

# Cryptography and Security

Serge Vaudenay



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

<http://lasec.epfl.ch/>

LASEC

- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

- 1 Ancient Cryptography**
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

## 1 Ancient Cryptography

- Summary of this Chapter
- Terminology
- Cryptography Prehistory
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

# A 3-Phases Evolution

- **Prehistory**

cryptography before communication systems (**confidentiality**)

- **Industrial era**

communication and information systems (**confidentiality**)

- **Modern cryptography**

since 1976

mass communication

academic research

(**confidentiality, integrity, authentication, privacy, non-repudiation, fairness, access control, timestamping, etc**)

# Milestones of Prehistory

- 1 **security by obscurity**: private encryption algorithms  
several techniques: *substitutions* and *transpositions*
- 2 encryption with a configurable **secret key**  
e.g., Vigenère
- 3 **Kerckhoffs principle**  
→ security should rely on the secrecy of the key only  
(not on the secrecy of the algorithm)

# From Industrial Era to Modern Crypto

- **communicating**  
information theory  
mass communication (radio)
- **computing**  
computer science  
automata (electromechanic devices)

# Early Milestones of Modern Cryptography

- **Kerckhoffs** (1883): principles of modern crypto
- **Shannon** (1949): an info-theoretical approach of cryptography
- **Diffie-Hellman** (1976): public-key cryptography
- **DES** (1977): encryption standard for non-military applications



## 1 Ancient Cryptography

- Summary of this Chapter
- Terminology
- Cryptography Prehistory
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

# Cryptography vs Coding Theory

- **Code**

*a system of symbols which represent information*

- **Coding theory**

*science of code transformation which enables to send information through a communication channel in a reliable and efficient way (→ dummy adversary)*

- **Cryptography**

*(a bit obsolete) the science of secret codes, enabling the confidentiality of communication through an insecure channel (→ malicious adversary)*

- **Cipher**

*secret code, enabling the expression of a public code by a secret one by making the related information confidential*

# Cryptanalysis

- **Cryptanalysis, cryptographic analysis, cryptoanalysis**  
*theory of security analysis of cryptographic systems*
- **To cryptanalyze a cryptosystem** ( $\neq$  to break it)  
*to prove or to disprove the security provided by a cryptosystem*
- **To break a cryptosystem**  
*to prove insecurity (= to disprove security)*
- **Cryptology**  $\neq$  cryptography  
*science of cryptography and cryptanalysis (sometimes also includes steganography)*
- **Steganography**  $\neq$  cryptography  
*science of information hiding*

# Once Upon a Time, in the XIX-th Century



Alfred de Musset



George Sand

# Steganography

[censored]

[censored]

[censored]

# Cryptographic Problems

In ancient time:

- encryption

In modern cryptography:

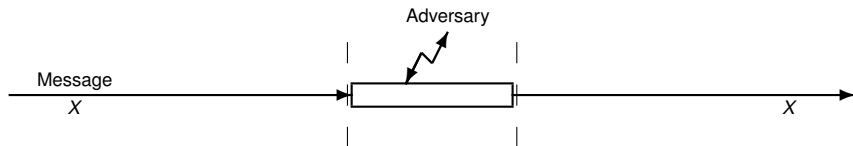
- encryption
- detection malicious modification of information
- data authentication
- access control
- timestamping
- fair exchange
- digital rights management
- privacy



# Applications

- bank cards
- Internet (e-commerce)
- mobile telephony (DECT, GSM, GPRS, EDGE, 3GPP...)
- e-passport
- mobile communication (Bluetooth, WiFi...)
- traceability, logistic & supply chains (RFID)
- pay-TV, DRM
- access control (car lock systems, metro...)
- payment (e-cash)
- electronic voting

# The Fundamental Trilogy



- **Confidentiality (C)**: only the legitimate receiver can get  $X$
- **Authentication + Integrity (A+I)**: only the legitimate sender can insert  $X$  and the received message must be equal to  $X$

# Basic Security Properties

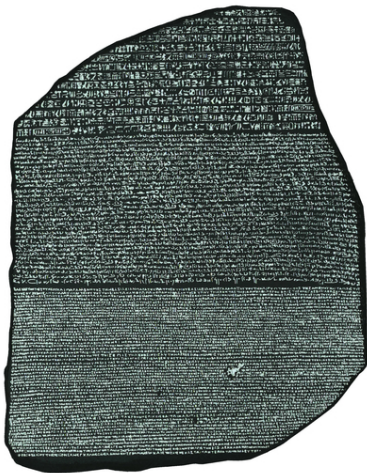
- **Confidentiality**  
the information should not leak to any unexpected party
- **Integrity**  
the information must be protected against any malicious modification
- **Authentication**  
the information should make clear who is its author

## 1 Ancient Cryptography

- Summary of this Chapter
- Terminology
- **Cryptography Prehistory**
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

# Secret Writing

Hieroglyphs!



# Transpositions

## Spartan scytales

this\_is\_a\_dummy\_message



t	h	i	s	_	i
s	_	a	_	d	u
m	m	y	_	m	e
s	s	a	g	e	



TSMH\_MSIAYAS\_G\_DMEIUE



# Simple Substitution: Caesar Cipher

a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	v	x
D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	V	X	A	B	C

caesar → FDHXDV

Quiz:

Q: How to break this?

A: ol fgngvfgvpny nanylfvf

▶ ROT13

# Simple Substitution: ROT13

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  
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

rot → EBG

Application: quiz

Q: Where can we find good quiz?

