50.020 Security
Lecture 2: Basic Encryption

Overview Encryption schemes

In 50.020, we will talk about the following encryption schemes

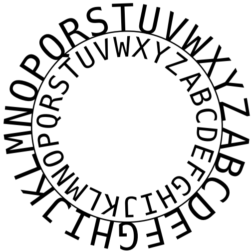
- Symmetric encryption schemes
  - Substitution ciphers
  - Stream ciphers
  - Block ciphers
- Asymmetric encryption schemes
  - RSA
  - Elliptic curve cryptography

Basics of substitution ciphers

- Historical ciphers, used until middle of last century
- Mono-alphabetic: plaintext and ciphertext based on alphabet (A-Z)
- Bijection (complete mapping) between both alphabets

Example (Caesar's cipher)

Shift all characters by k in alphabet
For $k = 3$:
'SECURITY'$\Rightarrow$'VHFXULWB'
Shift back to decrypt.



Security Assessment of Caesar's cipher

- System: Alice and Bob share key, no secure channel
- Attacker: Has ciphertext, does not have key, wants plain text
- Requirements: Confidentiality of plaintext, need key to decrypt
- How to attack? Effort?

Security Assessment of Caesar's cipher

- System: Alice and Bob share key, no secure channel
- Attacker: Has ciphertext, does not have key, wants plain text
- Requirements: Confidentiality of plaintext, need key to decrypt
- How to attack? Effort?

Brute force attack

- Try all possible values for keys (only 26)
- Derive which of the plaintexts is the correct one
- How can we make attacks harder?

Improving substitution ciphers

- The keyspace of Caesar's cipher is extremely small.
- Improve by a random mapping between the 26 in/output characters
- E.g., A$\rightarrow$ X, B$\rightarrow$ D, C$\rightarrow$ M,. . .
- How many different mappings exist?

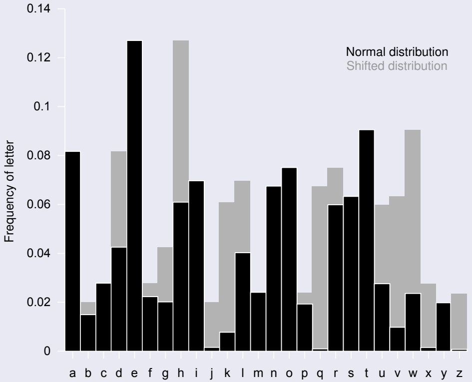
Improving substitution ciphers

- The keyspace of Caesar's cipher is extremely small.
- Improve by a random mapping between the 26 in/output characters
- E.g., A$\rightarrow$ X, B$\rightarrow$ D, C$\rightarrow$ M,...
- How many different mappings exist?

- 26! $\approx 4 \cdot 10^{26} \approx 2^{88}$
- But there are better ways to attack than brute force

Frequency analysis of ciphertext

Frequency of letters in English



Language-specific distribution can be used to identify substitutions

Advanced Substitution schemes

Examples to break up known frequency distribution:

- Have several alternative replacements for 'e', choose randomly
- Intentionally misspell or use dialect
- Insert 'red herring' characters to mislead analysis
- Treat 'et' as a new character, map it to a new symbol α
- Substitutions are still part of modern ciphers, but must operate on alphabets with uniform likelihood

Vigenère cipher

- Published in 1553 by Giovan Battista Bellaso
- Changes the substitution mapping in period pattern
- Key is a word that defines that pattern

	a	b	c	d	e	f	.
A	a	b	c	d	e	f	.
B	b	c	d	e	f	g	.
C	c	d	e	f	g	h	.
.	.	.	.	.	.	.	.

Plaintext:	`dead beef`
Key "cab":	`CABC ABCA`
Ciphertext:	`febf bfgf`

Breaking the Vigenère cipher

- Direct frequency analysis will not be successful any more
- Frequent character "peaks" are distributed

Breaking the Vigenère cipher

- Direct frequency analysis will not be successful any more
- Frequent character "peaks" are distributed

- Solution: as key has fixed length and is repeated often:
  - For different key lengths n, compute distributions for each n'th character
  - For the right key length, you will see characteristic distributions again
  - Similar to n Caesar's ciphers used to encrypt the plaintext
  - Break each character of key separately, then break the encryption

Transposition ciphers basics

- Letters do not get replaced, but their sequence is changed
- Shared key determines new sequence
- With message "This is secret" and "bar" as password:

key	B	A	R
order	2	1	18
text	T	H	I
	S	I	S
	S	E	C
	R	E	T

The ciphertext is "HIEETSSRISCT"

- How to attack this?

Character encodings (ASCII)

In practise, data is not represented by the Latin alphabet

- Kerkhoffs already mentioned telegraphs (Morse code)
- Computing systems use binary representations, e.g. ASCII
- ASCII represents 128 Latin & control characters in 7 bits
- Example: 0x61=a, 0x41=A, "Hello"=0x48656C6C6F
- From now on, we will operate on binary data (=integers)

Substitutions on binary data

How can the substitution principle be applied to binary data?

Substitutions on binary data

How can the substitution principle be applied to binary data?

Based on single bits

- inversion, 2 different keys possible (one encrypts as plaintext!)

