ECE 443/518 – Computer Cyber Security
Lecture 02 Cryptography

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

August 20, 2025

Outline

Reading Assignment

- This lecture: UC 1
- Next lecture: UC 2

Outline

Cryptography

- “secret writing”
- Old and new
  - As early as 2000 B.C. in ancient Egypt
  - Turing vs. Enigma machine in World War II
  - Academic research and commercial adoption since 1970’s
- Essential for computer cyber security.
  - Provide good examples for us to learn to identify threats and to design defense mechanisms in a formal (mathematical) setting.
  - Many security constructs are impossible without advances in cryptography.

Basic Model

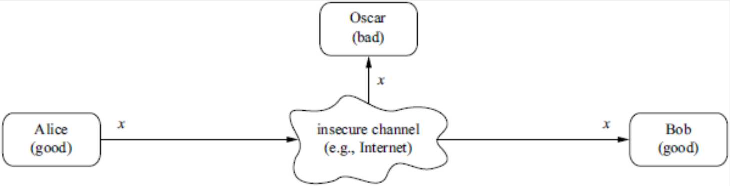


Fig. 1.4 Communication over an insecure channel

(Paar and Pelzl)

- Recall our example of king and general.
- Alice and Bob
  - For "good" parties like the king and the general.
  - Instead of using meaningless symbols like A and B.
- The opponent (attacker) Oscar who is "bad".
- The message x passing through the "insecure" channel for communication.
- What do "good", "insecure", and "bad" mean?
  - If we need to discuss security requirements like confidentiality and integrity?

Assumptions

- "Good" parties
  - We trust that Alice and Bob will faithfully follow the mechanism that we will design later.
  - If they use computers, we trust the computers to faithfully follow the mechanism.
- "Insecure" channel
  - We treat the channel as a blackbox that receives messages from Alice and sends messages to Bob.
  - We leave what is allowed and what is not allowed to happen in the channel to the "bad" opponent.
- "Bad" opponent, i.e. adversary
  - Address security requirements by defining behavior of attackers.
  - Passive adversary: break confidentiality by reading messages passing through the channel – but cannot do anything else like modifying messages or inserting messages.
  - And many other types of adversaries.

Outline

Symmetric Cryptography

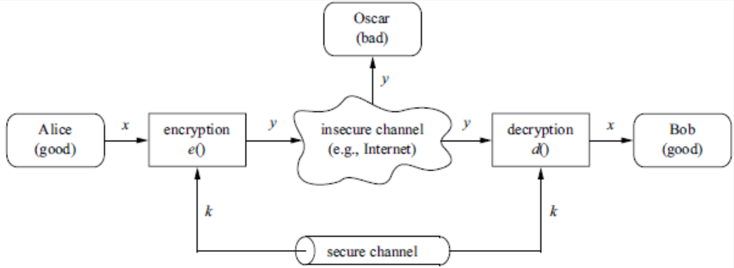


Fig. 1.5 Symmetric-key cryptosystem

(Paar and Pelzl)

- A mechanism for confidentiality
- plaintext x, ciphertext y, and the key k
- $e()$: encryption such that $y = e_k(x)$
- $d()$: decryption such that $x = d_k(y)$
- "Symmetric": both Alice and Bob know k.
  - If you feel uncomfortable with the secure channel to establish k between Alice and Bob, you are not alone – this motivated the discovery of public-key cryptography.

Assumptions

- Adversaries know y.
- No "security by obscurity"
  - We should assume adversaries to know $e()$ and $d()$.
  - Attackers will eventually know $e()$ and $d()$.
  - History showed that to break the system from there was easy.
  - No matter there is additional secret (Enigma) or not (DVD/CSS).
- Adversaries cannot know k directly.
  - But might be able to derive k from y, $e()$, and $d()$
  - Plus any other information explicitly allowed, e.g. x.

Problem Formulation: Ciphertext-only Attack (COA)

Given y, $e()$, and $d()$, find x and k such that:

$$y = e_k(x), \text{ and } x = d_k(y).$$

- Use mathematics to model how passive adverseries attack symmetric cryptography.
- Ciphertext-only attack using brute-force
  - Key space K: the set of all possible keys
  - For each $k \in K$, compute $x = d_k(y)$ and report k if x is meaningful.
  - What does "meaningful" mean?