A: va pnenzone pnaqvr

▶ ROT13



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

crypto  $\rightarrow$  LCNMRT

Number of possible tables:  $26! \approx 2^{88.4}$

# Probabilities of Occurrence in English

letter	probability	letter	probability	letter	probability
A	0.082	J	0.002	S	0.063
B	0.015	K	0.008	T	0.091
C	0.028	L	0.040	U	0.028
D	0.043	M	0.024	V	0.010
E	0.127	N	0.067	W	0.023
F	0.022	O	0.075	X	0.001
G	0.020	P	0.019	Y	0.020
H	0.061	Q	0.001	Z	0.001
I	0.070	R	0.060		

# Rough Frequencies in English

- 1 most frequent: E
- 2 very frequent: T A O I N S H R
- 3 frequent: D L
- 4 rare: C U M W F G Y P B
- 5 very rare: V K J X Q Z

30 most common digrams (in decreasing order):

TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND,  
OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI *and* OF.

12 most common trigrams (in decreasing order):

THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR *and* DTH.

# A Simple Substitution Cipher (from Stinson)

-----  
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

-----  
NDIFEFMZCZDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

-----  
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-----  
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

# Step I: Frequency Analysis

letter	frequency	letter	frequency	letter	frequency
A	0	J	11	S	3
B	1	K	1	T	2
C	15	L	0	U	5
D	13	M	16	V	5
E	7	N	9	W	8
F	11	O	0	X	6
G	1	P	1	Y	10
H	4	Q	4	Z	20
I	5	R	10		

# Solution

[homework]

# Vigenère Cipher

**Plaintext:** this is a dummy message

**Key:** ABC

		this		is		a		dummy		message
+		ABCA		BC		A		BCABC		ABCABCA
<hr/>										
=		TIKS		JU		A		EWMNA		MFUSBIE

**Ciphertext:** TIKSJUAEWMNAMFUSBIE

*e.g.*  $y + C = A$ .

# Character Addition Rule

+	a	b	c	d	e	f	g	...
A	A	B	C	D	E	F	G	...
B	B	C	D	E	F	G	H	...
C	C	D	E	F	G	H	I	...
D	D	E	F	G	H	I	J	...
E	E	F	G	H	I	J	K	...
F	F	G	H	I	J	K	L	...
G	G	H	I	J	K	L	M	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

cultural remark: using the mapping (isomorphism)  $a \leftrightarrow 0, b \leftrightarrow 1, c \leftrightarrow 2, \dots$  this is the addition modulo 26  
(group  $\mathbf{Z}_{26}$ )



# Column-Dependent Substitution

A	B	C		A	B	C
t	h	i		T	I	K
s	i	s		S	J	U
a	d	u		A	E	W
m	m	y	→	M	N	A
m	e	s		M	F	U
s	a	g		S	B	I
e				E		

# Kasiski Test

to check a guess  $n$  for the key length

- look at repeating patterns at a distance multiple of  $n$
- check that this is significant

# Kasiski Test Example

→ look at unexpectedly frequent patterns

CHR EEVOAHMAERATB IAXXWTNXBEEOPHBSBQMQUEQERBW  
RVXUOAKXAOSXXWEAHBWGJMMQMKNKGRFVGXWTRZXWIAK  
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX  
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR  
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBI EQJT  
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHA EYE VTAQE BB I  
PEEWEVKAKOEWA DREMXMTBHHCHRTKDNVRZCHRCLQOHP  
WQA I IWXNRMGWO I I FKEE

CHR occurs at 1, 166, 236, 276, 286.

# Question

In a random string of 313 characters from an alphabet of 26 letters, is it common to observe 5 occurrences of the same trigram?

# Reminders on Combinatorics

- number of  $k$ -tuples of elements in a set of size  $z$ :  
example  $z = 3, k = 2$ : 00, 01, 02, 10, 11, 12, 20, 21, 22

$$z^k$$

Application ( $k = 3, z = 26$ ): #possible trigrams is  $26^3 = 17\,576$

- number of possible subsets of  $t$  elements in a set of size  $n$ :  
example  $n = 3, t = 2$ :  $\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$

$$\binom{n}{t} = \frac{n!}{t!(n-t)!} = \frac{n \cdot (n-1) \cdots (n-t+1)}{t \cdot (t-1) \cdots 1}$$

Application: probability to get ball  $u$  (drawn with probability  $p$ ) exactly  $t$  times out of  $n$  samples:  $\binom{n}{t} p^t (1-p)^{n-t}$   
(binomial distribution)

# Are 5 Occurrences Significant? — i

In a truly random sequence of 313 characters  $x_1 x_2 \dots x_{313}$  with alphabet of 26 letters

- there are  $n = 311$  trigrams  $t_1 = x_1 x_2 x_3$ ,  $t_2 = x_2 x_3 x_4$ , ...  
 $t_n = x_n x_{n+1} x_{n+2}$
- every possible trigram  $abc$  has a number of occurrences  
 $n_{abc} = \sum_{i=1}^n 1_{t_i=abc}$
- approximation: all  $t_i$ 's are independent and uniformly distributed in a set of  $\frac{1}{p} = 26^3 = 17\,576$  possibilities
- $\Pr[n_{abc} = t] = \binom{n}{t} p^t (1-p)^{n-t}$   
Note:  $n \times p$  is small

