CSc 466/566

Computer Security

4 : Cryptography — Introduction

Version: 2012/02/06 16:06:39

Department of Computer Science
University of Arizona

collberg@gmail.com

Christian Collberg

Outline

Introduction

- In this section we introduce some classical symmetric ciphers.
- We also discuss various attacks against ciphers.

Outline

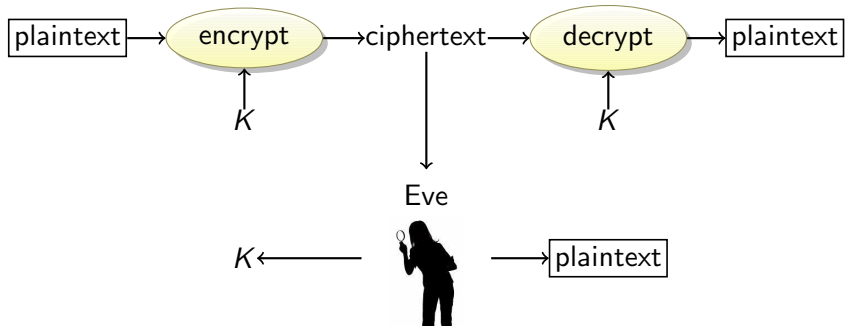
Attacks Against Cryptosystems

Definition (cryptanalysis)

The science of attacking cryptosystems.

- A cryptanalyst attacks cryptosystems.
- We assume the cryptanalyst knows the algorithms involved.
- He wants to discover plaintext or keys.

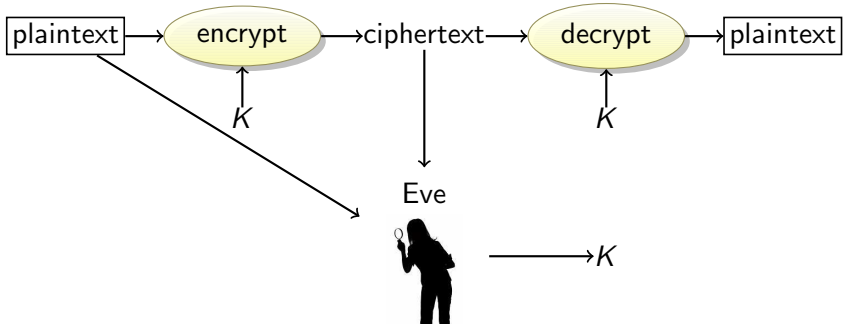
Ciphertext-only attack



We have: the ciphertext of several messages that have been encrypted with the same key, K.

We recover: the plaintexts, or K.

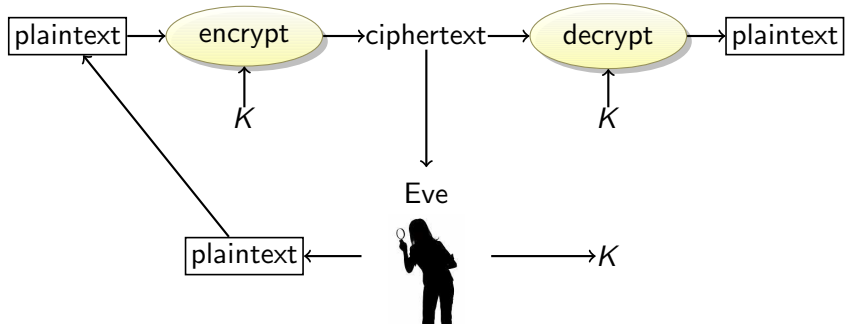
Known-plaintext attack



We have: the ciphertexts and corresponding plaintexts of several messages, all encrypted with the same key K.

We recover: the key K.

Chosen-plaintext attack

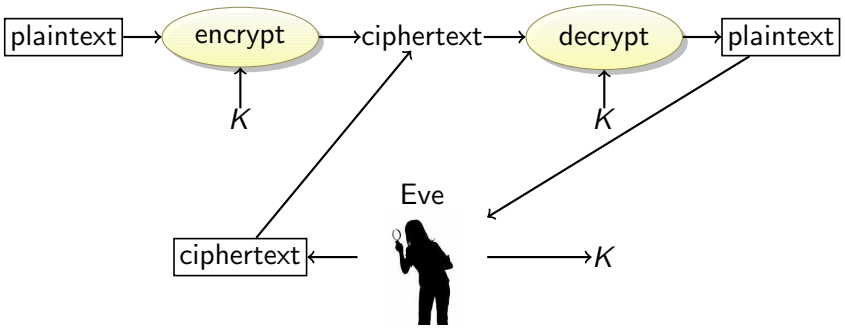


We have: the ciphertext of several messages that have been encrypted with the same key K, such that we get to choose the plaintexts.

