CPSC 467: Cryptography and Security

Michael J. Fischer

Lecture 5
September 15, 2020

Classical Private-Key Ciphers

Substitution Ciphers

Rotor Machines

Polygraphic Ciphers

Adversary Powers
- Computationally limited adversaries
- Kinds of attacks

Classical Private-Key Ciphers

Basic principles of classical cryptosystems

Classical ciphers are built from two principles:

1. *Confusion* Substitute a letter or a block of letters with another letter or block.
2. *Diffusion* Change the position of letters within a block or message.

Substitution Ciphers

Permuting the alphabet

A *substitution cipher* permutes the alphabet. Each letter is replaced by its image under the permutation. This is how a classical cryptogram works.

The Caesar cipher is a particularly simple substitution cipher, where the permutation is simply a *shift* (rotation) of the alphabet.

Affine ciphers

Affine ciphers generalize simple shift ciphers such as Caesar.

Let α and β be two integers with $\gcd(\alpha, 26) = 1$.

A key is a pair $k = (\alpha, \beta)$.
There are 12 possible choices for α (1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25) and 26 possibilites for β, so $|\mathcal{K}| = 12 \times 26 = 312$.

Encryption: $E_k(m) = \alpha m + \beta \bmod 26$.

Decryption: $D_k(c) = \alpha^{-1}(c - \beta) \bmod 26$.

Here, α^{-1} is the multiplicative inverse of α in the ring of integers $\mathbf{Z}_{26}$. For example, $5^{-1} = 21$ since $21 \times 5 = 105 \equiv 1 \pmod{26}$.

α^{-1} exists precisely when $\gcd(\alpha, 26) = 1$.

Encrypting Longer Messages

Substitution ciphers are often extended to longer messages by applying the substitution to each letter of the message.

This method of extending a cryptosystem on single letters to work with multiletter messages is called *Electronic Codebook (ECB)* mode.

The Full Caesar Cipher was obtained from the Basic Caesar Cipher by using it in ECB mode.

Polyalphabetic ciphers

A *polyalphabetic substitution cipher* allows a *different* substitution to be applied to each plaintext letter, depending on its position i in the message.

- Choose r different alphabet permutations $\pi_1, \ldots, \pi_r$ for some number r.
- Use π_1 for the first letter of m, π_2 for the second letter, etc.
- Repeat this sequence after every r letters.

Vigenère cipher

The Vigenère cipher is a simplified polyalphabetic cipher in which each substitution is a simple rotation of the alphabet as with the Caesar cipher.

The key is the tuple $(r, k_0, \ldots, k_{r-1})$.

The i plaintext letter is encrypted using the Caesar cipher with key k_s, where $s = i \bmod r$.

Vigenère example

Suppose $k = (3, 5, 2, 3)$ and $m =$ "et tu brute".

Plaintext	`ettub rute`
Sub-key	`52352 3523`
Ciphertext	`jvwzd uzvh`

Breaking polyalphabetic ciphers

Polyalphabetic ciphers are much harder to break than monoalphabetic ciphers, and many are secure enough against manual attacks to have been used at various times in the past.

Nevertheless, all can be broken using letter frequency analysis given long enough messages. This is because every r^{th} letter is encrypted using the same permutation, so the submessage consisting of just those letters still exhibits normal English language letter frequencies.

Rotor Machines

Rotor machines

Rotor machines are mechanical polyalphabetic cipher devices that have a very large value of r and that generate the permutations used to encode successive letters in a deterministic way.

They were invented about 100 years ago and were used into the 1980's.

See Wikipedia page on rotor machines for a summary of the many such machines that have been used during the past century.

The German Enigma machines

- Enigma machines are rotor machines invented by German engineer Arthur Scherbius.
- They played an important role during World War 2.
- The Germans believed their Enigma machines were unbreakable.
- The Allies, with great effort, succeeded in breaking them and in reading many top-secret military communications.
- This is said to have changed the course of the war.

Image from Wikipedia

How a rotor machine works

- Uses electrical switches to create a permutation of 26 input wires to 26 output wires.
- Each input wire is attached to a key on a keyboard.
- Each output wire is attached to a lamp.
- The keys are associated with letters just like on a computer keyboard.
- Each lamp is also labeled by a letter from the alphabet.
- Pressing a key on the keyboard causes a lamp to light, indicating the corresponding ciphertext character.

