Lecture 1

Basic Definitions

Definition 1.1 (plaintext, cleartext, in clear). The message or data to be encoded. Usually written in lowercase when alphabetic.

Definition 1.2 (encipher, encrypt). To render plaintext unintelligible except to the intended recipient.

Definition 1.3 (cipher). A particular method of encryption.

Definition 1.4 (ciphertext, cryptogram). The message after encryption. Usually written in uppercase when alphabetic.

Example 1.1.
Plaintext: I came, I saw, I conquered.

Cipher: Take each letter of plaintext and substitute the third subsequent letter of the alphabet, cycling from z to a.

Ciphertext: L FDPH, L VDZ, L FRQTXHUHG.

Note: This cipher is one of the oldest ciphers known. It is usually referred to as the "Julian" or "Caesar" cipher, because it was first recorded as being used by Julius Caesar.

The General Case

Suppose a transmitter generates a plaintext message M, which is to be communicated to a legitimate receiver over an insecure channel. To prevent an eavesdropper from learning the contents of M, the transmitter enciphers M with an *invertible* transformation E_k to produce cipertext $C = E_k(M)$.

C is sent along the insecure channel. When the receiver obtains C, he deciphers it by applying the inverse transformation D_k to C to obtain $M = D_k(C)$.

Note. $D_k E_k = I$, the *identity transformation*, and D_k is a *left inverse* of E_k (i.e., $D_k = E_k^{-1}$)

Note. The *cipher* or *cryptosystem*, E, defines a family of related transformations, indexed by k, the *cryptographic key*, and E_k belongs to that family.

We assume that E, the cryptographic system, is not secret, but that k, the key, *is* secret.

Definition 1.5 (Cryptosystem). A single-parameter family, denoted $\{E_k\}_{k\in\mathcal{K}}$, of invertible transformations

$$\begin{aligned} E_k &: \mathcal{M} \to \mathcal{C} \\ &M \mapsto E_k(M) = C \quad (M \in \mathcal{M}, C \in \mathcal{C}), \end{aligned}$$

where E_k acts on a *message-space* $\mathcal{M}$ of plaintext messages, and injects it into a *cipher-space* $\mathcal{C}$ of ciphertext messages, where the parameter or key k is selected from a finite set $\mathcal{K}$ called the *key space*.

Example 1.2 (The general Caesar cipher).
Plaintext: $m_1 m_2 m_3 \dots m_n$

Each m_i is the numerical equivalent of a letter of the alphabet

0	1	2	3	...	25
a	b	c	d	...	z

Ciphertext: $E_k(m_1)E_k(m_2)E_k(m_3)\dots E_k(m_n)$

$E_k(m_i) = (m_i + k) \bmod 26$ (represented as a letter of the alphabet)

$\mathcal{K} = \{0, 1, 2, \dots, 25\}$, $|\mathcal{K}| = 26$

(For the Caesar cipher, $k = 3$)

If you know the Caesar cipher is being used, you can solve it by a "brute force attack" simply by trying each key in turn.

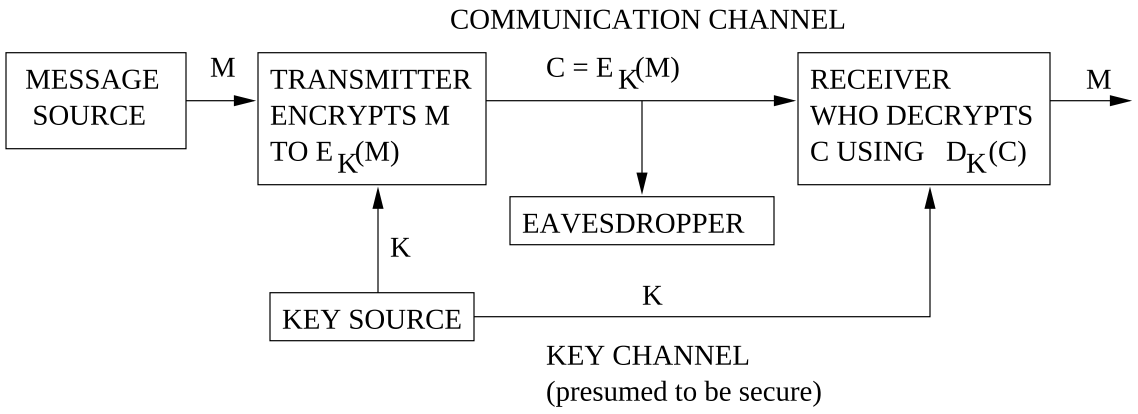
- since $|\mathcal{K}| = 26$, this won't take long
- this system is said to have a *small keyspace*.

How small is "small?" With modern technology, $10^{17} \approx 2^{56}$ is small (DES has $|\mathcal{K}| = 2^{56}$)

Lecture 2

Requirements for Cryptosystems

Schematic of a conventional (one-key or symmetric) cryptosystem


COMMUNICATION CHANNEL
MESSAGE SOURCE
M
TRANSMITTER ENCRYPTS M TO E K(M)
C = E K(M)
RECEIVER WHO DECRYPTS C USING DK(C)
M
EAVESDROPPER
K
KEY SOURCE
K
KEY CHANNEL (presumed to be secure)

Note. In order for the encryption to be effective, the *key channels* must be absolutely secure, as must the channel from the source to the transmitter. In the real world, this usually means expensive. For example, the keys to the Moscow-Washington hotline are transmitted by means of highly paid couriers, who fly there and back every week.

It would be nice to dispense with the key channel. Why bother encrypting when we have a secure channel already?

- cost
- convenience
- speed
- bandwidth

First General Principle It is generally assumed in cryptography that the knowledge of the general system or algorithm of encryption, $\{E_k\}_{k\in\mathcal{K}}$, is known to the eavesdropper. Only the key k is assumed not to be in his/her possession.

Shannon's desiderata for a "good" cryptosystem

Shannon 1940: A good cryptosystem should:

1. Be highly secure against it's designer, i.e., the designer of the system should not have any advantage in decryption;
2. Use a short, easily-changed key;
3. Require only simple encryption/decryption;
4. Not introduce error propagation;
5. Not entail message expansion.

Note. 2–5 were included because encryption used to be done by low-paid cipher clerks, and mistakes often happened. Since computers were applied to this purpose, these are less important.

Modern desiderata

1. As before
2. As before — key must be small (but small can be many digits long, eg., for storage on a magnetic card)
3. As before — operations must be simple (what computers do best), but we can do many of them
4. Error propagation is now desirable: since we can assume a computer won't make an error, we want the enemy to get hit with error propagation
5. Message expansion is now acceptable (in moderation!), because of the lower cost of transmission.

Lecture 3

Cryptanalysis

Definition 3.1 (Cryptanalysis). The process by which an unauthorized receiver of a ciphertext $E_k(M)$ determines M without prior knowledge of the key k.

Definition 3.2 (Cryptanalyst). The opponent of a cryptosystem — an unauthorized receiver who practices the "black art" of cryptanalysis.

Objectives of the cryptanalyst:

1. *Passive threat*: listening and reading the transmissions
2. *Active threat*: forgery, sending false transmissions, altering or deleting transmissions

Definition 3.3 (Privacy system). A system that protects against the unauthorized extraction of information from a public channel.

Definition 3.4 (Authentication system). A system that protects against the unauthorized modification or insertion of information in a public channel.

Types of Cryptanalytic Attack

Ciphertext Only Attack (COA)

The cryptanalyst possesses only the ciphertext.

The cryptanalyst may also have a *frequency distribution*, a statistical analysis of the relative frequencies of letters in similar text. For example:

- E occurs about 13% of the time in English
- THE is the most frequent 3-letter group in English

However, the cryptanalyst's knowledge of the plaintext is at best partial.

Known Text Attack (KTA)

The cryptanalyst has *some* corresponding plaintext and ciphertext, and must find the encryption key. For example:

- Diplomatic proposal: it is known that such proposals are usually sent back by the foreign diplomat to her capital word for word in the original language. If you know which ciphertext corresponds to which diplomatic proposal, you have knowledge of some corresponding plaintext and ciphertext.
- Timed press release: big corporations often wish a press release to be issued simultaneously world-wide, so they transmit it to their offices in encrypted form, together with information on when to decode it. If you have a copy of this ciphertext, you can (perhaps) match it up with the press release.

Chosen Text Attack (CTA)

The same as KTA, but the cryptanalyst is given plaintext and corresponding ciphertext *of his own choosing.* For example:

- Diplomatic proposal: if you take an unexpected action, you can be reasonably certain that it will be reported by the foreign diplomat, and you can probably manipulate the form of the report.

Definition 3.5 (Certified secure). A cryptosystem is *certified secure* when it has withstood a concerted attack by cryptanalysts under conditions favorable to them (KTA, CTA) for a long period of time up to the present. (Note: this is *not* a precise definition, rather functional)

Means of Cryptographic Attack

1. mathematical
   - statistics (frequency distributions, etc.)
   - number theory
   - group theory
   - combinatorics (and graph theory)
   - information theory
   - etc...
2. side information
   - linguistic (human language is full of redundancy)
   - formatting ("Dear John," etc...)
   - subject
   - known words or phrases (eg. military rank)
3. clandestine
   - theft
   - bribery
   - blackmail
   - sex and violence

Lecture 4

Substitution Ciphers

Classical Ciphers

These will be used as a paradigm for cryptanalysis. They fall into two categories:

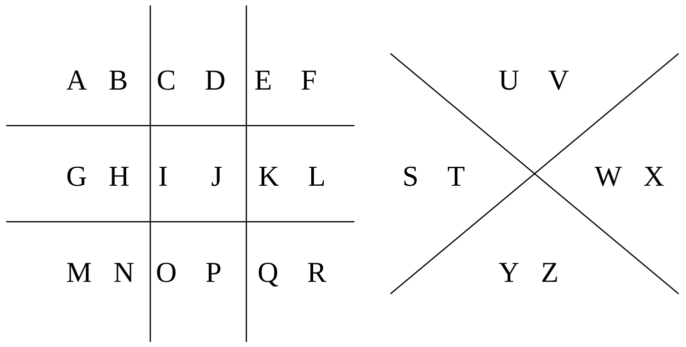
1. Substitution: the most familiar
2. Transposition: less familiar, but older

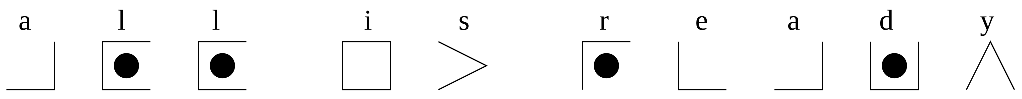
Definition 4.1 (Substitution Cipher). A *substitution cipher* is a cipher in which we replace each character of plaintext by a substitution cipher symbol: these symbols occur in the same order as the corresponding character(s) in the message.

Example 4.1. Julian (Caesar) cipher (already discussed)

Example 4.2. *Pig Pen cipher.* This cipher was actually used in the U.S. Civil War, but it is in fact even less secure than the Caesar cipher.

Each character of plaintext is replaced by the lines surrounding it, plus a dot if the plaintext character is the second character in the box.





The Pig Pen cipher depends entirely upon the *method*, and *not the key*.

Example 4.3. *Monoalphabetic Substitution cipher.* Each letter of the plaintext is replaced by a letter of a *single* substitution alphabet (eg. cryptograms and cryptoquotes from the newspaper). Note that both the Caesar and Pig Pen ciphers are monoalphabetic.

One method for construction monoalphabetic substitution ciphers

```
1. Take the key:                Dan Roscoe
2. Eliminate duplicates:        DANROSCE
3. Add balance of alphabet:     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
                                D A N R O S C E B F G H I J K L M P Q T U V W X Y Z
```

If this were the cipher, we would have seven collisions (letters that encrypt to themselves). In order to reduce this number, we can reorganize the cipher alphabet

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

and construct the cipher from the columns:

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

We now have four collisions, and using other rearrangements it is possible to reduce this even further.

We encode:

```
come at once attack commencing
MOHA DV OTMA DVVDMQ MOHHATMNTP
```

Definition 4.2 (Informal Cipher). ciphertext has the same spacing and punctuation as the plaintext

Informal ciphers give the cryptanalyst a lot of information about the plaintext via lengths of words and punctuation. In order to keep this information from the cryptanalyst, *formal ciphers* are typically used.

Definition 4.3 (Formal Cipher). ciphertext contains no punctuation, symbols appear in groups of five

Example 4.4. The previous example as a formal cipher:

```
MOHAD VOTMA DVVDM QMOHH ATMNT PXUVW
```

Note the addition of four *nulls* at the end. These are used to pad the final group of ciphertext symbols to five characters.

Security of Monoalphabetic Substitution Ciphers

Note that we have $|\mathcal{K}| = 26! \approx 4 \times 10^{26} \approx 2^{89}$, so exhaustive search is currently infeasible. Nevertheless, these ciphers are in general completely insecure.

Weaknesses:

1. highly vulnerable to KTA. Each bit of corresponding plaintext and ciphertext reveals some of the cipher.
2. Redundancy in any language generally yields the key with sufficient ciphertext. This is because the frequency distribution of the plaintext alphabet in a given language can be established statistically and compared with the ciphertext.

Note. In order to use frequency distributions, it is important that there be sufficient ciphertext. If the ciphertext is too short, it may not be possible to cryptanalyze solely using redundancy. In fact, there are still examples of ciphertexts that have yet to be broken (often allegedly giving the location of buried treasure).

Definition 4.4 (n-gram, n-graph). a sequence of n letters

The following table contains a sample of expected frequency distributions based on military text (taken from "Manual for the Solution of Military Cipers" by Parker Hitt).

Table 4.1: Frequencies of letters, digraphs, and trigraphs in English

Single Letters (based on 10000 letters)		**Digraphs** (based on 2000 letters)		**Trigraphs** (based on 10000 letters)	
E	1277	TH	50	THE	89
T	855	ER	40	AND	54
O	807	ON	39	THA	47
A	778	AN	38	ENT	39
N	686	RE	36	ION	36
I	667	HE	33	TIO	33
R	651	IN	31	FOR	33
S	622	ED	30	NDE	31
H	595	ND	30	HAS	28
D	402	HA	26	NCE	27
L	372	AT	25	EDT	27
U	308	EN	25	TIS	25
C	296	ES	25		
M	288	OF	25		
P	223	OR	25		
F	197	NT	24		
Y	196				
W	176				
G	174				
B	141				
V	112				
K	74				
J	51				
X	27				
Z	17				
Q	8				

Note. Most frequent pairs are high frequency vowels with high frequency consonants.

Note. Repeated letters can also give us a lot of information. For example, the vowels O and E are commonly repeated, while A, I, and U are rarely repeated, if ever.

Lecture 5

Example Cryptanalysis and Codes

Example 5.1. Consider the following message from Edgar Allen Poe's "The Gold Bug." The original was written with strange symbols, which have been converted to letters below:

```
DYZZB  YQDRR  FPGAM  HFRAZ  TRAZR

GMQFP  GAMBM  OFQRR  MDGGI  MPGUZ

PMBMY  NMMRD  PBGAF  NGMMP  SFPJG

MRPZN  GAMDR  GDPBH  UPZNG  ASDFP

HNDPV  ARMOM  PGAQF  SHMDR  GRFBM

RAZZG  CNZSG  AMQMC  GMUMZ  CGAMB

MDGAR  AMDBD  HMMQF  PMCNZ  SGAMG

NMMGA  NZJYA  GAMRA  ZGCFC  GUCMM

GZJGX
```

This ciphertext yields the following frequency distribution:

Single Letters		**Digraphs**		**Trigraphs**		**Double Letters**	
M	34	GA	12	GAM	7	MM	5
G	27	AM	8	RAZ	4	ZZ	2
A	19	RA	5	PGA	3	RR	2
R	16	FP	5			GG	1
Z	15	MD	5				
P	14	MM	5				
D	13	GM	4				
F	11	PG	4				

Based on our frequency counts, we suspect GAM = the, and the rest of the cryptanalysis follows relatively easily.

5.1 Codes

We could try to get around the weaknesses of monoalphabetic substitution ciphers by having several different symbols for frequently occurring words, letters, etc... (called *homophones*).

We could also have special symbols for frequently occurring words and special names. This was done in the period 1400-1850 (especially diplomatic codes). Such lists of symbols, words, and names are called *nomenclators*, and an encryption scheme based on them is called a *code*.

A *code* consists usually of thousands of words, phrases, letters, and symbols with the code-words or code numbers (code groups) that replace these elements. Note that a code can be considered as a very large cipher alphabet, with "symbols" of varying length.

There are two types of codes:

1. *1 part code*: can be used to both encode and decode. Both plaintext and ciphertext symbols are in lexicographic order. Example:

`ship`	`MXTWI`
`ship is`	`MXTXA`
`ship is not`	`MXTXC`
`ship is not to`	`MXTYJ`
`ship is not to be`	`MXUAA`

2. *2 part code*: one book used to encode, another to decode. The encoding book has the plaintexts in order, and the decoding book has the ciphertexts in order. Both books contain exactly the same pairs of plaintext/ciphertext — only the order is different (in order to aid encoding and decoding). Example:

Part 1		**Part 2**	
`ship`	`UPUFO`	`DAFET`	`will be`
`ship is`	`BECEC`	`DAFEW`	`26 October`
`ship is not`	`ATOHI`	`DAFEX`	`New York`
`ship is not to`	`BUHAG`	`DAFEY`	`null`
`ship is not to be`	`ZENDA`		

Advantages:

1. frequency distibutions of single characters, digraphs, etc. are not useful

Weaknesses:

1. code books can be stolen. Eg. British navy recoving code books from sunken German submarine in WWII.
2. essentially a monoalphabetic substitution cipher with very large plaintext alphabet (same weaknesses: frequency distibution and KTA). Eg. code 0075 (2 part, 10000 code words) and the Zimmerman telegraph.

Codes are no longer considered secure. However, they are and have been frequently used for *compression* (eg. telegraph). *Superencipherment* is typically used for security — simply encrypt the ciphertext which is obtained from applying the code.

Lecture 6

n-gram Encipherment

Enciphering several letters at a time rather than single letters is known as *n-gram encipherment* — we encipher n letters as a group. This smooths out the frequency counts that are useful in breaking monoalphabetic systems.

Consider digraphic encipherment ($n = 2$):

Plaintext	**Monoalphabetic**	**Digraphic**
se	RM	ER
ae	DM	FI

Note that in the monoalphabetic cipher, e always encrypts to M, whereas it varies in the digraphic cipher.

Advantages (digraphic):

1. need more text to get useful frequency data
2. larger plaintext alphabet (676 digraphs as opposed to only 26 letters)

6.1 The Playfair Cipher

- 1854 - Charles Wheatstone, named after his friend Baron Playfair
- digraphic cipher
- possibly used in the Boer War by the British
- originally considered too difficult for attachés to use!

1. Select a keyword to rearrange the alphabet (as usual, drop any duplicate letters)
   Keyword: MAGNETIC

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

2. Drop "J" from our initial cipher alphabet (so I and J will be considered as the same letter). Arrange the remaining 25 letters in a 5x5 square:

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

3. Insert an “x” (i.e. a null) between double plaintext letters. Note that “xx” never occurs in English, so this is safe.

```
Lord Granvilles letter
Lo rd Gr an vi lx le sl et xt er
```

4. Encrypt pairs of letters (digrams) as follows:

   (a) The letters are in the same column of the square: replace them by the letters immediately below them. Cycle as needed.

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

   (the digraph “rd” encrypts to “UR,” “cz” encrypts to “PH”)

   (b) The letters are in the same row of the square: replace them by the letters immediately to their right. Cycle as needed.

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

   (the digraph “lo” encrypts to “WX,” “ne” encrypts to “HR”)

   (c) Otherwise: replace each letter by that which lies in its row and stands in the column occupied by the other plaintext letter. The letter in the row of the first plaintext letter is taken first to preserve order.

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

   (the digraph “gr” encrypts to “DS,” “ty” encrypts to “VB”)

Thus, we have:

```
Lord Granvilles letter
lo rd gr an vi lx le sl et xt er
WX UR DS BE IU WL XR RO NI WI RN
WXURD SBEIU WLXRR ONIWI RNXAC
```

(Note the nulls “XAC”)

To decrypt, construct the same grid (based on the keyword!) and use Step 4, modified to reverse a. and b.

Lecture 7

Polyalphabetic Substitution Ciphers

7.1 General n-grams and the Hill Cipher

The Playfair cipher is an example of a digraphic encipherment technique. One type of general n-gram encipherment (for arbitrary n) is called the *Hill cipher*.

1929 — Lester S. Hill (math professor in New York)

The Hill cipher is based on linear algebra. Treat each n-gram as a vector (numerical equivalents of plaintext letters). Set up a linear transformation matrix and multiply to produce the ciphertext.

Unfortunately, the Hill cipher is vulnerable to attacks based on linear algebra.

No practical use (too complicated, too much error propagation), but important in establishing the link between cryptography and mathematics.

