Chapter 8
Mathematical Key Postprocessing

In both QKD and RKD, raw keys are generated that are subject to errors. In order to remove errors from the keys, i.e., to ensure that both sides have identical keys, error correction is necessary after the actual key exchange. To perform this error correction, the two parties who carried out the key exchange need to exchange additional data. This data could fall into the hands of a listening attacker and provide them with information that could help them guess the true key more quickly and reliably. To prevent this, privacy amplification is also performed in QKD and RKD technologies after error correction.

8.1 Error Estimation

Before the actual error correction, the quality of the raw data generated is estimated. The aim of this step is to determine the extent to which the raw bit sequences available to the two communication partners differ and whether key generation can be continued securely under the given conditions.

For the security assessment, it is conservatively assumed that every error that is later eliminated by error correction is to be regarded as information that has potentially become known to an attacker. Even though it is clear that a significant proportion of these errors are caused in practice by noise, attenuation, and other physical effects.

To this end, selected parts of the raw data or statistical parameters derived from it are exchanged via the public communication channel. On this basis, an error or deviation rate is determined, which indicates how many bits of the two raw sequences are likely to not match. In QKD, the quantum bit error rate (QBER) is typically used for this purpose, while in RKD a corresponding bit deviation rate derived from the measured values is used. The underlying principle is identical in both cases.

E. Piller and H. Schölnast, *Data Encryption at the Intersection of Mathematics and Physics*, SpringerBriefs in Information Security and Cryptography,
https://doi.org/10.1007/978-3-032-24764-3_8

The quality indicator found determines both the effort required for error correction and the necessary reduction in key length within the framework of privacy amplification.

8.2 Error Correction

Alice and Bob send each other bit sequences that are much shorter than the raw keys, but contain statistical information about these keys that the respective other party can use for error correction. The information transmitted for the purpose of error correction can essentially be thought of as parity information or, more generally, redundancy information, i.e., in the broadest sense, something like check digits. In practice, interactive methods such as cascade [Tup23][1] or unilateral methods such as LDPC/Polar codes [Kik20][2] are used. However, a detailed description of these methods will not be provided here.

With the help of the exchanged parity or redundancy information, Alice and Bob can identify the positions where they have differing bits in their bit sequences, and they can also use this information to agree on identical bit values at these positions without directly transmitting the specific positions or bit values. At the end of the error correction process, they calculate short hash values from the bit sequences they have received, which they send to each other and compare in order to be absolutely sure that they now have exactly identical bit sequences. In fact, despite these measures, there remains a minimal residual risk of receiving different keys. However, the size of this residual risk is adjustable and is usually so extremely small that it can be considered practically secure.

Eve can read the exchanged parity information and hash values. This does not give her any information about specific individual bits in the key, but if Eve knew before the error correction that the key consisted of, for example, 300 bits, and the information exchanged for the error correction was 30 bits long, then the number of possible keys is reduced from 2^{300} to 2^{270}. This is still an astronomical number, but it is only about one billionth of the original number of possible keys. This reduction in the number of keys is systematically taken into account in the next step through privacy amplification.

8.3 Privacy Amplification

By publicly disclosing the error correction data, Eve obtains a certain amount of information that is also contained in the now corrected and identical raw keys. (In the example above, the information content of this amount of information is 30

[1] https://doi.org/10.1103/PhysRevApplied.20.064040.

[2] https://doi.org/10.1109/LCOMM.2020.3021142.

Shannon because it is 30 bits long and cannot be compressed.) Although this is essentially only general parity information, if you have enough of it, you can make well-founded assumptions about the finished key, which, at least in theory, make it a little easier for the attacker to reconstruct parts of the key.

To prevent this, the information-theoretical amount of information in the finished key must be reduced, at least by the amount of information contained in the error correction data. To stick with the example: The 300-bit-long non-compressible raw key has an information content of 300 Shannon, the error correction data contains 30 Shannon, so the key must be reduced to such an extent that it can only hold a maximum of 270 Shannon. This amounts to a reduction to a maximum of 270 bits, but you cannot simply delete 30 bits; instead, you must distribute the original information evenly across all remaining bits.

