CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 2
August 29, 2014

Course Overview

Secret Message Transmission

Symmetric Cryptography

Caesar cipher

Some other classical ciphers
- Generalized shift ciphers
- Polyalphabetic ciphers
- Polygraphic Ciphers

Course Overview

What is this course about?

This course is about *Applied Cryptography*.

Wikipedia says,

> *"Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). More generally, it is about constructing and analyzing protocols that overcome the influence of adversaries and which are related to various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation."*

`https://en.wikipedia.org/wiki/Cryptography`

Course modules

1. Security properties: Interests of various parties, motivation and capabilities of adversaries, knowledgable and provability.
2. Classical cryptography: Simple algorithms, information leakage, stream and block ciphers, DES, AES, message authentication codes.
3. Public key cryptography: 2-key systems, pseudorandom numbers, cryptographic hash functions, digital signatures.
4. Crypto toolbox: Multiparty protocols such as contract signing, oblivious transfer, zero-knowledge proofs, bit commitment, secret-splitting, and coin flipping.
5. Real-world applications such as SSH, SSL, WPA, encrypted email, PGP/GPG, bitcoin, etc.

Computer science, mathematics and cryptography

Cryptography cuts across both computer science and mathematics.

Computer science: Cryptographic algorithms must be implemented correctly and efficiently.

Mathematics: Underlies both algorithms and their analysis.

Many cryptographic primitives are based on:

- Number theoretic problems such as factoring and discrete log;
- Algebraic properties of structures such as elliptic curves.

Understanding and modeling security uses

- Probability theory and coding theory;
- Complexity theory.

Will explore in enough depth to provide insight for how algorithms work and why they are believed secure.

Organization

The main body of the course is organized around *cryptographic primitives*. For each one:

- What can be done with it? Study of cryptographic algorithms and protocols.
- What are its properties? Modeling and analysis. Requires complexity theory, probability theory, and statistics.
- How does it work? Requires some mathematics, particularly number theory and algebra.
- How is it implemented? Requires attention to detail, especially to prevent accidental leak of secret information.

What this course is not

This course is broad rather than deep.

- Only enough mathematics to understand algorithmes such as AES, RSA, ElGamal, and elliptic curves will be presented.
- It will only briefly touch on cryptanalysis, the flip side of cryptography.
- It will not go deeply into real-world security protocols.
- It will not talk about security mechanisms for computer and network devices and applications such as firewalls, operating system access controls, detecting software security holes, or dealing with web security vulnerabilities.

Secret Message Transmission

Secret message transmission problem

Alice wants to send Bob a private message m over the internet.

Eve is an *eavesdropper* who listens in and wants to learn m.

Alice and Bob want m to remain private and unknown to Eve.

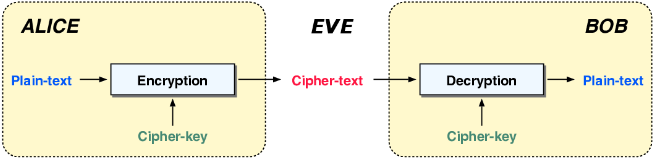


Image credit: Derived from image by Frank Kagan Gürkaynak,
`http://www.iis.ee.ethz.ch/~kgf/acacia/fig/alice_bob.png`

Solution using encryption

A *symmetric cryptosystem* (sometimes called a *private-key* or *one-key* system) is a pair of efficiently-computable functions E and D such that

- $E(k, m)$ *encrypts* *plaintext message* m using key k to produce a *ciphertext* c.
- $D(k, c)$ *decrypts* ciphertext c using k to produce a message m.

Requirements:

Correctness $D(k, E(k, m)) = m$ for all keys k and all messages m.

Security Given $c = E(k, m)$, it is hard to find m without knowing k.

The protocol

Protocol:

1. Alice and Bob share a common secret key k.
2. Alice computes $c = E(k, m)$ and sends c to Bob.
3. Bob receives c', computes $m' = D(k, c')$, and assumes m' to be Alice's message.