The operator types the message one character at a time and writes down the letter corresponding to the illuminated lamp.

The same process works for decryption since $E_{k_i} = D_{k_i}$.

Keystream generation

The encryption permutation.

- Each rotor is individually wired to produce some random-looking fixed permutation π.
- Several rotors stacked together produce the composition of the permutations implemented by the individual rotors.
- In addition, the rotors can rotate relative to each other, implementing in effect a rotation permutation (like the Caeser cipher uses).

Keystream generation (cont.)

Let $\rho_k(x) = (x + k) \bmod 26$. Then rotor in position k implements permutation $\rho_k \pi \rho_k^{-1}$. (Note that $\rho_k^{-1} = \rho_{-k}$.)

Several rotors stacked together implement the composition of the permutations computed by each.

For example, three rotors implementing permutations π_1, π_2, and π_3, placed in positions r_1, r_2, and r_3, respectively, would produce the permutation

$$\begin{aligned} & \rho_{r_1} \cdot \pi_1 \cdot \rho_{-r_1} \cdot \rho_{r_2} \cdot \pi_2 \cdot \rho_{-r_2} \cdot \rho_{r_3} \cdot \pi_3 \cdot \rho_{-r_3} \\ = \quad & \rho_{r_1} \cdot \pi_1 \cdot \rho_{r_2 - r_1} \cdot \pi_2 \cdot \rho_{r_3 - r_2} \cdot \pi_3 \cdot \rho_{-r_3} \end{aligned} \tag{1}$$

Changing the permutation

After each letter is typed, some of the rotors change position, much like the mechanical odometer used in older cars.

The period before the rotor positions repeat is quite long, allowing long messages to be sent without repeating the same permutation.

Thus, a rotor machine is implements a polyalphabetic substitution cipher with a very long period.

Unlike a pure polyalphabetic cipher, the successive permutations until the cycle repeats are not independent of each other but are related by equation (1).

This gives the first toehold into methods for breaking the cipher (which are far beyond the scope of this course).

History

Several different kinds of rotor machines were built and used, both by the Germans and by others, some of which work somewhat differently from what I described above.

However, the basic principles are the same.

The interested reader can find much detailed material on the web by searching for "enigma cipher machine" and "rotor cipher machine". Nice descriptions may be found at `http://en.wikipedia.org/wiki/Enigma_machine` and `http://www.quadibloc.com/crypto/intro.htm`.

Polygraphic Ciphers

Hill cipher

A *polygraphic cipher* encrypts several letters at a time.
It tends to mask the letter frequencies, making it much harder to break.

The Hill cipher is such an example based on linear algebra.

- The key is, say, a non-singular 3×3 matrix K.
- The message m is divided into vectors m_i of 3 letters each.
- Encryption is just the matrix-vector product $c_i = Km_i$.
- Decryption uses the matrix inverse, $m_i = K^{-1}c_i$.

An attack on the Hill cipher

A *known plaintext attack* assumes the attacker has prior knowledge of some plaintext-ciphertext pairs $(m_1, c_1), (m_2, c_2), \ldots$.

The Hill cipher succumbs to a known plaintext attack.

Given three linearly independent vectors m_1, m_2, and m_3 and the corresponding ciphertexts $c_i = Km_i$, $i = 1, 2, 3$, it is straightforward to solve for K.

Adversary Powers

Adversaries of unlimited power

A cryptosystem that can resist attack from an adversary of unlimited power is *information-theoretically* secure.

We saw last time that the Vernam cipher (or one-time pad) is information-theoretically secure.

- An adversary of unlimited power can always carry out a brute force attack.
- Every possible decryption can be enumerated.
- Security relies on the adversary being unable to distinguish correct from incorrect decryptions.

Short keys

Any cryptosystem with short keys automatically gives away a lot of information about the plaintext – namely, it is the decryption of the given ciphertext under one of the possible keys.

If the key space is small and the adversary has sufficient power, then the adversary can get considerable partial information about the message.