We recover: the key K.

- offline chosen-plaintext attack: the attacker must choose all

Chosen-ciphertext attack



We have: the plaintext of several messages that have been encrypted with the same key K, such that we get to choose the ciphertexts.

We recover: the key K.

Offline vs. Adaptive Attacks

- There are two variants of the chosen-plaintext attack:
  - Offline chosen-plaintext attack: the attacker must choose all plaintexts in advance;
  - Adaptive chosen-plaintext attack: the attacker can choose one plaintext at a time, and choose plaintexts based on previous choices.
- Similar for the chosen-ciphertex attack.

Rubber-hose cryptanalysis

We have: access to a person who can be threatened, blackmailed, tortured,...

We recover: Everything!

- Also purchase-key attack.

How to Recognize Plaintext

- In a brute-force attack we try every possible key until we find the right one.
- How do we know that we've found the right key?
  - Well, when we get something out which is plaintext.
  - Well, how do we know that it is plaintext?
  - Because it looks like plaintext!
- Plaintext could be:
  - English, Russian, Chinese (many different encoding);
  - A Microsoft Word file;
  - A `gzip` compressed file, ...
- Binary files usually have headers that are easy to recognize.
- Generally, when you decrypt with the wrong key, you get gibberish, when you have the right key the plaintext looks reasonable.

Unicity Distance: How Much Ciphertext do We Need?

Definition (unicity distance)

The unicity distance is the amount of the original ciphertext required such that there is only one reasonable plaintext, i.e. the expected amount of ciphertext needed such that there is exactly one key that produces a plaintext that makes sense.

- The unicity distance depends on the
  1. characteristics of the plaintext
  2. the key length of the encryption algorithm.
- Unicity distance of
  - Standard English text: $K/6.8$, where K is the key length. (6.8 is a measure of the redundancy of ASCII English text).
  - DES: 8.2 bytes.
  - 128-bit ciphers: ≈ 19 bytes.

Unicity Distance: How Much Ciphertext do We Need?...

- RC4 encrypts data in bytes.
- Example 1:
  - Plaintex: a single ASCII letter (0-25).
  - Ciphertext: a single byte (0-255).
  - Attacker tries to decrypt a ciphertext byte with a random key.
  - He has a 26/256 chance of producing a valid plaintext.
  - There's no way for him to tell the correct plaintext from the wrong plaintext.
- Example 2:
  - Plaintext: a 1K e-mail message.
  - The attacker tries to decrypt with random keys.
  - Eventually there's a plaintext that looks like an e-mail.
  - The odds are small that this is not the correct plaintext!
- The unicity distance determines when you can think like the second example instead of the first.

In-Class Exercise: Goodrich & Tamassia R-8.1-4

What type of attack is Eve employing here:

1. Eve tricks Alice into decrypting a bunch of ciphertexts that Alice encrypted last month.
2. Eve picks Alice's encrypted cell phone conversations.
3. Eve has given a bunch of messages to Alice for her to sign using the RSA signature scheme, which Alice does without looking at the messages and without using a one-way hash function. In fact, these messages are ciphertexts that Eve constructed to help her figure out Alice's RSA private key.
4. Eve has bet Bob that she can figure out the AES secret key he shares with Alice if he will simply encrypt 20 messages for Eve using that key. Bob agrees. Eve gives him 20 messages, which he then encrypts and emails back to Eve.

Outline

Substitution Ciphers

Definition (Substitution Cipher)

A method of encryption by which units of plaintext are replaced with ciphertext according to a regular system.

- The units can be single letters, pairs of letters, triplets of letters.
- The goal is confusion: ciphertext bits should depend on the cleartext bits in a very complex way.
- Easily broken: underlying letter frequencies are not hidden.
- The letter **E** occurs the most frequently in English.
- The letter in the ciphertext that occurs most often $\Rightarrow$ probably **E**!