Hill cipher:

1. Assign a number to each letter (a=0, b=1, etc...)
2. Chose an enciphering matrix $E_{n\times n}$ (which will function as the key) such that all entries are integers and
$$\gcd(|E|, L) = 1,$$
where $|E| = \det(E)$ and L is the size of the plaintext alphabet (in our case, $L = 26$).
3. Let D be a matrix such that
$$DE \equiv I \pmod{L}$$
(i.e., D is a left inverse of E mod L)
Note that since we know there exists x such that $x|E| \equiv 1 \pmod{L}$ (assuming $\gcd(|E|, L) = 1$), we can construct
$$D = x \cdot \operatorname{adj}(E),$$
and since $\operatorname{adj}(E)/|E| = E^{-1}$ and $x \equiv |E|^{-1} \pmod{L}$
$$D = x \cdot \operatorname{adj}(E) \equiv E^{-1} \pmod{L} \ .$$
4. To encrypt an n-gram, let x_i be the numerical equivalent of the ith letter of $\vec{x} = (x_1, x_2, \ldots, x_n)$, the plaintext n-gram. Compute the ciphertext n-gram $\vec{y} = (y_1, y_2, \ldots, y_n)$ using
$$E\vec{x} \equiv \vec{y} \pmod{L} \ .$$

5. To decrypt:
$$D\vec{y} \equiv DE\vec{x} \equiv I\vec{x} \equiv \vec{x} \pmod{L} .$$

Example 7.1. Let $n = 2$ ($L = 26$) and
$$E = \begin{pmatrix} 7 & 9 \\ 3 & 12 \end{pmatrix}, \quad |E| = 57$$

Plaintext:

$$\begin{array}{cccccccc} \text{g} & \text{o} & \text{o} & \text{d} & \text{t} & \text{i} & \text{m} & \text{e} \\ 6 & 14 & 14 & 3 & 19 & 8 & 12 & 4 \end{array}$$

To encrypt "go:"

$$\vec{x} = \begin{pmatrix} \text{g} \\ \text{o} \end{pmatrix} = \begin{pmatrix} 6 \\ 14 \end{pmatrix}$$
$$E\vec{x} = \begin{pmatrix} 7 & 9 \\ 3 & 12 \end{pmatrix} \begin{pmatrix} 6 \\ 14 \end{pmatrix} \equiv \begin{pmatrix} 12 \\ 4 \end{pmatrix} = \begin{pmatrix} \text{M} \\ \text{E} \end{pmatrix}$$

To encrypt "od:"

$$\vec{x} = \begin{pmatrix} \text{o} \\ \text{d} \end{pmatrix} = \begin{pmatrix} 14 \\ 3 \end{pmatrix}$$
$$E\vec{x} = \begin{pmatrix} 7 & 9 \\ 3 & 12 \end{pmatrix} \begin{pmatrix} 14 \\ 3 \end{pmatrix} \equiv \begin{pmatrix} 21 \\ 0 \end{pmatrix} = \begin{pmatrix} \text{V} \\ \text{A} \end{pmatrix}$$

Ciphertext:

$$\begin{array}{cccccccc} 12 & 4 & 21 & 0 & 23 & 23 & 16 & 6 \\ \text{M} & \text{E} & \text{V} & \text{A} & \text{X} & \text{X} & \text{Q} & \text{G} \end{array}$$
MEVAX XQGIL

(Note the nulls "IL")

Note that in this case
$$D = \begin{pmatrix} 18 & 19 \\ 15 & 17 \end{pmatrix}$$

Weaknesses of Playfair and Hill

1. Playfair: solved by using sufficient data in order that repetition of letter pairs can be used to reconstruct the square. Frequency counts are used and also the fact that if "ab" encrypts to "XY," then "ba" encrypts to "YX."

2. Hill: solved by using n-gram frequency distributions and linear algebra manipulation.

Both the Playfair and Hill ciphers fall almost immediately to a KTA.

Polyalphabetic Substitution Ciphers

Frequency statistics are smoothed out even further using a *polyalphabetic* substitution cipher.

Definition 7.1 (polyalphabetic substitution cipher). a cipher in which several cipher alphabets are used in the replacement of the plaintext characters.

Example 7.2. Suppose we use the key "BLACKSTONE." We produce a sequence of numbers from the key by writing the numbers representing the order in which the letters of the key would appear in the alphabet.

Here we have

B	L	A	C	K	S	T	O	N	E
2	6	1	3	5	9	10	8	7	4

To encrypt, write the key number sequence ($K_1, \ldots, K_{10}$ in this example) under the plaintext. Replace the plaintext letters above k_i by the letter k_i positions beyond it in the alphabet. (I.e., we are using 10 different Caesar ciphers.)

We encrypt the plaintext "stay in current position" as follows

s	t	a	y	i	n	c	u	r	r	e	n	t	p	o	s	i	t	i	o	n
18	19	0	24	8	13	2	20	17	17	4	13	19	15	14	18	8	19	8	14	13
B	L	A	C	K	S	T	O	N	E	B	L	A	C	K	S	T	O	N	E	B
2	6	1	3	5	9	10	8	7	4	2	6	1	3	5	9	10	8	7	4	2
20	25	1	1	13	22	12	2	24	21	6	19	20	18	19	1	18	1	15	18	15
U	Z	B	B	N	W	M	C	Y	V	G	T	U	S	T	B	S	B	P	S	P

yielding the ciphertext "UZBBN WMCYV GTUST BSBPS PXXXX."

Lecture 8

Vigenère Cipher

A more general version of the example from the previous lecture is the Vigenère cipher (from the 16th century). This cipher uses the numerical value of the letters of the keyword for successive Caesar ciphers.

Example 8.1. key = BLACKSTONE

plaintext: stay in current position

s	t	a	y	i	n	c	u	r	r	e	n	t	p	o	s	i	t	i	o	n
18	19	0	24	8	13	2	20	17	17	4	13	19	15	14	18	8	19	8	14	13
B	L	A	C	K	S	T	O	N	E	B	L	A	C	K	S	T	O	N	E	B
1	11	0	2	10	18	19	14	13	4	1	11	0	2	10	18	19	14	13	4	1
19	4	0	0	18	5	21	8	4	21	5	24	19	17	24	10	1	7	21	18	14
T	E	A	A	S	F	V	I	E	V	F	Y	T	R	Y	K	B	H	V	S	O

ciphertext: TEAAS FVIEV FYTRY KBHVS OXXXX

Originally, encryption was done by way of a Vigenère table (Table 8). To encrypt, use the key to select the row and the plaintext letter to select the column of the corresponding ciphertext character.

Table 8.1: The Vigenère Table

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

8.1 Analysis of the Vigenère Cipher

Assume we know the number n of cipher alphabets used (i.e. the length of the keyword). This assumption will be justified later.

We form n subtexts $T_0, T_1, \ldots, T_{n-1}$ by selecting the $(i+1)$st letter of the ciphertext and every nth letter thereafter (each of these subtexts is a Caesar cipher where the key letter is the ith letter of the keyword).

Using the fact that the most commonly occurring letters are E, T, O, A, and N we can usually find the key letter K_i corresponding to each subtext T_i.

Example 8.2. Consider the following ciphertext:

```
BIPIZ VYPVK JLXAD VUBPP QDVEY XTMIM TCLRV SIVKN SEBCP SNELX
WPCES CTMRD SIAGW ROCEL XWPCC YFCQP QDQGD BJVJF QBIMS YKACN
QBXEL XWPWZ VWPKE ODWAQ TGDZQ GGMRO ZTDTE KDSMF IPGYX KSCQB
KEMGC JWDLW AIJGM ZQWHU QBPEU RMUCT FDTQP PZVEP BKYYV WFCKB
PWEDZ GZCSL TKVSZ RLWIP IZYEP GPYHL SKMAY FGSCV QGAVG IMEDT
RXDZO KEMGC AVYCI VDVAY FVHGM OSDIK QGRRD CARIN WPEKJ ZGCEL
SITKW TXSRK GCDXG PCVRZ VAOMF ZPSHA MUSXM DPZNI TFEWI TNHEJ
TIPND SXIEC PBTJD LWMEW ZPDGP PELJZ GCELS CKCXM IMHMF DZMVT
VVSQC SCLER PGCIP GKFXZ DZKJL XADVQ PAIGE TGDCC ACOVY REACI
EMPWK IWCCJ WLTUC XOMLH QPPZV EPBKY YRGLB JOCIA HIYKJ XGEWT
DPGLX VHYCQ SIQQX PZWCN WBELW GBJOT FERZA ZESYG IRRTG KJJUI
DXWBK CXPBL TVFNL XSRWP DCSDP VFCZS LTKVS ZRLDB JOOEL PKQWX
YFXKC DTSFH BGBXM FPTUK YHDXV MCELS IAROP HACNQ BXELX WPPCS
EDVGV ZGSIQ QXESS CWVRP VAICU ODEKD XJSDX ARIVO OEDVW TSELE
PAVBT GLHMV YQVMA MUDZI FRZAZ ESJHK TKXFD TLCDL FWUWT OTXAH
AVYCI VDZVB LRKBQ VDPHL DIPYE LWGTQ MLXAD VCXOH WRZAZ EMLLP
GXYIW SMFPZ VHGWE ODWAC OKDPQ HAWAC PRUGG RDTSF IMERY MIJMU
DSEFR IPBPH MRMKX QSJBI VSZRW MXQCF VWHEK DSMFN WWBNS EBCPS
NELXW PCYIL LWTUL WOTTN KDTJD DKNPE KNAVO XFSHM HYCXZ TLGFP
PGEUG XESXT VEBJT LXWPZ CSYGI OCELW XJOMC CHIWI BLTZX KUEMW
QHBGW TWSKM TCLXA AMVYZ PXDZE YYXJD TNSYK SCLRB ZXWRB KXRMF
UWTWL XADVV RCSMV PGXNV QEBKY YFQPK QWMMF PBKYY SXEZQ QCEEB
QPQLR VHVCD PVEXV CVSEJ SECBP JWPBW BPWAI KCXPR UGGRD LRVSM
EBJTL XVHYC QSIQQ WLYLD UCDTG SATAK YHOXB JYFXA CBGBG IFIQQ
XMCLW MVOCQ ACINE DIJDZ CZAPA RIVSZ RMHQP QLRSA OQBTX ZBIPN
LOWNE JSNLA CLKFT HMPTK JPWLW MCVRS JXBJW ELWHC DCJWL TUGXN
VQEBU KATDX KCDTS FXVHY CQSIQ QXMIX DZGSE MKHMP DQVGB IVOCQ
ACINY CGGBX WDPVD DKCDT SFPVF OYXWG AAYFV VPBCM ZQEJV KMLXA
DVUXP XODZM KEXZT ZGMPM NXVID PVEXV CVZVU DUREE IJAWE KEMGC
BJODE ETSGI TWMHM FDZHW RZAZE XZTQP PZVEP BKYYE XIMTS EPWPD
GCELW CMVGZ VCXVC NOMLX WPDZX ZTINQ ZVAIP ODSIA QUUEM WQHBG
WAVGK QFODO WNOGX PVSIQ QXVIQ BIPKR IETVV FPVAU QEKEM GCIPN
ZTWGI VSZRS ANGKE YJTAV RLXWC PCXNI LWMDK DMURZ AZESY GIRRT
GKTKW BTXQD NVRPW MQAAC EIE
```

Assume that we have deduced that the keyword has length 7 (i.e., that 7 subalphabets were used). We do a separate frequency count for each subtext as follows:

Subtext 0		**Subtext 1**		**Subtext 2**		**Subtext 3**		**Subtext 4**		**Subtext 5**		**Subtext 6**	
D	27	M	27	G	26	D	28	P	31	V	26	A	29
C	23	V	26	P	26	Y	24	Z	29	X	26	L	27
I	21	I	25	V	26	B	20	E	28	S	23	W	24
T	21	W	21	C	21	X	19	C	20	E	19	S	21
X	21	B	20	K	18	C	18	L	19	I	19	E	18
P	20	Z	18	Q	17	O	17	T	19	M	18	G	15
H	17	Q	16	E	12	S	17	D	15	W	16	F	14
R	14	K	15	U	12	K	16	Y	14	R	14	J	11
G	13	T	9	A	11	R	10	X	9	L	11	K	9
B	12	A	8	R	11	W	10	N	8	Q	10	M	9
A	8	U	7	T	10	Z	9	M	6	T	9	Q	9
E	6	L	6	J	9	M	8	J	6	G	9	D	9
J	5	P	6	F	8	P	8	S	6	F	8	Z	8
W	5	C	6	N	7	N	7	F	6	H	8	V	7
Q	4	N	5	W	7	Q	5	O	5	P	5	X	7
S	4	E	5	O	5	I	5	A	5	C	5	U	5
N	4	G	4	H	4	V	5	Q	4	Y	5	H	5
K	3	O	4	M	3	E	4	R	4	K	3	Y	4
U	3	J	4	I	2	F	3	H	2	O	2	C	3
L	3	D	3	D	2	J	3	G	1	B	1	O	3
V	2	X	3	B	1	G	2	V	1	J	1	N	1
F	1	S	1	Y	1	U	1	B	1	Z	1	I	1
O	1	R	0	L	0	T	0	W	0	D	0	B	0
M	1	H	0	X	0	A	0	I	0	N	0	T	0
Y	0	Y	0	S	0	L	0	K	0	U	0	R	0
Z	0	F	0	Z	0	H	0	U	0	A	0	P	0

Notice that the most commonly occurring letters (and their numerical equivalents) in T_0 are

D	C	I	T	X	P	H	R	G
3	2	8	19	23	15	7	17	6

The letters E, T, A, and O correspond to the numbers 4, 19, 0, and 14. Let K_0 be the numerical equivalent of the key letter for T_0. Then "e" is encrypted to $4 + K_0 \pmod{26}$. Since $\{3, 2, 8, 19, 23, 15, 7, 17, 6\}$ are the numerical values of the most frequent ciphertext characters in T_0, with high probability we have

$$\begin{aligned} 4 + K_0 &\in \{3, 2, 8, 19, 23, 15, 7, 17, 6\} \\ K_0 &\in \{25, 24, 4, 15, 19, 11, 3, 13, 2\} \ . \end{aligned}$$

Similarly, "t" is encrypted to $19 + K_0 \pmod{26}$ and we expect that

$$\begin{aligned} 19 + K_0 &\in \{3, 2, 8, 19, 23, 15, 7, 17, 6\} \\ K_0 &\in \{10, 9, 15, 0, 4, 22, 14, 25, 13\} \ . \end{aligned}$$

By considering "o" and "a" we obtain

$$\begin{aligned} 14 + K_0 &\in \{3, 2, 8, 19, 23, 15, 7, 17, 6\} \\ K_0 &\in \{15, 14, 20, 5, 9, 1, 19, 3, 18\} \\ 0 + K_0 &\in \{3, 2, 8, 19, 23, 15, 7, 17, 6\} \\ K_0 &\in \{3, 2, 8, 19, 23, 15, 7, 17, 6\} \ . \end{aligned}$$

Since 15 is common to all these sets, we suspect that $K_0 = 15 \implies$ the first letter of the keyword is "P."

Note. The method described above is known as *Kerckhoff's shortcut*.

Lecture 9

The ϕ-statistic

The variation in frequencies of the occurences of elements in a non-random text is greater than those for random text. ϕ is a measure of this variation. The value of ϕ will aid us in deciding whether or not a polyalphabetic cipher has ben resolved into its basic components (i.e., we know the number of alphabets used).

Definition 9.1. Denote by:

N	number of characters in the text
n	number of characters in the alphabet ($n = 26$ for english)
f_i	number of times the ith alphabetic character occurs in the text.

(observe that $\sum f_i = N$). We define the *ϕ-statistic* as

$$\phi = \sum_{i=1}^{n} f_i(f_i - 1) \ .$$

Note. For a given text of length N, $\phi/(N(N-1))$ is the probability that two randomly-chosen ciphertext characters are identical. This is often refered to as the *index of coincidence*, denoted $I_C(N)$.

1) # pairs containing only char i: $\binom{f_i}{2} = f_i(f_i - 1)/2$

2) # pairs containing double chars: $\sum f_i(f_i - 1)/2$

3) # total pairs: $\binom{N}{2} = N(N-1)/2$

$I_C(N) =$ equation 2/equation 3

Theorem 9.1. *The expected value of ϕ, $E(\phi)$, is given by*

$$E(\phi) = S_2 N(N-1)$$

where

$$S_2 = \sum_{i-1}^{n} P_i^2$$

and P_i is the relative frequency of the ith character in the alphabet.

(Proof omitted)

Note. For random text

$$P_i = \frac{1}{n}, \quad S_2 = \sum_{i=1}^{n} \frac{1}{n^2} = \frac{1}{n} .$$

Thus, if $n = 26$, then for random text $S_2 \approx 0.0385$. (Note that this is the *minimum* value — less random text has larger values)

For non-random English text $S_2 \approx 0.0661$.

Idea:

- Compute ϕ (observed value)
- Compare with $E(\phi)$ for random and english text

Example 9.1. Ciphertext: IBMQO PBIUO MBBGA TCZOF MUUQB ($N = 25$)

Compute ϕ :

	f_i	$f_i - 1$	$f_i(f_i - 1)$	k	$k(f_i)(f_i - 1)$
B	5	4	20	1	20
M,O,U	3	2	6	3	18
I, Q	2	1	2	2	4
Others	1	0	0	7	0
					$\phi = 42$

For English, $E(\phi) = S_2 N(N-1) \approx 0.0661(25)(24) \approx 39.66$.

For random text, $E(\phi) \approx 0.0385(25)(24) \approx 23.1$.

Since $\phi = 42$ for the example, this ciphertext is probably a monoalphabetic substitution cipher.

Example 9.2. Ciphertext: HKWZA RRPVQ BIVYS MPDMQ MBUDC ($N = 25$)

$\phi = 18$

Since 18 is closer to 23.1 than to 39.66, this is likely random text, and at least not a monoalphabetic substitution cipher.

Note. $0.0385N(N-1)$ is the minimum *expected value.* The *observed* values could be less (minimum is 0 — only singly-occurring characters)

Lecture 10

Determining the Number of Alphabets

10.1 ϕ-statistic Method

For sufficient ciphertext, the ϕ statistic will resolve the number of alphabets used, even if it is large, because the ϕ-statistic can be averaged over all alphabets.

Example 10.1. (Naive statistically — usually make use of various normal probability distributions)

Consider the following ciphertext, which is known to be enciphered polyalphabetically with a number of alphabets between 40 and 50 :

```
HSKUS PMFHD UJJIX MSPTP OIPCI WKZVU
YPPNE USAIG BOOGA OPGPR HBOUC SHPVG
HQXZS ACKRK VBGHM VSFRY YTKHK VWZXV
LIJHW ARLKF IJSLT MHKAH QTUVT XSMEC
FCSKT GOOYB XZVLI JRYAC DWEJM SCAFP
IEAXO KAQDW EXPYP QHDNO JIXNZ JGNUD
OARFU ERJOY BDOKE IKDUV TDVEV LETDO
AFROU NYNBD VQOBE GGSHQ HXOPU ZCOCU
KKZLT PHKRT CCOAS BZUGB UBBUN OVTPO
VMIZD EPQFV KZ
```

Assuming the 50 alphabets were used, the message would be rewritten as

			1		2		3		4	
	01234	56789	01234	56789	01234	56789	01234	56789	01234	56789
1	HSKUS	PMFHD	UJJIX	MSPTP	OIPCI	WKZVU	YPPNE	USAIG	BOOGA	OPGPR
2	HBOUC	SHPVG	HQXZS	ACKRK	VBGHM	VSFRY	YTKHK	VWZXV	LIJHW	ARLKF
3	IJSLT	MHKAH	QTUVT	XSMEC	FCSKT	GOOYB	XZVLI	JRYAC	DWEJM	SCAFP
4	IEAXO	KAQDW	EXPYP	QHDNO	JIXNZ	JGNUD	OARFU	ERJOY	BDOKE	IKDUV
5	TDVEV	LETDO	AFROU	NYNBD	VQOBE	GGSHQ	HXOPU	ZCOCU	KKZLT	PHKRT
6	CCOAS	BZUGB	UBBUN	OVTPO	VMIZD	EPQFV	KZ			
ϕ	40222	02020	20000	02002	62000	22000	22002	02000	20200	00000

In the last line of the previous table, we calculate ϕ for each of the 50 subtexts, some with 6 characters and some with 5. We average all the ϕ values for the 6 character subtexts, and likewise for the 5 character subtexts:

$N=6$			$N=5$		
ϕ_i	w_i	$w_i\phi$	ϕ_i	w_i	$w_i\phi$
0	17	0	0	14	0
2	13	26	2	4	8
4	1	4	4	0	0
6	1	6	6	0	0
$\sum$	32	36	$\sum$	18	8

(w_i is the number of subtexts having $\phi = \phi_i$).