# Reminders on Calculus

- Stirling Formula:

$$n! \approx \sqrt{2\pi n} \times n^n e^{-n}$$

weaker formula:  $\log n! \approx n(\log n - 1)$

- for  $\lambda = n \times p \ll 1$ :

(example:  $n = 311$ ,  $p = \frac{1}{17\,576}$ ,  $t \leq 5$ )

$$\text{(binomial)} \quad \binom{n}{t} p^t (1-p)^{n-t} \approx \frac{\lambda^t}{t!} e^{-\lambda} \quad \text{(Poisson)}$$

- Taylor development on  $e^\lambda$ :

$$e^\lambda = \sum_{i=0}^{t-1} \frac{\lambda^i}{i!} + \int_0^\lambda \frac{(\lambda-x)^{t-1}}{(t-1)!} e^x dx$$

## Are 5 Occurrences Significant? — ii

$$e^\lambda = \sum_{i=0}^{t-1} \frac{\lambda^i}{i!} + \int_0^\lambda \frac{(\lambda - x)^{t-1}}{(t-1)!} e^x dx$$

- $\Pr[n_{abc} = t] \approx \frac{\lambda^t}{t!} e^{-\lambda}$  with  $\lambda = \frac{311}{17576}$
- we have

$$\begin{aligned} \Pr[n_{abc} \geq t] &\approx 1 - \sum_{i=0}^{t-1} \frac{\lambda^i}{i!} e^{-\lambda} \\ &= e^{-\lambda} \int_0^\lambda \frac{(\lambda - x)^{t-1}}{(t-1)!} e^x dx \\ &\leq \int_0^\lambda \frac{(\lambda - x)^{t-1}}{(t-1)!} dx \\ &= \frac{\lambda^t}{t!} \end{aligned}$$



## Are 5 Occurrences Significant? — iii

$$\Pr[n_{abc} \geq t] \leq \frac{\lambda^t}{t!}$$

- maximize over all  $abc$ :

$$\max_{abc} \Pr[n_{abc} \geq t] \leq \sum_{abc} \Pr[n_{abc} \geq t]$$

- with  $t = 5$  we have

$$\max_{abc} \Pr[n_{abc} \geq t] \leq 26^3 \frac{\lambda^t}{t!} \leq 10^{-6}$$

so the probability to get at least 5 occurrences of the same trigram is less than  $10^{-6}$

- conclusion:

observing 5 occurrences of CHR is significantly odd

# Where does CHR Come From?

key of length multiple of 5 + frequent trigram

·	·	·	·	·		·	·	·	·	·
<hr/>						<hr/>				
t	h	e	·	·		C	H	R	·	·
·	·	·	·	·		·	·	·	·	·
·	·	·	·	·		·	·	·	·	·
·	·	·	·	·	→	·	·	·	·	·
t	h	e	·	·		C	H	R	·	·
·	·	·	·	·		·	·	·	·	·
t	h	e	·	·		C	H	R	·	·

# Index of Coincidence

$$\text{Index}(x_1, \dots, x_n) = \Pr[x_I = x_J | I < J] = \sum_{c \in Z} \frac{n_c(n_c - 1)}{n(n - 1)}$$

where  $I, J \in \{1, \dots, n\}$  are independent and uniformly distributed

## Proposition

For any permutation  $\sigma$  over  $Z$ , we have

$$\text{Index}(\sigma(x_1), \dots, \sigma(x_n)) = \text{Index}(x_1, \dots, x_n)$$

For any permutation  $\sigma$  of  $\{1, \dots, n\}$ , we have

$$\text{Index}(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = \text{Index}(x_1, \dots, x_n)$$

the index of coincidence is invariant by substitution and transposition

- $\text{Index}(\text{English text}) \rightarrow 0.065$  when  $n \rightarrow +\infty$
- $\text{Index}(\text{Random string}) \rightarrow 0.038$  when  $n \rightarrow +\infty$

# Application to the Vigenère Cipher

With the example TIKSJUAEWMNAMFUSBIE, if we guess that the key is of length 3, we can write

T	I	K
S	J	U
A	E	W
M	N	A
M	F	U
S	B	I
E		

so we can compute the index of coincidence of TSAMMSE, IJENFB and KUWAUI.

## Example — i

guess the key is of length 4

C	H	R	E
E	V	O	A
H	M	A	E
R	A	T	B
I	A	X	X
W	T	N	X
⋮	⋮	⋮	⋮

first column:

CEHRIWBPBEBXKSEWMKVTLSTDTXIGSXLVUNWGXLSXLZLSNRMGEABJRRNXMNHAVEPEKAMBHDZCHAXGIE

(string of 79 characters)

$$\text{Index}(\text{col}) = \text{Index}(A^4 B^5 C^2 D^2 E^7 G^4 H^4 I^3 J^1 K^3 L^5 M^4 N^4 P^2 R^4 S^5 T^3 U^1 V^3 W^4 X^7 Z^2)$$

which is 0.0422: this is too low

## Example — ii

guess the key is of length 5

C	H	R	E	E
V	O	A	H	M
A	E	R	A	T
B	I	A	X	X
W	T	N	X	B
E	E	O	P	H
⋮	⋮	⋮	⋮	⋮

first column:

CVABWEBQBUAWWQRWWXANTBDPXXRDWBFAXCWMNJJFAIACNRNCATBWKDMCDCQQXWK

(string of 63 characters)

$$\text{Index}(\text{col}) = \text{Index}(A^7 B^6 C^6 D^4 E^1 F^2 I^1 J^2 K^2 M^2 N^4 P^1 Q^4 R^3 T^2 U^1 V^1 W^9 X^5) = 0.0630$$

this is high enough!

