1.1 General Discussion of Bitstream Ciphers

As a first example of a bitstream cipher we encountered XOR in Part I of these lectures. SageMath code is in Appendix E.1 of Part II.

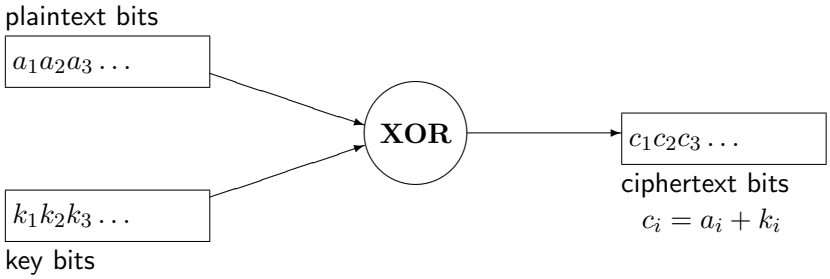

Figure 1.2: The principle of XOR encryption

In the twenties of the 20th century XOR ciphers were invented to encrypt teleprinter messages. These messages were written on five-hole punched tapes as in Figure 1.3, each column representing a five-bit-block. Another punched tape provided the key stream. Vernam filed this procedure as a U. S. patent in 1918. He used a key tape whose ends were glued together, resulting in a periodic key stream. Mauborgne immediately recognized that a nonperiodic key is obligatory for security.

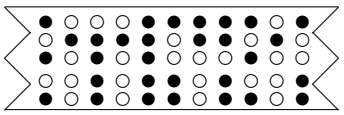

Figure 1.3: Punched tape—each column represents a five-bit character

In its strongest form, the one-time pad, XOR encryption is an example of perfect security in the sense of Shannon, see Part I, Section 10. As algorithm A5 or E_0 XOR helps to secure mobile phones or the Bluetooth protocol for wireless data transmission. As RC4 it is part of the SSL protocol that (often) encrypts client-server communication in the World Wide Web, and of the PKZIP compression software. There are many other current applications, not all of them satisfying the expected security requirements.

> *The scope of XOR encryption ranges from simple ciphers that are trivially broken to unbreakable ciphers.*

Advantages of XOR ciphers

- Encryption and decryption are done by the same algorithm: Since $c_i = a_i + k_i$ also $a_i = c_i + k_i$. Thus decryption consists of adding key stream and ciphertext (elementwise binary).
- The method is extremely simple to understand and to implement
- ... and very fast—provided that the key stream is available. For high transfer rates one may precompute the key stream at both ends of the line.
- If the key stream is properly chosen the security is high.

Pitfalls

- XOR ciphers are vulnerable under known plaintext attacks: each correctly guessed plaintext bit reveals a key bit.
- If the attacker knows a piece of plaintext she also knows the corresponding piece of the key stream, and then is able to exchange this plaintext at will. For example she might replace "I love you" by "I hate you", or replace an amout of 1000$ by 9999$. In other words the integrity of the message is poorly protected. (To protect message integrity the sender has to implement an extended procedure.)
- XOR ciphers provide no diffusion in the sense of SHANNON's criteria since each plaintext bit affects the one corresponding plaintext bit only.
- Each reuse of a part of the key sequence (also in form of a periodic repetition) opens the door for an attack. The historical successes in breaking stream ciphers almost always used this effect, for example the attacks on encrypted teleprinters in World War II, or the project VENONA during the Cold War.

A remark on the first item, the vulnerability for attacks with known plaintext: The common ISO character set for texts has a systematic weakness. The 8-bit codes of the lower-case letters `a..z` all start with `011`, of the upper-case letters `A..Z`, with `010`. A supposed sequence of six lower-case letters (no matter which) reveals $6 \cdot 3 = 18$ key bits.

> By the way the appearance of many zeroes in the leading bits of the bytes is an important identifying feature of texts in many European languages.

In other words: We cannot prevent the attacker from getting or guessing a good portion of the plaintext. Thus the security against an attack with

known plaintext is a fundamental requirement for an XOR cipher, even more than for any other cryptographic procedure.

This being said the crucial question for a pseudorandom sequence, or for the pseudorandom generator producing it, is:

> *Is it possible to determine some more bits from a (maybe fragmented) chunk of the sequence?*

The answer for the "classic" pseudorandom generators will be YES. But we'll also learn about pseudorandom generators that—supposedly—are cryptographically secure in this sense.