To compute the average ϕ (denoted by $\overline{\phi}$):

$$\overline{\phi} = \frac{\sum w_i \phi_i}{\sum w_i} = \frac{\sum \phi}{\#\text{ of subtexts}}$$

$N=6$	$N=5$
$\overline{\phi} = 36/32 = 1.125$	$\overline{\phi} = 8/18 \approx 0.44$
$E(\phi)_E = 0.0661(6)(5) \approx 1.98$	$E(\phi)_E = 0.0661(5)(4) \approx 1.32$
$E(\phi)_R = 0.0385(6)(5) \approx 1.15$	$E(\phi)_R = 0.0385(5)(4) \approx 0.77$

This indicates that the number of cipher alphabets used is probably *not* 50, since the average ϕ values are closer to the expected values for random text ($E(\phi)_R$) than to English ($E(\phi)_E$). So, we try 49, 48, 47, etc... until we try 43.

Assuming the 43 alphabets were used, the message would be rewritten as

			1		2		3		4
	01234	56789	01234	56789	01234	56789	01234	56789	012
1	HSKUS	PMFHD	UJJIX	MSPTP	OIPCI	WKZVU	YPPNE	USAIG	BOO
2	GAOPG	PRHBO	UCSHP	VGHQX	ZSACK	RKVBG	HMVSF	RYYTK	HKV
3	WZXVL	IJHWA	RLKFI	JSLTM	HKAHQ	TUVTX	SMECF	CSKTG	OOY
4	BXZVL	IJRYA	CDWEJ	MSCAF	PIEAX	OKAQD	WEXPY	PQHDN	OJI
5	XNZJG	NUDOA	RFUER	JOYBD	OKEIK	DUVTD	VEVLE	TDOAF	ROU
6	NYNBD	VQOBE	GGSHQ	HXOPU	ZCOCU	KKZLT	PHKRT	CCOAS	BZU
7	GBUBB	UNOVT	POVMI	ZDEPQ	FVKZ				
ϕ					00000	01000			
	20244	42426	40242	46040	44462	04822	04204	22242	462

We calculate ϕ for each of the subtexts and average over all subtexts of the same length:

$N=7$			$N=6$		
ϕ_i	w_i	$w_i\phi$	ϕ_i	w_i	$w_i\phi$
0	4	0	0	3	0
2	6	12	2	9	18
4	11	44	4	4	16
6	3	18	6	1	6
			8	1	8
			14	1	14
$\sum$	24	74	$\sum$	19	62
		$\overline{\phi} = 74/24 \approx 3.08$			$\overline{\phi} = 62/19 \approx 3.26$
		$E(\phi)_E = 0.0661(7)(6) \approx 2.77$			$E(\phi)_E = 0.0661(6)(5) \approx 1.98$
		$E(\phi)_R = 0.0385(7)(6) \approx 1.61$			$E(\phi)_R = 0.0385(6)(5) \approx 1.15$

These values are a good indication that the number of cipher alphabets is 43, since the $\overline{\phi}$ values are closer to the expected values for English than those for random text.

Conclusion: much more redundancy than for random text $\Longrightarrow$ not random $\Longrightarrow$ probably English.

10.2 Factoring Method

The factoring method (also known as the Kasiski method), can be used when the number of alphabets is small, or there is a lot of ciphertext.

The basic idea is the following: if k alphabets are used and an n-gram G_n in position i is encrypted to C_n, then that same n-gram in position $i+mk$ will also encrypt to C_n. In other words, if two identical segments of plaintext are enciphered by identical segments of the key, the resulting ciphertext segments will be identical.

Example 10.2. In this example, the plaintext tx is encryted by AT each time. The distances between repititions of the ciphertext DC are multiples of 4, the length of the keyword.

```
axtlf btleg ...tl
BOATB OATBO ...AT
--DC- -DC-- ...DC
```

One can take advantage of this by looking at all repititions in the ciphertext and looking at the distances between individual repitions. The number of alphabets will likely be equal to the gcd of these distances.

Example 10.3. Consider the following ciphertext:

```
SIJYU MNVCA ISPJL RBZEY QWYEU LWMGW ICJCI MTZEI MIBKN
QWBRI VWYIG BWNBQ QCGQH IWJKA GEGXN IDMRU VEZYG QIGVN
CTGYO BPDBL VCGXG BKZZG IVXCU NTZAO BWFEQ QLFCO MTYZT
CCBYQ OPDKA GDGIG VPWMR QIIEW ICGXG BLGQQ VBGRS MYJJY
QVFWY RWNFL GXNFW MCJKX IDDRU OPJQQ ZRHCN VWDYQ RDGDG
BXDBN PXFPU YXNFG MPJEL SANCD SEZZG IBEYU KDHCA MBJJF
KILCJ MFDZT CTJRD MIYZQ ACJRR SBGZN QYAHQ VEDCQ LXNCL
LVVCS QWBII IVJRN WNBRI VPJEL TAGDN IRGQP ATYEW CBYZT
EVGQU VPYHL LRZNQ XINBA IKWJQ RDZYF KWFZL GWFJQ QWJYQ
IBWRX
```

The principal repititions of three or more letters have been underlined in the message and the factors (up to 20 only) of the intervals between them are as follows:

Fragment	Distance	Factors
CGXGB	60	2,3,4,5,6,10,12,15,20
PJEL	95	5,19
BRI	285	3,5,15,19
QRD	165	3,5,15
QWB	275	5,11
WIC	130	2,5,10,13
XNF	45	3,5,9,15
YZT	225	3,5,15
ZGI	145	5

The factor 5 is common to all of these repititions, and there seems to be every indication that five alphabets are involved. Certainly, this is not a proof that five alphabets were used — it is only a working hypothesis.

Note. Factoring may not work if there are coincidental repitions of ciphertext. In this case, look for the most common factor amongst the repitions (trial and error).

Lecture 11

Mixed Polyalphabetic Ciphers

11.1 Mixed Vigenère

One major problem with the Vigenère cipher is that every subtext is simply a Caesar cipher, and Kerckhoff's shortcut can be used very effectively to find the keyword. This was addressed by using, instead of a plain alphabet in the Vigenère Table, a disordered alphabet.

Example 11.1. As with regular Vigenère, select the column according to the plaintext symbol and the row according to the key.

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

Analysis

More difficult than regular Vigenère, must nevertheless insecure:

1. resolve the number of alphabets (as before — each is still monoalphabetic)
2. use frequency counts to determine the most common characters in each cipher alphabet. Attempt to separate the vowels from the consonants in each subtext.
3. use the principal of "symmetry of position"

Symmetry of Position

Each row of a particular cipher alphabet is a cyclic shift of any of the others. Hence, if the relative position of a pair of characters is known in one row, determination of either in another row allows one to fix the position of the other character in that row.

Example 11.2. Suppose that as the result of an analysis based upon considerations of frequency, we have assumed the following values in a given cryptogram:

Plain	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
Cipher 1					G										Y					V						
Cipner 2					N										G					P						
Cipher 3					L										B					I						
Cipher 4					W										I					Q						

Note that the letter G is common to cipher alphabets 1 and 2. In alphabet 2, we note that N occupies the 10th position to the left of G, and the letter P occupies the 5th position to the right of G. We may therefore place these letters, N and P, in their proper positions in alphabet 1, the letter N being placed 10 letters before G, and the letter P, 5 letters after G. Thus:

Plain	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
Cipher 1					G					P					Y					V	N					
Cipher 2					N										G					P						
Cipher 3					L										B					I						
Cipher 4					W										I					Q						

Using the same G, we can also map Y and V into alphabet 2:

Plain	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
Cipher 1					G					P					Y					V	N					
Cipher 2				V	N										G					P					Y	
Cipher 3					L										B					I						
Cipher 4					W										I					Q						

Similarly, we can use symmetry of position to compare alphabets 3 and 4:

Plain	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
Cipher 1					G					P					Y					V	N					
Cipher 2				V	N										G					P					Y	
Cipher 3					L					W					B					I					Q	
Cipher 4					W					B					I					Q						L

Use the new information to fill in more parts of the plaintext, and try to guess at new words. Then, go back to the table and use symmetry of position, etc.

11.2 Coherent Running Key Ciphers

Observation: the longer the Vigenère key, the safer the cipher.

Use a *running* key, text taken from a readily available book or other coherent text (eg. Declaration of Independence). Use the running key in the same way as Vigenère (each key letter specifies a Caesar cipher).

Example 11.3.

```
 plaintext:  thism ateri alise nciph ered
       key:  ONANO NINTE RFERI NGBAS ISOV
-----------------------------------------
ciphertext:  HUIFA NBRKM RQMJM AIJPZ MJSYX
```

To encrypt t in "this:" $t + O = 19 + 14 = 33 \equiv 7 \pmod{26} = H$

Analysis

1. The coherent running key has a letter frequency similar to that of the plaintext. Thus, plaintext/key combinations like i/E, e/T, e/E, etc. are common, and common letters of ciphertext may represent such combinations (eg. cipher character M in the previous example). If we can successfully guess enough letters in the key, we may be able to recognize the entire key (coherent text).
2. (Bazeries-Friedman solution) Use a probable plaintext word and subtract it from the ciphertext to see if a meaningful key results. Try completing unfinished words in the key.

Coherent running-key cipher is certainly harder to break than Vigenère and Mixed Vigenère, but is nevertheless insecure.

What happens if we do multiple levels of encryption (2 or more coherent running keys)? It has been *proven* that using 4 separate and distinct coherent running keys will (statistically) result in a secure cipher.

What do we mean by "secure?" Information Theory provides mechanisms to evaluate statistical security.

Lecture 12

Entropy

Information theory measures the *amount of information* in a message by the average number of bits needed to encode all possible messages in an optimal prefix-free encoding. Here, *optimal* means the average number of bits is as small as possible.

Example 12.1. The four messages

UP, DOWN, LEFT, RIGHT

could be encoded in the following ways:

String	Character	Numeric	Binary
"UP"	"U"	1	00
"DOWN"	"D"	2	01
"LEFT"	"L"	3	10
"RIGHT"	"R"	4	11
(40 bits)	(8 bits)	(16 bits)	(2 bits)
(5 char string)	8-bit ASCII	(2 byte integer)	2 bits

All encodings carry the same information (which we will be able to measure), but some are more efficient (in terms of the number of bits required) than others. Huffmann encoding can be used to improve on this if the directions occur with different probabilities.

The amount of information in a message is formally measured by the *entropy* of the message. The entropy is a function of the probability distribution over the set of possible messages.

Definition 12.1. Let $\{X_1, X_2, \dots, X_n\}$ be a set of n possible messages occurring with probabilities

$$p(X_1), p(X_2), \dots, p(X_n) \quad \text{where} \quad \sum_{i=1}^{n} p(X_i) = 1$$

(i.e., the $p(X_i)$ form a probability distribution). The *entropy* of a discrete random variable X (X can be any of the X_i) is defined by the weighted average

$$H(X) = \sum_{\substack{i=1 \\ p(X_i)\neq 0}}^{n} p(X_i) \log_2 \frac{1}{p(X_i)} = - \sum_{\substack{i=1 \\ p(X_i)\neq 0}}^{n} p(X_i) \log_2 p(X_i) \ .$$

Example 12.2. Suppose $n > 1$. Then

$$\begin{gathered} 0 < p(X_i) < 1 \quad (i = 1, 2, \ldots, n) \\ \frac{1}{p(X_i)} > 1 \\ \log_2 \frac{1}{p(X_i)} > 0, \end{gathered}$$

hence $H(X) > 0$ if $n > 1$. If there are at least 2 messages, then receiving either of them conveys information.

Example 12.3. Suppose $n = 1$. Then

$$p(X_1) = 1, \quad \frac{1}{p(X_1)} = 1, \quad \log_2 \frac{1}{p(X_1)} = 0 \implies H(X) = 0 \ .$$

If there is only one possible message, no new information is gained upon receiving it.

Example 12.4. Suppose there are two possible messages which are equally likely:

$$\begin{aligned} p(\text{male}) = p(\text{female}) &= \frac{1}{2}, \\ H(X) &= \frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2 = 1 \ . \end{aligned}$$

Receiving either message conveys exactly 1 bit of information (male or female).

Example 12.5. Suppose we have

$$p(UP) = \frac{1}{2}, \quad p(DOWN) = \frac{1}{4}, \quad p(LEFT) = \frac{1}{8}, \quad p(RIGHT) = \frac{1}{8} \ .$$

Then

$$\begin{aligned} H(X) &= \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 8 \\ &= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} \\ &= \frac{14}{8} = \frac{7}{4} = 1.75 \ . \end{aligned}$$

An optimal encoding is

$$UP = 0, \quad DOWN = 10, \quad LEFT = 110, \quad RIGHT = 111 \ .$$

Because UP is more probable than the other messages, receiving UP conveys less information than receiving one of the other messages. The *average* amount of information received is 1.75 bits.

Intuitively, $\log_2(1/p(X))$ represents the number of bits needed to encode X in an optimal encoding. The weighted average $H(X)$ gives the average number of bits per character in an optimally encoded message.

Example 12.6. Suppose we have n messages which are equally likely: $p(X_i) = 1/n$.

$$H(X) = \sum_{i=1}^{n} \frac{1}{n} \log_2 n = \log_2 n \ .$$

If $n = 2^k$, then $H(X) = k$. In other words, if all messages are equally likely, then $H(X) = \log_2 n$.

Lecture 13

Rate and Redundancy

Theorem 13.1. *Given n, $H(X)$ is maximized for $p(X_i) = 1/n$ for $i = 1 \leq i \leq n$. Furthermore, $H(X)$ decreases as the distribution of messages becomes increasingly skewed, reaching a minimum when $n = 1 \implies H(X) = 0$.*

Idea of proof: (This is *not* a complete proof! The full proof uses induction on n)

Suppose $p(X_1) > 1/n$, $p(X_2) < 1/n$, and $p(X_i)$ is fixed for $i > 2$. Set $p = p(X_1)$, then $p(X_2) = 1 - p - \epsilon$ and

$$\begin{aligned} H &= -p \log p - (1 - p - \epsilon) \log(1 - p - \epsilon) + k \\ \frac{dH}{dp} &= -\log p - 1 + \log(1 - p - \epsilon) + 1 \\ &= \log \frac{1 - p - \epsilon}{p} < 0 \quad \text{since } 0 < p < 1 \end{aligned}$$

Thus, H decreases as a function of p as p increases. Prove that H is maximized when $p = 1/n$ by induction on n (A2 question!) **QED**

Rate and Redundancy

Definition 13.1. For a given language, let X be the set of all messages N characters long. The *rate* of the language for messages of length N is defined to be

$$r = \frac{H(X)}{N} = \frac{\text{avg. info}}{\text{number of chars}};$$

that is, the average number of bits of information in each character.

Example 13.1. In English, for large N

$$1 \leq r \leq 1.5 \ .$$

Definition 13.2. The *absolute rate* of a language is defined to be the maximum number of bits of information that could be encoded in each character, assuming that all possible sequences of characters are equally likely. If there are L characters in the language, then the absolute rate is given by

$$R = \log_2 L,$$

the maximum entropy of individual characters.

Example 13.2. In English, $R = \log_2 26 \approx 4.7$ bits/character.

Note that the actual rate of English (1.5) is much less than the absolute rate (English is highly redundant).

Definition 13.3. The *redundancy* of a language with rate r and absolute rate R is defined by

$$D = R - r \quad (\approx 4.7 - 1.5 = 3.2 \text{ for English.}$$

Redundancy is more commonly used in the form

$$D/R = \frac{R-r}{R} = 1 - \frac{r}{R} \ .$$

By this measure, English has redundancy

$$0.68 < D/R < 0.79,$$

i.e., English is between 68% and 79% redundant.

Lecture 14

Equivocation and Perfect Security

The uncertainty of messages may be reduced when we have additional information (side information). For example, let X be a 32 bit integer, with all values equally likely; then $H(X) = 32$. But if we know that X is even, then the entropy is reduced by one bit ($H(X) = 31$) because the low order bit of X must be 0.

Given a message Y in the set $Y_1, Y_2, \ldots, Y_m$ where $\sum_{i=1}^{m} p(Y_i) = 1$, let $p(X \mid Y)$ be the conditional probability of a message X, given Y (the messages Y_i are the side information). Let $p(X,Y)$ be the joint probability of message X and message Y. Then

$$p(X,Y) = p(X \mid Y)p(Y) \ .$$

Definition 14.1. The *equivocation* is the conditional entropy of X given Y :

$$\begin{aligned} H(X \mid Y) &= -\sum_{X,Y} p(X,Y) \log_2 p(X \mid Y) \\ &= \sum_{Y} p(Y) \sum_{X} p(X \mid Y) \log_2 \frac{1}{p(X \mid Y)} \ . \end{aligned}$$

Example 14.1. $n = 4$, $p(X_i) = 1/4$ (all four messages are equally likely). Thus, $H(X) = 2$. Similarly, let $m = 4$, $p(Y_i) = 1/4$ for each message Y_i. Now, suppose that each message Y_i narrows the choice of X to two of the four possibilities are shown:

Y_1	X_1 or X_2
Y_2	X_3 or X_4
Y_3	X_2 or X_3
Y_4	X_4 or X_1

Then for each Y_i, $p(X_i \mid Y_i) = 1/2$ for two of the X_i and $p(X_i \mid Y_i) = 0$ for the other two X_i. Thus

$$H(X \mid Y) = 4\left(\frac{1}{4} \cdot 2 \left(\frac{1}{2} \log_2 2\right)\right) = 1 \ .$$

Thus, there is only one bit of uncertainty in X, given Y.

Example 14.2. Consider X to be the result of throwing a fair die (i.e., all results are equally probable, $p(X) = 1/6$ and $H(X) = \log_2 6$). Suppose

$$Y = \begin{cases} 1 & \text{if } X \text{ is odd} \\ 0 & \text{if } X \text{ is even.} \end{cases}$$

Then $H(Y) = 1$ and $p(X \mid Y) = 0$ or $1/3$. What is $H(X \mid Y)$?

$$\begin{aligned}
H(X \mid Y) &= \sum_Y p(Y) \sum_X p(X \mid Y) \log_2 \frac{1}{p(X \mid Y)} \\
&= \frac{1}{2}\left(p(2 \mid 0) \log_2 \frac{1}{p(2 \mid 0)} + p(4 \mid 0) \log_2 \frac{1}{p(4 \mid 0)} + p(6 \mid 0) \log_2 \frac{1}{p(6 \mid 0)}\right) \\
&\quad + \frac{1}{2}\left(p(1 \mid 1) \log_2 \frac{1}{p(1 \mid 1)} + p(3 \mid 1) \log_2 \frac{1}{p(3 \mid 1)} + p(5 \mid 1) \log_2 \frac{1}{p(5 \mid 1)}\right) \\
&= \frac{1}{2}\left(6 \frac{1}{3} \log_2 3\right) \\
&= \log_2 3
\end{aligned}$$

Perfect Security

Suppose we have the following:

1. Plaintext messages M occur with probabilities $p(M)$ and $\sum_M p(M) = 1$.
2. Ciphertext messages C appear with probabilities $p(C)$ such that $\sum_C p(C) = 1$.
3. Keys are selected with prior probabilities $p(K)$ where $\sum_K p(K) = 1$.

Let $p(M \mid C)$ be the probability that the message M was sent, given that the ciphertext C was received.

Definition 14.2. *Perfect security* is defined by the condition

$$p(M \mid C) = p(M),$$

i.e., the fact that we know the ciphertext gives us no information about M.

Lecture 15

Perfect Security

Consider $p(C \mid M)$, the probability of receiving ciphertext C given that M was sent. We have

$$p(C \mid M) = \sum_{\substack{K \\ E_K(M)=C}} p(K) \ .$$

That is, $p(C \mid M)$ is the sum of probabilities $p(K)$ of the keys K that encipher M to C. Usually there is at most 1 key such that $E_K(M) = C$ for given M and C, but some ciphers can transform the same plaintext into the same ciphertext with different keys. For example, a monoalphabetic cipher will transfrom a message into the same ciphertext with different keys if the only differences between the keys occur for characters which do not appear in the message.

A necessary and sufficient condition for perfect security is

$$p(C \mid M) = p(C) \quad \text{for all } M,$$

i.e., the probability of receiving a particular ciphertext C, given that M was sent (enciphered with some key K) is the same as the probability of receiving C given that some other message M was sent (enciphered under another key). Thus, the following are equivalent:

$$p(M \mid C) = p(M) \iff p(C \mid M) = p(C) \quad \text{for all } M, C$$

To see this, note the following:

$$\begin{aligned} p(M,C) &= p(C,M) && \text{joint probabilities} \\ p(M,C) &= p(M \mid C)p(C) && \text{identity} \\ p(C,M) &= p(M)p(C \mid M) && \text{identity} \\ p(M)p(C \mid M) &= p(M \mid C)p(C), \end{aligned}$$

but if we have perfect security, by definition $p(M) = p(M \mid C)$, so those two terms cancel and we have

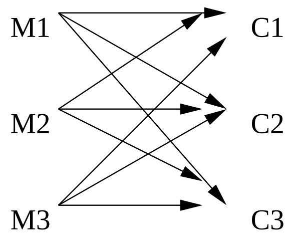
$$p(C \mid M) = p(C)$$

as claimed.