## Example — iii

- next step: find the first character of the key
- note that  $W$  is frequent while  $U$  and  $V$  are much less frequent and  $Y$  and  $Z$  are inexistent
- in English,  $h$  is frequent while  $f$  and  $g$  are much less frequent and  $j$  and  $k$  are sparse
- idea: guess that  $W$  is the encryption of  $h$
- $h + P = W$
- the first character of the key may be  $P$

## 1 Ancient Cryptography

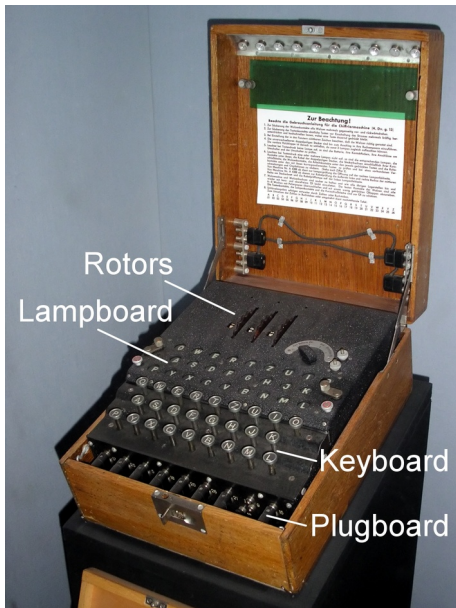
- Summary of this Chapter
- Terminology
- Cryptography Prehistory
- **Pre-Modern Industrial Cryptography**
- Cryptography and Information Theory



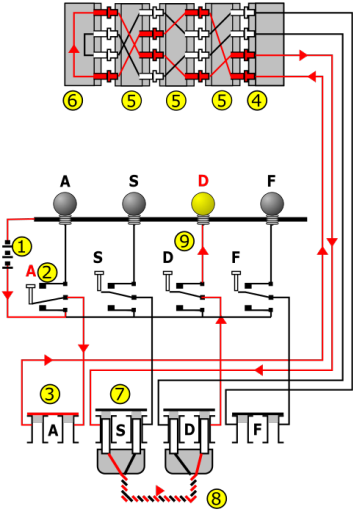
# Enigma

- electro-mechanical encryption device (typewriter)
- could be plugged to a radio transmitter
- patented (1918)
- developed to be secure even with public specifications (Kerckhoffs principle), in hostile environment (battlefield)
- used by German armies in WW2
- preliminary attacks by polish mathematician Rejewski in 1932 (before Anschluss)
- “industrial” (over 2000 messages decrypted per day) attack by UK intelligence at Bletchley Park during WW2 (performing: Turing)

# Picture of Enigma

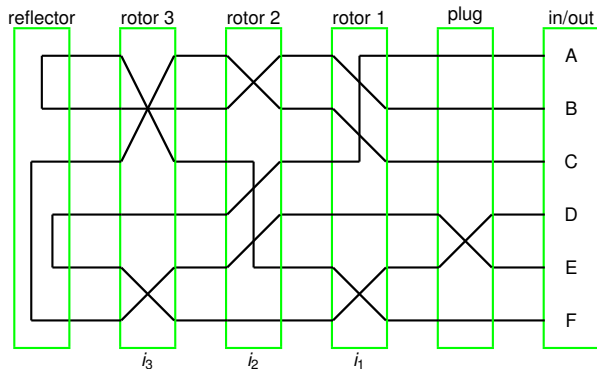


# Enigma Circuit



[https://en.wikipedia.org/wiki/Enigma\\_machine](https://en.wikipedia.org/wiki/Enigma_machine)

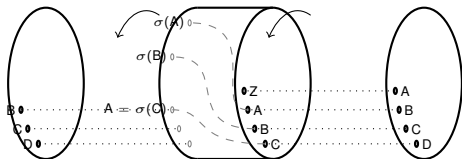
# Example: DEAD BEEF



deadbeef → AADCCBBB

# Enigma Building Blocks

- given a permutation  $\sigma$  over  $\mathcal{Z} = \{A, B, \dots, Z\}$ , a **fixed point** is an element  $x \in \mathcal{Z}$  such that  $\sigma(x) = x$
- an **involution** over  $\mathcal{Z}$  is a permutation  $\sigma$  of  $\mathcal{Z}$  such that  $\sigma(\sigma(x)) = x$  for all  $x$ .  
Examples: reflector, plug board
- a **rotor**  $\sigma$  defines a set of permutations  $\sigma_0, \dots, \sigma_{25}$  over  $\mathcal{Z}$  the rotor in position  $i$  implements permutation  $\sigma_i$  such that  $\sigma_i = \rho^i \circ \sigma \circ \rho^{-i}$  where  $\rho(A) = B, \rho(B) = C, \dots, \rho(Z) = A$



# The Enigma Cipher (Mathematically) — i

We define permutations over the 26-character alphabet.

**Reflexion.**  $\pi$  is a fixed involution with no fixed points.

**Rotors.**  $S$  be a set of five permutations over the alphabet.  
 $\rho$  is the circular rotation over the alphabet by one position.

$\rho^i$  thus denotes the circular rotation over the alphabet by  $i$  positions.

