Understanding Cryptography: From Caesar to RSA

Nicola Chiapolini

UZH

Inverted CERN School of Computing, 3-4 March 2011

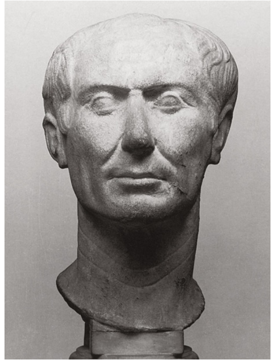
Outline

1. Basic Substitution & Attacks
2. More Complex Methods
3. Something Completely Different?
4. Two Modern Algorithms

Caesar

if there was occasion for secrecy, he wrote in cyphers; that is, he used the alphabet in such a manner, that not a single word could be made out. The way to ***decipher*** *those epistles was to* ***substitute*** *the fourth for the first letter, as* ***d for a****, and so for the other letters respectively*

„The Lives of the Twelve Caesars”
Gaius Suetonius



Caesar

The resulting pair of plain and cipher alphabet is:

plain: ABCD**E**FGHIJKLMNOPQRS**T**UV**W**XYZ
cipher: XYZA**B**CDEFGHIJKLMNOP**Q**RS**T**UVW

Example

We try this simple message here

plaintext: **WET**RY THISS IMPLE MESSA GEHER E
ciphertext: **TBQ**OV QEFPP FJMIB JBPPX DBEBO B

Caesar - Remarks

CERN School of Computing

Key

- shift defines cipher completely (e.g. 3 letters).
- key space very small (brute force by hand)

Rot13

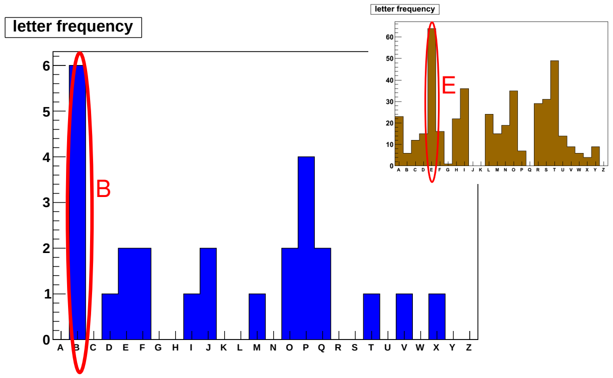
- encryption identical to decryption ($A \xrightarrow{13} N \xrightarrow{13} A$)
- e.g to prevent spoilers

Caesar - Attack

CERN School of Computing

Letter Frequency

T**B**QOV QEFPP FJMI**B** J**B**PPX D**B**E**B**O **B**



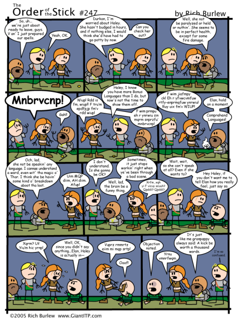
Caesar - Attack

CERN School of Computing

Plain text

- we know/guess that the cipher text contains `message`
- there is only one possible position

```
TBQOV QEFPP FJMIB JBPPX DBEBO B
                  MESSA GE
```

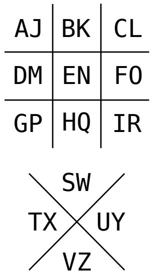

The Order of the Stick #247
by Rich Burlew
Mnbrvcnp!
©2005 Rich Burlew www.GiantITP.com

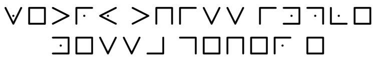
Mnbrvcnp
Treasure

Randomised Substitution

CERN School of Computing

- use permutations
- big key-space (26!)
- could use symbols



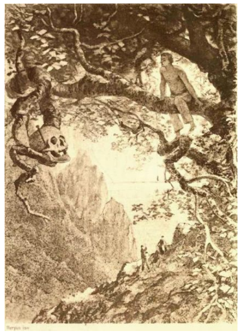


Randomised Substitution - Attack

CERN *i*CSC School *of* Computing

Attacks

- plain text attack
- frequency analysis for letters: `ETAON`
- frequency analysis for larger groups
- use partially decrypted words