Example 15.1. Suppose we have 3 messages M_1, M_2, M_3 and three ciphertexts C_1, C_2, C_3, and all occur with equal probability ($p(M_1) = p(M_2) = p(M_3) = 1/3$ and $p(C_1) = p(C_2) = p(C_3) = 1/3$). Also, suppose that we have perfect security, i.e.,

$$\begin{aligned} p(M \mid C) &= p(M) = 1/3 \\ p(C \mid M) &= p(C) = 1/3 \ . \end{aligned}$$

This means that C_i corresponds to M_j with equal probability for all i, j.



Perfect security requires that the number of random keys must be as great as the number of possible messages. Otherwise, there would be some message M such that for a given C, no K decrypts C into M, implying $p(M \mid C) = 0$. The cryptanalyst could thereby eliminate certain possible plaintext messages from consideration, increasing the chances of breaking the cipher.

Theorem 15.1. *For a cryptosystem to have perfect security, it is necessary for there to be at least as many keys as there are messages.*

Proof. Let $n = |\mathcal{M}|$ (the number of messages). Consider some key K_r and $S = \{E_{K_r}(M) \mid M \in \mathcal{M}\}$. These elements must all be distinct, so $|S| = n$. For $M_j \in \mathcal{M}$, let $C_j = E_{K_r}(M_j)$, then

$$p(C_j) = p(C_j \mid M_j) > 0 \ .$$

For any other M_i,

$$p(C_j \mid M_i) = p(C_j) > 0 \ .$$

So, pick a particular C_j. Then $\exists K_i$ such that $E_{K_i}(M_i) = C_j$. We can't have $K_i = K_r$ if $i \neq r$, since we would have one key mapping two messages to the same ciphertext. Hence there are at least n keys. **QED**

Example 15.2. With an alphabet of size 26, if n is large enough that $26^n > 26!$, monoalphabetic substitution does not have perfect security.

Lecture 16

The One-time Pad (Vernam coding)

Consider the alphabet to be replaced by 5-bit binary equivalents (or ASCII, EBCDIC, or whatever), thus converting the message to a bitstream. Define (XOR)

$$a \oplus b = \begin{cases} 0 & a = b \\ 1 & a \neq b, \end{cases}$$

and if $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ then

$$A \oplus B = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n)$$

(component-wise XOR).

Let M be any n-bit message and K be any n-bit (random) key. Let C be the ciphertext given by $C = M \oplus K$. This cipher is called the *one-time pad*, and is provably secure as long as the following hold:

1. K must be random
2. K must be as long as M
3. K must be used only once

Suppose K were used twice:

$$\begin{aligned} C_1 &= M_1 \oplus K \\ C_2 &= M_2 \oplus K \\ \Longrightarrow\ C_1 \oplus C_2 &= M_1 \oplus M_2 \quad \text{since } K \oplus K = (0, 0, \ldots, 0)\ . \end{aligned}$$

Note that $C_1 \oplus C_2 = M_1 \oplus M_2$ is nothing more than a coherent running key cipher (adding two coherent texts, M_1 and M_2), which as we have seen is insecure.

The one-time pad is an example of an unconditionally secure cipher. Such ciphers resist all cryptanalytic attacks. Even if the cryptanalyst tries all possible keys, he finds multiple solutions.

There are disadvantages to such schemes. For example, the one-time pad requires a random key which is as long as the message, and each key can be used only once. However, one-time schemes are used when unconditional security is crucial, for example, Moscow-Washington hotline.

The major problem with the one-time pad is the cost. As a result, we generally rely on *computationally secure* ciphers. These ciphers would succumb to exhaustive search, because there is a unique "meaningful" decipherment. The computational difficulty of finding this solution foils the cryptanalyst. *Proof* of security does not exist for any proposed computationally secure system.

16.1 Key Equivocation and Perfect Security

Shannon measured the security of a cipher in terms of the *key equivocation* $H(K \mid C)$. That is the amount of uncertainty about K, given C. If $H(K \mid C) = 0$, then there is no uncertainty and the ciphertext can theoretically be broken, given enough resources. As N, the length of the ciphertext increases, usually $H(K \mid C)$ decreases.

Definition 16.1. The *unicity distance* is the smallest N such that $H(K \mid C)$ is close to zero; that is, the amount of ciphertext needed to uniquely determine K.

Note. A cipher is *unconditionally secure* if $H(K \mid C)$ never approaches zero, even for large N.

Lecture 17

Unicity Distance

Most ciphers are too complex to determine the probabilities needed to evaluate the unicity distance. It is possible to approximate it for certain ciphers using a random cipher model (Hellman).

Assume each plaintext and ciphertext comes from a finite alphabet of L symbols. Thus, there are $L^N = 2^{RN}$ ($R = \log_2 L$ is the absolute rate of the language) possible messages of length N. We partition the messages into two subsets:

1. a set of $2^{rN} = 2^{H(M)}$ meaningful messages
2. a set of $2^{RN} - 2^{rN}$ meaningless messages.

All meaningful messages will be assumed to have the same probability 2^{-rN}, while all meaningless messages have probability 0. We will also assume that we have $2^{H(K)}$ keys, all equally likely, where $H(K)$ is the key entropy (number of bits in the key). The probability of any given key K is given by $p(K) = 2^{-H(K)}$.

A *random cipher* is one in which for each key K and ciphertext C, the decipherment $D_K(C)$ is an independent random variable uniformly distributed over all 2^{RN} messages, both meaningful and not. Intuitively, this means that for a given K and C, $D_K(C)$ is as likely to produce one plaintext message as it is to produce any other. (Not totally independent, since $D_K(C) \neq D_K(C')$ for $C \neq C'$.)

Consider the ciphertext $C = E_K(M)$ for given K and M. A *spurious key decipherment* or *false solution* arises whenever encipherment under another key K' *could* produce C; that is,

$$C = E_{K'}(M)$$

for the same message or

$$C = E_K(M')$$

for another *meaningful* message M'. A cryptanalyst intercepting one of these ciphertexts would be unable to break the cipher since there is no way to select the correct key or message. We are *not* concerned with decipherments which produce meaningless messages because the cryptanalyst can immediately reject these solutions.

For every correct solution to a particular cipher, there are $2^{H(K)} - 1$ remaining keys, each of which has the same probability q of yielding a spurious key decipherment. Because each plaintext message is equally likely, the probability of getting a meaningful message and, therefore, a false solution is given by

$$q = \frac{2^{rN}}{2^{RN}} = 2^{-DN} \quad (D \text{ is the redundancy}).$$

Let F denote the expected number of false solutions

$$F = \left(2^{H(K)} - 1\right) q = \left(2^{H(K)} - 1\right) 2^{-DN} \approx 2^{H(K)-DN} \;.$$

Because of the rapid decrease in the exponent with increasing N

$$\log_2 F \approx H(K) - DN = 0$$

is taken as the point where the number of false solutions is sufficiently small that the cipher can be broken. Thus

$$N = \frac{H(K)}{D}$$

approximates the *unicity distance* — the average amount of text needed to uniquely determine the key (break the cipher).

Example 17.1. If for a given N, the number of possible keys is at least as large as the number of messages, then

$$H(K) \geq \log_2(2^{RN}) = RN \ .$$

Thus

$$H(K) - DN \geq RN - DN = rN \neq 0$$

The cipher is theoretically unbreakable (eg. one-time pad).

Example 17.2. Caesar cipher. $H(K) = \log_2 26 \approx 4.7$. Unicity distance:

$$N = \frac{H(K)}{D} \approx \frac{4.7}{4.7 - 1.5} = \frac{4.7}{3.2} \approx 1.5 \ .$$

Theoretically we only need 1 or 2 characters to break a Caesar cipher — certainly not true in practice.

Difficulties with unicity distance:

- $r = 1.5$ is true for large values of N, but not necessarily true for small N.
- all messages are not equally likely (as in the random model)

One should regard the unicity distance derived here as a conservative lower bound.

Example 17.3. Digraph substitution. $|\mathcal{K}| = (26^2)!$, $H(K) = \log_2((26^2)!) \approx 5419$ (derived by *Stirling's formula*:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \ .$$

Unicity distance:

$$N = \frac{H(K)}{D} \approx \frac{5419}{3.2} \approx 1700 \ .$$

Thus, theoretically we need at least 1700 characters to break a digraphic cipher.

Lecture 18

Transposition Ciphers

A *transposition* cipher is a rearrangement of the letters in the plaintext according to some specific system and key (i.e., a permutation of the plaintext). They are generally insecure.

Definition 18.1. *Route ciphers* usually involve arranging the plaintext into a geometric figure and then rearranging the plaintext according to some route through the figure

Example 18.1. Plaintext: Now is the time for all good men

System: arrange the plaintext by rows into a rectangle of K columns and extract the ciphertext by the columns.

For $K = 5$:

```
N O W I S
T H E T I
M E F O R
A L L G O
O D M E N
```

Ciphertext: NTMAO OHELD WEFLM ITOGE SIRON

Transposition ciphers can be detected by frequency counts — will be the same as for English text.

Definition 18.2. In a *columnar transposition*, the message is arranged horizontally in a rectangle. The key is used to generate a permutation of the columns. The ciphertext is read vertically from the permuted columns.

Example 18.2. key: SCHMID

We use the relative order of the key letters:

```
S C H M I D
6 1 3 5 4 2
```

plaintext: sell all stock on Monday

```
6  1  3  5  4  2
S  E  L  L  A  L
L  S  T  O  C  K
O  N  M  O  N  D
A  Y
```

ciphertext: ESNYL KDLTM ACNLO OSLOA

Analysis

Since these ciphers represent permutations of the plaintext, we can detect such a cipher by using the frequency distribution for the plaintext alphabet. There should be a close correlation.

Example 18.3. Consider the following ciphertext:

```
EOEYE  GTRNP  SECEH  HETYH  SNGND  ODDET
OCRAE  RAEMH  TECSE  USIAR  WKDRI  RNYAC
ANUEY  ICNTT  CEIET  US
```

Note that this ciphertext has 77 letters. This number usually suggests the dimensions of the rectangle, in this case 7×11 or 11×7. However, it could also be, for example, a ragged 8×10 rectangle with the last 3 letters missing. To determine the correct number of rows, we look at all possible values. Since the correct number of rows will result in arranging letters of the same plaintext words in the same row, the ratio of vowels to consonants per row gives a good indication whether our guess is correct. For example, we expect in military text 40% of the letters to be vowels.

For our example, 11 rows has the best vowel frequency.

```
E E G A E R C
O C N E U N N
E E D R S Y T
Y H D A I A T
E H D E A R C
G E D M R A E
T T E H W N I
R Y T T K U E
N H O E D E T
P S C C R Y U
S N R S I I S
```

Once the text is arranged into the suspected rectangle, we need to determine the order in which the columns were taken. We select one column and consider the digrams formed by pairing the remaining columns to its right. For each column, sum the expected frequencies for the digrams formed. The column yielding the highest sum likely indicates the correct pairing.

Table 18.3 contains the distribution of digraphs based on 50000 letters of government plaintext telegrams (reduced to 5000 digraphs)

The following table shows the frequencies of the digraphs formed by putting columns $1, 2, \ldots, 6$ to the right of column 7

1		2		3		4		5		6		7
E	32	E	32	G	0	A	20	E	32	R	4	C
O	18	C	19	N	8	E	57	U	7	N	8	N
E	71	E	71	D	6	R	17	S	19	Y	41	T
Y	41	H	78	D	6	A	28	I	45	A	28	T
E	32	H	14	D	1	E	32	A	20	R	4	C
G	4	E	42	D	60	M	14	R	87	A	35	E
T	27	T	27	E	13	H	0	W	0	N	75	I
R	87	Y	4	T	37	T	37	K	0	U	3	E
N	7	H	78	O	50	E	71	D	6	E	71	T
P	2	S	12	C	3	C	3	R	31	Y	0	U
S	19	N	4	R	5	S	19	I	34	I	34	S
340		381		189		298		281		303		

From our analysis is appears that column 2 follows column 7.

First	Second Letter																									
Letter	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
A	3	6	14	27	1	4	6	2	17	1	2	32	14	64	2	12		44	41	47	13	7	3		12	
B	4				18				2	1		6	1		4			2	1	1	2				7	
C	20		3	1	32	1	14	7		4	5	1	1	41			4	1	14	4		1		1		
D	32	4	4	8	33	8	2	2	27	1		3	5	4	16	5	2	12	13	15	5	3	4		1	
E	35	4	32	60	42	18	4	7	27	1		29	14	111	12	20	12	87	54	37	3	20	7	7	4	1
F	5		2	1	10	11	1		39			2	1		40	1		9	3	11	3		1		1	
G	7		2	1	14	2	1	20	5	1		2	1	3	6	2		5	3	4	2		1			
H	20	1	3	2	20	5			33			1	2	3	20	1	1	17	4	28	8		1		1	
I	8	2	22	6	13	10	19				2	23	9	75	41	7		27	35	27		25		15		2
J	1				2										2						2					
K	1		1		6				2			1		1					1							
L	28	3	3	9	37	3	1	1	20			27	2	1	13	3		2	6	8	2	2	2		10	
M	36	6	3	1	26	1		1	9				13		10	8		2	4	2	2				2	
N	26	2	19	52	57	9	27	4	30	1	2	5	5	8	18	3	1	4	24	82	7	3	3		5	
O	7	4	8	12	3	25	2	3	5	1	2	19	25	77	6	25		64	14	19	37	7	8	1	2	
P	14	1	1	1	23	2		3	6			13	4	1	17	11		18	6	8	3	1	1		1	
Q													1					1			15					
R	39	2	9	17	98	6	7	3	30	1	1	5	9	7	28	13		11	31	42	5	5	4		9	
S	24	3	13	5	49	12	2	26	34		1	2	3	4	15	10		5	19	63	11	1	4		1	
T	28	3	6	6	71	7	1	78	45			5	6	7	50	2	1	17	19	19	5		36		41	1
U	5	3	3	3	11	1	8		5			6	5	21	1	2		31	12	12		1				
V	6				57				12						1					1						
W	12				22			4	13			1		2	19			1	1						1	
X	2		2	1	1	1		1	2					1	1	2		1	1	7						
Y	6	2	4	4	9	11	1	1	3			2	2	6	10	3		4	11	15	1		1			
Z	1				2				1																	

Lecture 19

Product Ciphers

Definition 19.1. The *product* of two ciphers is the result of applying one cipher followed by the other.

Note. The product of two substitution ciphers is a substitution cipher. The product of two transposition ciphers is a transposition cipher.

Shannon suggested applying two simple ciphers with a fixed mixing transformation in between to *diffuse* redundancy into long term statistics, and to confuse the cryptanalyst by making the relation between the redundancy of the ciphertext C and the description of the key K very complex.

Definition 19.2 (Confusion). Applying substitutions in order to make the relationship between the key and ciphertext as complex as possible.

Definition 19.3 (Diffusion). Applying transformations that dissipate the statistical properties of the plaintext across the ciphertext.

Example 19.1. Hayhanen Cipher — named after the Russian spy caught using it in New York. (50's — FBI couldn't break it!)

Plaintext: feb/4/havecontactedfred.willconsultbeforeinvestingincoverbusiness.nat

1. The Hayhanen cipher begins with substitution according to a keysquare, designed so that some letters encode to 1 digit and some to 2 digits, but are unambiguous when read. Letters in the first row (keyword) are represented by only a single number. Letters in the second and third rows (remainder of plaintext alphabet) use two digits, the one on the left first followed by the one above. Note that the numbered rows use the numbers that are *not* used in the first row.

	3	9	6	4	0	8	5	1	7	2
	o	r	i	e	n	t	a	l		
7	b	c	d	f	g	h	j	k	m	p
2	q	s	u	v	w	x	y	z	/	.

   Thus, e encodes to 4, and v encodes to 24.

   We substitute all the letters according to this keysquare:

f	e	b	/	4	/	h	a	v	e	c	o	n	t	a	c	t	e
74	4	73	27	444	27	78	5	24	4	79	3	o	8	5	79	8	4
d	f	r	e	d	.	w	i	l	l	c	o	n	s	u	l	t	
76	74	9	4	76	22	20	6	1	1	79	3	0	29	26	1	8	
b	e	f	o	r	e	i	n	v	e	s	t	i	n	g	i	n	
73	4	74	3	9	4	6	0	24	4	29	8	6	0	70	6	0	
c	o	v	e	r	b	u	s	i	n	e	s	s	.	n	a	t	
79	3	24	4	9	73	26	29	6	0	4	29	29	22	0	5	8	

2. The resulting string of digits is written into a transposition block row by row:

4	7	5	8	2	9	6	1	3
7	4	4	7	3	2	7	4	4
4	2	7	7	8	5	2	4	4
7	9	3	0	8	5	7	9	8
4	7	6	7	4	9	4	7	6
2	2	2	0	6	1	1	7	9
3	0	2	9	2	6	1	8	7
3	4	7	4	3	9	4	6	0
2	4	4	2	9	8	6	0	7
0	6	0	7	9	3	2	4	4
9	7	3	2	6	2	9	6	0
4	2	9	2	9	2	2	0	5
8								

3. The digits are taken out of the block by columns (in the order of the key numbers) and written into a second block using a route cipher. We fill in the area to the left of the line first, then the areas to the right. The line in block forms a step-like pattern: start to the left of column 1 and step to the right until the edge is reached, jump to the left of column 2, etc.

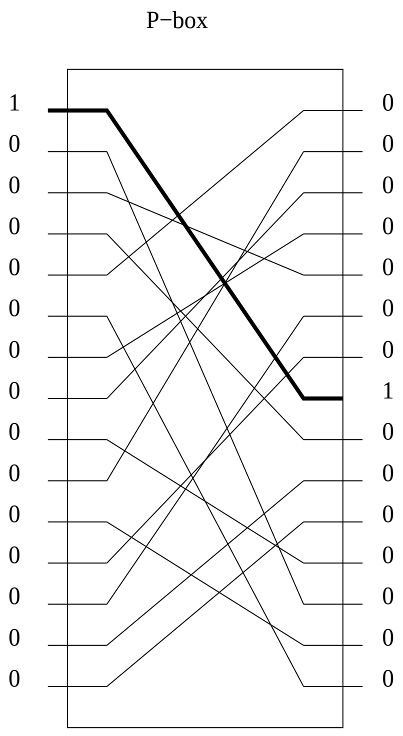
4	6	2	1	5	3
4	**4**	**9**	6	2	9
7	**7**	**8**	**6**	2	4
0	**4**	**6**	**0**	**3**	2
8	**8**	**4**	**6**	**2**	**3**
9	**9**	9	7	2	0
6	**9**	**4**	4	4	6
4	**8**	**6**	**9**	7	2
7	**0**	**7**	**4**	**0**	7
5	**7**	**4**	**7**	**4**	**2**
3	**3**	**2**	**0**	**9**	7
4	**8**	**4**	**7**	**3**	**6**
0	7	0	9	4	2
2	7	2	2	2	5
2	**7**	5	9	1	6
4	**0**	**3**	9	8	3
9	**7**	**2**	**7**	2	2
4	**1**	**1**	**4**		

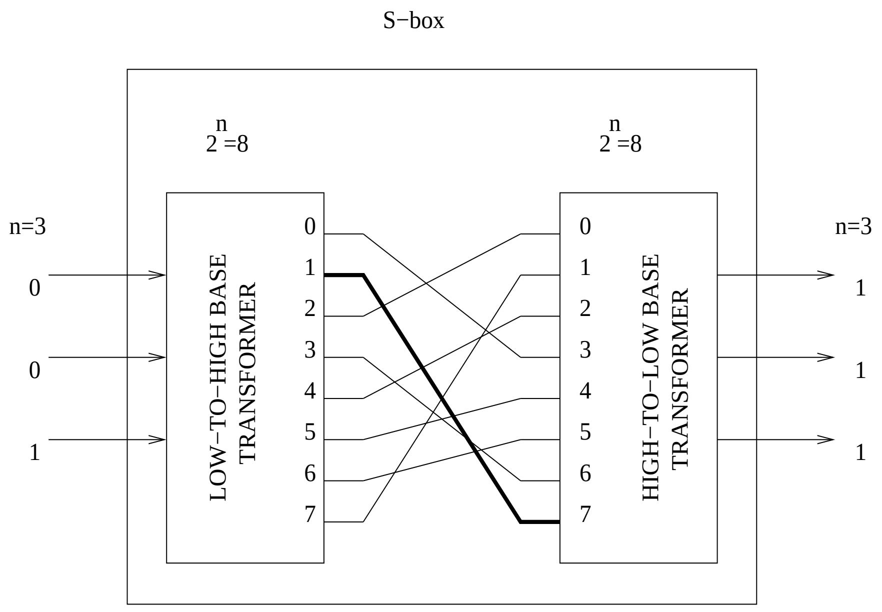
4. Finally the digits are transcribed by columns into groups of five.