Assumptions:

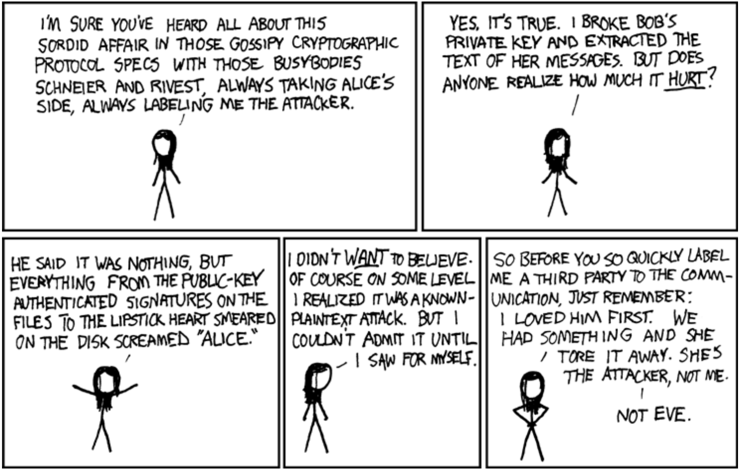
- Eve learns nothing except for c during the protocol.
- The channel is perfect, so $c' = c$. The real world is not so perfect, so we must ask what happens if $c' \neq c$?
- Eve is a *passive eavesdropper* who can read c but not modify it.

Requirements

What do we require of E, D, and the computing environment?

- Given c, it is hard to find m without also knowing k.
- k is not initially known to Eve.
- Eve can guess k with at most negligible success probability. (k must be chosen randomly from a large key space.)
- Alice and Bob successfully keep k secret. (Their computers have not been compromised; Eve can't find k on their computers even if she is a legitimate user, etc.)
- Eve can't obtain k in other ways, e.g., by social engineering, using binoculars to watch Alice or Bob's keyboard, etc.

Eve's side of the story


I'M SURE YOU'VE HEARD ALL ABOUT THIS SORDID AFFAIR IN THOSE GOSSIPY CRYPTOGRAPHIC PROTOCOL SPECS WITH THOSE BUSYBODIES SCHNEIER AND RIVEST, ALWAYS TAKING ALICE'S SIDE, ALWAYS LABELING ME THE ATTACKER.
YES, IT'S TRUE. I BROKE BOB'S PRIVATE KEY AND EXTRACTED THE TEXT OF HER MESSAGES. BUT DOES ANYONE REALIZE HOW MUCH IT HURT?
HE SAID IT WAS NOTHING, BUT EVERYTHING FROM THE PUBLIC-KEY AUTHENTICATED SIGNATURES ON THE FILES TO THE LIPSTICK HEART SMEARED ON THE DISK SCREAMED "ALICE."
I DIDN'T WANT TO BELIEVE. OF COURSE ON SOME LEVEL I REALIZED IT WAS A KNOWN-PLAINTEXT ATTACK. BUT I COULDN'T ADMIT IT UNTIL I SAW FOR MYSELF.
SO BEFORE YOU SO QUICKLY LABEL ME A THIRD PARTY TO THE COMMUNICATION, JUST REMEMBER: I LOVED HIM FIRST. WE HAD SOMETHING AND SHE TORE IT AWAY. SHE'S THE ATTACKER, NOT ME.
NOT EVE.

Cartoon by Randall Munroe, `https://www.xkcd.com/177/`

Symmetric Cryptography

Formalizing what a cryptosystem is

A *symmetric cryptosystem* consists of

- a set $\mathcal{M}$ of *plaintext messages*,
- a set $\mathcal{C}$ of *ciphertexts*,
- a set $\mathcal{K}$ of keys,
- an *encryption* function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- a *decryption* function $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$.

We often write $E_k(m) = E(k, m)$ and $D_k(c) = D(k, c)$.

Desired properties

Decipherability $\forall m \in \mathcal{M}, \forall k \in \mathcal{K}, D_k(E_k(m)) = m$. In other words, D_k is the left inverse of E_k.