„The Gold-Bug"
E.A.Poe

Multiple Cipher Equivalents

CERN *i*CSC School *of* Computing

Goal: flatten the letter distributions

- define multiple options for the cipher alphabet
- need symbols or numbers to extend this idea further

plain: ABCD**E**FGH**IJ**KLMNOPQRSTUVWXYZ

cipher: BEIN**T**ACW**YY**QMJPKOGSLXRZDHVU

F

Multiple Cipher Equivalents - Attack

CERN *i*CSC School *of* Computing

GMAZD VACKC ARHPG IYTZW RYKHY OTDEK GWDCS

EATJU HEOKA MAHAI GETGR YKSEH ADRKZ AHYCU

JXAKH YAKJU HKYGV DTHYU TCGZM DCKYE MYAWA

HAKAC RYHYR ZAKPG ARHDT GCKJU HKPDW UHAYB

GMDTA ARWAM AWUEZ KYUCG PACUH XYCKD BBYTH

CKYTG XYMDG MAZBT YXPAX CDZBE RWBTY XECZE

TFGEO ATOZD ECPGO ERARB ZUDRO G**ERWH** **PDXYT**

GPDGV DTOAC GCPAX CDZBA RKPAC WUHAK PGXYT

DIACG ZAPDW ATGOH CPACD BBYTK C**ERWH** **PGXYT**

DCUOO GCCBU ZKPDC GDBBY THCET GKPDX YTGPD

IAZZS TYJEJ ZAAXS TYMGE RWDVE ZHPAC YIRXA

RW**ERW** **KPGXY** **TD**OYX SZGHD ZAWYG CPDES SGETK

YBUZB AZHPD IAZZY BPACO TGEKY T

Summary of the Basics

CERN *i*CSC School *of* Computing

Ideas

1. shift alphabet
2. randomise alphabet
3. use multiple cipher equivalents

Attack Methods

- brute force
- plaintext
- frequencies (letters, pairs, trigrams)
- repeated segments

Vigenere Cipher - Algorithm

Idea use more than one full cipher alphabet.

```
plain:     ABCDEFGHIJKLMNOPQRSTUVWXYZ
cipher C:  CDEFGHIJKLMNOPQRSTUVWXYZAB
cipher E:  EFGHIJKLMNOPQRSTUVWXYZABCD
cipher R:  RSTUVWXYZABCDEFGHIJKLMNOPQ
cipher N:  NOPQRSTUVWXYZABCDEFGHIJKLM

plaintext:  WETRY THISS IMPLE MESSA GEHER E
key:        CERNC ERNCE RNCER NCERN CERNC E
ciphertext: YIKEA XYVUW ZZRPV ZGWJN IIYRT I
```

Vigenere Cipher - Attack

1. **determine key-length n**
2. split into n columns
3. use letter frequency for each column separately

Coincidence Counting

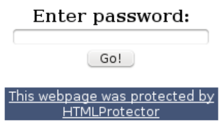
```
ciphertext: YIKEA XYVUW ZZRPV ZGWJN IIYRT I
shift 2:      YIK EAXYV UWZZR PVZGW JNIIY RT I
shift 4:        Y IKEAX YVUWZ ZRPVZ GWJNI IYRT I
```

key-length: **4**

CSC 2010: HTML riddle

Code

1. Take password and create hash
2. check hash against given value
3. if check passed, decipher secret message by XOR with password letters



Enter password:

Go!

This webpage was protected by HTMLProtector

CSC 2010: HTML riddle - Key Length

```
0x68 0x56 0x42 0x18 0x50 0x4B 0x52 0x18 0x47
                    0x68 0x56 0x42 0x18 0x50

0x5C 0x45 0x41 0x11 0x5A 0x42 0x4A 0x58 0x56
0x4B 0x52 0x18 0x47 0x5C 0x45 0x41 0x11 0x5A

0x42 0x4B 0x11 0x49 0x52 0x4A 0x42 0x56 0x59
0x42 0x4A 0x58 0x56 0x42 0x4B 0x11 0x49 0x52

0x19 0x11 0x76 0x7C 0x16 0x11 0x70 0x17 0x54
0x4A 0x42 0x56 0x59 0x19 0x11 0x76 0x7C 0x16

0x58 0x4F 0x52 0x18
0x11 0x70 0x17 0x54 0x58 0x4F 0x52 0x18
```