Another reason for this step is the errors themselves. Although it is known that most errors occur without any involvement from an attacker, it cannot be ruled out with certainty for any single different bit that it may have been caused by eavesdropping. Therefore, it is always decided to assume that every error is information that has fallen into Eve's hands. This is handled in a similar way to error correction data, which is also assumed to be known to the attacker: the amount of information in the keys is reduced by at least the maximum amount of information that an attacker could have obtained. This step is called "privacy amplification." The preceding estimation of the amount of information that an attacker could have obtained in the worst case (i.e., the number of bits by which the key must be shortened) is usually referred to as "leakage accounting" [Tom12].[3]

The Basic Procedure Is as Follows:

First, Alice and Bob exchange small parts of their keys to determine the quantum bit error rate (QBER, see above), and they count how much information they have exchanged with each other due to error correction. They subtract these two values and an additional security buffer from the raw key length to obtain the desired length of the final key.

Alice then uses a random number generator to create a random bit sequence that must not be related to the key currently being processed. This means that it must not be derived from this key in the form of a hash value or by other methods, but must be independent of it. However, it may be derived from a previously exchanged key, which may no longer be used for encryption afterwards. This random bit sequence is as long as the input key length plus the output key length. Alice transmits this bit sequence to Bob via the public classical side channel. From this, Alice and Bob generate the same matrix (a Toeplitz matrix over the residue field Z2 [WP–Toe,[4] WP–Res[5]]). They interpret the key they received after error correction as a vector, multiply it by the Toeplitz matrix, and thereby obtain a new vector in the desired shorter length, which they interpret again as a bit sequence. This matrix multiplication acts as a

[3] https://doi.org/10.1038/ncomms1631.

[4] https://en.wikipedia.org/wiki/Toeplitz_matrix.

[5] https://en.wikipedia.org/wiki/Residue_field.

Fig. 8.1 General form of a Toeplitz matrix

$$\begin{pmatrix} a & b & c & d & e \\ x & a & b & c & d \\ y & x & a & b & c \\ z & y & x & a & b \end{pmatrix}$$

universal hash function here. If the matrix was chosen randomly and independently of the key to be processed, each bit of the finished key depends equally on all bits of the input key, and the original amount of information was distributed completely evenly across all result bits. (Changing a single bit of the input key will, in the most likely case, flip exactly 50% of all bits in the finished key. The following also applies: if you want to change only a single bit of the result, you normally have to flip around 50% of all bits of the input key.)

Even if Eve was able to make certain assumptions about individual bits or bit combinations after error correction, she now has no information whatsoever about individual bits or bit combinations in the finished key.

Example of Multiplication with a Toeplitz Matrix

A Toeplitz matrix is completely defined by its values on the left and top edges. The value of every other cell is identical to the value of its neighbor diagonally above the cell, as shown in Fig. 8.1.

In privacy amplification, all entries in the Toeplitz matrix are bit values (0 or 1), with the values on the left and top edges being random values and the others being as described.

In the following example, the key after error correction is 5 bits long and is interpreted as a column vector. It is multiplied from the left by a 3x5 Toeplitz matrix with the properties described above in order to obtain a 3-bit-long secure key after privacy amplification. (The length is thus reduced by 2 bits.)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \bmod 2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

In a “normal” matrix multiplication, the result would have been $\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$, but “mod 2” takes the remainder when dividing by 2, resulting in the result shown.

Since these are bit values that can only be 0 or 1, the individual elementary multiplications can also be interpreted as logical AND operations. And if, instead of the usual summation, an XOR operation is performed on the intermediate results, the desired bit values are obtained immediately. The entire process can therefore be

reduced to very simple logical operations that can be performed at very high speed in specialized hardware and also deliver very high performance in standard processor chips.

References

[Tup23] Tupkary, Devashish, L. Norbert, Using cascade in quantum key distribution. Phys. Rev. Appl. **20**(6), 064040 (2023). https://doi.org/10.1103/PhysRevApplied.20.064040

[Kik20] Kiktenko, O. Evgeniy, et al., Blind information reconciliation with polar codes for quantum key distribution. IEEE Commun. Lett. **25**(1), 79–83 (2020). https://doi.org/10.1109/LCOMM.2020.3021142

[Tom12] Tomamichel, Marco, et al., Tight finite-key analysis for quantum cryptography. Nat. Commun. **3**(1), 634 (2012). https://doi.org/10.1038/ncomms1631

[WP-Toe] Wikipedia contributors, "Toeplitz matrix," Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Toeplitz_matrix

[WP-Res] Wikipedia contributors, "Residue field," Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Residue_field