Number Theory in Cryptography

Introduction

September 20, 2006
Universidad de los Andes

Guessing Numbers

Guessing Numbers

$$\text{(person } x\text{)} \quad \longmapsto \quad \text{(last 6 digits of phone number of } x\text{)}$$

Guessing Numbers

$$(\text{person } x) \quad \longmapsto \quad (\text{last 6 digits of phone number of } x)$$

A **Hash Function** is a function f from A to B such that

- It is easy to compute $f(x)$ for any $x \in A$.
- For any $y \in B$, it is hard to find an $x \in A$ with $f(x) = y$.
- It is hard to find $x, x' \in A$ with $x \neq x'$ and $f(x) = f(x')$.

Caesar Cipher

VIXYVR XS VSQI

Caesar Cipher

VIXYVR XS VSQI

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
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

Caesar Cipher

VIXYVR XS VSQI

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
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

RETURN TO ROME

Caesar Cipher

VIXYVR XS VSQI

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
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

RETURN TO ROME

Breaking the code: just try all 26 shifts.

Substitution Cipher

MQWE WE B YXM QBLHGL

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

Substitution Cipher

MQWE WE B YXM QBLHGL

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

THIS IS A LOT HARDER

Substitution Cipher

MQWE WE B YXM QBLHGL

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

THIS IS A LOT HARDER

Breaking the code:

Can not try $26! = 403291461126605635584000000$ permutations...

Solution: Letter Frequencies

	English	Spanish
A	82	125
B	14	14
C	28	47
D	38	59
E	131	137
F	29	7
G	20	10
H	53	7
I	63	62
J	1	4
K	4	0
L	34	50
M	25	31

	English	Spanish
N	71	67
O	80	86
P	20	25
Q	1	9
R	68	69
S	61	79
T	105	46
U	25	39
V	9	9
W	15	0
X	2	2
Y	20	9
Z	1	5

out of 1000 letters

Viginere Cipher

HVD PZAHSQ JMLEIDRXPSG ZVZ UCH OVZZZSFUIY

Viginere Cipher

HVD PZAHSQ JMLEIDRXPSG ZVZ UCH OVZZSFUIY

Shift the letters of the encrypted message according to the value of the letters of the secret keyword "**LLAVES**." (a= 1, b= 2, ...).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

```
HVD PZAHSQ JMLEIDRXPSG ZVZ UCH OVZZSFUIY
LLA VESLLA VESLLAVESLL AVE SLL AVESLLAVE
THE LETTER FREQUENCIES ARE NOT PRESERVED
```

Viginere Cipher

HVD PZAHSQ JMLEIDRXPSG ZVZ UCH OVZZSFUIY

Shift the letters of the encrypted message according to the value of the letters of the secret keyword "**LLAVES**." (a= 1, b= 2, ...).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

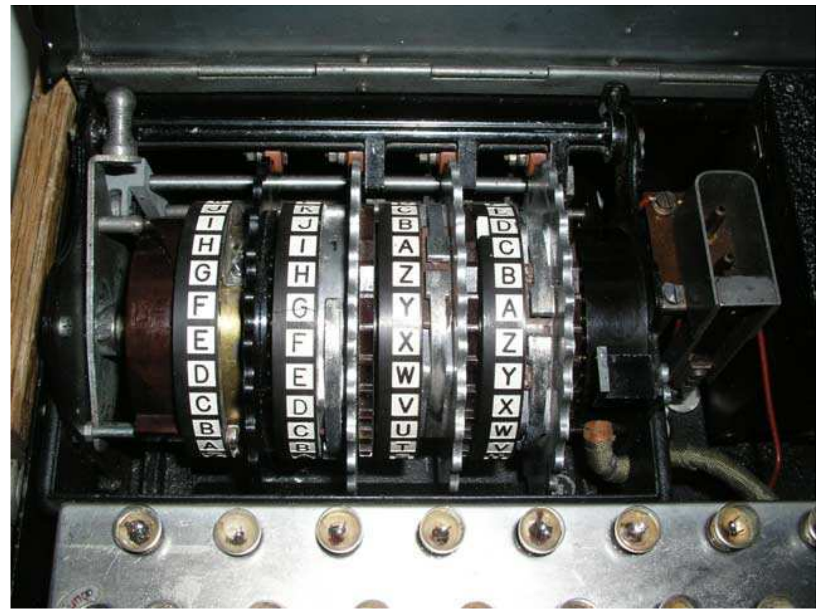
```
HVD PZAHSQ JMLEIDRXPSG ZVZ UCH OVZZSFUIY
LLA VESLLA VESLLAVESLL AVE SLL AVESLLAVE
THE LETTER FREQUENCIES ARE NOT PRESERVED
                 EN   ES    E N     ES
```

Repeated bigrams stay **repeated bigrams**
if their distance is a multiple of the length of the key.

