6:     Bob computes $p_B = \sum_{i=1}^{n} M_i \cdot B_i \bmod 2$
7:     Alice and Bob exchange the masked parities $p_A$ and $p_B$
8:     if $p_A \neq p_B$ then
9:         $k = 0$
10:        $A_M = A[M == 1]$, masked substring of $A$
11:        $B_M = B[M == 1]$, masked substring of $B$
12:        ErrorCorrection$(A_M, B_M, 1, m)$
13:    else
14:        $k = k + 1$
15:    end if
16: until $k = c \log n$
```

Algorithm 11.9 ErrorCorrection

Input: Bitstrings A for Alice and B for Bob, with differing parity, left border l (initially 1), right border r (initially m)
Result: Bob corrects one bit that differs from Alice's corresponding bit

```
1: if $l == r$ then
2:     $B_l = A_l$ (Bob changes his value to Alice's value)
3: else
4:     $\mu = \frac{l+r}{2}$
5:     Alice computes $p_A = \sum_{i=l}^{\mu} A_i \bmod 2$
6:     Bob computes $p_B = \sum_{i=l}^{\mu} B_i \bmod 2$
7:     if $p_A \neq p_B$ then
8:         ErrorCorrection$(A, B, l, \mu)$
9:     else
10:        ErrorCorrection$(A, B, \mu + 1, r)$
11:    end if
12: end if
```

Lemma 11.10. *After termination of Algorithm 11.8, Alice and Bob share the same bitstring with an arbitrarily high probability* $\frac{1}{n^c}$.

Proof. Let $\delta_1 \ldots \delta_k$ be the indices where $A_{\delta_i} \neq B_{\delta_i}$. The parities of the two masked strings differ if the mask contains an odd number of these positions. Therefore, the probability of not detecting an existing error is equal to picking an even number of $\delta_1 \dots \delta_k$ into the mask. Let $m_{\delta_1}, \dots m_{\delta_{k-1}}$ be the choice for M for all but the last position. If an even number of these m_i are 1, the probability of keeping an even number is $\frac{1}{2}$ (picking $m_{\delta_k} = 0$). If an odd number of m_i are 1, the probability of having an even number is also $\frac{1}{2}$ (this time picking $m_{\delta_k} = 1$). Hence, our error probability of creating $c \log n$ masks in a row with even errors is $\frac{1}{2^{c \log n}} = \frac{1}{n^c}$. □

Remarks:

- This algorithm reveals *a lot* of information to Eve:

Lemma 11.11. *For correcting f errors between A, B, Eve learns $f \cdot (\log n + 2) + c \cdot \log n$ bits.*

Proof. For each of the f errors, Eve learns the actual bit that is corrected, and its neighboring bit. Eve also receives the parity information of the recursive calls before getting down to the single wrong bit. So on each power of two, Eve learns the equivalent information of (at most) one bit. So fixing a single bit costs the equivalent of revealing $\log n + 1$ bits. On top of this, there are rounds were Bob is not correcting any bits. Apart from the final $c \cdot \log n$ rounds, the first test in Algorithm 11.8 also gives $p_A = p_B$ with probability 1/2, so in expectation we have f more rounds where Eve learns a parity. In each of these rounds, Eve learns one parity of a mask, so the equivalent of (at most) one bit. □

Remarks:

- There are approaches that leak less information to Eve, but are more complicated or less efficient.
- Alice and Bob want to prevent Eve from using the leaked information to (partially) decrypt their messages. Therefore, they transform the shared key into a smaller key using a *hash function* where Eve does not have information about the values of any particular bits.

11.5 Privacy Amplification

Definition 11.12 (Universal Family)**.** *Let $\mathcal{H} \subseteq \{h : U \to M\}$ be a family of hash functions from U to M. If for all pairs of distinct keys $k \neq k' \in U$, the probability of a collision is $\Pr[h(k) = h(k')] \leq \frac{1}{m}$ when we choose $h \in \mathcal{H}$ uniformly, then $\mathcal{H}$ is called a **universal family (of hash functions)**.*

Remarks:

- If we choose a hash function from a universal family, we can expect the hashes to be distributed well, regardless of the key set.
- There are $(2^n)^{2^r}$ many possible functions, so we would need $2^r \cdot n$ bits to decode any of those.