$\alpha_i$  denotes  $\rho^i \circ \alpha \circ \rho^{-i}$

**Wire connection.**  $\sigma$  is a configurable involution with 6 pairs (14 fixed points)

# The Enigma Cipher (Mathematically) — Example

$x$ :	$A$	$B$	$C$	$D$	$E$	$F$
$\rho(x)$ :	$B$	$C$	$D$	$E$	$F$	$A$
$\alpha(x)$ :	$C$	$A$	$B$	$D$	$F$	$E$
$\alpha_0(x)$ :	$C$	$A$	$B$	$D$	$F$	$E$
$\alpha_1(x)$ :	$F$	$D$	$B$	$C$	$E$	$A$
$\alpha_2(x)$ :	$B$	$A$	$E$	$C$	$D$	$F$
$\alpha_3(x)$ :	$A$	$C$	$B$	$F$	$D$	$E$
$\alpha_4(x)$ :	$F$	$B$	$D$	$C$	$A$	$E$
$\alpha_5(x)$ :	$F$	$A$	$C$	$E$	$D$	$B$

$$\alpha_i = \rho^i \circ \alpha \circ \rho^{-i}$$

# The Enigma Cipher (Mathematically) — ii

**Secret key:** 3 components:

- $\sigma$
- an ordered choice  $\alpha, \beta, \gamma \in S$  of pairwise different permutations
- a number  $a$

**Plaintext:**  $x = x_1, \dots, x_m$

**Ciphertext:**  $y = y_1, \dots, y_m$

**Encryption:**

$$y_i = \sigma^{-1} \circ \alpha_{i_1}^{-1} \circ \beta_{i_2}^{-1} \circ \gamma_{i_3}^{-1} \circ \pi \circ \gamma_{i_3} \circ \beta_{i_2} \circ \alpha_{i_1} \circ \sigma(x_i)$$

where  $i_3 i_2 i_1$  are the last three digits of the basis 26 numeration of  $i + a$ .



# Key Entropy in Enigma

- $\sigma$ : number of involutions with 14 fixed points

$$\begin{aligned} & \binom{26}{14} \times 11 \times 9 \times 7 \times \dots \times 1 \\ &= 9\,657\,700 \times 11 \times 9 \times 7 \times \dots \times 1 \\ &= 100\,391\,791\,500 \\ &\approx 2^{37} \end{aligned}$$

- $\alpha, \beta, \gamma$ : number of choices for the rotors

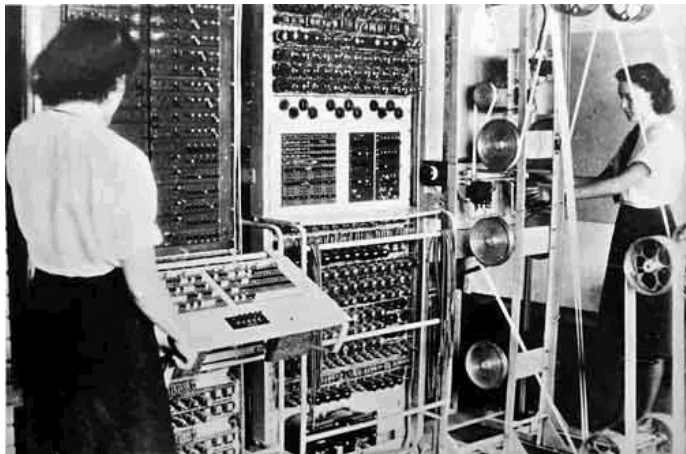
$$5 \times 4 \times 3 = 60 \approx 2^6$$

- $a$ : number of initial positions

$$26^3 = 17\,576 \approx 2^{14}$$

total: 57 bits

# A Turing Machine



Can we reasonably assume that the adversary ignores the cryptosystem?

# The Laws of Modern Cryptography

## Law I: the Kerckhoffs Principle

security should not rely on the secrecy of the cryptosystem itself

- motivation:  
the adversary may get some information about the system (e.g. by reverse engineering, corruption, etc)
- meaning:  
security analysis must assume that the adversary knows the cryptosystem
- does not mean:  
cryptosystem must be public



# Kerckhoffs Principles

## Kerckhoffs Principles

- 1 Le système doit être matériellement, sinon mathématiquement, indéchiffrable;
- 2 **Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi;**
- 3 La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants;
- 4 Il faut qu'il soit applicable à la correspondance télégraphique;
- 5 Il faut qu'il soit portatif et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes;
- 6 Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

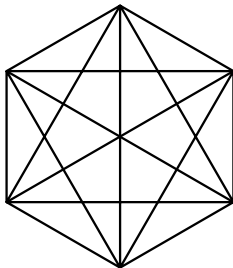
# The Laws of Modern Cryptography

## Law II: the $n^2$ Problem

in a network of  $n$  users, there is a number of potential pairs of users within the order of magnitude of  $n^2$

- we cannot assume that every pair of users share a secret key
- we must find a way for any pair of users to establish a shared secret key

# How Many Symmetric Keys?



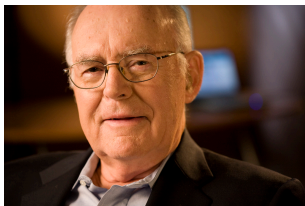
for  $n$  users we may need up to  $\frac{n(n-1)}{2}$  symmetric keys