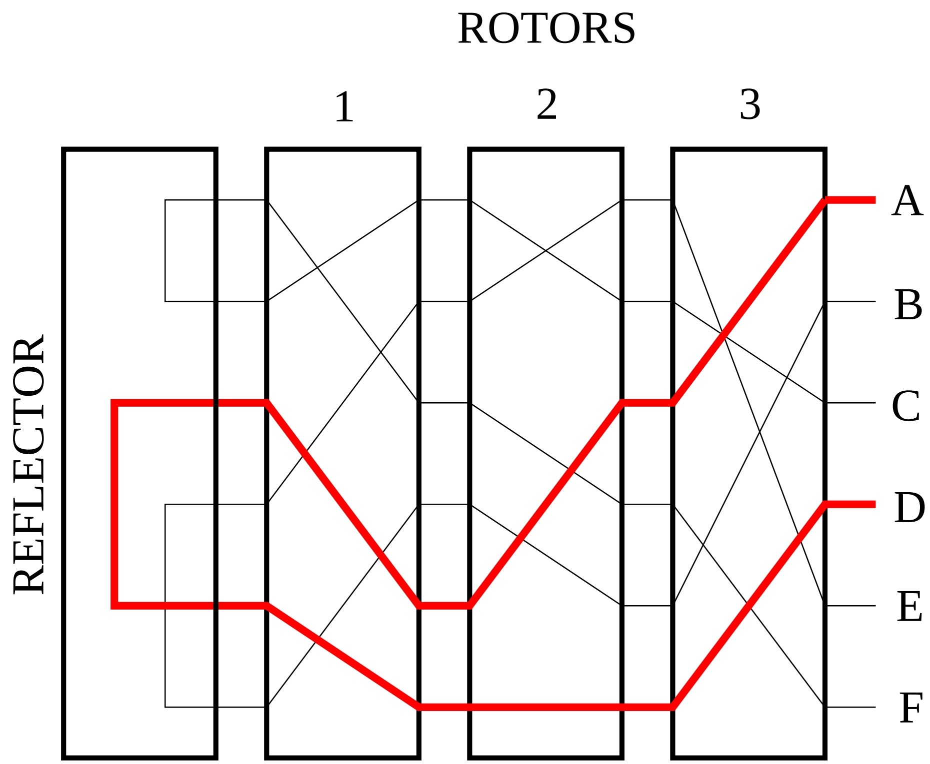
Security

All these ciphers are **breakable**
once the enemy knows
the type of encryption.