English Letter Frequency

Letter	Frequency
E	12.02%
T	9.10%
A	8.12%
O	7.68%
I	7.31%
N	6.95%
S	6.28%
R	6.02%
H	5.92%
D	4.32%
L	3.98%
U	2.88%
C	2.71%

Letter	Frequency
M	2.61%
F	2.30%
Y	2.11%
W	2.09%
G	2.03%
P	1.82%
B	1.49%
V	1.11%
K	0.69%
X	0.17%
Q	0.11%
J	0.10%
Z	0.07%

http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

Monoalphabetic Substitution Ciphers

- In a monoalphabetic cipher each character of the plaintext is mapped to a corresponding character of the ciphertext:

$$A \to 9, B \to 11, \ldots$$

Caesar Cipher: Add 3 to the ASCII value of each character, mod 26:

$$A \to D, B \to E, X \to A, \ldots$$

ROT13: Unix utility used on Usenet. Adds 13 mod 26 to each letter.

$$P = \mathrm{ROT13}(\mathrm{ROT13}(P))$$

- These methods are simple to break: use the fact that different letters in the English alphabet occur with different frequencies.

Encoding

- In these simple ciphers we typically
  1. convert all letters to upper case;
  2. remove spaces;
  3. remove punctuation;
  4. break into blocks of the same size (typically 5 letters);
  5. add some unusual letter (like Z) to the last block, if necessary.
- Example:

```
It wAs A DArk and sTormY NighT ...
```

turns into

```
ITWAS ADARK ANDST ORMYN IGHTZ
```

- Knowing word boundaries can help with cryptanalysis.

Homophonic Substitution Ciphers

- In a homophonic cipher each character of the plaintext is mapped to several characters of the ciphertext:

$$A \to \{9, 10, 11\}, B \to \{3, 1, 8\}, \ldots$$

- Address the letter-frequency attack that can be used against monoalphabetic ciphers.
- Assign each plaintext letter a set of symbols proportional to it's frequency.
- For example,
  - E → 14, 16, 24, 44, 46, 55, 57, 64, 74, 82, 87, 98
  - H → 23, 39, 50, 56, 65, 58
  - L → 26, 37, 51, 84
  - O → 00, 05, 07, 54, 72, 90, 99
  - Z → 02
- Notice E maps to a lot more symbols than Z

Polyalphabetic Substitution Ciphers

- In a polyalphabetic cipher you have several keys, each one used to encrypt one letter of the plaintext. We recycle keys when we run out of them:

K_1	K_2	K_3	K_1	K_2	K_3	K_1	K_2	K_3
a	*t*	*t*	*a*	*c*	*k*		*a*	*t*
x	*v*	*d*	*x*	*t*	*d*	*r*	*p*	*d*

  The number of keys is called the *period*.
- In a running-key cipher (AKA book cipher) one text is used to encrypt another.

Polygraphic Substitution Ciphers

- In a polygram cipher blocks of characters in the plaintext are mapped to blocks of characters in the ciphertext:

$$\texttt{ARF} \to \texttt{RTW}, \texttt{ING} \to \texttt{PWQ}, \ldots$$

- We represent the cipher with a Substitution Box (S-Box):

	A	**B**	**C**	**D**	**E**	**F**
A	BA	CA	DC	DD	DE	FB
B	EA	AB	EC	BD	BE	AF
C	AA	BB	AC	ED	CE	BF
D	EB	DB	BC	CD	DF	FC
E	DA	CB	CC	AD	AE	FF
F	FA	CF	EE	FD	EF	FE

- Examples: `AA` → `BA`, `AB` → `CA`, `EF` → `FF`

Polygraphic Substitution Ciphers: Playfair

- Create a jumbled 5 x 5 square of jumbled letters:

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

- Convert letters a pair at a time: TI → RF, TW → VC
- To use in the heat of battle we want it to be simple to
  1. generate the table;
  2. memorize the table;
  3. encrypt/decrypt.

Polygraphic Substitution Ciphers: Playfair...

- How do we create the table (the cipher key)?
  1. Select a key phrase;
  2. Fill in the spaces of the table, starting top left (omitting duplicate letters), with the letters from the key phrase;
  3. Fill in the remaining spaces with the remaining letters of the alphabet, in order.
- Omit **Q** to make the alphabet fit, or merge **I/J** into one entry.
- Example (key phrase: DIAMONDRING):

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

DIAMONDRING

Alphabet: ABCDEFGHIJKLMNOPRSTUVWXYZ

Polygraphic Substitution Ciphers: Playfair...

- To encrypt, start by breaking the message into digraphs:

> It wAs A DArk and sTormY NighT ...

turns into

> IT WA SA DA RK AN DS TO RM YN IG HT

- We use the two letters of the digraph to create a rectangle in the key table.

Polygraphic Substitution Ciphers: Playfair...