Simple Symmetric Encryption: The Substitution Cipher

- For illustration purposes only.
- x consist of upper case letters and spaces.
- k is a mapping from upper case letters to lower case letters.
  - E.g. $A \rightarrow k$, $B \rightarrow d$, $C \rightarrow w$, $\ldots$
- $e()$ uses k to substitute upper case letters in x.
  - E.g. for $x = ABBA\ C$ we have $y = kddk\ w$.
- k needs to be one-to-one for $d()$ to work properly.
- Can we apply brute-force to find k and x for the ciphertext y below?

  ```
  iq ifcc vqqr fb rdq vfllcq na rdq cfjwhwz hr bnnb
  hcc hwwhbsqvqbre hwq vhlq
  ```

Key Space Matters

- There are $26 * 25 * \cdots * 1 \approx 2^{88}$ possible keys for the passive adversary to try using brute-force.
  - Need a few billions years if a computer can try a key in a nanosecond.
- What if the attackers get lucky?
  - They could find the key without trying all possible keys.
  - We indeed need to consider the expected running time conditioned under a certain probability of successful attacks.
  - We omit such analysis for simplicity because of the strong correlation between the conditional expectations and the key space size if keys are uniformly distributed.

Practical Limitation of Computational Power

- We claim the substitution cipher is computationally secure against ciphertext-only attack using brute-force.
  - Assume the passive adversary is computationally bounded instead of unbounded.
- Can a computationally bounded passive adversary apply another attack to break the substitution cipher?
- Is there a cipher secure against ciphertext-only attack using brute-force for computationally unbounded passive adversaries?

Cryptanalysis

`iq ifcc vqqr fb rdq vfllcq na rdq`

`cfjwhwz hr bnnb hcc hwwhbsqvqbre hwq vhlq`

- Instead of treating the substitution cipher as a blackbox, adversaries may exploit how it encrypts messages.
- Spaces are preserved so adversaries can identify words.
  - In particular those short words.
  - Any good guess of what is `rdq`?
- Adversaries may work with a key known only partially.
  - What is `hr` if adversaries can decrypt `rdq`?
  - And then `hcc` and `hwq`? And then everything?

Cryptanalysis (Cont.)

`iqifccvqqrfbrdqvfllcqnardqcfjwh`
`wzhrbnnbhcchwwhbsqvqbrehwqvhlq`

What if we preprocess the plaintext to remove spaces?

- With some effort, we can still read the message.
- Adversaries cannot decrypt by identifying short words first.
- However, as the same upper case letter maps to the same lower case letter, the letter frequencies will match those for English.
  - E.g. E, T, A are most probable.
- Adversaries may still obtain x without first knowing k.

Lesson Learned

- Key space need to be large enough to resist brute-force for computationally bounded adversaries.
- Good ciphers should not allow to decrypt partially with partially known keys.
- Good ciphers should hide the statistical properties of the encrypted plaintext.
  - Preprocess the plaintext to remove any statistical properties, e.g. compression, will further help.
- Don't design ciphers by yourself and expect them to be good!

Other Attacks

- Even if ciphers themselves are secure ...
- Probalistic methods
  - If keys are not uniformly distributed, the attacker can start with keys that are more probable and reduces the expected running time of brute-force.
  - A practical concern with defective or compromised random number generators.
- Implementation Attacks
  - Even if a mechanism is secure, implementations may leak x and k through a side-channel.
  - Usually associated with signals in the physical world.
- Social Engineering Attacks
  - As utimately human beings manage the secret key, adversaries may exploit our weakness to obtain the key.
  - Via violence, deception, system/software bugs etc.

Outline

Cryptography

Symmetric Cryptography

Modular Arithmetic

Retrospection

- Without computers, ancient ciphers are limited to simple rules that can be followed by human beings.
  - Usually simplified substitution ciphers.
  - Can be described by mathematics, especially those dealing with arithmetics, known today as elementary number theory.
- With a computer, it turned out elementary number theory still plays a very important role in designing cryptosystem with surprising properties.
- Let's start with modular arithmetic.

Integer Division with Remainder

Given a (dividend) and $m > 0$ (divisor), there exist unique q (quotiant) and r (remainder) such that:

$$a = qm + r, \text{ and } 0 \leq r < m,$$

where a, m, q, r are all integers.

- m divides a iff (if and only if) $r = 0$, written as $m|a$.
  - In such case, we also call m a factor, or a divisor of a.
  - Obviously $1|a$ and $a|a$. a is a prime number iff a has no other divisor.
- We use $a \bmod m$ to emphasize the process to compute r from a and m.
  - We don't care about the quotiant most of the time.
  - Most programming languages use %. But be aware of the difference when a is negative.
  - Anyway, cryptography nowadays uses extremely large integers so we always need to rely on library functions.