# The Laws of Modern Cryptography

## Law III: the Moore Law

the speed of CPUs doubles every 18–24 months

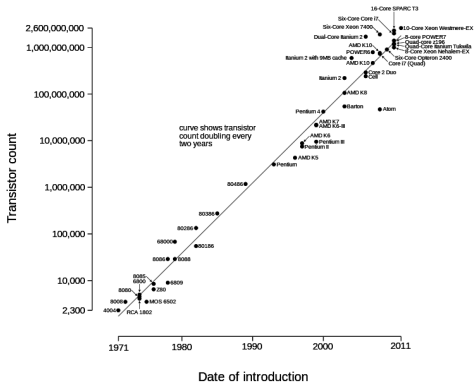
- we should wonder how long a system must remain secure
- we must estimate the speed of CPU at the end of this period
- we assess security against brute force attacks





# Moore's Law

Microprocessor Transistor Counts 1971-2011 & Moore's Law



$$f_t \approx 10^9 \times 2^{\frac{1}{2 \text{ years}}(t-2004)} \times \text{cste}$$

number of keys per second which can be tested in an exhaustive search with technology at time  $t$

# Security by Key Length

- to offer security between current time  $t_0$  until time  $t_0 + \Delta$ , the key length must be **at least**

$$\text{margin} + \log_2 \left( \int_{t_0}^{t_0 + \Delta} f_t dt \right)$$

- assuming that  $f_t$  is exponential, the key length must be  $\Omega(\Delta)$

# A 128-Bit Key

11000000	10010011	00000011	01001001
11010011	11110010	01111011	10100101
10101001	00110001	00110000	11011110
00101110	01001110	00011111	00100001

number of possible combinations:

$$\begin{aligned} & \overbrace{2 \times 2 \times 2 \times \cdots \times 2}^{128 \text{ times}} \\ = & 2^{128} \\ = & \underbrace{340\,282\,366\,920\,938\,463\,463\,374\,607\,431\,768\,211\,456}_{39 \text{ digits}} \end{aligned}$$

# Exhaustive Search on 128 Bits

- in 2007, a standard PC could test 1 000 000 keys per second
- to run exhaustive search within 15 billion years, we need 720 000 billions of 2007-PCs!
- if the Moore law goes on, in 2 174, a single PC will do in within a second
- better create the Big Bang and take **15 billion years of vacations** to solve the problem within a second!

# The Laws of Modern Cryptography

## Law IV: the Murphy Law

if there is a single security hole, the system will fall into it

- never leave a security hole
- don't bet on security, rather prove it



## 1 Ancient Cryptography

- Summary of this Chapter
- Terminology
- Cryptography Prehistory
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

# Bitwise Exclusive Or

- exclusive or (XOR) of two bits:

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

- XOR: binary addition where carry bits are ignored
- XOR: addition modulo 2
- bitwise XOR of two bitstrings:

$$\begin{array}{r} 10010 \\ \oplus 00111 \\ \hline = 10101 \end{array}$$

- XOR properties
  - closure: the XOR of bitstrings is a bitstring
  - associative:  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
  - commutative:  $a \oplus b = b \oplus a$
  - neutral element:  $a \oplus [00 \cdots 0] = a$
  - (self-)invertibility:  $a \oplus a = [00 \cdots 0]$  (or  $+ = -$ )

# Vernam Cipher

- we use a uniformly distributed random key  $K$  (a bitstring)
- every message  $X$  requires a new  $K$  of same size (one-time pad)
- Encrypting  $X$  with  $K$ : compute  $X \oplus K$
- Decrypting  $Y$  with  $K$ : compute  $Y \oplus K$

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

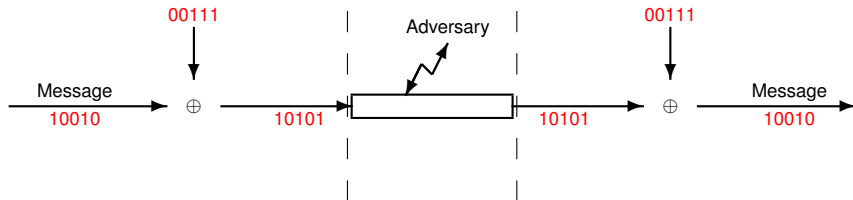
$$\begin{array}{r} \oplus \quad (X) \quad 10010 \\ \quad (K) \quad 00111 \\ \hline = \quad (Y) \quad 10101 \end{array}$$
  
$$\begin{array}{r} \oplus \quad (K) \quad 00111 \\ \quad (X) \quad 10010 \\ \hline = \quad (Y) \quad 10101 \end{array}$$



# Vernam Cipher



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

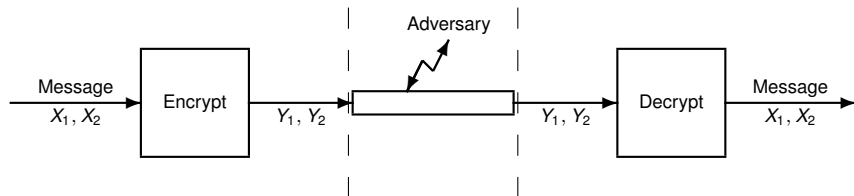


When is this insecure?