In real-life situations, the adversary does not have unlimited time and space in order to break the cipher. The goal of the cipher is to make it costly for the adversary but not necessarily impossible.

Measuring computational difficulty

We want a notion of how much time is required to carry out a computational task.

Why not use actual running time?

- It depends on the speed of the computer as well as on the algorithm for computing the function.
- It varies from one input to another.
- It is difficult to analyze at a fine grained level of detail.

Role of complexity theory

Complexity theory allows one to make meaningful statements about the *asymptotic* difficulty of computational problems, independent of the particular computer or programming language.

Complexity measures *rate of growth* of worst-case running time as a function of the length n of the inputs.

An algorithm runs in time $T(n)$ if its running time on all but finitely many inputs x is at most $T(|x|)$.

An algorithm runs in *polynomial time* if it runs in time $p(n)$ for some polynomial function $p(n)$.

A function f is *polynomial time* if it is computable by some polynomial time algorithm.

Feasibility

The computational complexity of a cryptosystem measures how the time to encrypt and decrypt grows as a function of an underlying *security parameter* s.

Polynomial time functions are said to be *feasible*.

Feasibility is a minimal requirement.

In practice, we care about the actual run time for fixed values of the security parameter (such as $s = 512$).

Eve's information

Until now, we've implicitly assumed that Eve has no information about the cryptosystem except for the encryption and decryption methods and the ciphertext c.

In practice, Eve might know much more.

- She probably knows (or has a good idea) of the message distribution.
- She might have obtained several other ciphertexts.
- She might have learned the decryptions of earlier ciphertexts.
- She might have even chosen the earlier messages or ciphertexts herself.

This leads us to consider several attack scenarios.

Attack scenarios

Ciphertext-only attack Eve knows only c and tries to recover m.

Known plaintext attack Eve knows c and a sequence of plaintext-ciphertext pairs $(m_1, c_1), \ldots, (m_r, c_r)$ where $c \notin \{c_1, \ldots, c_r\}$. She tries to recover m.

Known plaintext attacks

A known plaintext attack can occur when

1. Alice uses the same key to encrypt several messages;
2. Eve later learns or successfully guesses the corresponding plaintexts.

Some ways that Eve learns plaintexts.

- The plaintext might be publicly revealed at a later time, e.g., sealed bid auctions.
- The plaintext might be guessable, e.g., an email header.
- Eve might later discover the decrypted message on Bob's computer.

Chosen text attack scenarios

Still stronger attack scenarios allow Eve to choose one element of a plaintext-ciphertext pair and obtain the other.

Chosen plaintext attack
: Like a known plaintext attack, except that Eve chooses messages $m_1, \ldots, m_r$ before getting c and Alice (or Bob) encrypts them for her.

Chosen ciphertext attack
: Like a known plaintext attack, except that Eve chooses ciphertexts $c_1, \ldots, c_r$ before getting c and Alice (or Bob) decrypts them for her.

Mixed chosen plaintext/chosen ciphertext attack
: Eve chooses some plaintexts and some ciphertexts and gets the corresponding decryptions or encryptions.

Why would Alice cooperate in a chosen plaintext attack?

- Eve might be authorized to generate messages that are then encrypted and sent to Bob, but she isn't authorized to read other people's messages.[1]
- Alice might be an internet server, not a person, that encrypts messages received in the course of carrying out a more complicated cryptographic protocol.[2]
- Eve might gain access to Alice's computer, perhaps only for a short time, when Alice steps away from her desk.

[1] Nothing we have said implies that Eve is unknown to Alice and Bob or that she isn't also a legitimate participant in the protocol.

[2] We will see such protocols later in the course.

Adaptive chosen text attack scenarios

Adaptive versions of chosen text protocols are when Eve chooses her texts one at a time after learning the response to her previous text.

Adaptive chosen plaintext attack Eve chooses the messages $m_1, m_2, \ldots$ one at a time rather than all at once. Thus, m_2 depends on (m_1, c_1), m_3 depends on both (m_1, c_1) and (m_2, c_2), etc.

Adaptive chosen ciphertext and adaptive mixed attacks are defined similarly.