Definition 11.13 (Universal Hashing)**.** *Let m be prime and $s \in \mathbb{N}$. Let $U = \{0, \dots, b-1\}^s$ and let $M = \{0, \dots, m-1\}$ with $b \leq m$. For a key $k =$*

$(k_0, \dots, k_{s-1}) \in U$ *and coefficient tuple* $a = (a_0, \dots, a_{s-1}) \in \{0, \dots, m-1\}^s$, *define*

$$h_a(k_0, \dots, k_{s-1}) = \sum_{i=0}^{s-1} a_i \cdot k_i \bmod m.$$

Then $\mathcal{H} := \{h_a : a \in \{0, \dots, m-1\}^s\}$ *is a universal family of hash functions.*

Remarks:

- We pick a universal hash function. Universal hashing gives us a general method for picking hash functions from a universal family in an efficient manner. We need a prime number m and uniformly at random some integers $a_0, \dots, a_r \in \{0, 1, \dots m-1\}$.
- Note that privacy amplification makes the key more secure without actually making it more secure. Why? Before hashing, the key had n bits, out of which Eve knew an equivalent of b (because of her satellite data plus information learnt from Algorithm 11.8). Before hashing, Eve could test the remaining 2^{n-b} key candidates and this attack is still possible now.
- For example with $\alpha = \beta = 0.1, \epsilon = 0.05$: If Alice and Bob run two rounds of information reconciliation and collect $n = 1024$ bits, they can expect to have about $f = 2$ errors, whereas Eve has about 62 errors. During information reconciliation (with $c = 2$), Eve can expect to learn only $4 \log n + 4 = 44$ bits. That leaves Eve with 2^{18} key candidates, which she would need to hash and test.
- However, the key is still more secure, since Eve loses the positional information of her knowledge. All that remains is a brute-force attack.
- Algorithm 11.14 summarizes the whole procedure:

Algorithm 11.14 Correlated Randomness Key Exchange

Input: Bitstrings A for Alice and B for Bob, $0 < \alpha, \beta, \epsilon < 0.5$
Result: Alice and Bob have a shared key with high probability

```
1: Alice and Bob determine the number of rounds based on $\alpha, \beta$ and $\epsilon$
2: $\hat{A}, \hat{B}$ = AdvantageDistillation(A,B) as in in Algorithm 11.5
3: $\tilde{A}, \tilde{B}$ = InformationReconciliation($\hat{A}, \hat{B}$) as in in Algorithm 11.8
4: $\bar{A}, \bar{B}$ = hash($\tilde{A}$), hash($\tilde{B}$), as in Definition 11.13
```

Remarks:

- Overall, this protocol is very complicated and takes many messages to agree on a key. Is there a better way?

11.6 Diffie Hellman Key Exchange

Definition 11.15 (Primitive Root)**.** *Let $p \in \mathbb{N}$ be a prime. $g \in \mathbb{N}$ is a primitive root of p if the following holds: For every $h \in \mathbb{N}$, with $1 \leq h < p$, there is a $k \in \mathbb{N}$ s.t. $g^k = h \mod p$.*

Example 11.16. *$g = 2$ is a primitive root of $p = 5$, because $2^1 = 2 \mod 5$, $2^2 = 4 \mod 5$, $2^3 = 3 \mod 5$, and $2^4 = 1 \mod 5$. There exists one more primitive root of 5.*

Algorithm 11.17 Diffie-Hellman Key Exchange

```
Input: Publicly known prime $p$ and a primitive root $g$ of $p$.
Result: Alice and Bob agree on a common secret key.

1: Alice picks a secret key $k_A \in \{1, 2, \ldots, p-1\}$ and sends $A = g^{k_A} \mod p$ to
   Bob.
2: Bob picks a secret key $k_B \in \{1, 2, \ldots, p-1\}$ and sends $B = g^{k_B} \mod p$ to
   Alice.