Practices

- $13 \bmod 5$
- $17 \bmod 5$
- $(13 * 17) \bmod 5$
- $((13 \bmod 5) * 17) \bmod 5$
- $(13 * (17 \bmod 5)) \bmod 5$
- $((13 \bmod 5) * (17 \bmod 5)) \bmod 5$
- The last 4 equations give the same result.
  - There is a better way to reason with remainders without computing them everytime.

Congruence

If $a \bmod m$ and $b \bmod m$ is the same, we write:

$$a \equiv b \pmod{m}.$$

- That is equivalent to $m|a-b$.
- In comparison to the textbook, we use the extra parenthesis around $\pmod{m}$ to emphasize $\equiv$ works like $=$.
  - Addition, subtraction, and multiplication just work.
  - E.g. since $13 \equiv 3 \pmod{5}$ and $17 \equiv 2 \pmod{5}$, we have

$$13 * 17 \equiv 3 * 2 \equiv 6 \equiv 1 \pmod{5}.$$

  - This kind of structures is called a ring.
  - What about divisions?

Algebra

- What is $\frac{1}{2}$?
  - 0.5. Not an integer.
  - Or we can use algebra: $\frac{1}{2}$ is a solution to $2x = 1$.
    - If this doesn't make sense, then think of $\sqrt{2}$.
- Now consider congruence and treat $\equiv$ as $=$.
  - Does $2x \equiv 1 \pmod 5$ have an integer solution?
  - Yes, $x \equiv 3 \pmod 5$, infinite many integers.
- When does $ax \equiv b \pmod m$ have solutions?
  - Assume $a \not\equiv 0 \pmod m$.
  - If m is a prime number, then always there are solutions.
    - This is an example of finite field (a.k.a. Galois field).
  - What about $4x \equiv 1 \pmod 6$? $4x \equiv 2 \pmod 6$?

More on Algebra

Solve the following for the unknown integer x.

- Linear equation

$$ax \equiv b \pmod{m}.$$

- System of congruences

$$\begin{aligned} x &\equiv a_1 \pmod{m_1}, \\ x &\equiv a_2 \pmod{m_2}, \\ &\ldots, \\ x &\equiv a_n \pmod{m_n}. \end{aligned}$$

- n-th root

$$x^n \equiv a \pmod{m}.$$

- Discrete logarithm

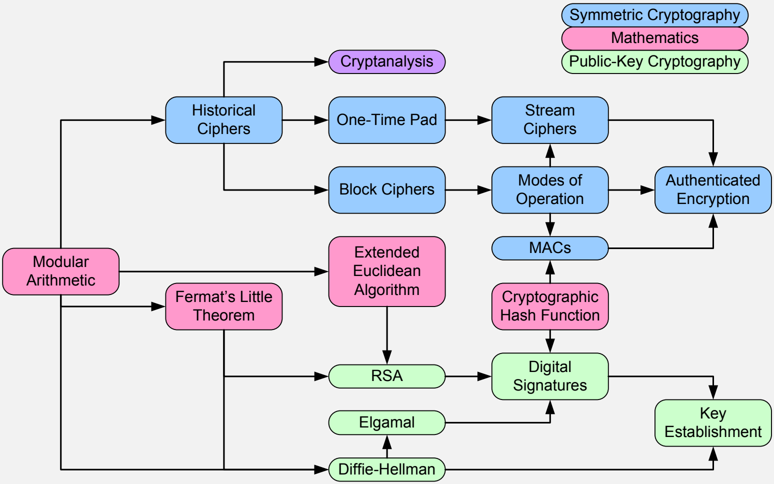
$$a^x \equiv b \pmod{m}.$$

- They serve as the foundation for the current practice of public-key cryptography.

Historical Ciphers

- Message encoding
  - Upper case letters only, each as an integer between 0 and 25.
  - Plaintext and ciphertext are both strings of integers.
- Caesar Cipher, a.k.a. Shift Cipher
  - Choose an integer key k
  - $e()$: substitute each plaintext letter x with $x + k \bmod 26$.
  - $d()$: substitute each ciphertext letter y with $y - k \bmod 26$.
  - Can be extended to Affine Cipher with a pair of integers (a, b) as the key where $e(x) = ax + b \bmod 26$.
- The key space is too small to even resist brute-force attack.
  - For Caesar cipher, any $k' \equiv k \pmod{26}$ will work – adversaries only need to try 26 keys.
  - For affine cipher, at most $25 * 26$ keys since a cannot be 0.

Introductory Cryptography Readmap



- The midterm exam will cover most of them.