CSC 2010: HTML riddle - Columns

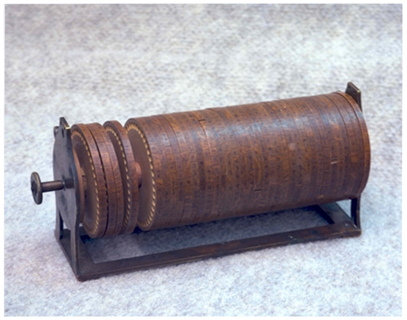
```
0x68 0x56 0x42 0x18 0x50 0x4B 0x52 0x18
0x47 0x5C 0x45 0x41 0x11 0x5A 0x42 0x4A
0x58 0x56 0x42 0x4B 0x11 0x49 0x52 0x4A
0x42 0x56 0x59 0x19 0x11 0x76 0x7C 0x16
0x11 0x70 0x17 0x54 0x58 0x4F 0x52 0x18
0x58 0x57 0x17 0x5B 0x58 0x4D 0x4E 0x18
0x7A 0x78 0x7A 0x7D 0x7F 0x6A 0x7C 0x15
0x64 0x6B 0x76 0x74 0x62 0x72 0x7E 0x61
0x1D 0x19 0x62 0x4A 0x50 0x55 0x17 0x4A
0x54 0x5E 0x5E 0x57 0x5F 0x17 0x17 0x71
0x11 0x58 0x5A 0x18 0x03 0x0C 0x17 0x41
0x54 0x58 0x45 0x4B 0x11 0x56 0x5B 0x5C
0x1F 0x19 0x7A 0x41 0x11 0x5F 0x56 0x4E
0x5E 0x4B 0x5E 0x4C 0x54 0x19 0x43 0x50
```

CSC 2010: HTML riddle - Key Letter

```
0x11 = 0001 0001
0x20 = 0010 0000
       0011 0001 = 0x31 = '1'
```

http://www.physik.uzh.ch/~nchiapol/icsc

Extending Vigenere: Wheel Cipher



- 36 disks
- key: disk ordering
- assemble plaintext in one line
- read off ciphertext in any other line

Extending Vigenere: Plaintext Auto-Key

```
plaintext:  WETRY THISS IMPLE MESSA GEHER E
key:        CERNW ETRYT HISSI MPLEM ESSAG E
ciphertext: YIKEU XAZQL PUHDM YTDWM KWZEX I
```

Weakness

- The key has the properties of plaintext.
- Strings in key and plaintext at fixed offset.

Extending Vigenere: Key Auto-Key

- simple rule (sum of last 3 key letters)
- hard to do manually
- simple rules are probably weak
- mechanical devices
- pseudo random number generator

```
plaintext:  WETRY THISS IMPLE MESSA GEHER E
key:        24734 41944 75689 30257 46770 4
ciphertext: YIAUC XIRWW PRVTN PEUXH KKOLR I
```

Extending Vigenere: The Enigma

The Enigma

- randomised alphabets
- generates different alphabet for each letter
- sequence defined by rotors configuration
- huge periods:

$$26^{nRotors}$$

A provably secure cipher

The One-Time Pad

- generate truly random key with same length as message
- use key only once

© D. Rijmenants 2009

Block Ciphers

- ciphers worked on one letter at a time
- could use groups instead

CERN School of Computing

Playfair

Playfair

- encrypt pairs of letters at a time
- pair in row: next to right
- pair in column: next below
- else: take diagonally opposed

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

Example

```
plaintext:  WETRY THISS IMPLE MESSA GEHER E
ciphertext: EDSNN YBOXX KOSIN KRQUR DNDAN R
```

CERN School of Computing

Playfair - Attack

- needs a lot of cipher text
- use statistics of letter pairs
- hope for plaintext segments
- algorithm has typical properties e.g. ER → RN

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

CERN School of Computing

Summary Complex Methods

Ideas

1. use multiple alphabets
2. generate the key during encryption
3. encrypt groups of letters

Attack Methods

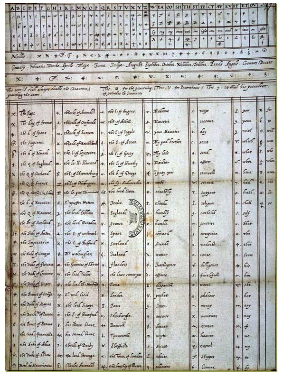
- coincidence counting
- reuse of key
- knowing the rules

CERN School of Computing

Codes

Instead of letters or groups of letters we could replace whole words.

- limited possibilities
- need large code books
- useful for compression



Steganography

- mark letters with pinhole
- invisible ink
- microdots
- selected bits in image file

Transposition Methods

Example

`plaintext: WE TRY THIS` **`SIMPLE`** `MESSAGE HERE`

W	E	T	R	Y
T	H	I	S	**S**
I	**M**	**P**	**L**	**E**
M	E	S	S	A
G	E	H	E	R
E	T	A	O	N

`ciphertext: WT`**`I`**`MG EEH`**`M`**`E ETTI`**`P`**` SHARS `**`L`**`SEOY `**`SE`**`ARN`