3: Alice calculates $K = B^{k_A} \mod p$
4: Bob calculates $K = A^{k_B} \mod p$
```

Example 11.18 (Algorithm 11.17 with $p = 5$ and $g = 2$)**.** *Let's assume that Alice picks $k_A = 2$ and Bob picks $k_B = 3$. Thus, Bob receives $A = 2^2 \mod 5 = 4$ and Alice receives $B = 2^3 \mod 5 = 3$. Then, Bob calculates $4^3 \mod 5 = 4$, and Alice calculates $3^2 \mod 5 = 4$. Hence, Alice and Bob have agreed on the common secret key of 4.*

Theorem 11.19. *In Algorithm 11.17, Alice and Bob agree on the same key K.*

Proof. Everything mod p, we have

$$K = B^{k_A} = \left(g^{k_B}\right)^{k_A} = g^{k_B k_A} = g^{k_A k_B} = \left(g^{k_A}\right)^{k_B} = A^{k_B} = K \mod p.$$

□

Remarks:

- There are sophisticated methods to quickly find primitive roots, but they are beyond the material covered in this chapter.
- How secure is Algorithm 11.17? It relies on the **assumption** that the computational problem if finding the discrete logarithm is hard to solve.

Definition 11.20 (Discrete Logarithm Problem)**.** *Let $p \in \mathbb{N}$ be a prime, and let $g, a \in \mathbb{N}$ with $1 \leq g, a < p$. The discrete logarithm problem is defined as finding an $x \in \mathbb{N}$ with $g^x = a \mod p$.*

Remarks:

- Intuitively, the best approach to calculate the common secret key of Algorithm 11.17 from the publicly known p, g, g^{k_A}, g^{k_B} is to solve the discrete logarithm problem. This is also the best known attack.
- However, for some classes of primes there are better attacks, which is why one often resorts to so-called safe primes p, where $p' = (p-1)/2$ is also a prime.
- The game changes with quantum computing: Shor's algorithm for quantum computers solves the discrete logarithm in polynomial time.
- Shor's algorithm also solves the factorization problem, which is used in RSA systems. In other words, both dominant asymmetric cryptographic systems will be broken with the availability of a quantum computer.

11.7 More Crypto

Is everything solved now that we exchanged a key? Nope. Additionally, crypto systems implement at least these functions:

- **Encryption**: How do we use an exchanged key to make our message unreadable to an attacker? There are several different techniques that could be used, for instance, One Time Pad, Bulk Encryption, Cypher Block Chaining, Advanced Encryption Standard.
- **Authentication**: How does the recipient know that the message was not tempered with? Hash-Based Message Authentication Code, Signatures.
- **Certification**: How can the recipient know that the sender is the person they pretend to be? Certificate Authorities, Web of Trust.

11.8 Post Quantum Cryptography

With quantum computers solving the discrete logarithm, do we have to go back to information theoretic approaches?

Definition 11.21 (Lattice). *Let $b_1, \dots b_n \in \mathbb{R}^n$ be a basis of vectors. We define the lattice $L \subset \mathbb{R}^n$ to be all linear combinations of this basis with **integer** coefficients.*

Definition 11.22 (Shortest Vector Problem). *Given a lattice L for a basis $b_1, \dots b_n \in \mathbb{R}^n$. The shortest vector problem constitutes finding the vector in the lattice with minimal euclidean norm to the origin. This problem is **believed** to be hard, even with a quantum computer.*

Remarks:

- We can formulate new computational encryption schemes where the best known attack requires solving the Shortest Vector Problem.
- Alternatively, Alice and Bob can generate a key using quantum coherence.

Chapter Notes

The basis of Algorithm 11.14 is by Cisszár and Körner [1], where Alice sends bits to Bob, and Bob as well as Eve receive a noisy version of these bits. In this setting, Alice and Bob can generate a key if Bob's noise is smaller than Eve's. Ueli Maurer [2] extended the scenario to the satellite setting and allowed Alice and Bob to communicate two way. Now they can also handle a superior Eve. In [3] this setting was analyzed. One famous algorithm for solving the discrete logarithm problem with a quantum computer was already formulated by Peter Shor in the last Millenium [4].

This chapter is based on text and figures from Stefan Wolf, and written in collaboration with Lukas Faber.

Bibliography

[1] I. Csiszar and J. Korner. Broadcast channels with confidential messages. *IEEE Transactions on Information Theory*, 24(3):339–348, May 1978.

[2] U. M. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39(3):733–742, May 1993.

[3] Ueli Maurer and Stefan Wolf. Information-theoretic key agreement: From weak to strong secrecy for free. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 351–368. Springer, 2000.

[4] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.