   Ciphertext: 66067 49470 79299 74986 49467 42402 53219 42306 27276 25632 47089 64753 40224 94223 22470 49342 18247 48998 07387 77071

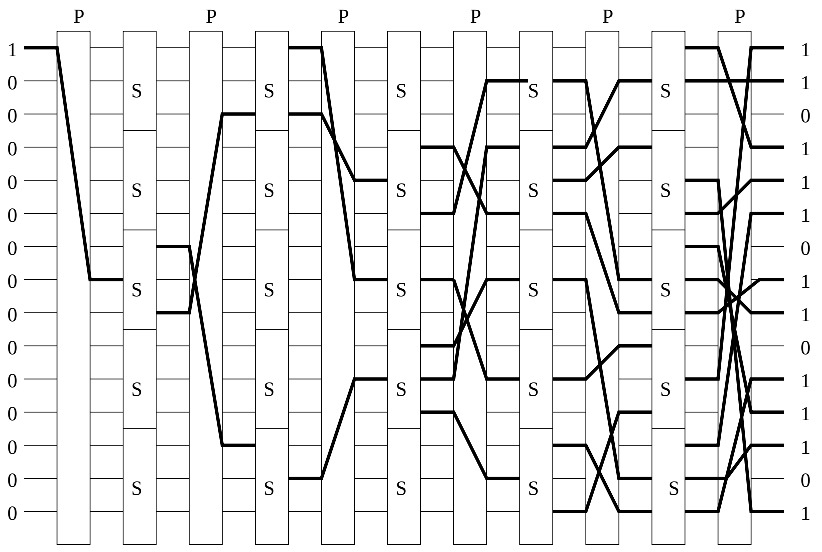
Example 19.2. IBM's Lucifer system. This system uses permutations (transpositions) on large blocks for the mixing transformation, and substitution on small blocks for confusion.

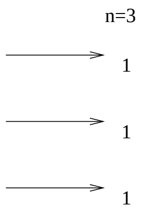
Since this system was set up in hardware, they called the chips which did the permutation "P-boxes" and those that did the substitution "S-boxes."





The Lucifer system simply consisted of a number of P and S boxes in alternation.





Lecture 20

The Data Encryption Standard (DES)

DES was developed by IBM around 1972. The National Bureau of Standards (NBS — now NIST) made solicitations to IBM in 1973/1974 concerning the use of this as a public standard, in response to corporate needs for securing information.

IBM and the National Security Agency (NSA) secretly evaluated DES for security. DES was approved in 1978, and it is still in use. The Advanced Encryption Standard, approved in October 2000, will eventually replace it.

NBS assumptions for DES:

1. The algorithm is assumed completely known by everyone including adversaries (Shannon).
2. An adversary would have a substantial quantity of matched plaintext and ciphertext using a specific, unknown key.

NBS requirements for DES:

1. No method of recovering the key was known other than exhaustive search
2. Such a search was not economically possible (computational security)

Description of DES

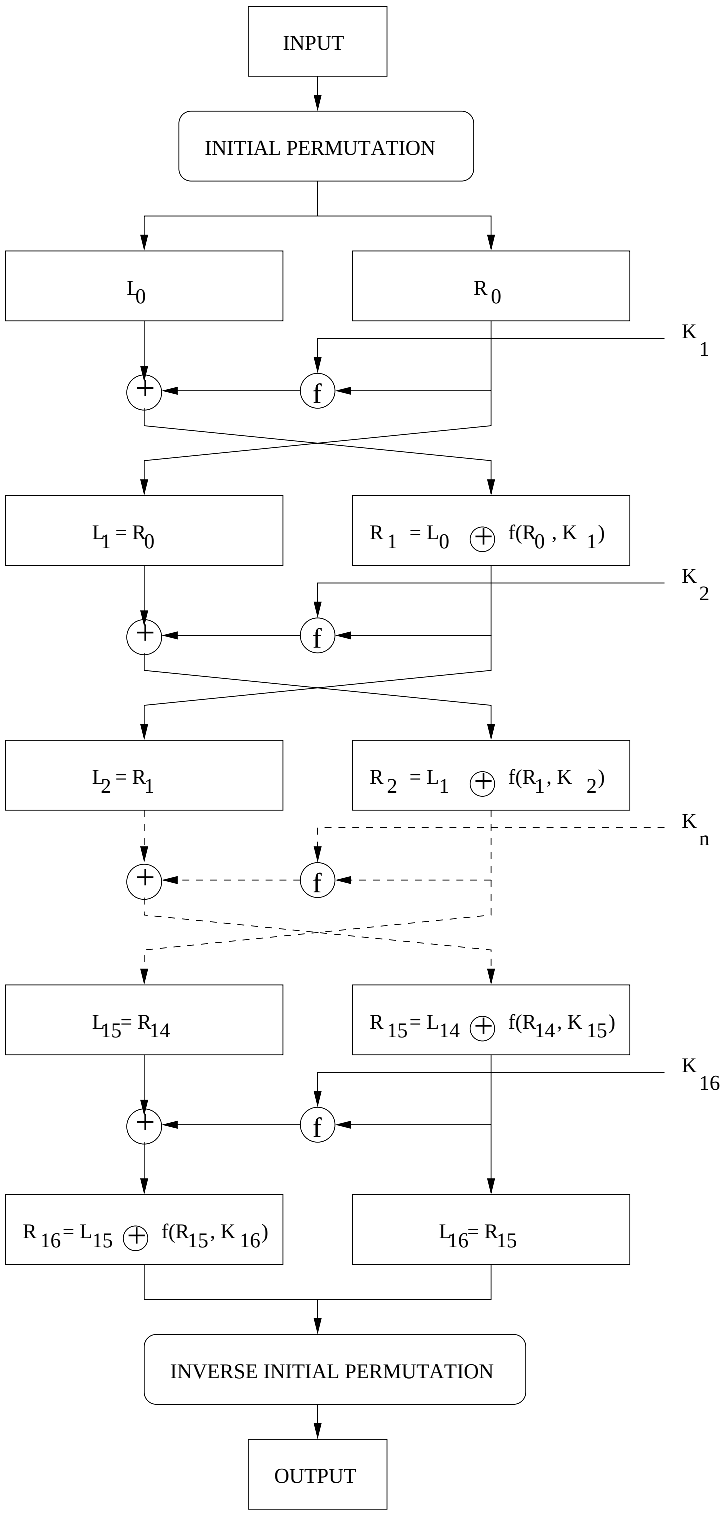
DES encrypts 64-bit blocks of data (i.e., plaintexts have 64 bits) using 64-bit keys. Note that 8 of the key bits are parity bits, resulting in 56 actual bits of the key used for security purposes.

1. The 64 plaintext bits are permuted in a fixed order (transposition cipher).
2. The block is divided into two 32-bit words L_0 and R_0.
3. The block undergoes 16 substitution "rounds." In each round, one word is transformed using XOR and a substitution function, after which the two words are swapped.
4. In the last round, the two words are not swapped.
5. The original permutation is reversed.

Thus

$$DES_{key}(M) = IP^{-1}(S_{16}(S_{15}(\ldots(S_2(S_1(IP(M))))\ldots))) \ .$$

Diagram of DES:


INPUT
INITIAL PERMUTATION
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
f
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
K_n
f
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
f
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
INVERSE INITIAL PERMUTATION
OUTPUT

Initial Permutation IP

See FIPS publication. Notation: first bit of the output is the 58th bit of the input.

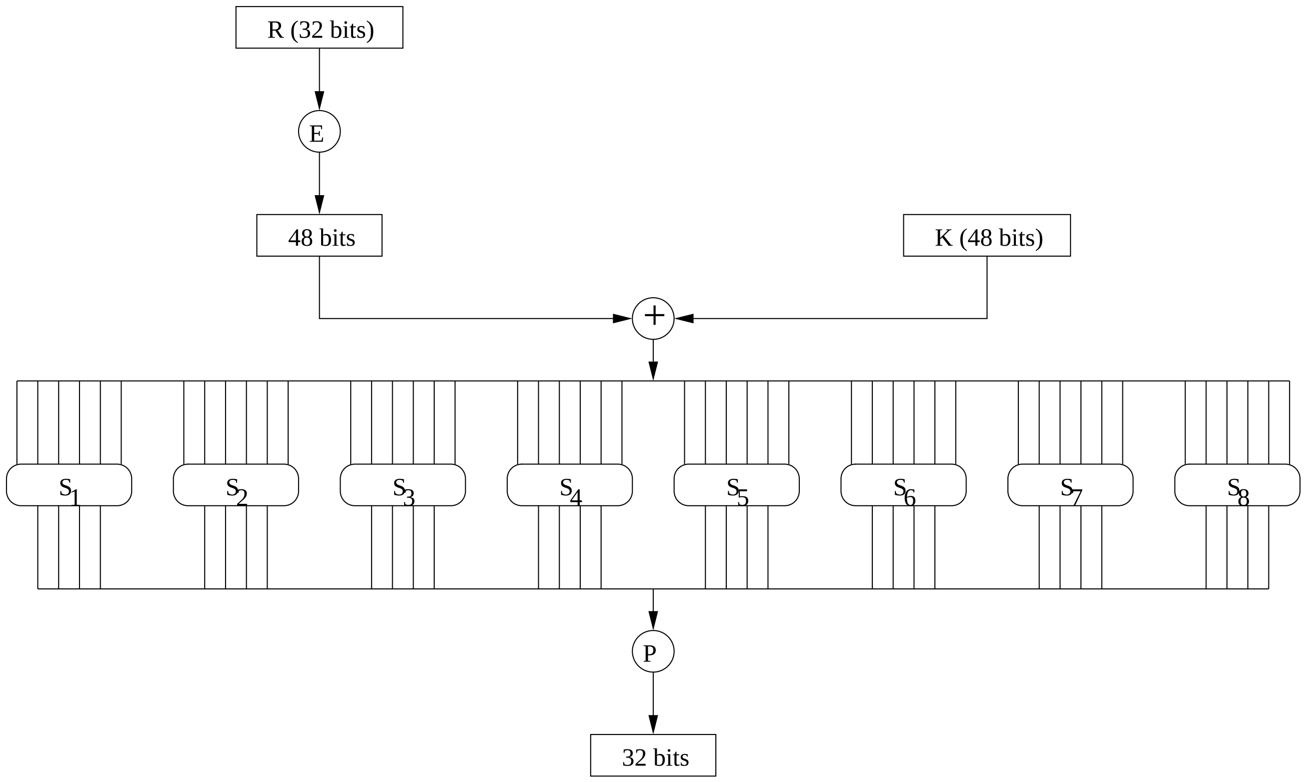
Substitution Rounds

In round i, the right word R_{i-1} is combined with the ith subkey K_i via the function f. The output of f is XORed with L_{i-1} to form R_i. The next left word L_i is the previous right word ($L_i = R_{i-1}$).

The Function f

f accepts as input R_i (32 bits) and the ith round subkey K_i (48 bits). The subkey generation is described below. The function f works as follows:

1. R_i is expanded to the 48 bit $R_i^{'}$ via the expansion function E, which simply repeats some of the bits of R_i in generating $R_i^{'}$ (see the FIPS publication for the specification of E).
2. $R_i^{'}$ is XORed with K_i (both are 48 bits long).
3. $R_i^{'} \oplus K_i$ is broken into 8 6-bit words. Each of these words is replaced by a 4-bit word according to the 8 (different) S-boxes $S_1, S_2, \ldots, S_8$. The result of applying the S-boxes is a 32-bit string.
4. The 32-bit string is permuted according to the fixed permutation P (see the FIPS publication).



Generation of the Subkeys K_i

1. The key K (56 bits) is permuted according to the fixed permutation "PERMUTED CHOICE 1" (see FIPS publication) and separated into two 28-bit words C_0 and D_0.
2. Each word is rotated either one or two places to the left according to the fixed schedule below, yielding C_1 and D_1.

Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of left shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3. K_1 is obtained from C_1 and D_1 via "PERMUTED CHOICE 2," which selects 48 bits from C_1 and D_1 according to a fixed ordering.
4. Steps 2 and 3 are repeated with C_i and D_i to obtain the remaining 15 subkeys.

Formal Notation and Decryption

$K_n = KS(n, key)$ (K_n is the nth subkey, KS is the key schedule)

Note:

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_{i+1}) \quad i = 0, 1, \ldots, 15 \\ L_i &= R_{i+1} \oplus f(R_i, K_{i+1}) \\ C &= IP^{-1}(R_{16}, L_{16}) \\ IP(C) &= (R_{16}, L_{16}) \end{aligned}$$

(IP^{-1} is the inverse of the initial permutation function)

Suppose $DES_{key}(M) = C$. Denote by

$$K_i^{'} = KS(17 - i, key), \quad i = 1, 2, \ldots, 16$$

(the key schedule in reverse order). Now run the DES device on C, using $K_i^{\prime}$ instead of K_i. We have $IP(C) = (R_{16}, L_{16})$, and thus

$$\begin{aligned} L_0^{'} &= R_{16} \\ L_1^{'} &= R_0^{'} = L_{16} \\ &= R_{15} \end{aligned} \qquad \begin{aligned} R_0^{'} &= L_{16} \\ R_1^{'} &= L_0^1 \oplus f(R_0^{'}, K_1^{'}) \\ &= R_{16} \oplus f(L_{16}, K_{16}) \\ &= R_{16} \oplus f(R_{15}, K_{16}) \\ &= R_{16} \oplus R_{16} \oplus L_{15} \\ &= L_{15} \end{aligned}$$

In fact, by continuing this argument we get

$$L_i^{'} = R_{16-i} \quad R_i^{'} = L_{16-i}$$

and hence

$$IP^{-1}(R_{16}^{'}, L_{16}^{'}) = IP^{-1}(L_0, R_0) = M$$

Decryption of DES is simply running the DES algorithm on C with the reverse key schedule.

Note. The invertibility of DES is *independent* of the function f. Regardless of what function is used for f, decryption of DES works exactly as described above. This works largely because the individual parts of DES are *involutions* — functions that are their own inverses ($g(g(x)) = x$)

Lecture 21

Strengths and Weaknesses of DES

Under DES, encryption of 0 (plaintext consisting of 64 zeros) under the key 0 produces:

```
Plaintext   0000  0000  0000  0000 (in hex)
L1 R1       0000  0000  D8D8  DBBC
L2 R2       D8D8  DBBC  E73A  ED4F
                    ⋮
L16 R16     BBEA  0DC2  1C20  87FC
Ciphertext  8CA6  4DE9  C1B1  23A7
```

Error Propagation

Modern desiderata: small modifications in the plaintext should cause a lot of modification in the corresponding ciphertext. In particular, we want that changing one bit of plaintext causes one half of the ciphertext bits to change on average.

In the following example, the keys are the same and the plaintexts differ only in the last bit:

$K = 0^{56}$	$P = (10)^{32}$	$C =$ 3AE7 1695 4DC0 4E25
$K = 0^{56}$	$P = (10)^{31}(11)$	$C =$ 17D8 E9C3 74D1 4494

Note that the ciphertexts differ in 34 bits. Statistically, they are as far apart from each other as random sequences.

Similarly, we want that changing one key bit causes one half of the ciphertext bits to change on average:

$K = 0^{56}$	$P = (01)^{32}$	$C =$ B109 FD80 3EB2 D05E
$K = 0^{55}1$	$P = (01)^{32}$	$C =$ 451F 0C33 F24F B8DC

Again, 34 bits differ between the two ciphertexts.

21.1 Cryptanalysis of DES

1. Exhaustive search: There are $2^{56} \approx 10^{17}$ possible keys. If one key is tested every μsec, the complete search would take 2.2 centuries in the worst case (1.1 centuries on average). Old estimate — RIJNDAEL encrypts a 128-bit block in less than 0.5 μsec.

2. Parallelism: A search machine consisting of 10^6 chips each testing one key per μsec (total of 10^{12} keys per second) would find a key in one day. Electronic Frontier Foundation has built a DES cracker for \$250000 which finds a single DES key in 56 hours (tests 8800 keys per μsec). A combination of the DES cracker and 100000 PC's on the internet has found a DES key in 22.25 hours (tests 245000 keys per μsec).

3. Time-memory tradeoff (Hellman): Shorten time by using a lot of memory — applies to any brute-force attack on a cryptosystem.

   Fix the following notation:

$$\begin{aligned} P &= \text{64-bit plaintext} \\ C &= \text{64-bit ciphertext} \\ K &= \text{56-bit key} \\ DES: C &= S_K(P) \end{aligned}$$

   Let P_0 be a fixed plaintext block, for example, 8 ASCII blanks (this is a chosen-text attack).

   Define $f(K) = R(S_K(P_0))$, where R is a reducing function which throws away 8 bits.

   Start with m starting points $SP_1, SP_2, \ldots, SP_m$ selected at random from the key space $\{0, 1, \ldots, 2^{56} - 1\}$. For $1 \le i \le m$ let $X_{i,0} = SP_i$. Compute $X_{i,j} = f(X_{i,j-1})$ for $1 \le j \le t$:

$$\begin{aligned} SP_1 = X_{1,0} &\to X_{1,2} \to \cdots \to X_{1,t} = EP_1 \\ SP_2 = X_{2,0} &\to X_{2,2} \to \cdots \to X_{2,t} = EP_2 \\ &\vdots \\ SP_m = X_{m,0} &\to X_{m,2} \to \cdots \to X_{m,t} = EP_m \end{aligned}$$

   The endpoints $EP_i = f^{(t)}(SP_i)$. Discard all the intermediate points and sort

$$S = \{(SP_i, EP_i) \mid i = 1, 2, \ldots, m\}$$

   on the end points in ascending order. The steps up to this point amount to a precomputation *before* starting the real attack.

   Now, suppose someone selects a key K, and the cryptanalyst intercepts $C_0 = S_K(P_0)$ (i.e., he knows which ciphertext block C_0 corresponds to P_0). Let $Y_1 = R(C_0) = f(K)$. He then checks whether Y_1 is one of the endpoints in S (very fast since S is sorted). If $Y_i = EP_i$, then either $K = X_{i,t-1}$ or EP_i has more than one inverse under f. The cryptanalyst can then check for a false solution by checking that $C_0 = S_K(P_0)$ for $K = X_{i,t-1}$ (he needs P_0 at this point). If no endpoint is Y_i or we have a false solution, try $Y_2 = f(Y_1)$ and compare Y_2 to the endpoints. If $Y_2 = EP_i$ then $K = X_{i,t-2}$ or we have a false solution. Continue the process on Y_3, etc. until we find K.

Theorem 21.1. *(Hellman) If $f(i)$ is a random function and $t^2 m = N = 2^{56}$, the expected probability of success by this method is*

$$\frac{mt}{N} = \frac{1}{t} \ .$$

Here $N = 2^{56}$, so if we set $t = \sqrt[3]{N}$ we get $1/t \approx 10^{-6}$ (small probability of success). Thus, in order to get an expected probability of success closer to 1, we compute $t \approx 10^6$ tables like S, where for each of these tables we use a different reducing function R. This yields:

Expected time:	$tt = t^2$
Expected memory:	$mt = t^2$ (since $m = t = \sqrt[3]{N}$)

Lecture 22

Analytic Attacks on DES

If a cryptosystem is *linear* then

$$C = AP + BK$$

where C is the ciphertext, P is the plaintext, K is the key, and A and B are matrices determined by the system (A and B are public). Note that B may or may not be square. To decrypt:

$$P = A^{-1}(C - BK) \ .$$

A cryptanalyst can easily mount a known text attack (P and C known) as follows:

$$\begin{aligned} BK &= C - AP \\ B^T BK &= B^T(C - AP) \\ K &= (B^T B)^{-1} B^T (C - AP) \end{aligned}$$

Thus, linear cryptosystems are not secure (Hill cipher is a simple example). What about DES?

If DES were linear, we would have the following matrix sizes:

$$A : 64 \times 64, \quad B : 64 \times 56,$$
$$K : 56 \times 1, \quad P : 64 \times 1, \quad C : 64 \times 1$$

In DES, the expansion operation E, the permutations, and XOR are all linear. All that is left is the S-boxes — if they are also linear, then DES is linear (and hence insecure).

If the S-boxes are *affine*, i.e., $y = Gx + h$, where

G is a 4×6 matrix
h is a 4×1 matrix
x is the 6×1 input
y is the 4×1 output,

then we can find a matrix H such that for DES

$$\begin{aligned} C &= AP + BK + H \\ K &= (B^T B)^{-1} B^T (C - AP - H) \ . \end{aligned}$$

Thus, an affine cipher is as easy to break as a linear cipher using KTA (note A, B, and H are public).