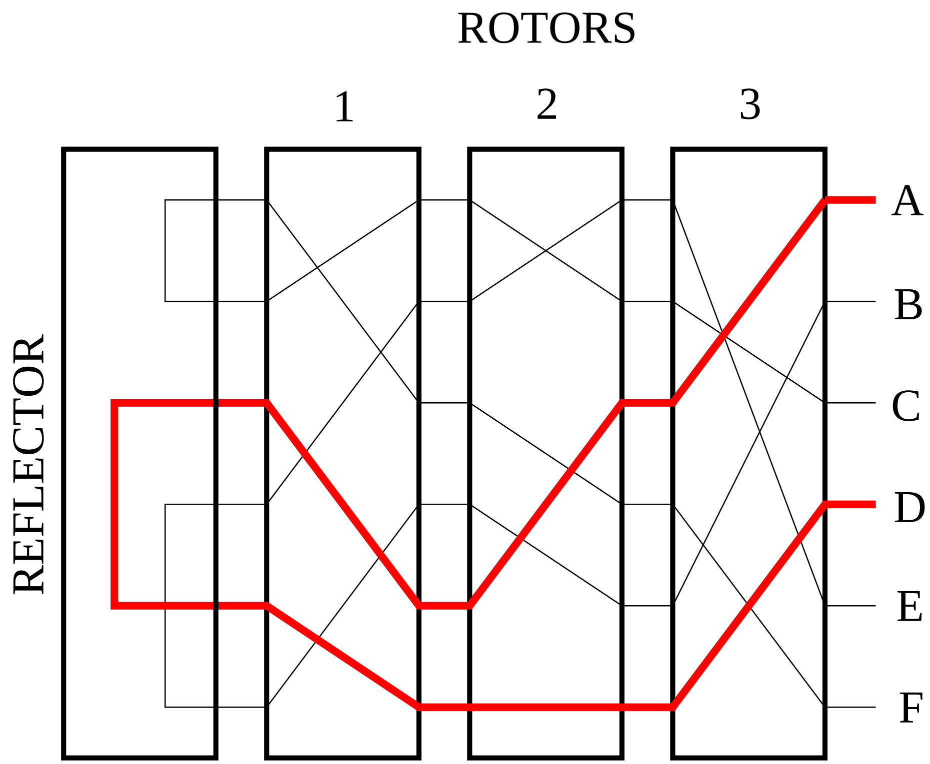
Enigma



A German WW-II encryption machine, broken by the allies



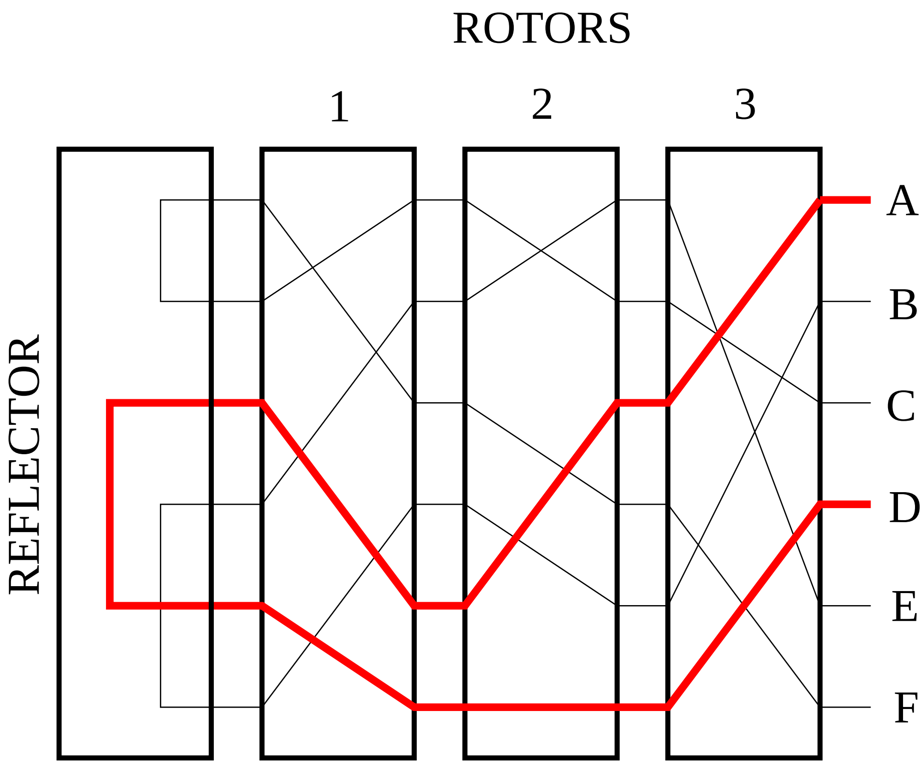
Period of 26^3 substitutions



Period of 26^3 substitutions

Weaknesses:

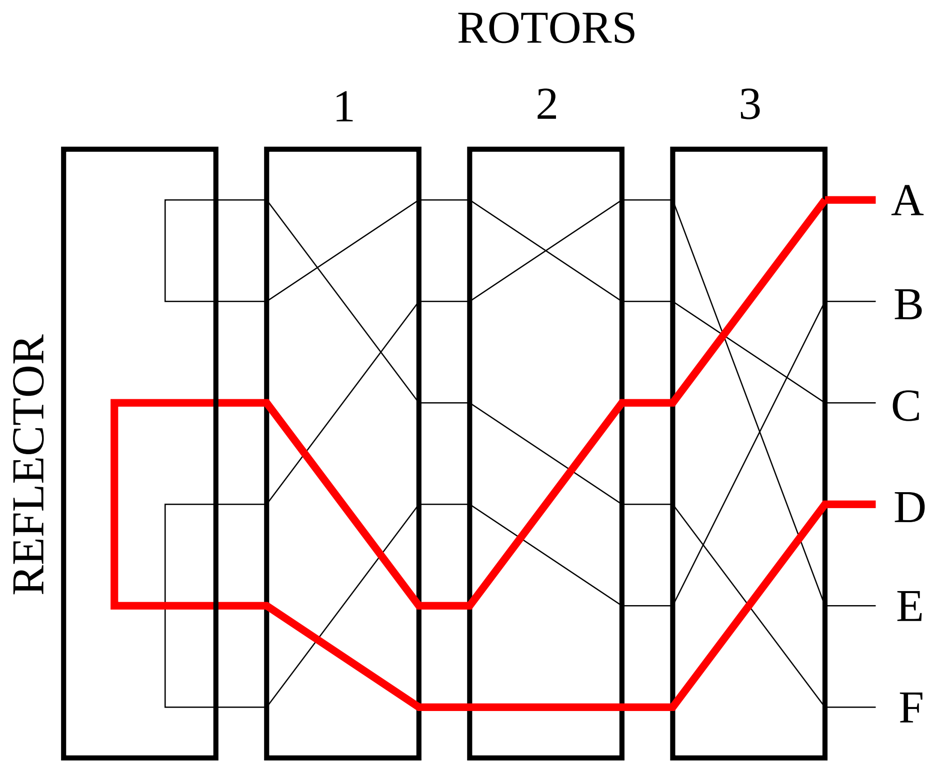
- Permutations are involutions
- Letter x does **not** map to x
- Rotors can be stolen
- Book of initial settings too



Period of 26^3 substitutions

Weaknesses:

- Permutations are involutions
- Letter x does **not** map to x
- Rotors can be stolen
- Book of initial settings too
- User errors:
  - repeated initial 3 letters
  - nonrandom initial 3 letters
  - test message with only T's