Data Encryption Standard

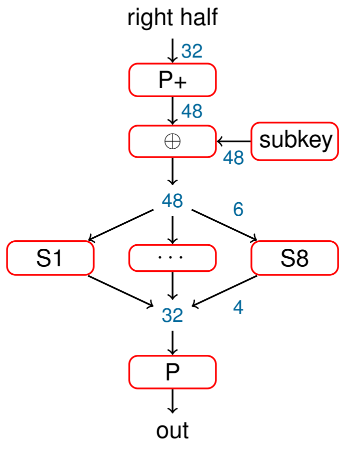
- Block Cipher
- Blocks of 64 Bits
- Key: 64 Bits (56 used)

Generating Key Stream

- use plaintext & key
- XOR
- Permutations
- Substitution

Data Encryption Standard

1. split block in halves
2. prepare subkey
3. **combine subkey and right half**
4. xor result and left half
5. switch halves
6. restart at 2 (16x)



Data Encryption Standard

Result

Each bit in each block depends on every other bit of the block and all the key bits.

- no statistical properties remain
- prevents differential cryptanalysis (injecting plaintexts with minimal differences)

Public Key

The Basic Idea

use different keys for encryption and decryption

- need a problem that is hard to solve
- but easy with additional information

Example

Factorise: $7031 = 79 \cdot 89$

Public Key - RSA

Step by Step: Generate Key

1. choose two prime numbers p, q
2. calculate $n = p \cdot q$
3. calculate $f = (p-1) \cdot (q-1)$
4. find two numbers e, d such that $e \cdot d = 1 \mod f$
5. publish e and n, keep d secret