# Using the Same Key Twice

$$Y_1 = X_1 \oplus K$$

$$Y_2 = X_2 \oplus K$$

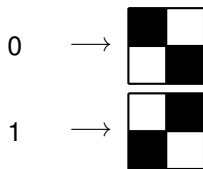


$$Y_1 \oplus Y_2 = (X_1 \oplus K) \oplus (X_2 \oplus K) = (X_1 \oplus X_2) \oplus (K \oplus K) = X_1 \oplus X_2$$

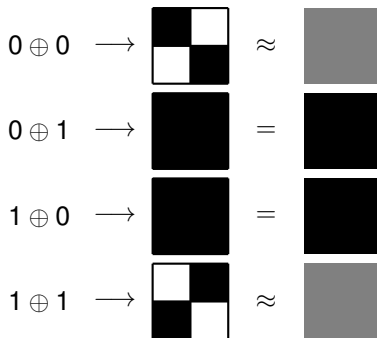
leakage of the  $X_1 \oplus X_2$  value

# Visual Cryptography

Pixel coding



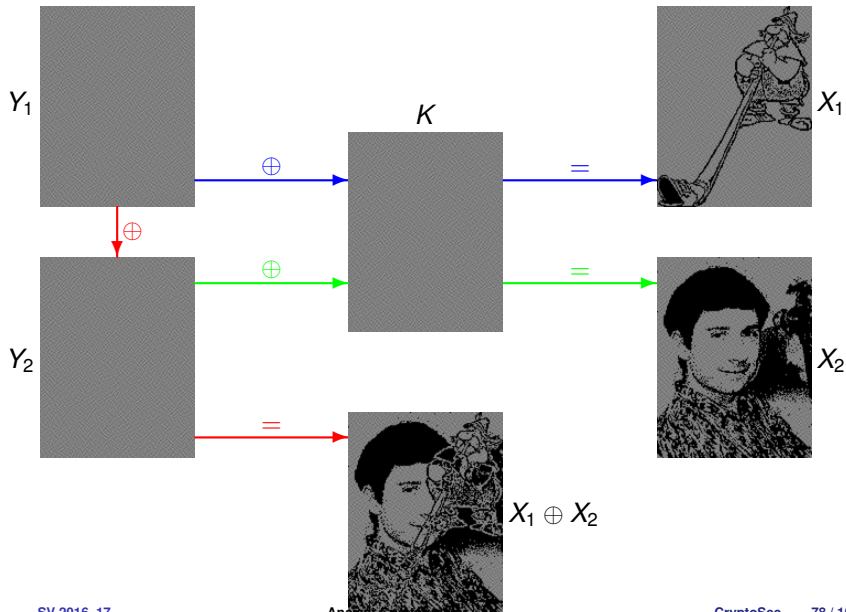
Pixel XOR



# Example



# Using the Same Key Twice



# Insecurity Cases in the Vernam Cipher

- if  $K$  is smaller than  $X$

$$Y = (X_L \oplus K) || X_R$$

→ insecure

- if  $K$  is not uniformly distributed

$$\Pr[K = k] \text{ high} \implies \Pr[X = y \oplus k] \text{ high}$$

→ insecure

- if  $K$  is used twice and messages are redundant

$$Y_1 \oplus Y_2 = X_1 \oplus X_2 \implies \text{information about } X_1 \text{ and } X_2$$

→ insecure

# Summary of Security Requirements

- the key must have (at least) the same length of the message
- the key must be uniformly distributed
- the key must be thrown away after usage
  
- 😞: this makes no sense for most of applications!
- 😊: this provides perfect security
- makes sense to prepare emergency communication (red telephone)
  - keys are exchanged (through slow channels) before the messages to transmit are known
- bad news for other application: there is essentially no better cipher with this strong security property



# Intuition on Why it is Perfectly Secure

- if the adversary gets  $Y = y$  then for any  $x$

$$\Pr[X = x | Y = y] = \Pr[X = x | X \oplus K = y] = \Pr[X = x]$$

because  $X$  and  $X \oplus K$  are statistically independent

the adversary gets no information about  $X$  in knowing that  $Y = y$

# Abelian Group Laws

## Definition

An **Abelian group** is a set  $G$  together with a mapping from  $G \times G$  to  $G$  which maps  $(a, b)$  to an element denoted  $a + b$  and such that

1. [closure] for any  $a, b \in G$ , we have  $a + b \in G$
2. [associativity] for any  $a, b, c$ , we have  $(a + b) + c = a + (b + c)$   
(notation:  $n.a$  means  $a + a + \dots + a$  ( $n$  times))
3. [neutral element] there exists an element denoted by  $0$  s.t. for any  $a$ ,  $a + 0 = 0 + a = a$
4. [invertibility] for any  $a$  there exists an element denoted by  $-a$  s.t.  $a + (-a) = (-a) + a = 0$  (notation:  $a - b$  means  $a + (-b)$ )
5. [commutativity] for any  $a, b \in G$ , we have  $a + b = b + a$

- $\mathbf{Z}$  with the regular addition
- $\{0, 1\}^n$  with  $\oplus$
- $\{0, 1, \dots, n-1\}$  with  $(a, b) \mapsto \begin{cases} a + b & \text{if } a + b < n \\ a + b - n & \text{otherwise} \end{cases}$

# Useful Lemma

## Lemma

*Let  $X$  and  $K$  be two independent random variables in a given group. If  $K$  is uniformly distributed, then  $Y = K + X$  is uniformly distributed and independent from  $X$ .*

### Proof.

For any  $x$  and  $y$ :

$$\begin{aligned}\Pr[X = x, Y = y] &= \Pr[X = x, K = y - x] \\ &= \Pr[X = x] \times \Pr[K = y - x] \\ &= \Pr[X = x] \frac{1}{\#\text{group}} \\ \Pr[Y = y] &= \sum_x \Pr[X = x, Y = y] \\ &= \frac{1}{\#\text{group}}\end{aligned}$$