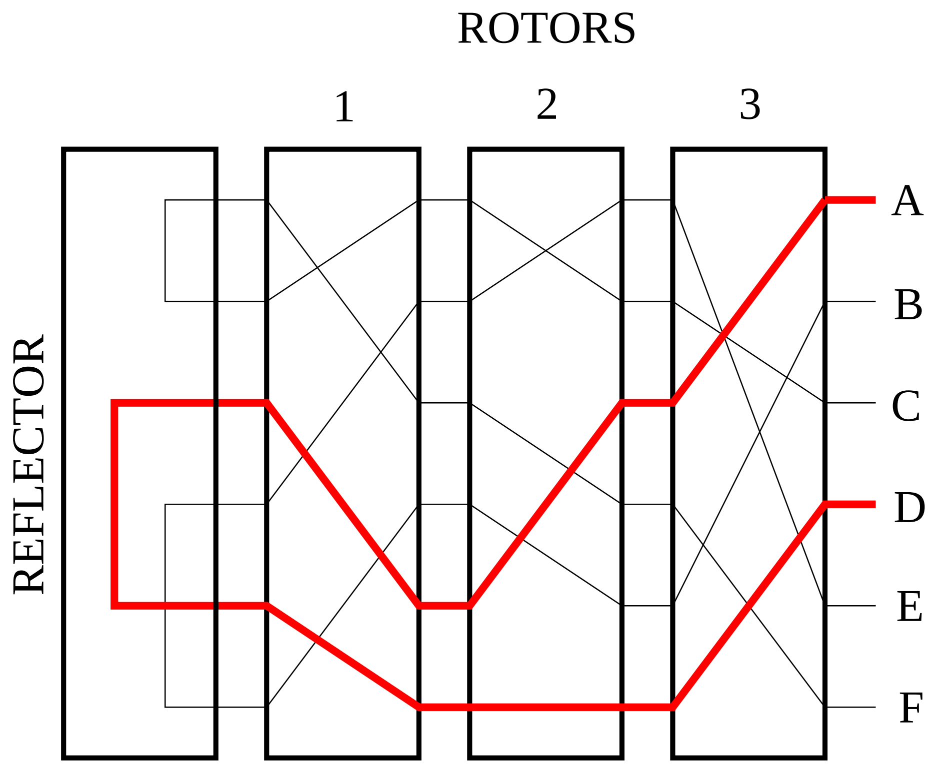


Period of 26^3 substitutions

Weaknesses:

- Permutations are involutions
- Letter x does **not** map to x
- Rotors can be stolen
- Book of initial settings too
- User errors:
  - repeated initial 3 letters
  - nonrandom initial 3 letters
  - test message with only T's

British could decipher until 1932, then extra keyboard permutation.



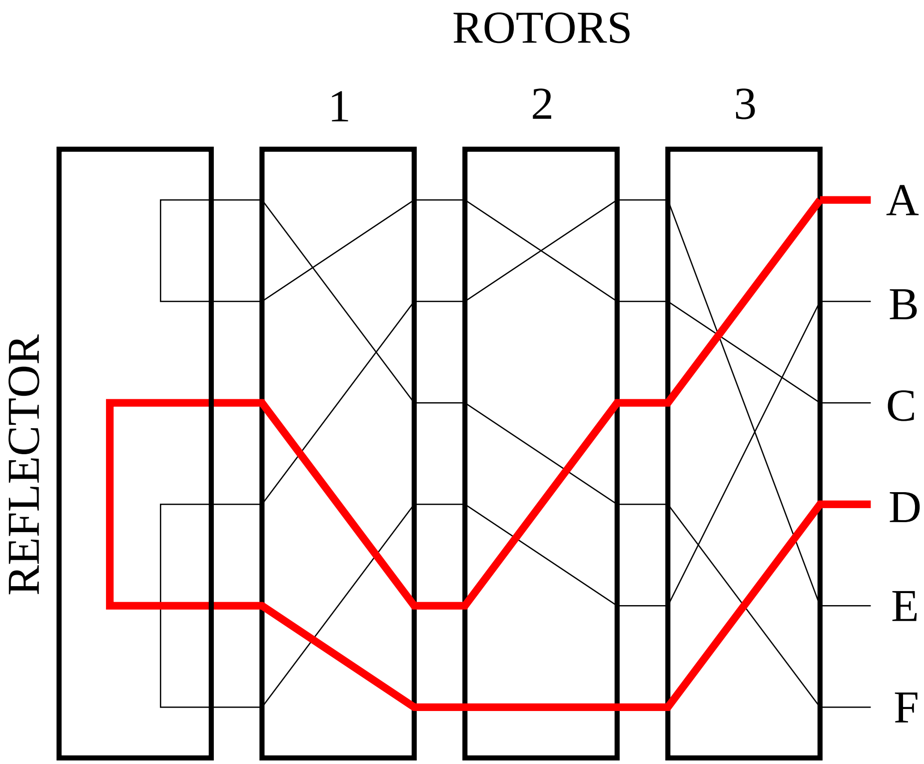
Period of 26^3 substitutions

Weaknesses:

- Permutations are involutions
- Letter x does **not** map to x
- Rotors can be stolen
- Book of initial settings too
- User errors:
  - repeated initial 3 letters
  - nonrandom initial 3 letters
  - test message with only T's

British could decipher until 1932, then extra keyboard permutation.

Polish until 1939, then extra rotors, no repeated 3 letters.



Period of 26^3 substitutions

Weaknesses:

- Permutations are involutions
- Letter x does **not** map to x
- Rotors can be stolen
- Book of initial settings too
- User errors:
  - repeated initial 3 letters
  - nonrandom initial 3 letters
  - test message with only T's

British could decipher until 1932, then extra keyboard permutation.

Polish until 1939, then extra rotors, no repeated 3 letters.

At the end of the war all messages could be deciphered in 2 days.

The Germans were still confident about ENIGMA.

Lesson learned

A crypto system should be safe even if

- the enemy knows your encryption algorithm
- the enemy knows lots of plain texts together with their encryptions
  (no chosen plain text attacks)

Lesson learned

A crypto system should be safe even if

- the enemy knows your encryption algorithm
- the enemy knows lots of plain texts together with their encryptions
  (no chosen plain text attacks)

Solution

- Use a public algorithm with a secret key.

Data Encryption Standard (DES, 1974)

Xor:

$\oplus$	0	1
0	0	1
1	1	0

$$(x \oplus y) \oplus y = x$$

Data Encryption Standard (DES, 1974)

Xor:

$\oplus$	0	1
0	0	1
1	1	0

$$(x \oplus y) \oplus y = x$$

message	1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1
key	0 1 1 0 1 0 0 1 0 0 0 1 0 0 1 0 $\oplus$
encryption	1 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1

Data Encryption Standard (DES, 1974)

Xor:

$\oplus$	0	1
0	0	1
1	1	0

$$(x \oplus y) \oplus y = x$$

message	1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1
key	0 1 1 0 1 0 0 1 0 0 0 1 0 0 1 0 $\oplus$
encryption	1 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1

encryption $\oplus$ key $=$ message

Data Encryption Standard (DES, 1974)

Xor:

$\oplus$	0	1
0	0	1
1	1	0

$(x \oplus y) \oplus y = x$

message	1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1
key	0 1 1 0 1 0 0 1 0 0 0 1 0 0 1 0 $\oplus$
encryption	1 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1

encryption $\oplus$ key $=$ message

message $\oplus$ encryption $=$ key **!DANGER!**

Data Encryption Standard (DES, 1974)

- Pick a secret shared key of 64 bits.
- Divide the message in blocks of 64 bits.
- Encrypting one block consists of a combination of
  repeated $\oplus$ with parts of the key, permutations,
  breaking up in subblocks, and small functions by table.