Suppose the DES S-boxes were affine. If we modify only 4 entries in each of them, the resulting S boxes need no longer be affine, but affine cryptanalysis would nevertheless yield a solution one time in 3870. To see this, note that there are 60/64 unmodified entries in each S-box, 8 S-boxes in total, and 16 rounds, so the probability of running DES without accessing any of the modified S-box entries is

$$\left(\frac{60}{64}\right)^{8\times 16} \approx \frac{1}{3870} \ .$$

Thus, linearity or affineness can be exploited if the cryptosystem is *close* to being affine or linear. This technique is called *linear cryptanalysis* (M. Matusi, EUROCRYPT 1993), and will break DES in time 2^{43} (better than exhaustive search), and Matusi actually used this method to become the first person to recover a DES key in 50 days using twelve workstations. However, this attack is a known text attack and requires 2^{43} matched pairs of plaintext and ciphertext (not very practical in general).

Differential cryptanalysis (Biham and Shamir, Journal of Cryptology, 1991) can also be used to break DES slightly faster than by brute force (2^{47}), but it is a chosen text attack requiring 2^{47} chosen plaintext/ciphertext pairs.

Large-scale, parallel, brute-force attack is the most practical attack.

22.1 Strengths of DES

1. No S-box is affine
2. No portion of an S-box is affine.
3. The overall algorithm is not affine.
4. No S-box is translation invariant ($S(x) \neq S(x+E)$)
5. Changing a single input bit to any S-box causes at least two output bits to change.
6. P and E are coupled to guarantee that the four outputs of each S-box are taken to six different S-boxes at the next round.
7. Rapid error-propagation hinders a key clustering attack (testing a smaller set of similar keys)

22.2 Improving DES

1. Use less structured S-boxes. Less structure $\Longrightarrow$ more confusion.
2. Increase the number of rounds. This will relax the need for structured S-boxes.
3. Introduce the key in a more complex manner than XOR and add other non-linearity besides the S-boxes.
4. Have PERMUTATED CHOICE 2 mix the C and D registers.
5. Make the key scheduling algorithm non-linear.
6. Increase the key size (128 bits)

Extra security for DES can also be obtained by multiple encryption with different keys:

$$C = S_{K_3}(S_{K_2}^{-1}(S_{K_1}(P))),$$

since $S_{K_2}S_{K_1} = S_{K_3}$ does not seem to occur for DES.

Lecture 23

Modes of Operation for Block Ciphers

Definition 23.1. A *block cipher* divides the plaintext into blocks (usually of a fixed size) and operates on (encrypts/decrypts) each block independently. Thus, a particular plaintext block will always be encrypted into the same ciphertext.

Note. The Caesar cipher (blocks of one character), DES (blocks of 64-bits), and RIJNDAEL (blocks of 128, 192, or 256 bits) are all examples of block ciphers.

A block cipher is a substitution cipher, and it must have a *large* alphabet to foil frequency analysis. In any cipher, each bit of ciphertext should be a function involving all bits of the plaintext and key. The cipher should be designed so that changing a single bit of the plaintext or the key causes 50% of the ciphertext bits to change on average (error propagation). This is useful in authentication and makes it improbable that an opponent can make undetected modifications to encrypted data unless he knows the key.

23.1 Cipher Block Chaining (CBC)

CBC is one method of providing greater security to block ciphers. It prevents repeated plaintext blocks from having the same ciphertext.

We start with an initial *random* block C_0, which is sent in encrypted form to the intended recipient. Each plaintext block is then encrypted using

$$C_i = E_K(P_i \oplus C_{i-1}) \quad i \geq 1$$

and decryption is performed using

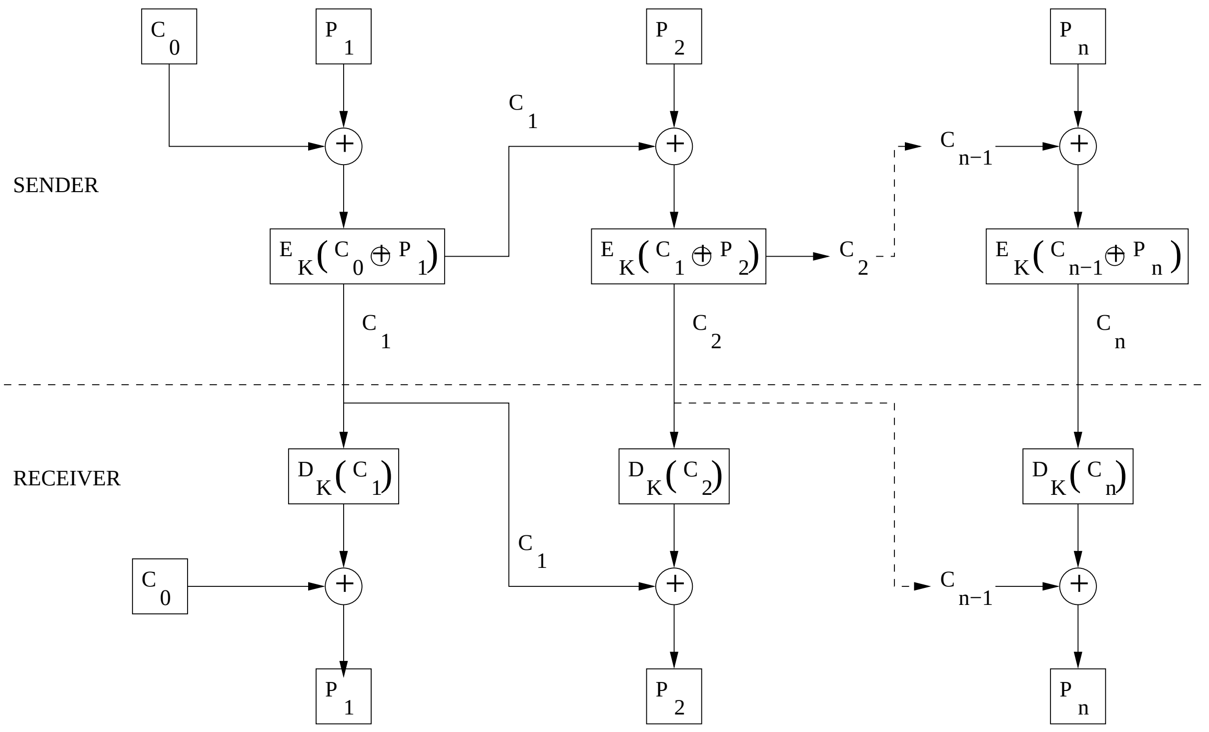
$$P_i = D_K(C_i) \oplus C_{i-1}$$

The process of combining P_i with C_{i-1} in this manner is called *pre-whitening*, since it scrambles the message with random data before encryption.

Features of CBC

1. Repeated plaintext sequences will be encrypted differently in different repetitions.
2. By varying C_0, we encrypt the same message differently.
3. CBC has limited error propagation in decrypting. A modification in decrypting a block propagates only to the next block.

Diagram of CBC



23.2 Plaintext Block Chaining

Similar to CBC, but has unlimited error propagation.

Send random P_0 (in encrypted form) to the intended recipient. Then

$$C_i = E_K(P_i \oplus P_{i-1})$$
$$P_i = D_K(C_i) \oplus P_{i-1} \quad .$$

Features of PBC

1. Repeated groups of plaintext blocks are encrypted the same after a one-block delay. Suppose we have the following sequence of plaintext blocks

$$P_0 P_1 P_2 \ldots P_{k-1} P_k P_{k+1} P_{k+2} \ldots$$

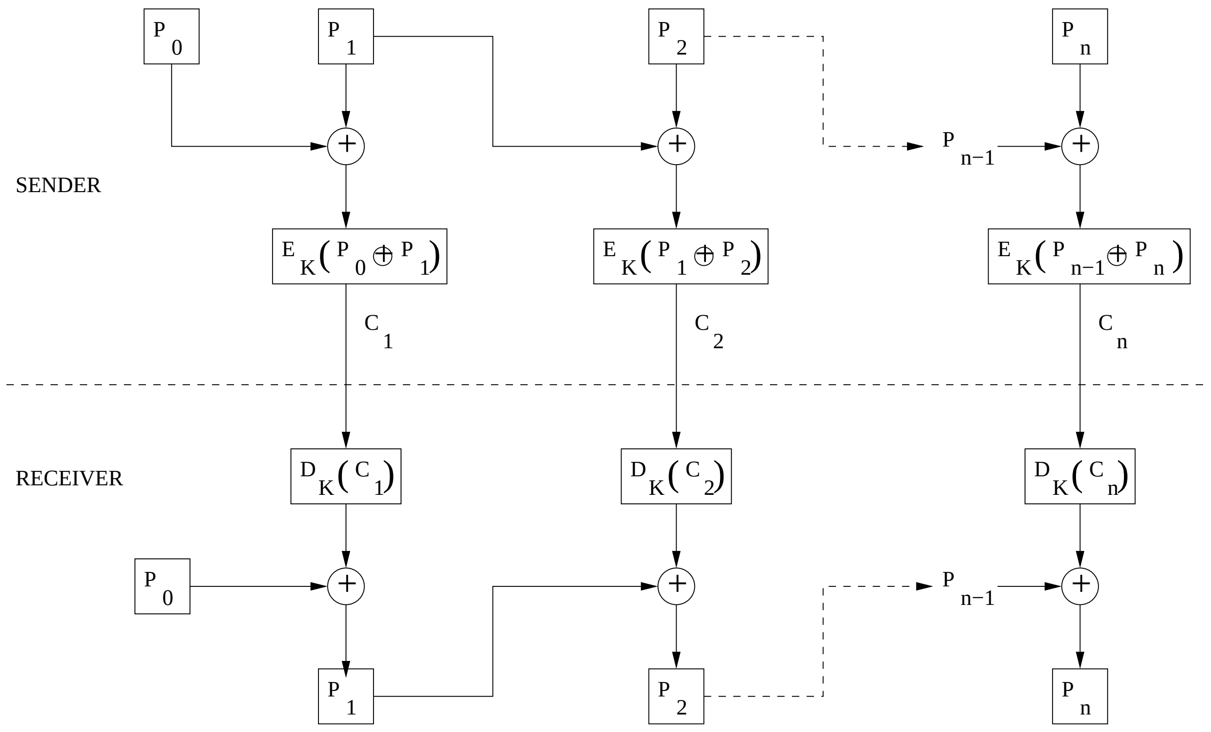
and that $P_1 = P_{k+1}$, $P_2 = P_{k+2}$, etc... To encrypt P_1 we encrypt $P_0 \oplus P_1$ and to encrypt P_{k+1} we encrypt $P_k \oplus P_{k+1}$. Since P_0 and P_k are different, the first block of repeated plaintext is encrypted differently. However, for P_2 and P_{k+2} we encrypt $P_1 \oplus P_2$ and $P_{k+1} \oplus P_{k+2}$, which will result in the same ciphertext since

$$P_1 \oplus P_2 = P_{k+1} \oplus P_{k+2}$$

($P_1 = P_{k+1}$ and $P_2 = P_{k+2}$).

2. Unlimited error propagation — good set up to protect against the playback threat (unauthorized re-use of an encrypted message). Simply add data/time authentication field at the end of the message (to make sure the message was not scrambled during transmission).

Diagram of PBC



To avoid the disadvantages of CBC and PBC, we can combine them. Send encrypted P_0 and C_0 (both random). Then

$$\begin{aligned} C_i &= E_K(P_i \oplus C_{i-1} \oplus P_{i-1}) \\ P_i &= D_K(C_i) \oplus C_{i-1} \oplus P_{i-1} \quad . \end{aligned}$$

Lecture 24

Stream Ciphers

Stream ciphers do not treat incoming characters independently. Each character accepted as input is enciphered into an output character in a manner which depends on the internal state of the device. After each character is enciphered, the device changes state according to some rule. Therefore, two occurrences of the same plaintext character will usually not result in the same ciphertext character.

There are two types of such ciphers, *synchronous* and *self-synchronous* stream ciphers.

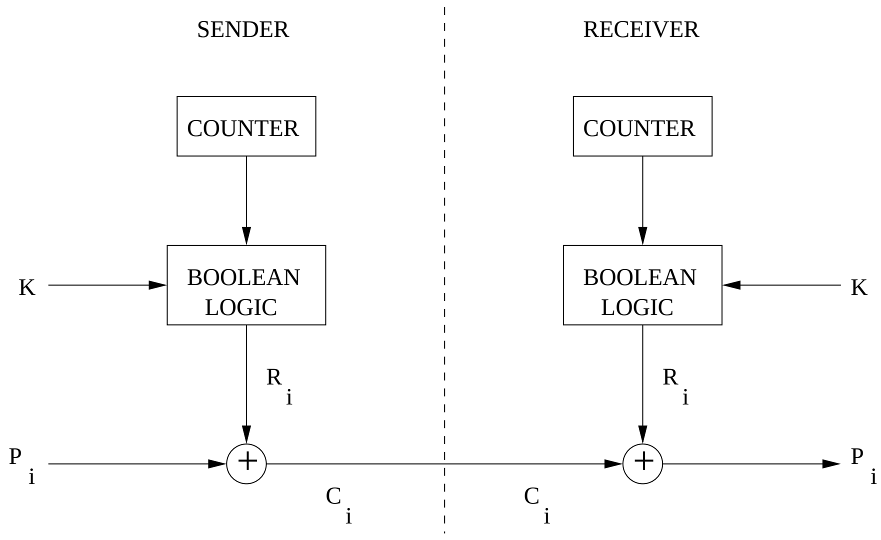
24.1 Synchronous Stream Ciphers (SSC)

In a SSC, the next state depends only on the previous state and not on the input, so that the progression of states is independent of the sequence of characters received. The output corresponding to a particular input depends only on the input and its position in the input sequence, not on the characters enciphered before or after it.

Note. The one-time pad, in a sense, is an SSC.

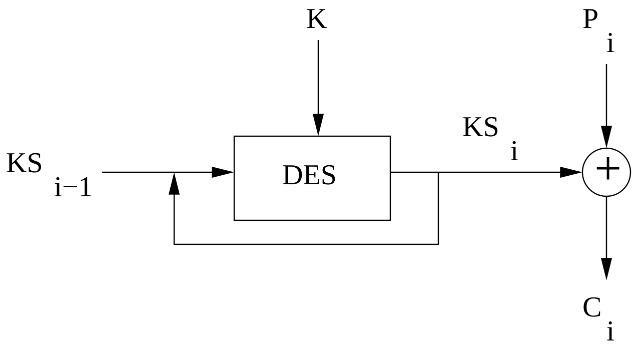
Diagram of a SSC

The boolean logic should produce a pseudo-random sequence synchronized by the key.

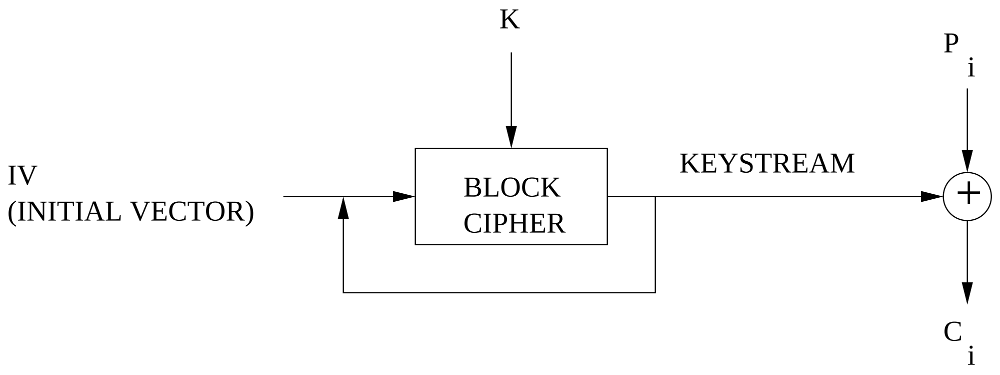


DES as an SSC

Start with a random keystream KS_0 sent to the receiver in the clear. Compute $KS_i = E_K(KS_{i-1})$. Then $C_i = P_i \oplus KS_i$.



Note that we can create an SSC from any block cipher in this fashion, substituting the block cipher for DES in the above diagram. (Example: Cryptonomicon)

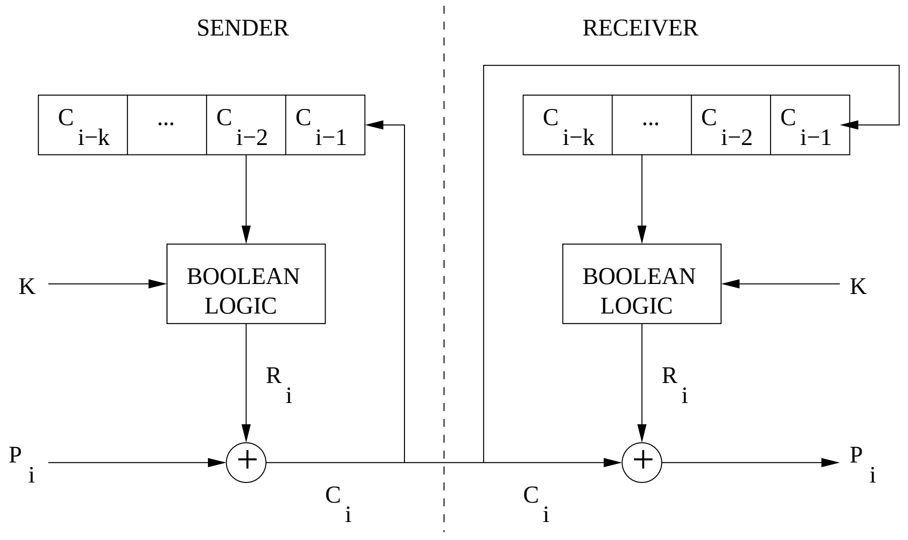


Problems:

1. No error propagation
2. Loss of one character between sender and receiver destroys synchronization (no memory)

Self-Synchronizing Stream Ciphers

Similar to synchonous stream ciphers, except the counter is replaced by a register containg the previous k ciphertexts. Limited error propagation, self-synchronizing after an initial start-up period (k steps). Can also be implemented with a block cipher as above.



Error Propagation

1. A single error in a cipher text block causes the deciphered block to be totally garbled. Error propagation is necessary in a block system to foil a key clustering attack.
2. A synchronous system entails no error propagation.
3. A self-synchronizing system has limited error propagation (k steps).

24.2 Certified DES Modes

DES was certified by the NSA to be used in three modes:

1. Electronic Codebook Mode (ECB): straightforward block-by-block encryption using DES. Because the same text produces the same ciphertext, this mode is only recommended for sending isolated blocks of random or pseudo-random data.
2. CBC (cipher block chaining): Start with a 64-bit random initialization vector (IV), which is sent to the receiver in ECB form. IV can be used repeatedly as long as the DES key does not change.
3. CFB (cipher feedback): Usually k bits are fed back, where $k = 8$. IV is at least 48 random bits, right-justified, padded with 0. Each cryptographic session requires a different IV, by these may be sent in the clear.

24.3 Authentication

Problem: prevent unauthorized individuals from injecting information in a public channel.

Let A_k be an authentication function. Desired properties:

1. Given K and D, $A_K(D)$ is easy to evaluate.

2. $A_K(D)$ should be difficult to evaluate without K.

3. Given $A_K(D)$ and D it should be very difficult to find another piece of data D' for which the authentication function is the same $(A_K(D) = A_K(D'))$.

Authentication Modes for DES

1. CBC Authentication (CBCA): Encrypt the message using CBC, throwing away all the cipher blocks except the last. The three left-most bytes of this block form the *message authentication code* (MAC, or hash value). Send the plaintext plus the MAC. The recipient separately encrypts the message, and compares the left-most three bytes of the last cipher block she obtains with the MAC. If they match, the message that was sent is authentic.

2. CFB Authentication (CFBA): Same as CBCA, but use CFB for encryption.

Lecture 25

Some Number Theoretic Results

25.1 Linear Diophantine Equations

Solve

$$ax + by = 1 \tag{25.1}$$

given $a, b \in \mathbb{Z}$, $b > 0$, and $\gcd(a, b) = 1$. If $\gcd(a, b) \neq 1$, (25.1) is insoluble. If $b < 0$, use $-b$ and solve for $(x, -y)$.

Euclidean Algorithm

$$\begin{aligned}
a &= bq_0 + r_0 & q_0 &= \lfloor a/b \rfloor, 0 < r_0 < b \\
b &= r_0 q_1 + r_1 & q_1 &= \lfloor b/r_0 \rfloor, 0 < r_1 < r_0 \\
r_0 &= r_1 q_2 + r_2 & q_2 &= \lfloor r_0/r_1 \rfloor, 0 < r_2 < r_1 \\
&\vdots \\
r_{n-3} &= r_{n-2} q_{n-1} + r_{n-1} & r_{n-1} &= \gcd(a, b) \\
r_{n-2} &= r_{n-1} q_n + r_n & r_n &= 0
\end{aligned}$$

Repeated division with remainder. Notice that the sequence of remainders (the r_i) is strictly decreasing, and thus the sequence is finite (algorithm terminates).