Substitutions on binary data

How can the substitution principle be applied to binary data?

Based on single bits

- inversion, 2 different keys possible (one encrypts as plaintext!)

Based on double bits

- every two bits are replaced, 4! possible keys

Substitutions on binary data (continue)

Based on n bit *blocks*

- $2^n!$ possible keys

Frequency distribution of n-bit blocks

- Depending on the character coding and n, some blocks might still be more frequent
- This would enable attacks again

Overview Modern Ciphers

Data processing

- Ciphers operate on streams or blocks
- Stream ciphers operate on single characters at a time
- Blocks have fixed length, are processed in one go

Basic operations

- Mostly XOR, shifts (performance reasons)
- Some ciphers use algebraic operations such as (+*^), x mod n
- All operations are operating on finite sets of numbers

Overview Modern Ciphers (Continue)

Cipher can be symmetric or asymmetric

- Symmetric: same key for enc,dec
- Asymmetric: different keys for enc,dec (aka public-key crypto)

Stream ciphers vs. block ciphers

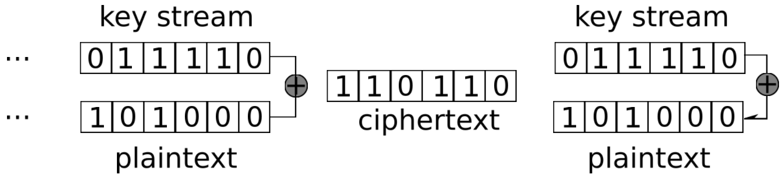
Stream ciphers

- operate on single elements of the input (single characters, bits)
- Well suited for (audio) signal transmission
- Pro: low processing delay for low data rate input
- Con: Not as efficient (throughput) for high data rates

Block ciphers

- operate on fixed length blocks of input (e.g., 256 bit)
- Well suited for packet-based communication
- Pro: Parallelization possible, higher throughput
- Con: Data has to fit blocks, padding required, lower efficiency

(Ideal) Stream ciphers



- Encryption and decryption require the same key
- Operations are completely symmetric
- Requires random symmetric key stream of same length as input
- Will frequency analysis of the ciphertext work, why?

Frequency analysis of ideal stream cipher

- Ideal key stream s has P(s_0)=P(s_1)=0.5, independently of p
- s_0,s_1,p_0,p_1 are random events (key bit,plaintext bit)

s	p	$c = s \oplus p$	P($\cdot$)
0	0	0	$P(s_0) * P(p_0) = 0.5 * P(p_0)$
1	0	1	$P(s_1) * P(p_1) = 0.5 * P(p_0)$
1	1	0	$P(s_1) * P(p_0) = 0.5 * P(p_1)$
0	1	1	$P(s_0) * P(p_1) = 0.5 * P(p_1)$

So, $P(c_0)$ is
$0.5 * P(p_0) + 0.5 * P(p_1) = 0.5 * (P(p_0) + P(p_1)) = 0.5$

- The initial distribution of plaintext is hidden in ciphertext

One-Time Pad

- Stream ciphers are very secure if long *random* key is available
  - It is impossible to recover the plaintext from ciphertext (even with infinite resources for attacker)
  - Key can only be used once
- This ideal cipher is called One-Time Pad
  - Has been used in practise, e.g. to encrypt "red" telephone line between Russia and US
- Problem: key as long as message, must be exchanged securely
  - Assumes secure channel to exchange key
  - Why not exchange message over that channel?

Why can't we brute-force OTP ciphertext?

- OTP is one of the few ciphers where brute force attacks are impossible

Why can't we brute-force OTP ciphertext?

- OTP is one of the few ciphers where brute force attacks are impossible

---

- Brute force search through 2^n keyspace will create 2^n potential plaintexts (all possible values)
- It is impossible to determine which one was the original plaintext

Why not re-use the key?

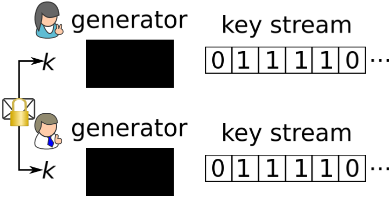
- We could be more efficient by encrypting twice with same key?
- Example: m_1 and m_2, key stream s. $c_1 = E(m_1, s)$ and $c_2 = E(m_2, s)$
- Problem?

Why not re-use the key?

- We could be more efficient by encrypting twice with same key?
- Example: m_1 and m_2, key stream s. $c_1 = E(m_1, s)$ and $c_2 = E(m_2, s)$
- Problem?

- As $E(m, s) = m \oplus s$,
  - $c_1 \oplus c_2 = (m_1 \oplus s) \oplus (m_2 \oplus s) = m_1 \oplus m_2$
- Bad if alphabet of m has some frequency distribution.
- Really bad if either m_1 or m_2 are known to attacker!

Generating the key stream from short key



- We need a way to generate a long *pseudo-random* sequence
- Both parties exchange the short key over secure channel
- Both parties then separately generate long key stream to de/encrypt
- Key stream must be unpredictable (generating function is public)

Conclusion

- Substitution ciphers and frequency analysis
- Transposition ciphers
- Substitution ciphers on binary data
- Stream ciphers vs. block ciphers