Data Encryption Standard (DES, 1974)

- Pick a secret shared key of 64 bits.
- Divide the message in blocks of 64 bits.
- Encrypting one block consists of a combination of
  repeated $\oplus$ with parts of the key, permutations,
  breaking up in subblocks, and small functions by table.

Disadvantage: Need to agree on a key before hand...
System uses a **secret shared key**

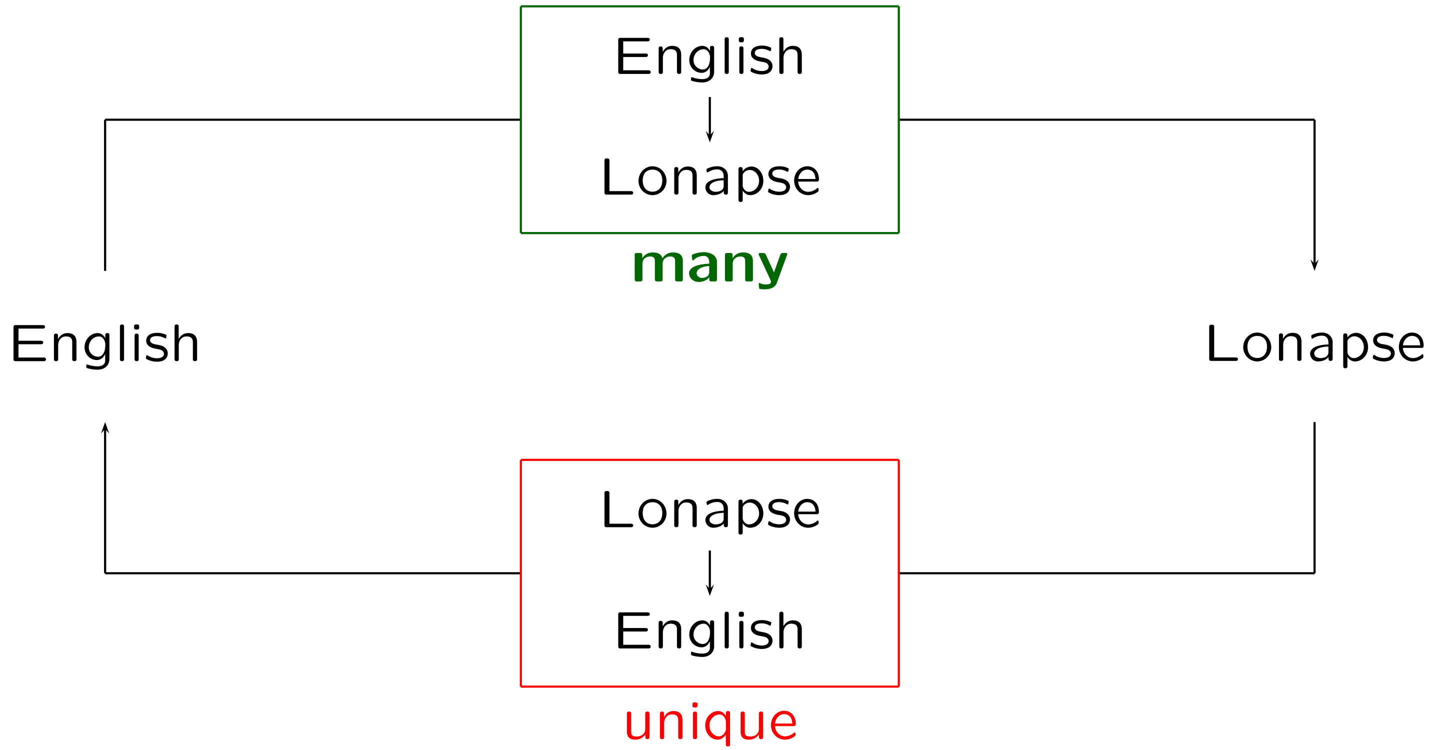
Data Encryption Standard (DES, 1974)

- Pick a secret shared key of 64 bits.
- Divide the message in blocks of 64 bits.
- Encrypting one block consists of a combination of
  repeated $\oplus$ with parts of the key, permutations,
  breaking up in subblocks, and small functions by table.