- Rules to encrypt the digraph $\alpha\beta$:
  1. If $\alpha = \beta$, add an **X**, encrypt the new pair.
  2. If one letter is left, add an **X**, encrypt the new pair.
  3. If α, β are in the same row:

*	*	*	*	*
*	*	*	*	*
α	X	*	β	Y
*	*	*	*	*
*	*	*	*	*

$\Rightarrow \quad \alpha\beta \to XY$

     If necessary, wrap around.
  4. If $\alpha\beta$ occur in the same column:

*	*	*	*	*
*	*	α	*	*
*	*	X	*	*
*	*	β	*	*
*	*	Y	*	*

$\Rightarrow \quad \alpha\beta \to XY$

Polygraphic Substitution Ciphers: Playfair...

- And the final rule:
  5. If the letters are not on the same row or column:

X	*	*	α	*
*	*	*	*	*
*	*	*	*	*
β	*	Y	*	*
*	*	*	*	*

$\Rightarrow \quad \alpha\beta \to XY$

     Order matters: X is on the same row as α.
- To decrypt:
  1. Use the inverse of the last three rules.
  2. Drop any **X**s that don't make sense.

Polygraphic Substitution Ciphers: Playfair...

- Example plaintext:

IT WA SA DA RK AN DS TO RM YN IG HT

- IT→MP

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

- WA→XI

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

Polygraphic Substitution Ciphers: Playfair...

- SA→XG

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

- DA→IM

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

In-Class Exercise

1. Construct a Playfair table using the key phrase BLINKENLIGHTS.
2. Encode the message Run, RAbbit, Run!
3. Encrypt the plaintext message from 2.
4. Decrypt the ciphertext message from 3.

Outline

Transposition Ciphers

Definition (Transposition Cipher)

A method of encryption by which units of plaintext are rearranged to form the ciphertext.

- In a transposition cipher the original characters of the plaintext are not changed, but simply moved around in the ciphertext.
- Letter frequencies don't change.
- The ciphertext is a permutation of the cleartext.
- The goal is diffusion: spreading the information from the plaintext across the ciphertext.

Columnar Transposition Cipher

- In a simple columnar transposition cipher we write the plaintext horizontally in a fixed width table, and read it off vertically.
- The plaintext `attack␣at␣dawn` could be enciphered into `actwtk␣nt␣f␣aaa␣`, using this table:

```
a t t a
c k   a
t   d a
w n
```


Outline

S-Boxes

- We can extend the substitution box idea to binary words.
- Here's a 4×4 S-box that maps 4 bits to 4 bits:

S	**00**	**01**	**10**	**11**
00	0011	1000	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

S	**0**	**1**	**2**	**3**
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

- Examples: $0000 \rightarrow 0011$, $0001 \rightarrow 0100$, $1010 \rightarrow 0100$

Inverse S-Boxes

- If S is an S-box with unique substitutions there exists an inverse S-box S^{-1} that reverses the substitution:

S	**00**	**01**	**10**	**11**
00	0011	1000	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

⇓

S^{-1}	**00**	**01**	**10**	**11**
00	1101	0011	1011	0000
01	1010	0110	0101	1100
10	0001	1110	0100	0111
11	1111	1001	1000	0010

S	**0**	**1**	**2**	**3**
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

⇓

S	**0**	**1**	**2**	**3**
0	13	3	11	0
1	10	6	5	12
2	1	14	4	7
3	15	8	9	2

Inverse S-Boxes. . .

- Examples:

$$0000 \xrightarrow{S} 0011 \xrightarrow{S^{-1}} 0000$$
$$1111 \xrightarrow{S} 1100 \xrightarrow{S^{-1}} 1111$$
$$1010 \xrightarrow{S} 0100 \xrightarrow{S^{-1}} 1010$$

- Desirable properties of S-boxes:
  1. changing one input bit ⇒ about half of the output bits will change (avalanche effect);
  2. each output bit will depend on every input bit.

P-Boxes

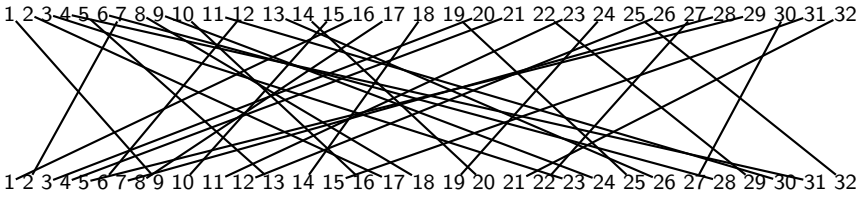
- We can extend the transposition cipher idea to binary words.
- Here's a 32-bit P-box that is used by the DES cipher:

P	**moved to position**							
1-8	9	17	23	31	13	28	2	18
9-16	24	16	30	6	26	20	10	1
17-24	8	14	25	3	4	29	11	19
25-32	32	12	22	7	5	27	15	21

P-Boxes. . .