Theorem 25.1 (Lamé, 1844). $n < 5 \log_{10} \min(a, b)$ *(i.e., Euclidean algorithm is* $O(\log \min(a, b))$*)*

Let $A_{-2} = 0$, $A_{-1} = 1$, $B_{-2} = 1$, $B_{-1} = 0$ and

$$\begin{aligned}
A_k &= q_k A_{k-1} + A_{k-2} \\
B_k &= q_k B_{k-1} + B_{k-2}
\end{aligned}$$

for $k = 0, 1, \ldots$. We have $A_n = a$ and $B_n = b$ (n from above), and

$$A_k B_{k-1} - B_k A_{k-1} = (-1)^{k-1} \ .$$

Putting $k = n$ yields

$$\begin{aligned}
A_n B_{n-1} - B_n A_{n-1} &= (-1)^{n-1} \\
a(-1)^{n-1} B_{n-1} + b(-1)^n A_{n-1} &= 1 \ .
\end{aligned}$$

Thus, a solution of (25.1) is given by

$$x = (-1)^{n-1} B_{n-1}, \quad y = (-1)^n A_{n-1} \ .$$

Computing x, y in this manner is known as the *Extended Euclidean Algorithm.*

Consider the linear congruence $ax \equiv 1 \pmod{m}$ given $a, m \in \mathbb{Z}$, $\gcd(a, m) = 1$ (i.e., finding $a^{-1} \pmod{m}$). We want x such that

$$m \mid ax - 1 \implies ax - 1 = ym \implies ax - my = 1$$

(can be solved using Euclidean Algorithm).

Example 25.1. For $412x \equiv 1 \pmod{317}$ we obtain $x = -10$, so $x \equiv 307 \pmod{317}$ is a solution.

25.2 The Power Algorithm (Binary Exponentiation)

Evaluate $a^n \pmod{m}$ given a, n, m.

Let $n = b_0 2^k + b_1 2^{k-1} + \cdots + b_{k-1} 2 + b_k$ be the binary expansion of n, i.e., $b_0 = 1$, $b_i \in \{0, 1\}$, $1 \le i \le k$, $k = \lfloor \log_2 n \rfloor$. Given $b_0, \ldots, b_k$, we can evaluate n efficiently ($O(\log_2 n)$) using *Horner's Method*:

$$n = 2(\ldots(2(2b_0 + b_1) + b_2) \cdots + b_{k-1}) + b_k \ .$$

Define $s_0 = b_0$, $s_{i+1} = 2s_i + b_{i+1}$. Then

$$\begin{aligned}
s_0 &= b_0 \\
s_1 &= 2b_0 + b_1 \\
s_2 &= 2(2b_0 + b_1) + b_2 = 2^2 b_0 + 2b_1 + b_2 \\
s_3 &= 2(2(2b_0 + b_1) + b_2) + b_3 = 2^3 b_0 + 2^2 b_1 + 2b_2 + b_3 \\
&\vdots \\
s_k &= n \ .
\end{aligned}$$

Define $r_0 \equiv a^{s_0} \pmod{m}$ and $r_i \equiv a^{s_i} \pmod{m}$. Then $r_k \equiv a^{s_k} \equiv a^n \pmod{m}$ and we can compute r_k iteratively as follows:

$$\begin{aligned}
r_0 &\equiv a^{s_0} \equiv a \pmod{m} \\
r_1 &\equiv a^{s_1} \equiv a^{2b_0 + b_1} \equiv (a^{s_0})^2 a^{b_1} \equiv (r_0)^2 a^{b_1} \pmod{m} \\
&\vdots \\
r_{i+1} &\equiv a^{s_{i+1}} \equiv a^{2s_i + b_{i+1}} \equiv (a^{s_i})^2 a^{b_{i+1}} \equiv (r_i)^2 a^{b_{i+1}} \pmod{m} \ .
\end{aligned}$$

This gives us an $O(\log_2 n)$ algorithm to compute a^n — start with $r_0 = a$ and for $1 \le i < k$ compute

$$r_{i+1} = \begin{cases} r_i^2 \mod m & \text{if } b_{i+1} = 0 \\ r_i^2 a \mod m & \text{if } b_{i+1} = 1 \end{cases} .$$

Note that there at most $2 \log_2 n$ multiplications, and no operands larger than m^2.

Lecture 26

More Number Theoretic Results

26.1 Euler's ϕ Function

Let m be a positive integer. We define

$$\phi(m) = |\{r \in \mathbb{Z} \mid 0 < r < m, \gcd(r, m) = 1\}| \ .$$

$\phi(m)$ is the number of integers between 1 and $m - 1$ which have multiplicative inverses modulo m.

Example 26.1. $\phi(42) = |\{1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41\}| = 12$

Let p be a prime. Then

$$\begin{aligned} \phi(p) &= p - 1 = p^0(p-1) \\ \phi(p^2) &= p^2 - p = p^1(p-1) \\ \phi(p^n) &= p^n - p^{n-1} = p^{n-1}(p-1) \ . \end{aligned}$$

Let m be composite.

Theorem 26.1. *If* $\gcd(m_1, m_2) = 1$, *then* $\phi(m_1 m_2) = \phi(m_1)\phi(m_2)$.

Proof. Omitted (uses Chinese Remainder Theorem). **QED**

Corollary 26.2. *If the prime factorization of m is given by*

$$m = \prod_{i=1}^{k} p_i^{\alpha_i}, \quad p_i \textit{ prime},$$

then

$$\phi(m) = \prod_{i=1}^{k} \phi(p_i^{\alpha_i}) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1) \ .$$

Example 26.2. $\phi(42) = \phi(2 \times 3 \times 7) = \phi(2)\phi(3)\phi(7) = (1)(2)(6) = 12$.

Theorem 26.3 (Euler). *If* $\gcd(a, m) = 1$, *then*

$$a^{\phi(m)} \equiv 1 \pmod{m} \ .$$

Theorem 26.4 (Fermat). *If p is prime, then $\phi(p) = p - 1$ and if $p \not| a$*

$$a^{p-1} \equiv 1 \pmod{p} \ .$$

Fermat's Theorem gives rise to a fast probabilistic primality test using binary exponentiation:

- If $a^{N-1} \equiv 1 \pmod{N}$ for a few small primes $a \not| N$, then N is probably prime (base a pseudoprime).
- If $a^{N-1} \not\equiv 1 \pmod{N}$ for any prime $a \not| N$, then N is composite.

Unfortunately, there are composite numbers (called *Charmichael numbers*) for which $a^{N-1} \equiv 1 \pmod{N}$ for all primes a. An example is 1729. Thus, this method cannot prove primality.

26.2 Primitive Roots

Definition 26.1. For a prime p, a ***primitive root modulo*** p is a number g, $0 < g < p$, such that $p - 1$ is the smallest positive integer k such that $g^k \equiv 1 \pmod{p}$.

Note. A primitive root g is alternatively called a generator of the cyclic group of residue classes modulo p. Generators yield the longest possible cycle of powers modulo p.

Example 26.3. Is $a = 3$ a generator modulo $p = 7$? By tabulating the powers of $a \bmod p$ we get

$$3^0 = 1, \quad 3^1 = 3, \quad 3^2 = 2, \quad 3^3 = 6, \quad 3^4 = 4, \quad 3^5 = 5, \quad 3^6 = 1, \quad 3^7 = 3, \ldots$$

(sequence repeats). Since 6 is the smallest power of a yielding 1, 3 is a primitive root modulo 7. 5 is also a primitive root modulo 7. There are no others (e.g. 2 has period 3).

Note. If g is a primitive root and $\gcd(a, p) = 1$, then $g^i \equiv a \pmod{p}$ for some i with $0 \leq i < p - 1$.

Theorem 26.5. *If p is an odd prime, there are exactly $\phi(p-1)$ primitive roots of p.*

Proof. Omitted. **QED**

Computing Primitive Roots

Suppose p is a large prime (otherwise exhaustive search is easier). Select any small prime q and calculate $q^{(p-1)/r_i} \pmod{p}$ for each prime divisor r_i of $p-1$. If $q^{(p-1)/r_i} \not\equiv 1 \pmod{p}$ for each r_i, then q is a primitive root of p.

Example 26.4. $p = 19$. Select $q = 2$. $p - 1 = 18 = 2 \times 3^2$ and

$$2^{(19-1)/2} = 2^9 \equiv 18 \pmod{19}$$
$$2^{(19-1)/3} = 2^6 \equiv 7 \pmod{19} \ .$$

Thus, 2 is a primitive root of 19.

Lecture 27

One-way Functions

A function f is said to be a *one-way function* if:

1. $f(x)$ is easy to evaluate for a given x.
2. Given $y = f(x)$, it is computationally infeasible to find x.

Example 27.1. A secure cryptosystem (computationally infeasible to find the key) provides a one-way function. Define $y = f(x) = E_x(P_0)$, where P_0 is a known piece of plaintext. Given P_0 and y (KTA) it should not be easy to compute the key x. We could also use $f(x) = E_x(x)$.

Example 27.2. Let p be a large prime ($\approx 2^{1027}$) and g be a primitive root. Define

$$f(x) \equiv g^x \pmod{p}, \quad 0 < x, f(x) < p \ .$$

This seems to be a one-way function if $p - 1$ has at least one large prime divisor. Computing x given $f(x)$ and g is known as the *discrete logarithm problem.*

Example 27.3. Consider

$$f(x) \equiv a_0 x^n + a_1 x^{n_1} + a_2 x^{n_2} + \cdots + a_k \pmod{p}, \quad a_0 \not\equiv 0 \pmod{p}$$

where p is a large prime, $n > n_1 > n_2 \ldots$, n is large and k is small — large sparse polynomial to a large degree. In 1977 the following was suggested:

$$f(x) = a_0 x^n + a_1 x^{n_1} + a_2 x^2 + a_3 x + a_4 \pmod{p}$$

where $n = 2^{24} - 3$, $n_1 = 2^{24} - 17$, $p \approx 2^{60}$, $a_i \approx 10^{19}$. Today we would require a much larger value of p.

One-way functions provide a secure method of computer login. We no longer need to keep the password dataset in the computer. Instead, keep a table containing

$$f(P_1), f(P_2), \ldots, f(P_n)$$

where P_i is a password and f is a one-way function. The user submits his password P, the machine calculates $f(P)$ and determines whether it is in the stored table.

27.1 Key Distribution

A and B wish to exchange a key for encryption over a public channel in such a way that a tapper cannot determine the key.

Solution

Due to Diffie and Hellman (1976) — still used today.

A and B decide to use a large prime p and a primitive root g.

A	Public Channel	B
Select $x_1 < p$ randomly		Select $x_2 < p$ randomly
$y_1 \equiv g^{x_1} \pmod{p}$	$y_1 \longrightarrow$	y_1
y_2	$\longleftarrow y_2$	$y_2 \equiv g^{x_2} \pmod{p}$
$K = y_2^{x_1}$		$K = y_1^{x_2}$

Notice that

$$y_2^{x_1} \equiv (g^{x_2})^{x_1} \equiv g^{x_2 x_1} \equiv (g^{x_1})^{x_2} \equiv y_1^{x_2} \pmod{p},$$

i.e., A and B have the same K at the end of the protocol. Use K for the key (e.g. take 128 high order bits).

The tapper has g, p, y_1, y_2, but since the DLP is (presumably!) difficult he cannot determine x_1 or x_2.

Lecture 28

Public-Key Cryptography

28.1 One-way Trapdoor Functions

A function f is said to be a *one-way trapdoor function* if:

1. $f(x)$ is easy to compute for any x.
2. Given $y = f(x)$ it is computationally infeasible to determine x unless certain information used in the design of f is known. When this special information k is known, there exists a function g which is easy to compute such that $x = g(k, y)$.

Public-Key Cryptography

A *public-key cryptosystem* consists of two functions E_{K_1} and D_{K_2} with the following properties:

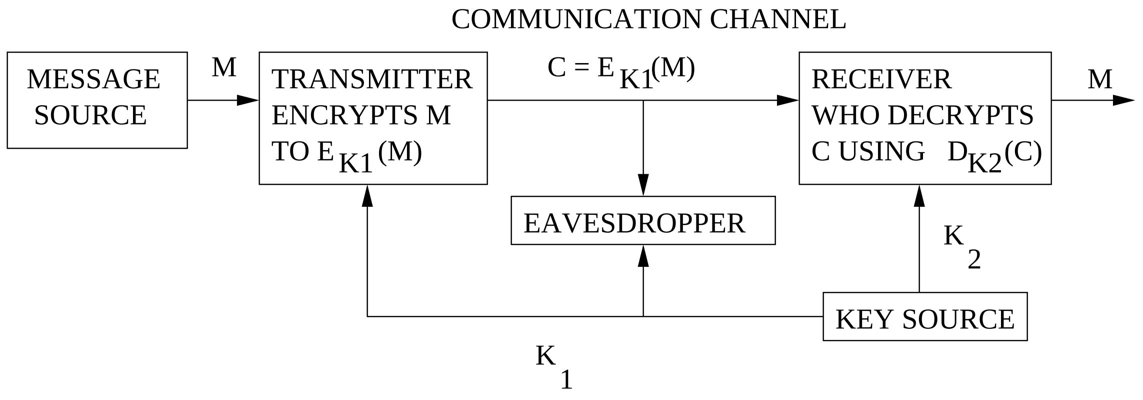
1. $E_{K_1}(M)$ and $D_{K_2}(C)$ are easy to compute when K_1, K_2 are known.
2. $D_{K_2}(E_{K_1}(M)) = M$.
3. Given K_1, E_{K_1}, $C = E_{K_1}(M)$ it is computationally infeasible to find M or K_2.

Properties 1,2,3 describe E_{K_1} as a one-way trapdoor function. For a signature system (authentication) we require:

4. $E_{K_1}(D_{K_2}(C)) = C$.

Public-key authentication: A computes $MAC = D_{K_2}(M)$ and B verifies $E_{K_1}(MAC) = E_{K_1}(D_{K_2}(M)) = M$.

Schematic of a Public-Key Cryptosystem



Note. In a public-key cryptosystem, it is *not* necessary for the key channel to be secure.

Lecture 29

The RSA Cryptosystem

Named after Ron Rivest, Adi Shamir, and Len Adelman.

Encryption: $C \equiv M^e \pmod{n}$

Decryption: $M \equiv C^d \pmod{n}$

The designer:

1. Selects two distinct large primes p and q (each around $2^{512} \approx 10^{155}$)
2. Let $n = pq$, $\phi(n) = (p-1)(q-1)$.
3. Select at random an integer e such that $\gcd(e, \phi(n)) = 1$ and $1 < e < n$ (from A4, e shouldn't be too small!).
4. Solves the linear congruence
$$de \equiv 1 \pmod{\phi(n)} \ .$$
5. Keeps d secret and makes n and e public, i.e., the *public key* $K_1 = n, e$ and the *private key* $K_2 = d, p, q$.

Suppose A wants to send a message M to the designer D. We assume that M is an integer and that $M < n$. If $M > n$, block it into less-than-n size blocks.

1. A computes $C \equiv M^e \pmod{n}$ where $0 < C < n$ (e and n are D's public key — everyone knows them)
2. A transmits C.
3. D computes $C^d \pmod{n}$ (d is D's private key — only D knows d)

Note. Encryption and decryption are done using the power algorithm (fast).

Why does this work? We have
$$C^d \equiv (M^e)^d \equiv M^{ed} \pmod{n},$$
but since d is chosen such that $ed \equiv 1 \pmod{\phi(n)}$ we have $ed = k\phi(n) + 1$ for some $k \in \mathbb{Z}$, and
$$M^{ed} \equiv M^{k\phi(n)+1} \equiv MM^{k\phi(n)} \equiv M(M^{\phi(n)})^k \pmod{n} \ .$$

Euler's Theorem states that $a^{\phi(n)} \equiv 1 \pmod{n}$, so we have
$$C^d \equiv M(M^{\phi(n)})^k \equiv M(1)^k \equiv M \pmod{n} \ .$$

Note. We have assumed that $\gcd(M, n) = 1$ in applying Euler's Theorem. The probability that $\gcd(M, n) \neq 1$ is $1/p + 1/q$, i.e., *very* small. Note that since $n = pq$ and $M < n$, $\gcd(M, n) \in \{1, p, q\}$, and thus in these rare cases we would likely find a factor of n.

Note. Security rests on the presumed difficulty of factoring n. The private key d is computed from the congruence $ed \equiv 1 \pmod{\phi(n)}$ — anyone who knows $\phi(n)$ can find d easily, but to compute $\phi(n)$ we need to know p and q.

It is easy to guarantee that $\gcd(M, n) = 1$ — simply take messages in blocks such that $M < p, q$.

29.1 Generating Random Primes

To generate a random k-bit prime number:

1. Select a random k-bit number.
2. Look at the set of numbers $r, r+1, r+2, \ldots, r+j$ (j determined below).
3. Use the Sieve of Eratosthenes (i.e., trial division) to eliminate those $r+i$ with small prime factors.
4. Test the smallest remaining number for primality. If it is not prime, try the next one, etc...

Questions:

1. How large should j be to guarantee that we find a prime?

   Let $\pi(x)$ denote the number of primes $\leq x$. It is known that $\pi(x) \sim x/\log x$. Thus, if $y = x + j$ such that $\pi(y) = \pi(x) + 1$, then there must be a prime between x and $x + j \implies j = \log n$.

2. How do we test a number for primality?

   Probable primes (using Fermat's Theorem) are good enough.

Lecture 30

Security of RSA

There is no proof that RSA is secure.

- If n can be factored, then RSA is broken. It is not proven whether factoring is hard.
- If d can be found, it can be used to factor n.
- It is not proven that other methods to compute M given C, e, n do not exist, which do not rely on factoring. I.e., it is not known whether breaking RSA is *equivalent* to factoring n.

Nevertheless, we need to design RSA systems such that $n = pq$ cannot be factored easily.

Features of p and q

1. p and q must both be large ($2^{512} \approx 10^{155}$, soon to be $2^{1024} \approx 10^{308}$). This means $n \approx 2^{1024}$ (or 2^{2048}).
2. $|p - q| \gg \sqrt[4]{n}$
3. $p - 1$, $q - 1$, $p + 1$, $q + 1$ must all have a large prime factor. There are factoring algorithms which can exploit these properties.

The best general purpose factoring algorithm currently available (the "number field sieve") can factor n in

$$\log n < e^{((\log n)^{1/3}(\log \log n)^{2/3})(1+o(1))} < n$$

operations. Massive parallelism, internet distributed computations, are applicable with this algorithm.

Current factoring record (integer without a special form): RSA-155, a 155-digit RSA modulus was factored in August 1999 (total time — 20 years on a single 450 MHz PC with 64 MB RAM).

30.1 Iterative Attack

Let $C_0 = C = f(M)$ (f is the encryption function). Define:

$$C_i = f(C_{i-1}) \ .$$

Stop when $C_m = C_0 \implies M = C_{m-1}$.

This technique is provable equivalent to factoring n under the assumption of the Extended Riemann Hypothesis.

Let

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

where $s = \sigma + ti$ (a complex number). When is $\zeta(s) = 0$?

- Trivial zeros: $s = -2n$
- Non-trivial zeros: Riemann hypothesis states that if s is a non-trivial root, then $\sigma = 1/2$.

Over 1.5 billion non-trivial zeros have been found, and for all of them $\sigma = 1/2$. The Riemann hypothesis has resisted proof since 1853.

The *extended Riemann hypothesis* says that

$$L(s,\chi) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s}$$

has its non-trivial zeros on the same line ($\sigma = 1/2$), and also remains unproved.

30.2 Summary of RSA

1. It seems to be secure.
2. The key size is "small" (two 310 digit numbers).
3. No message expansion.
4. It can be used as a signature scheme.

Disadvantages:

1. It is very slow compared with DES, RIJNDAEL, and other *private key* cryptosystems.
2. Finding keys is fairly expensive.
3. Security is unproven.

30.3 The El Gamal PKC

It is important to have other public-key cryptosystems (PKC) whose security relies on other hard problems. RSA relies on factoring. An alternative PKC is due to El Gamal.

B (the designer) chooses a large prime p and a primitive root $g \bmod p$, and selects some random x with $0 < x < p$. B then evaluates $y \equiv g^x \pmod{p}$ with $0 < y < p$ and publishes $\{g, p, y\}$. His private key is x.

Suppose A wants to send a message M $(< p)$ to B.

1. A selects a random k, $0 < k < p$.
2. A computes $K \equiv y^k \pmod{p}$, $0 < K < p$.

3. A sends $\{C_1, C_2\}$ to B where

$$C_1 \equiv g^k \pmod p, \quad 0 < C_1 < p,$$
$$C_2 \equiv KM \pmod p, \quad 0 < C_2 < p \ .$$

To decrypt, B computes

$$C_1^x \equiv g^{kx} \equiv (g^x)^k \equiv y^k \equiv K \pmod p$$

and solves

$$C_2 \equiv KM \pmod p$$

for M. Note that x is B's private key.