Feasibility E and D, regarded as functions of two arguments, should be computable using a feasible amount of time and storage.

Security (weak) It should be difficult to find m given $c = E_k(m)$ without knowing k.

What's wrong with this definition?

This definition leaves three important questions unanswered?

1. What is a "feasible" amount of time and storage?
2. What does it mean to be "difficult" to find m?
3. What does it mean to not "know" k?

Practical considerations

These questions are all critical in practice.

1. E and D must be practically computable by Alice and Bob or the cryptosystem can't be used. For most applications, this means computable in milliseconds, not minutes or days.
2. The confidentiality of m must be preserved, possibly for years, after Eve discovers c. How long is long enough?
3. The only way to be certain that Eve does not know k is to choose k at random from a random source to which Eve has no access. This is easy to get wrong.

Caesar cipher

Encoding single letters

The Caesar cipher is said to go back to Roman times.

It encodes the 26 letters of the Roman alphabet $A, B, \ldots, Z$.

Assume the letters are represented as $A = 0$, $B = 1$, $\ldots$, $Z = 25$.

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0, \ldots, 25\}$.

$$E_k(m) = (m + k) \bmod 26$$

$$D_k(c) = (c - k) \bmod 26.$$

Formally, we have a cryptosystem for 1-letter messages.

Encoding longer messages

The Caesar cipher encrypts longer messages by encrypting each letter separately.

How do we formalize this?

- What is the message space now?
- What is the ciphertext space?
- What is the key space?
- What is the encryption function?
- What is the decryption function?

Caesar cipher formally defined

For arbitrary strings, we have

$$\mathcal{M}' = \mathcal{C}' = \mathcal{M}^*$$

where $\mathcal{M}^*$ is the transitive closure of $\mathcal{M}$.

That is, $\mathcal{M}^*$ consists of all sequences of 0 or more letters from $\mathcal{M}$.

The encryption and decryption for length-r sequences are

$$E'_k(m_1 \dots m_r) = E_k(m_1) \dots E_k(m_r)$$

$$D'_k(c_1 \dots c_r) = D_k(c_1) \dots D_k(c_r).$$

A brute force attack on the Casear cipher

Ciphertext `HWWXE UXWH`

Decryption key	Plaintext
$k = 0$	hwwxe uxwh
$k = 1$	gvvwd twvg
$k = 2$	fuuvc svuf
$k = 3$	ettub rute
$k = 4$	dssta qtsd
$k = 5$	crrsz psrc
$\dots$	$\dots$

Which is the correct key?

Recognizing the correct key

Caesar's last words, "Et tu, Brute?"

[From William Shakespeare's play, *Julius Casear*, Act 3, Scene 1.]

Ciphertext `HWWXE UXWH`

Decryption key	Plaintext
k = 0	hwwxe uxwh
k = 1	gvvwd twvg
k = 2	fuuvc svuf
k = 3	ettub rute
k = 4	dssta qtsd
k = 5	crrsz psrc
...	...

How do you know when you've found the correct key?

You don't always know!

Suppose you intercept the ciphertext JXQ.
You quickly discover that $E_3(\mathrm{GUN}) = \mathrm{JXQ}$.
But is $k = 3$ and is GUN the correct decryption?

You then discover that $E_{23}(\mathrm{MAT}) = \mathrm{JXQ}$.
Now you are in a quandary. Which decryption is correct?

Have you broken the system or haven't you?

You haven't found the plaintext for sure, but you've reduced the possibilities down to a small set.

Terminology

A *shift cipher* uses a letter substitution defined by a rotation of the alphabet.

Any cipher that uses a substitution to replace a plaintext letter by a ciphertext letter is called a *substitution cipher*. A shift cipher is a special case of a substitution cipher.

Any cipher that encrypts a message by applying the same substitution to each letter of the message is called a *monoalphabetic* cipher.

Some other classical ciphers

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 $\gcd(\alpha, 26) = 1$.