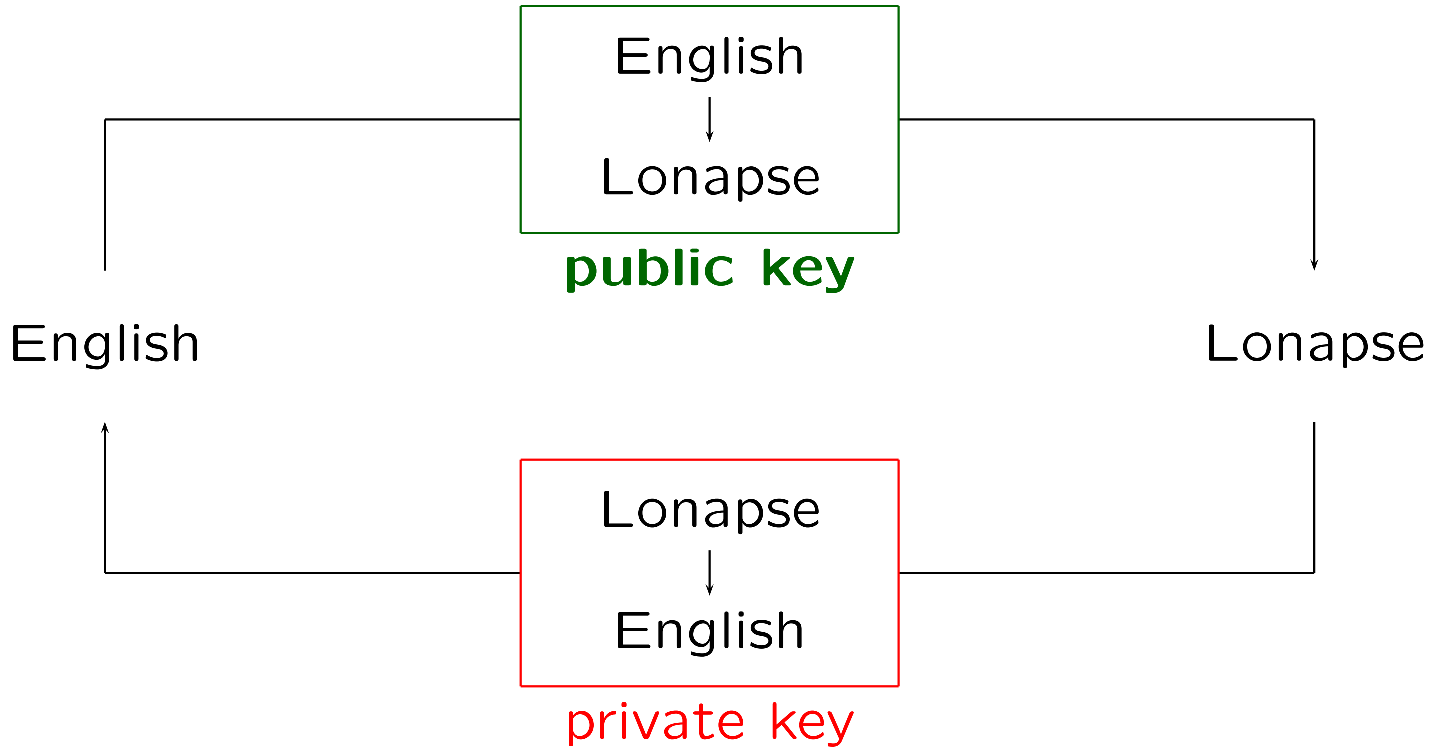
Disadvantage: Need to agree on a key before hand...
System uses a **secret shared key**

Problem: How do you prove a cryptography system is "secure"?

Public Keys


English
Lonapse
many
English
Lonapse
Lonapse
English
unique

Public Keys



English
Lonapse
public key
English
Lonapse
Lonapse
English
private key

Public Keys

$$
\begin{tikzcd}
M_E & M_L \arrow[l, "\text{L2E}"'] & \textbf{B} \\
M_E \arrow[r, "\text{E2L}"] & M_L \arrow[u] & \textbf{A}
\end{tikzcd}
$$

encrypting, sending,
and decrypting
a message

Public Keys

$$
\begin{tikzcd}
M_E & M_L \arrow[l, "\text{L2E}"] & \text{B} \\
M_E \arrow[r, "\text{E2L}"] & M_L \arrow[u] & \text{A}
\end{tikzcd}
$$

encrypting, sending,
and decrypting
a message

English and Lonapse have same words!

Public Keys

$$\begin{tikzcd}
M_E & M_L \arrow[l, "\text{L2E}"] & \text{B} & M_E \arrow[r, "\text{L2E}"'] & M_{NL} \arrow[d] \\
M_E \arrow[r, "\text{E2L}"] & M_L \arrow[u] & \text{A} & ?M_E? & M_{NL} \arrow[l, "\text{E2L}"']
\end{tikzcd}$$