The security of this system relies on the presumed difficulty of the discrete logarithm problem (no proof of equivalence). However, it *is* equivalent to the *Diffie-Hellman problem*: given $g^x \pmod p$ and $g^y \pmod p$, compute $g^{xy} \pmod p$.

Disadvantages:

1. Increased bandwidth — the communication channel must be twice as wide as the message being sent.
2. Twice as much computational work for encrypting and decrypting (as RSA).
3. A new random number, k, must be generated for each message.

Lecture 31

Authentication using PKC

Authentication is usually achieved by means of a *signature*. A signature is a simple means by which the recipient of a message can authenticate the identity of the sender. It should have two properties:

1. Only the sender can produce his signature.
2. *Anyone* should be easily able to verify the validity of the signature.

Note that this is *different* from a MAC — both sender and receiver can generate MAC's (eg. using DES).

31.1 Authentication Without Secrecy

A sender wishes to send M and a signature indicating that he produced M. If the sender uses a public-key cryptosystem which is also a signature system, he can sign his message.

Suppose A is the sender. She has a decryption scheme D_A and an encryption scheme E_A. She creates her signature by computing

$$S = D_A(M)$$

and sends $\langle M, A, S\rangle$. Anyone who receives $\langle M, A, S\rangle$ can find E_A (because it is public) and verify the message by computing $E_A(S)$ and comparing to M. If the message is authentic, then

$$E_A(S) = E_A(D_A(M)) = M \ .$$

This works with RSA because it is a signature system:

$$\begin{aligned} S = D_A(M) &\equiv M^{d_A} \pmod{n_A} \\ E_A(S) = E_A(D_A(M)) &\equiv (M^{d_A})^{e_A} \pmod{n_A} \\ &\equiv (M^{e_A})^{d_A} \pmod{n_A} \\ &\equiv M \ . \end{aligned}$$

31.2 Authentication With Secrecy

Problem: A wants to send a *secret* message M to B. B wants to be sure that A sent M.

Solution: A calculates $S = D_A(M)$ (S is the signature). He then uses B's encryption system to compute $E_B(S, M) = F$ and sends $\langle F, A\rangle$ to B.

B, on receiving $\langle F, A\rangle$ computes

$$D_B(F) = D_B(E_B(S, M)) = S, M$$

and then verifies that

$$E_A(S) = M \ .$$

31.3 Impersonation

Problem: A sends a message M and signature $S = D_A(M)$ to B. B calculates $E_A(S)$ and ensures that this is M. Could only A send M? Can C send a message to B in such a way that B thinks it came from A?

Yes. C knows E_A, so C can select some L and compute $E_A(L)$. C then sends $\langle E_A(L), L\rangle$ to B. B assumes that L is the signature, evaluates $E_A(L)$, and accepts $E_A(L)$ as the message.

Language redundancy usually foils this attack, since $E_A(L)$ will normally not be coherent English text. However, if A is supposed to be sending random information, this will be a problem.

Solution: A sends $\langle M, D_A(f(M))\rangle$, where f is a one-way function. B computes $G = E_A(D_A(f(M)))$ and $f(M)$, and checks that they are the same. Only A could send M.

Suppose C attemps to impersonate A as above. If C randomly chooses L, computes $X = f^{-1}(E_A(L))$, and sends $\langle X, L\rangle$ to B, then he will be successful since B will compute $E_A(L)$ and verify that this is equal to

$$f(X) = f(f^{-1}(E_A(L))) = E_A(L) \ .$$

However, if f is a one-way function, C cannot find a suitable X, and therefore will not be able to impersonate A.

Lecture 32

Applications of Authentication

High-Security Login Procedure

To foil eavesdropping, hacking, etc...

Suppose A wishes to login. He computes the following:

$$D_A(A, time, date)$$

where A is A's identity (e.g. user name) and *time* and *date* are used as a date-time stamp to foil the playback threat (reusing this authentic message at a later date). He then computes

$$P = E_M(A, D_A(A, time, date))$$

in isolation from the machine M. P is then sent to the machine M.

The host M, upon receiving P, evaluates

$$D_M(P) = (A, D_A(A, time, date)) \ .$$

After using the identity A to look up A's encryption function, M evaluates

$$E_A(D_A(A, time, date)) = (A, time, date) \ .$$

The machine can then verify that A is who she says she is, and the inclusion of the date and time spoils the playback threat.

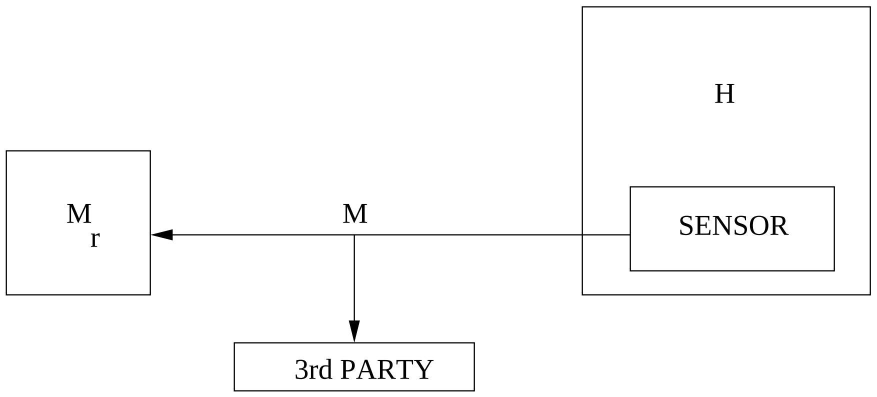
Unfortunately, this scheme is rather slow and expensive, so it is only used for high-security applications.

Remote Sensing

Examples:

- Monitoring radioactive hot cells in a reactor (to prevent theft)
- Video camera security (avoiding playback threat)
- Verification and compliance with nuclear test ban treaty

Problem: A monitor M_r is observing some information produced in a host's (H) environment.



This is done by means of a sensor S placed in H. S produces messages, M, which are examined by the monitor M_r. The monitor and the host don't trust each other.

Desirable features:

1. All parties should be able to *verify the authenticity* of any message.
2. *No part* of any message M can be concealed from the host.
3. No unilateral action on the part of any of the parties should *lessen the confidence* in the authenticity of M.
4. No party should be able to *forge messages* that could be taken as authentic.

To solve this problem, use RSA:

- The monitor produces e, d, n and gives (d, n) to the host and all other legitimate third parties.
- The sensor sends $C \equiv M^e \pmod{n}$ (only the sensor and monitor know e). Third parties and the host can authenticate by evaluating $M \equiv C^d \pmod{n}$.
- All parties can read C, but it cannot be modified. The host cannot forge because he doesn't know e. In fact, no one can forge a message $C' \equiv (M')^e \pmod{n}$ unless he knows e.

One problem: Unfortunately the monitor knows e. This means that H can do whatever he likes, and when the monitor complains, claim that the monitor has forged messages incriminating H.

Solution: Keep e secret from *all* parties. This means that the sensor S must produce e, d, n, reveal d and n to all legitimate parties, but reveal e to no one.

S must be capable of producing random information which cannot be inferred by anyone (including the monitor). One possible source of randomness: put a small quantity of radioactive material in S and let S count the number of radioactive "ticks."

- Start at time t.
- Count at time t_1 (mod2) gives one bit of the key.
- Count at time t_2 (mod2) gives second bit of the key.
- etc...

Lecture 33

Other Applications of Public-Key Cryptography

33.1 Secret Sharing

A (k, n) *threshold scheme* is defined as follows. We divide a piece of information n into n pieces $D_1, D_2, \dots, D_n$ such that:

1. Knowledge of any k or more pieces D_i allows one to compute D easily.
2. Knowledge of fewer than k pieces leaves D undetermined, with all possibilities for D equally likely.

Applications: bank vault key, nuclear launch codes

Constructing a (k, n) Threshold Scheme

We select a large prime $p > n$ and $p > D$, which may be made public. Select at random a set of integers $a_1, a_2, \dots, a_{k-1}$ such that $0 \leq a_i < p$, which are kept secret. Let $a_k = D$.

Evaluate the polynomial

$$f(x) = a_1 x^{k-1} + a_2 x^{k-2} + \cdots + a_k \pmod{p}$$

for $x = 1, 2, \dots, n$. Set $D_i \equiv f(i) \pmod{p}$, $i = 1, 2, \dots, n$. As usual, take D_i such that $0 \leq D_i < p$. We now have the n pieces $D_1, D_2, \dots, D_n$ that we require.

Given any k of the D_i's we obtain k linear equations in k unknowns (the a_i), from which we can find a_k. Alternatively, we can solve for $f(0) = a_k = D$ using Lagrange interpolation.

Lagrange interpolation works as follows. Given k points $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$, we can compute a degree $k - 1$ polynomial $f(x)$ such that $y_i = f(x_i)$ using

$$f(x) = \sum_{i=1}^{k} L_i(x) y_i$$

where

$$L_i(x) = \frac{(x - x_1)(x - x_2)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_k)}{(x_i - x_1)(x_i - x_2)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_k)} .$$

Notice that for each x_j, $1 \le j \le k$

$$L_i(x_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j, \end{cases}$$

which implies that $f(x_j) = y_j$.

In our case, we need $D \equiv f(0) \equiv \sum_{i=1}^{k} L_i(0) y_i \pmod{p}$. To find $L_i(0) \pmod{p}$ solve the linear congruence

$$(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_k)L_i(0) \\ \equiv (-x_1)(-x_2)\ldots(-x_{i-1})(-x_{i+}))\ldots(-x_k) \pmod{p} \ .$$

Thus, using Lagrange interpolation we can compute D given k or more of the D_i's.

33.2 The Oblivious Transfer Problem

Problem: A and B wish to generate a random bit in such a way that both parties are confident that the other side has not cheated.

We start with a number $n = pq$, where p and q are large primes with $p, q \equiv 3 \pmod{4}$. The protocol is based on the fact that the congruence $x^2 \equiv a \pmod{n}$ has 4 possible solutions in this case. To compute these solutions we proceed as follows. We have

$$\begin{aligned} x^2 &\equiv a \pmod{p} \\ (x^2)^{(p-1)/2} &\equiv a^{(p-1)/2} \pmod{p} \\ x^{p-1} &\equiv a^{(p-1)/2} \pmod{p} \\ 1 &\equiv a^{(p-1)/2} \pmod{p} \ . \end{aligned}$$

Let $x = a^{(p+1)/4}$. Then

$$x^2 \equiv a^{(p+1)/2} \equiv a^{(p-1)/2} a \equiv a \pmod{p},$$

i.e., $x \equiv \pm a^{(p+1)/4}$ are the two square roots of $a \bmod p$. Similarly, $\pm a^{(q+1)/4}$ are the two square roots of $a \bmod q$. We combine these results to compute the solutions of $x^2 \equiv a \pmod{n}$.

Example 33.1. Let $n = 33$, $p = 3$, $q = 11$. Solve $x^2 \equiv 31 \pmod{33}$.

We compute x modulo p and q. From $x^2 \equiv 31 \equiv 1 \pmod{3}$ we obtain

$$x \equiv 1^{(3+1)/4} \equiv \pm 1 \pmod{3}$$

and from $x^2 \equiv 31 \equiv 9 \pmod{11}$ we obtain

$$x \equiv 9^{(11+1)/4} \equiv \pm 3 \pmod{11} \ .$$

We pick one value modulo 3 and another modulo 11. Choosing 1 and 3 yields

$$x = 1 + 3u = 3 + 11v \implies 3u - 11v = 2 \ .$$

The four possible choices of x mod 3 and 11 yield the four solutions mod33 :

$$x = 8, 14, 19, 25 \ .$$

Bit commitment protocol:

1. A selects at random two large primes p and q. He computes $n = pq$ and sends n to B (keeping p and q secret).

2. B selects at random x with $0 < x < n$, computes $y \equiv x^2 \pmod{n}$ with $0 < y < n$, and sends y to A.
3. A computes z such that $z^2 \equiv y \pmod{n}$ (as above) and sends z to B. Note that A computes one of the four possible values of z.
4. B calculates $x^2 - z^2$ and determines the bit as follows.
   - Suppose $x = \pm z$. Then $x^2 \equiv z^2 \pmod{n} \implies n \mid x^2 - z^2$ and $\gcd(n, x+z) = p$ or q. Thus, if $x = z$, B can factor n and the bit is 1.
   - Suppose $x \neq \pm z$. Then $x^2 - z^2 = 0$ and B cannot factor n. The bit is set to 0.

A computes $z = \pm x$ with probability $1/2$, so the bit sent is random. B confirms the value of the bit sent by sending x to A. A confirms that both x and z are square roots of n.

Lecture 34

Problems and Complexity

Example problems

1. M : given $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ find ab (multiplication problem)
2. F : given $c \in \mathbb{Z}$ find $a, b > 1$ such that $ab = c$ (factoring problem)
3. K : given $a_1, a_2, \ldots, a_k$, $a_i \in \mathbb{Z}$, find $x_1, x_2, \ldots, x_k$, $x_i \in \{0, 1\}$ such that for a given $S \in \mathbb{Z}$
$$S = AX$$
where $A = (a_1, a_2, \ldots, a_k)$ and $X = (x_1, x_2, \ldots, x_k)$ (knapsack problem)

Definitions

An *instance* of a problem is obtained when particular values of all problem parameters are specified. E.g., for K the parameters are S, A, k.

The *size* of a problem instance is the amount of input data needed to describe that instance. E.g., the number of bits of input to a computer.

The total number of symbols needed in this description is called the *input length* n of the problem. E.g., for F, $n = \lceil \log_2 C \rceil$. For K, $n = \lceil \log_2 S \rceil + \lceil \log_2 k \rceil + \sum_{i=1}^{k} \lceil \log_2 a_i \rceil$.

Notation

$f(x)$ and $g(x)$ are functions.

We say that $f(x) = O(g(x))$ if there exists some constant c such that
$$|f(x)| \le c|g(x)|$$
for sufficiently large x.

Example 34.1. Time complexity of M (multiply a and b): Suppose we use a computer with fixed binary word length W and that a consists of r words and b consists of s words. Then $n = rw + sw$ (input length) and the amount of time required to find C is $rwsw < kn^2$. Thus, the time complexity of M is $O(n^2)$.

Note that the best result is $O(n \log n \log \log n)$.

As a matter of convenience we deal with decision problems — problems which have only one of two possible answers, “YES” or “NO.”

Example 34.2. To convert M to a decision problem we could ask: Is the product of a and b greater than a preassigned number B?

F : given c, B does there exist a such that $a \mid c$ and $a \geq B$?

K : does there exist some binary vector X such that given S and A, $S = AX$?

Lecture 35

$\mathcal{NP}$-completeness

Definition 35.1. A *deterministic* algorithm when applied to a decision problem Π halts in every instance of Π with an answer "YES" or "NO."

We say that a problem $\Pi \in \mathcal{P}$ when it can be solved by a deterministic algorithm of time complexity $O(n^c)$ where c is an absolute constant (i.e., independent of n). Such problems are said to be *tractable*.

Example 35.1. $M \in \mathcal{P}$.

Definition 35.2. A *non-deterministic* algorithm solves a decision problem Π if the following two properties hold for all instances of Π :

1. If the answer to the instance I is "YES," there exists some structure S such that when guessed for input I, it will lead the checking stage to respond "YES" for I and S.
2. If the answer to I is "NO," there does not exist any structure S that, when guessed for input I, will lead the checking stage to respond "YES" for I.

A polynomial-time non-deterministic algorithm:

- solves a problem Π if there exists a polynomial p such that for every instance which has answer "YES" there is some guess S that leads the deterministic checking stage to respond "YES" for I and S within time $p(n)$, when n is the input length of I.

$\mathcal{NP}$ — class of all decision problems that can be solved by polynomial time non-deterministic algorithms.

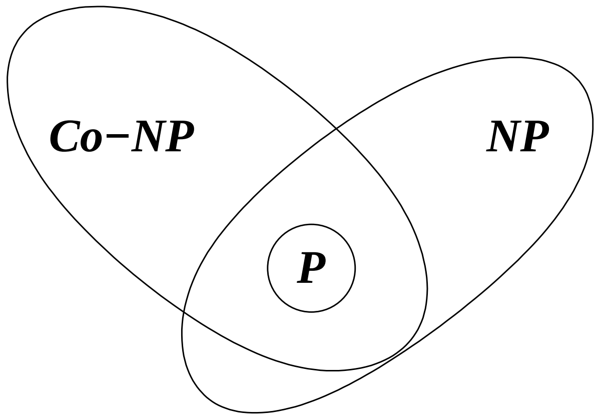
Clearly $\mathcal{P} \in \mathcal{NP}$. Also $F \in \mathcal{NP}$ and $K \in NP$.

$Co-\mathcal{NP}$ — class of all problems whose complements can be solved by polynomial-time non-deterministic algorithms.

Probably $K \notin Co-\mathcal{NP}$, $F \in Co-\mathcal{NP}$.

Short verification for answer "YES" — $\mathcal{NP}$

Short verification for answer "NO" — $Co-\mathcal{NP}$



Theorem 35.1. *There exists a subclass $\mathcal{NPC}$ of problems in NP such that if $\Pi \in \mathcal{NPC}$ and Π can be shown to be in $\mathcal{P}$, then all of the problems in $\mathcal{NP}$ are in $\mathcal{P}$. That is, $\mathcal{NP} = \mathcal{P}$. These are called the $\mathcal{NP}$-complete problems. They are the hardest problems in $\mathcal{NP}$. E.g., $K \in \mathcal{NPC}$.*

Lecture 36

Summary of Complexity Theory

What we don't know

Question	Consensus
$\mathcal{NP} = \mathcal{P}?$	No
$Co - \mathcal{NP} = \mathcal{NP}?$	No
$\mathcal{P} = Co - \mathcal{NP} \cap \mathcal{NP}?$	Probably not

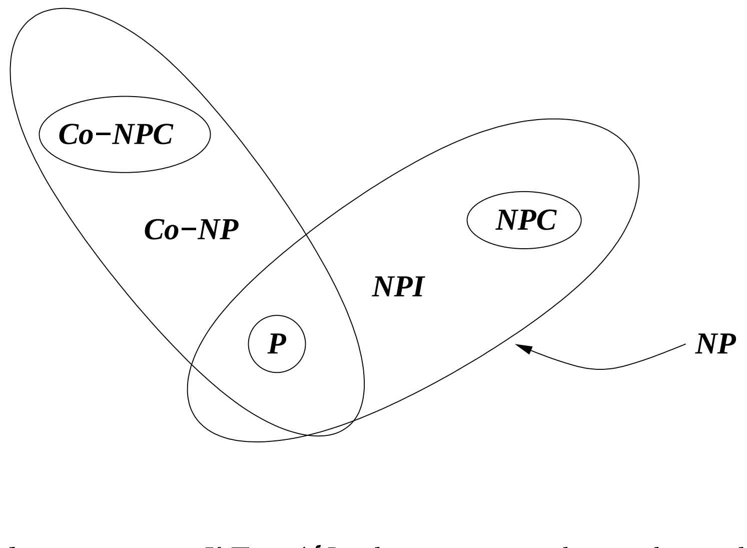
E.g. $F \in \mathcal{NP}$ and $F \in Co - \mathcal{NP}$.

What we know

Theorem 36.1. *If* $\Pi \in \mathcal{NPC}$ *and* $\Pi \in Co - \mathcal{NP}$, *then* $\mathcal{NP} = Co - \mathcal{NP}$.

This implies that $F \notin \mathcal{NPC}$ (assuming $Co - \mathcal{NP} \neq \mathcal{NP}$).

Theorem 36.2. *If* $\mathcal{P} \neq \mathcal{NP}$, *there exists a non-empty subclass* $\mathcal{NPI}$ *of* $\mathcal{NP}$ *such that if* $\Pi \in \mathcal{NPI}$, *then* $\Pi \notin \mathcal{P}$ *and* $\Pi \notin \mathcal{NPC}$. *Such a problem is called an* $\mathcal{NP}$*-intermediate problem.*



Theorem 36.3. *If* $\Pi \in \mathcal{NP}$, *there exists a polynomial* p *such that* Π *can be solved by a deterministic algorithm of time complexity* $O(2^{p(n)})$.

It appears to be a good idea to base a cryptosystem on problems that are “hard” in the sense described above. However, there are difficulties:

1. Complexity theory deals with the worst possible case of any problem. It could be that only one or a few instances of a problem are truly intractable. A cryptosystem based on such a problem would only occasionally be secure.
2. Complexity theorists assume that only a certain amount of information is available for the solution of a particular instance of a problem. Cryptanalysts frequently have a great deal more information at their disposal, such as corresponding plaintext and ciphertext. The existence of such extra side information is not taken into account in analyzing the complexity of problems.
3. Given any particular difficult problem, it is not always easy to convert it into a cryptosystem.

What are really needed are new measures of complexity especially tailored to the problem of cryptanalysis. When we can certify the security of cryptosystems according to such measures, the problem of secure communications will be solved.