$p = 11, q = 7$
$n = 77$
$f = 60$

$e = 23$

$$d = x \cdot \frac{f}{e} + \frac{1}{e} \qquad x \in \mathbb{N}$$

$$\Rightarrow d = 47 \qquad (x = 18)$$

Public Key - RSA

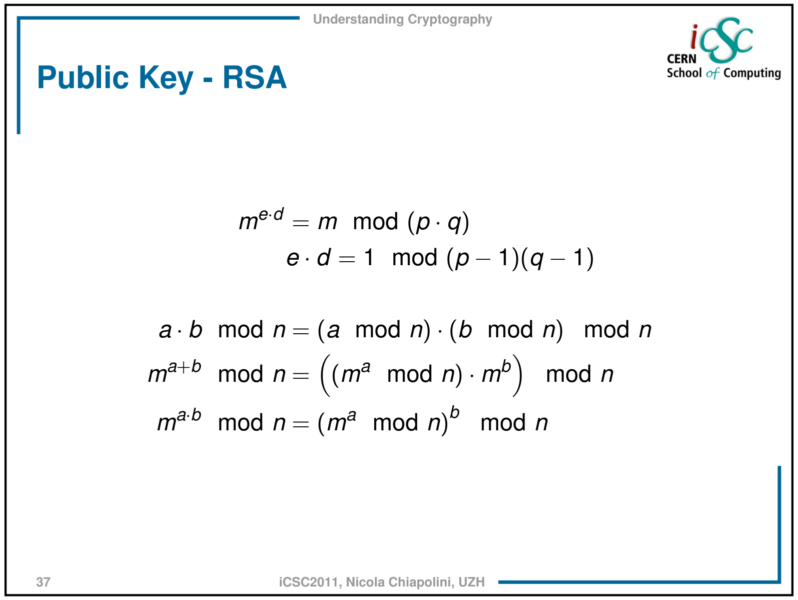
Step by Step: Usage

1. encrypt:
   $c = m^e \mod n$
2. decrypt:
   $m = c^d \mod n$

$m = 42 \quad (< n)$

$c = 42^{23} \mod 77 = 14$

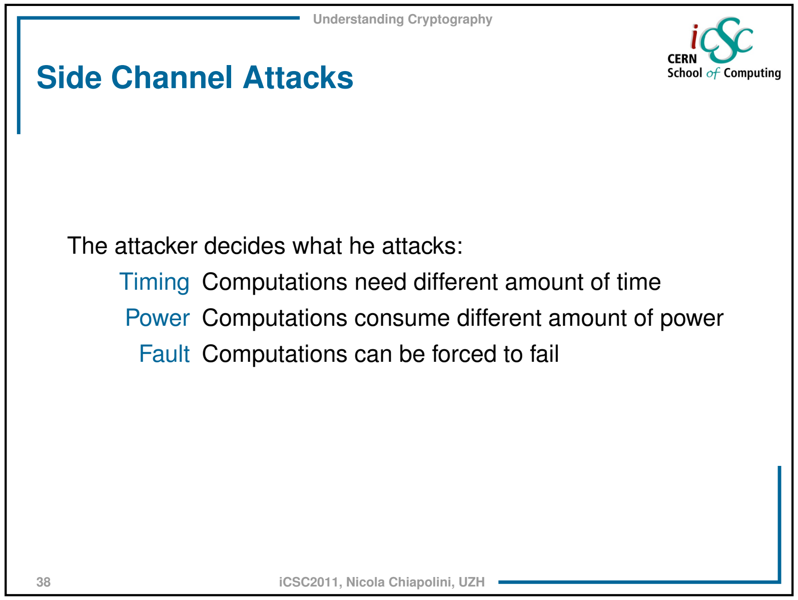
$m = 14^{47} \mod 77 = 42$



Public Key - RSA

$$m^{e\cdot d} = m \mod (p\cdot q)$$
$$e\cdot d = 1 \mod (p-1)(q-1)$$

$$a\cdot b \mod n = (a \mod n)\cdot(b \mod n) \mod n$$
$$m^{a+b} \mod n = \left((m^a \mod n)\cdot m^b\right) \mod n$$
$$m^{a\cdot b} \mod n = (m^a \mod n)^b \mod n$$



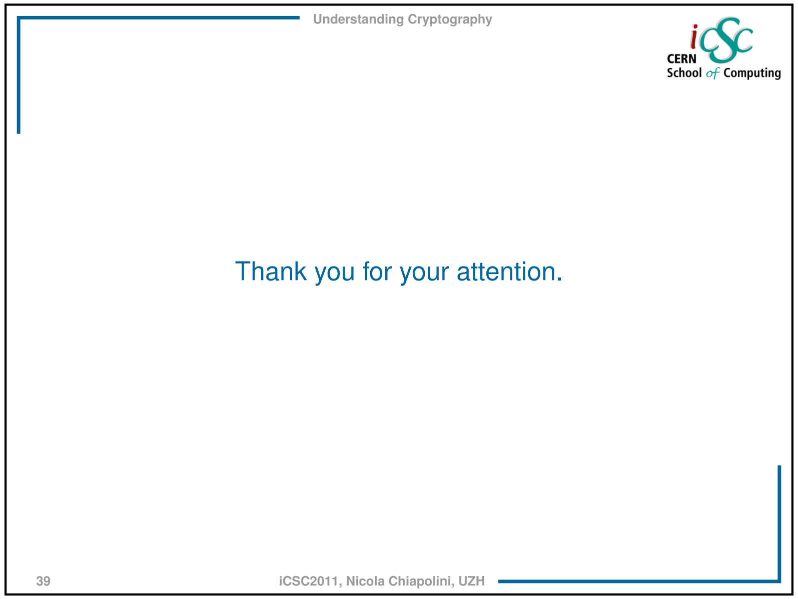
Side Channel Attacks

The attacker decides what he attacks:

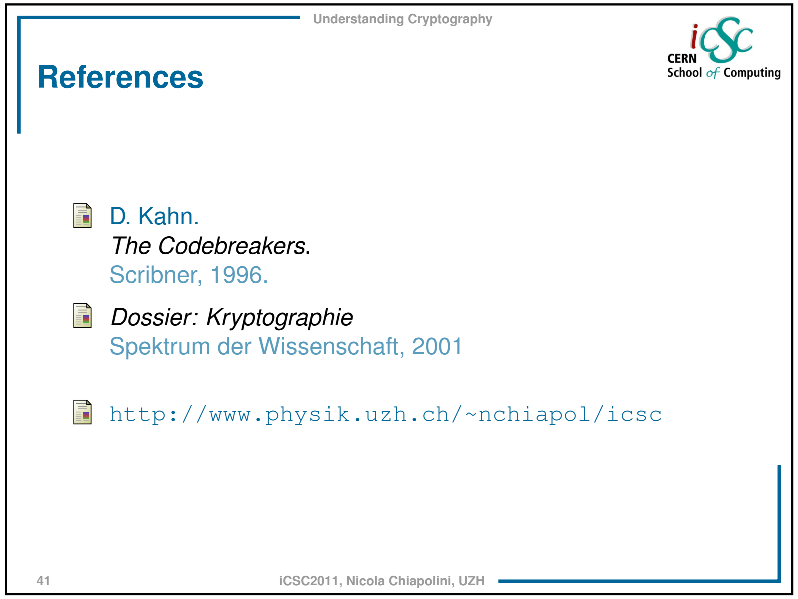
Timing Computations need different amount of time

Power Computations consume different amount of power

Fault Computations can be forced to fail



Thank you for your attention.



References

D. Kahn.
The Codebreakers.
Scribner, 1996.

Dossier: Kryptographie
Spektrum der Wissenschaft, 2001

http://www.physik.uzh.ch/~nchiapol/icsc