encrypting, sending,
and decrypting
a message

signing, sending,
and **checking the signature**
of a message

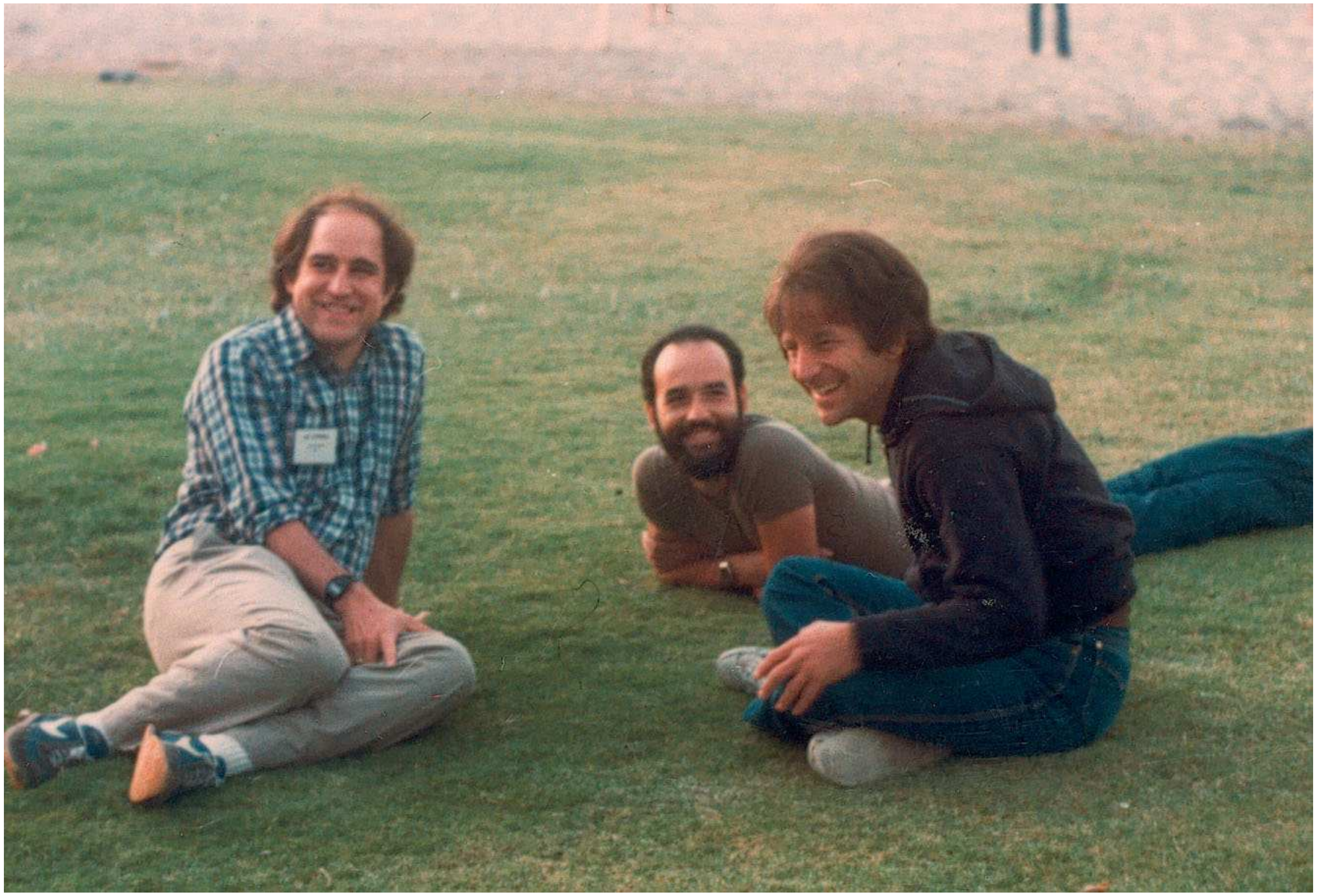
English and Lonapse have same words!

Public Keys (RSA)

RSA (Rivest, Shamir, Adleman):

An $n >> 0$, a **public** key e, and a private key d,

such that $x^{de} \equiv x \mod n$ for all x.



Public Keys (RSA)

$0 < M < n$
$x^{de} \equiv x \mod n$

$$
\begin{tikzcd}
M \equiv (M^e)^d & M^e \arrow[l] & \mathbf{B} & M \arrow[r] & M^d \arrow[d] \\
M \arrow[r] & M^e \arrow[u] & \mathbf{A} & M \stackrel{?}{\equiv} (M^d)^e & M^d \arrow[l]
\end{tikzcd}
$$

encrypting, sending,
and decrypting
a message M

signing, sending,
and **checking the signature**
of a message

Public Keys (RSA)

Security of this system is based on our inability to take e-th roots.

A factorization of n allows one to compute d from e.

It is believed that finding d is as hard as factorizing n.

So breaking this system would be as hard as factorizing n.

Public Keys (RSA)

Security of this system is based on our inability to take e-th roots.

A factorization of n allows one to compute d from e.

It is believed that finding d is as hard as factorizing n.

So breaking this system would be as hard as factorizing n.

Advantages:

compact, use in smart cards

both encryption and signing

Public Keys (RSA)

Security of this system is based on our inability to take e-th roots.

A factorization of n allows one to compute d from e.

It is believed that finding d is as hard as factorizing n.

So breaking this system would be as hard as factorizing n.

Advantages:

compact, use in smart cards

both encryption and signing

Disadvantages:

Computationally intensive

only small messages

man-in-the-middle attack
(weakness of public keys)

RSA only encripts small messages

For **signing**, you can just sign a hash-function of the message instead.

$$\begin{tikzcd}
\text{B} & & M \arrow[rr] & & {[M, H(M)^{d}]} \arrow[d] \\
\text{A} & H(M) \stackrel{?}{\equiv} (H(M)^{d})^{e} & & & {[M,\ H(M)^{d}]} \arrow[lll]
\end{tikzcd}$$

