3. Principles of modern cryptography©

3.1. Rigorous Analysis [tbd]

3.2. Kerckhoffs' principle

Cryptography is concerned with restricting manipulation of data, such as maintaining the secrecy of data against an eavesdropper (by encryption). When designing systems to prevent attacks and penetrations, it may seem preferable to maintain the secrecy of the defense methods and mechanisms. Indeed, obviously, attacking a system is easier if we know the design of its defense mechanisms.

However, weaknesses of cryptographic systems are often hard to identify. When the design of the cryptosystems is secret, it is not subject to scrutiny and cryptoanalysis, and weaknesses may exist, unknown to the users. On the other hand, the attacker may be able to learn the design anyway, e.g. by reverse engineering, and may detect some weakness. Furthermore, secret designs cannot be shared widely or standardized and therefore prevent wide deployment.

This motivates the Kerckhoffs' principle [K83]: cryptographic systems should be designed to remain secure, even for attackers who know every detail of the design, except the values of secret keys. This does not necessarily mean that the design must be made public, but at least that the security analysis must not assume that the design is secret.

3.3. Unconditional security [tbd - add here]

In a seminal paper [S49], Shannon defined security (of a cipher) with respect to an adversary with unlimited computational power. Specifically, a cipher is secure in this sense if the adversary gains no extra information from the ciphertext. The plaintext is assumed to be chosen randomly from some probability distribution, known to the adversary. The probability distribution must remain the same, for a plaintext message randomly chosen using this distribution, when given its encryption (the ciphertext).

This notion of security is usually referred to as information-theoretic security or as unconditional security, contrasting it with the more commonly used computational security notion (discussed next), which is based on assuming that the adversary has limited computational resources.

Shannon also proved that the *one time pad* cipher is unconditionally secure. In this cipher, the ciphertext is the result of exclusive-or of each bit of the plaintext with a corresponding

bit of the key. We use each bit of the key only once (exclusive-or with only one plaintext bit), therefore the length of the key should be at least the length of the plaintext. Namely, ciphertext bit *t*, denoted *c[t]*, is:

$$c[t] = p[t] \oplus k[t]$$

The reader is encouraged to confirm that knowing *c[t]* does not change the probability of either value of *p[t]*, whatever that is. More precisely:

Claim 1 For any integer *n* and any probability distribution M_n of *n*-bit strings, select $k \in_R \{0,1\}^n$ and $M \in_R M_n$. Then for every $x,y \in \{0,1\}^n$ *holds:* $P(M=x)=P(M=x|(k \oplus M)=y)$.

The one time pad is not very practical, mainly due to its vast key size (i.e., same as the plaintext). Shannon also showed that this is unavoidable, i.e. any unconditionally secure encryption will require a key of length at least as long as the plaintext. Therefore, practical cryptography, and this book in particular, is more concerned with computational security, in spite of the need to assume that the adversary is computationally bounded and that there are problems that are computationally intensive.

[TBD – add exact impossibility claim here]

3.4. Computational security [extend]

Computational security relies on assumptions on the computational limitations of the adversary, used together with assumption on the computational hardness of some problem. The hardness is a function of the length of the key or other inputs to the cryptographic mechanism, which we typically denote by *n*. We normally assume that the adversary computational abilities (storage and operations) are bounded by some polynomial in *n*, namely for some constant *c* and sufficiently large *n*, the maximal amount of work that the adversary can perform is bounded by n^c.

In addition to limiting the computational abilities of the attacker, when defining computational security we also allow a small probability of success for the attacker. This follows from the fact that the attacker can always simply guess the key, albeit with very small probability. Therefore, all of our claims and definitions allow the attacker to gain by the attack – but only a small, negligible advantage, e.g. by simply guessing a key. Such *negligible advantage* must be smaller than any polynomial in *n*, to prevent (computationally bounded) adversary from improving substantially by repeated attacks.

Discuss constructions, reductions, principle of using weakest possible assumptions/requirements, practical feasibility, quantitative security reductions

3.5. Clarity and prudence principles

Precise, formal analysis and design of cryptographic problems and schemes is difficult. It is tempting to use informal, intuitive arguments and definitions, or to assume particular attack scenarios and specific environment and usage. However, relying on informal arguments or

assumptions about the environment, and especially about the attack scenarios, may lead to false sense of security and to undetected weaknesses.

Attackers often use unexpected approaches, which may be outside a list of `expected` attack scenarios. Following Kerckhoffs' principle, we assume that the adversary knows the design of the system, and therefore will try to find attacks that are different from the ones that the designer envisioned. A list of known attack scenarios may be useful as a `checklist`, to gain intuition about a design or to quickly identify problems and weaknesses; but it is not a sufficient proof of security.

Similarly, many attacks use scenarios that differ from the environment that the designers considered or assumed, formally or intuitively. Often, the weaknesses are exactly in the `fuzzy areas` such as implicit assumptions. Many attacks exploit differences between the `real` goals and scenarios, and the informal notions. For example, designers may assume, implicitly or explicitly, uniformly distributed inputs, while in reality the inputs may be significantly biased. Similarly, when relevant, we assume that the adversary knows the probability distribution of all inputs.

To avoid such pitfalls and weaknesses, analysis should follow the *prudence* and *clarity* principles:

- *Clarity*: we must clearly state all assumptions about the environment and goals; claims for fulfilling the goals must follow clearly from the assumptions. The adversary is assumed to know all assumptions, including these of probability distributions of the inputs.
- *Prudence*: the only assumptions about the behavior of the adversary can be on its abilities, such as computational abilities (e.g., its computational abilities are bounded by a polynomial in the length of the key). Definitions of security must be as general as possible, to hold for any conceivable application and environment.

Unfortunately, the complexity of real-life problems and systems often force practitioners to compromise on formal, precise definitions and proofs. Furthermore, completely precise and formal treatment is quite complex and difficult.

In this book, we try to balance between precision and complexity and provide practitioners a simple methodology for understanding cryptographic protocols and systems, without requiring in-depth knowledge of number theory, complexity theory or probability. We limit ourselves to simplified definitions, claims and proofs. We believe this approach allows readers to understand and make simple and yet sound claims, appropriate for simple cryptographic protocols and systems such as dealt with in most of this book. Readers should be careful not to rely on security of protocols and systems, without precise and formal validation of the security claims.

In the simplified terminology we adopt, we attempt to capture the essence of the precise, formal theory. In particular:

- Statements are always in respect to the specified probability distributions (uniform if unspecified).
- When discussing abstract cryptographic mechanism (e.g. encryption), we assume that a `long enough` key size is used (to make it infeasible for the attacker to guess the key randomly with significant probability).
- We say that the attacker cannot do an operation (attack), or that it is infeasible, if an attacker, with `reasonably bounded` computational capabilities, cannot succeed in it with significant probability.
- We say that two scenarios or types of data are indistinguishable or equivalent, if an attack is feasible in one scenario or with one type of data, if and only if it is feasible in the other scenario or with the other type of data.

3.6. Exercises

1. Prove that the one-time-pad is unconditionally secure.
2. A company announces it developed an unbreakable encryption function, with a 256-bit key.
    a. The company explains that the only way to break the system is to try all 2^{256} possible keys, which is clearly infeasible. Criticize.
    b. The company claims that any attack on the system would imply NP=P. Criticize.
3. Assume that a polynomial solution is found to one of the NP-complete problems, showing that NP=P. Does this imply that all existing cryptographic functions are insecure?