Polyalphabetic ciphers

Another way to strengthen substitution ciphers is to use different substitutions for different letter positions.

- Choose r different alphabet permutations $\pi_1, \dots, \pi_r$ 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.

While this is much harder to break than monoalphabetic ciphers, letter frequency analysis can still be used.

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.

Vigenère cipher

The *Vigenère cipher* is a polyalphabetic cipher in which the number of different substitutions r is also part of the key. Thus, the adversary must determine r as well as discover the different substitutions.

All polyalphabetic ciphers can be broken using letter frequency analysis, but they are secure enough against manual attacks to have been used at various times in the past.

The German Enigma encryption machine used in the second world war is also based on a polyalphabetic cipher.

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.

Playfair cipher

The *Playfair* cipher, invented by Charles Wheatstone in 1854 but popularized by Lord Lyon Playfair, is another example of a polygraphic cipher [MvOV96, chapter 7, pp. 239-240] and [Wik].

Here, the key is a passphrase from which one constructs a 5×5 matrix of letters. Pairs of plaintext letters are then located in the matrix and used to produce a corresponding pair of ciphertext letters.

How Playfair works

Construct the matrix from the passphrase.

- Construct the matrix by writing the passphrase into the matrix cells from left to right and top to bottom.
- Omit any letters that have previously been used.
- Fill remaining cells with the letters of the alphabet that do not occur in the passphrase, in alphabetical order.
- In carrying out this process, "I" and "J" are identified, so we are effectively working over a 25-character alphabet.

Thus, each letter of the 25-character alphabet occurs exactly once in the resulting matrix.

Example Playfair matrix

Let the passphrase be

“CRYPTOGRAPHY REQUIRES STRONG KEYS”.

The resulting matrix is

C	R	Y	P	T
O	G	A	H	E
Q	U	I/J	S	N
K	B	D	F	L
M	V	W	X	Z

First occurrence of each letter in the passphrase shown in orange:

“CRYPTOGRAPHY REQUIRES STRONG KEYS”.

Letters not occurring in the passphrase: BDFLMVWXZ.

Encrypting in Playfair: preparing the message

To encrypt a message using Playfair:

- Construct the matrix.
- Remove spaces and pad the message with a trailing 'X', if necessary, to make the length even.
- Break up the message into pairs of letters.
- In case a pair of identical letters is about to be produced, insert an "X" to prevent that.

Examples:

- "MEET ME AT THE SUBWAY" becomes "ME" "ET" "ME" "AT" "TH" "ES" "UB" "WA" "YX".
- "A GOOD BOOK" becomes "AG", "OX", "OD" "BO", "OK".

Encrypting in Playfair: substituting the pairs

To encrypt pair *ab*, look at rectangle with *a* and *b* at its corners.

1. If *a* and *b* appear in different rows and different columns, replace each by the letter at the opposite end of the corresponding row. Example: replace "AT" by "EY":

Y	P	**T**
A	H	E

2. If *a* and *b* appear in the same row, then replace *a* by the next letter circularly to its right in the row, and similarly for *b*. For example, the encryption of "LK" is "KB".
3. If *a* and *b* appear in the same column, then replace *a* by the next letter circularly down in the column, and similarly for *b*.

Example: "MEET ME AT THE SUBWAY" encrypts as "ZONEZOEYPEHNBVYIPW".

Decrypting in Playfair

Decryption is by a similar procedure.

In decrypting, one must manually remove the spurious occurrences of “X” and resolve the “I/J” ambiguities.

See Trappe and Washington [TW06] or Wikipedia [Wik] for a discussion of how the system was successfully attacked by French cryptanalyst Georges Painvin and the Bureau du Chiffre.

References

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone.
Handbook of Applied Cryptography.
CRC Press, 1996.

Wade Trappe and Lawrence C. Washington.
Introduction to Cryptography with Coding Theory.
Prentice Hall, second edition, 2006.
ISBN 0-13-186239-1.

Wikipedia.
Playfair cipher.
URL `http://en.wikipedia.org/wiki/Playfair_cipher`.