signing, sending,
and **checking the signature**
of a message

RSA only encripts small messages

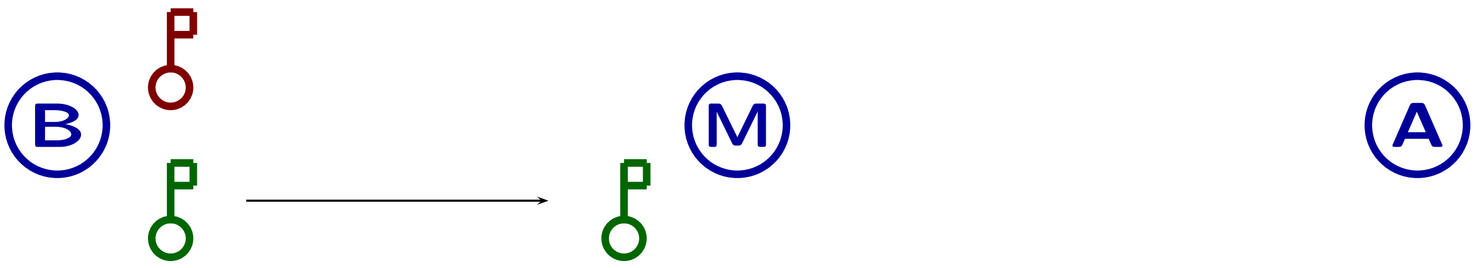
For **encryption**, one can use public-key systems to agree on a shared secret key for a more efficient encryption algorithm (like **triple-DES**).

A certain way of doing this is called **PGP** (Pretty Good Privacy)

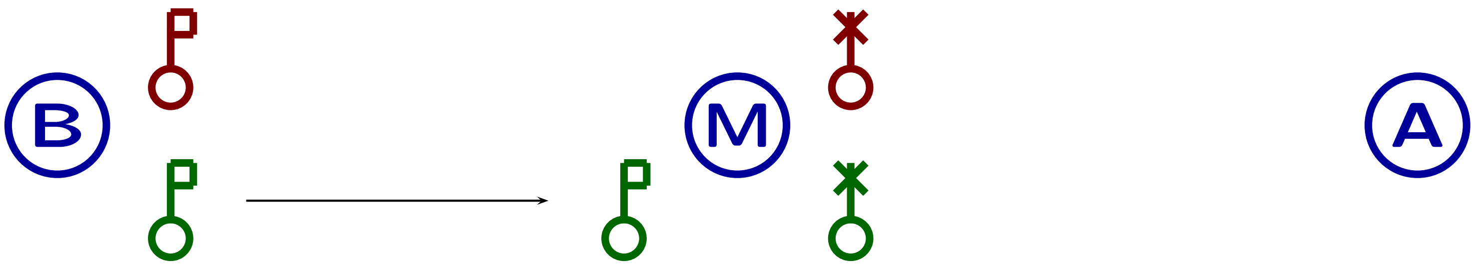
Public key systems and the man-in-the-middle attack

Public key systems and the man-in-the-middle attack

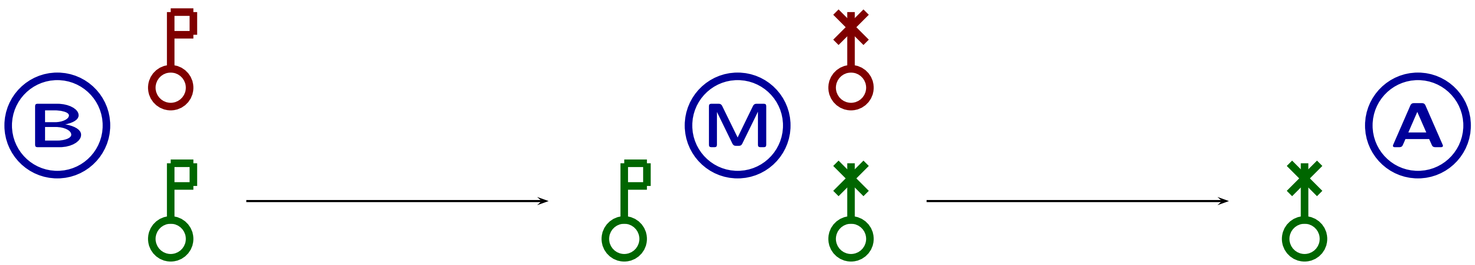
Public key systems and the man-in-the-middle attack



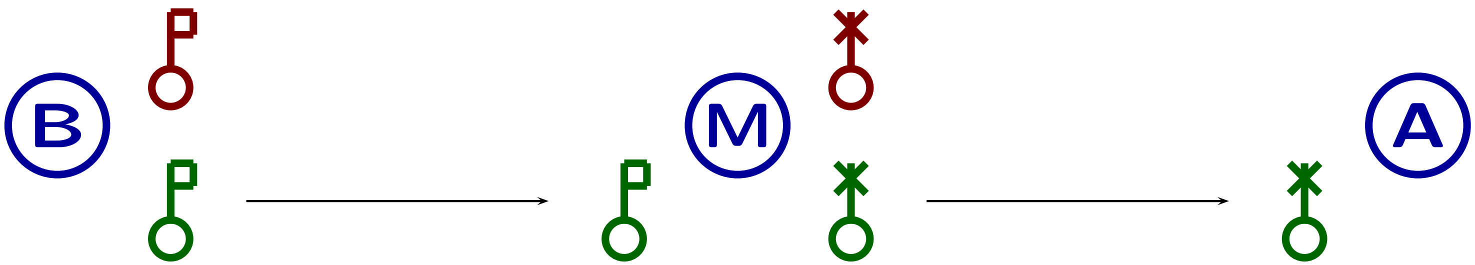
Public key systems and the man-in-the-middle attack



Public key systems and the man-in-the-middle attack



Public key systems and the man-in-the-middle attack



Solution: A trusted third party
(online companies that garantee you are you
by checking your credit card info)

Important

- Factorizing integers

Important

- Factorizing integers

- Discrete logarithms (tomorrow)

Important

- Factorizing integers
- Discrete logarithms (tomorrow)
- Coffee (now)