P	**moved to position**							
1-8	9	17	23	31	13	28	2	18
9-16	24	16	30	6	26	20	10	1
17-24	8	14	25	3	4	29	11	19
25-32	32	12	22	7	5	27	15	21

⇕



Product Ciphers

- In product ciphers we achieve both diffusion and and confusion by chaining together S-Boxes and P-Boxes.

Outline

One-Time Pads

- The pad is a large, non-repeating set of random key letters.
- To encrypt, add each plaintext letter to the next letter on the pad, mod 26. Decryption is done the same.
- This is provably secure, provided you have a truly random set of pad letters and never reuse the pad.
- Two problems:
  1. We need an infinite number of never-repeating keys;
  2. Alice and Bob need to be absolutely synchronized (at all times know which key they're using).

One-Time Pads: Example

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

pad :	*r*	*a*	*n*	*d*	*o*	*m*	*p*	*a*	*d*
	↓	↓	↓	↓	↓	↓	↓	↓	↓
numeric pad :	18	1	14	4	15	13	16	1	4
cleartext :	*a*	*t*	*t*	*a*	*c*	*k*	*a*	*t*	*d*
	↓	↓	↓	↓	↓	↓	↓	↓	↓
numeric cleartext :	1	20	20	1	3	11	1	20	4
	↓	↓	↓	↓	↓	↓	↓	↓	↓
add mod 26 :	19	21	8	5	18	24	17	21	8
ciphertext :	*s*	*u*	*h*	*e*	*r*	*w*	*q*	*u*	*h*

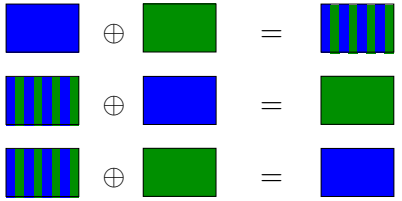
Exclusive-OR

$$\begin{array}{lllll} 0 & \oplus & 0 & = & 0 \\ 0 & \oplus & 1 & = & 1 \\ 1 & \oplus & 0 & = & 1 \\ 1 & \oplus & 1 & = & 0 \end{array} \quad \Bigg| \quad \begin{array}{lllllll} a & \oplus & a & & & = & 0 \\ a & \oplus & b & \oplus & b & = & a \\ a & \oplus & a & \oplus & a & = & a \end{array}$$

- Since xor-ing the same value twice gives us the original, we get a simple symmetric algorithm:

$$\begin{aligned} P \oplus K &= C \\ C \oplus K &= P \end{aligned}$$

Exclusive-OR in Sparkling Color



Pseudo-Random Number Generator (PRNG)

- A PRNG is seeded with a key K and generates a sequence of numbers such that
  - numbers are in the range $[0, n-1]$ for some $n > 0$;
  - the numbers are uniformly distributed;
  - having seen numbers $x_0, x_1, \ldots, x_i$ it's hard to predict x_{i+1}.
- Cryptographic PRNGs can be constructed from symmetric ciphers such as AES:
  1. Let K be the seed;
  2. $R \leftarrow E_{\text{AES}}(K)$
  3. Output R
  4. $K{+}{+}$
  5. Goto 2

Encryption with PRNG

- Let
  - key: K
  - plaintext message: $\langle M_0, M_1, M_2, \ldots \rangle$
  - ciphertext: $\langle C_0, C_1, C_2, \ldots \rangle$
  - sequence of pseudo-random numbers: $\langle P_0, P_1, P_2, \ldots \rangle$
- Encryption algorithm:
  1. Seed the PRNG with K;
  2. $C_i = M_i \oplus P_i$
- Decryption algorithm:
  1. Seed the PRNG with K;
  2. $M_i = C_i \oplus P_i$
- Make sure that:
  1. Only perform one encryption for a given key K.
  2. The length of the plaintext should be much smaller than the period of the PRNG.

Outline

Readings and References

- Chapter 8.1.1-8.1.5 in *Introduction to Computer Security*, by Goodrich and Tamassia.

Acknowledgments

Additional material and exercises have also been collected from these sources:

1. Igor Crk and Scott Baker, *620—Fall 2003—Basic Cryptography*.
2. Bruce Schneier, *How to Recognize Plaintext*, http://www.schneier.com/crypto-gram-9812.html#plaintext.
3. Pfleeger and Pfleeger, *Security in Computing*.