# Generalized Vernam Cipher

Let  $G$  be an Abelian group and consider an arbitrary plaintext source producing elements in  $G$

- let  $K$  be uniformly distributed in  $G$  and independent from the plaintext
- given  $X$ , the encryption of  $X$  with key  $K$  is  $Y = K + X$
- given  $Y$ , the decryption of  $Y$  with key  $K$  is  $X = (-K) + Y$
- the key is used only once

## Theorem

*For any distribution of  $X$  over  $G$ ,  $Y$  is independent from  $X$  and uniformly distributed.*

(perfect secrecy)

# Information Theory

Claude Shannon

[Claude Shannon]

## skip reminders on Shannon entropy

▶ skip

**CAUTION:** in cryptography, “entropy” is often used in an informal way by meaning some kind of “effective bit-length”

# Reminder on the Shannon Entropy — i

- $H(X)$ : number of bits of information to represent the value of  $X$
- $H(X, Y)$ : entropy of  $(X, Y)$
- $H(X|Y) = H(X, Y) - H(Y)$

$$H(X) = - \sum_x \Pr[X = x] \log_2 \Pr[X = x]$$

$$H(X, Y) = - \sum_{x,y} \Pr[X = x, Y = y] \log_2 \Pr[X = x, Y = y]$$

$$H(X|Y) = - \sum_{x,y} \Pr[X = x, Y = y] \log_2 \Pr[X = x|Y = y]$$

## Reminder on the Shannon Entropy — ii

- a real function  $f$  is convex on  $[a, b]$  iff

$$\forall \text{set } S \quad \forall t : S \rightarrow [a, b] \quad \forall p : S \rightarrow ]0, 1[$$

$$\sum_{x \in S} p_x = 1 \implies \sum_{x \in S} p_x f(t_x) \geq f\left(\sum_{x \in S} p_x t_x\right)$$

- it is strictly convex if we further have the property that equality implies all  $t_x$  are equal
- a real function  $f$  which has a second derivative on  $]a, b[$  is strictly convex on  $[a, b]$  iff its second derivative is always  $> 0$  on  $]a, b[$



## Reminder on the Shannon Entropy — iii

### Proposition

$H(X) \geq 0$  with equality if, and only if  $X$  is constant

### Proof.

- $f(t) = -\log_2 t$  is strictly convex on  $[0, 1]$   
take  $t_x = p_x = \Pr[X = x]$  and get

$$H(X) \geq -\log_2 \left( \sum_{x \in S} p_x^2 \right)$$

clearly,  $\sum_x p_x^2 \leq 1$  so this log is positive

- Assuming equality, we must have  $\sum_x p_x^2 = 1$  so all  $p_x$  must be equal to 1 so there must be a single  $x$  (we cannot have two different values with probability 1)  
(i.e.  $X$  is constant equal to this  $x$ )



## Reminder on the Shannon Entropy — iv

### Proposition

$H(X, Y) \geq H(X)$  with equality if, and only if  $Y$  can be written  $f(X)$

### Proof.

- We write

$$H(Y|X) = \sum_x \Pr[X = x] \sum_y \Pr[Y = y|X = x] \log_2 \Pr[Y = y|X = x]$$

We know that for each  $x$  the inner sum is  $\geq 0$  with equality iff there is a single  $y = f(x)$  for which  $\Pr[Y = y|X = x] > 0$

- Clearly:  $H(Y|X) \geq 0$
- Assuming equality, for each  $x$  we define  $y = f(x)$  and get  $\Pr[Y = f(x)|X = x] = 1$  for all  $x$   
so,  $\Pr[Y = f(X)] = 1$

□

$$H(Y|X) = - \sum_{x,y} \Pr[X = x, Y = y] \log_2 \Pr[Y = y|X = x]$$

# Reminder on the Shannon Entropy — v

## Proposition

$H(X, Y) \leq H(X) + H(Y)$  with equality if, and only if  $X$  and  $Y$  are independent.

## Proof.

- $t \mapsto t \ln t$  has second derivative  $\frac{1}{t}$  so it is convex and

$$-\sum_y \Pr[Y = y] t_y \log_2 t_y \leq -\left(\sum_y \Pr[Y = y] t_y\right) \log_2 \left(\sum_y \Pr[Y = y] t_y\right)$$

with equality iff all  $t_y$ 's for  $\Pr[Y = y] \neq 0$  are equal

- Applying this to  $t_y = \Pr[X = x | Y = y]$  yields

$$-\sum_y \Pr[X = x, Y = y] \log_2 \Pr[X = x | Y = y] \leq -\Pr[X = x] \log_2 \Pr[X = x]$$

with equality iff  $\Pr[X = x | Y = y]$  does not depend on  $y$

- summing up for all  $x$  leads to  $H(X|Y) \leq H(X)$  with equality iff  $X$  and  $Y$  are independent



# Reminder on the Shannon Entropy — vi

## Proposition

If  $\Pr[X = x] \neq 0$  for  $n$  values of  $x$  then  $H(X) \leq \log_2 n$  with equality if, and only if all non-zero  $\Pr[X = x]$  are equal to  $\frac{1}{n}$ .

## Proof.

- $t \mapsto -\ln t$  has second derivative  $\frac{1}{t^2}$  so is convex and

$$\sum_x \Pr[X = x] \log_2 t_x \leq \log_2 \left( \sum_x \Pr[X = x] t_x \right)$$

with equality iff all  $t_x$ 's for  $\Pr[X = x] \neq 0$  are equal

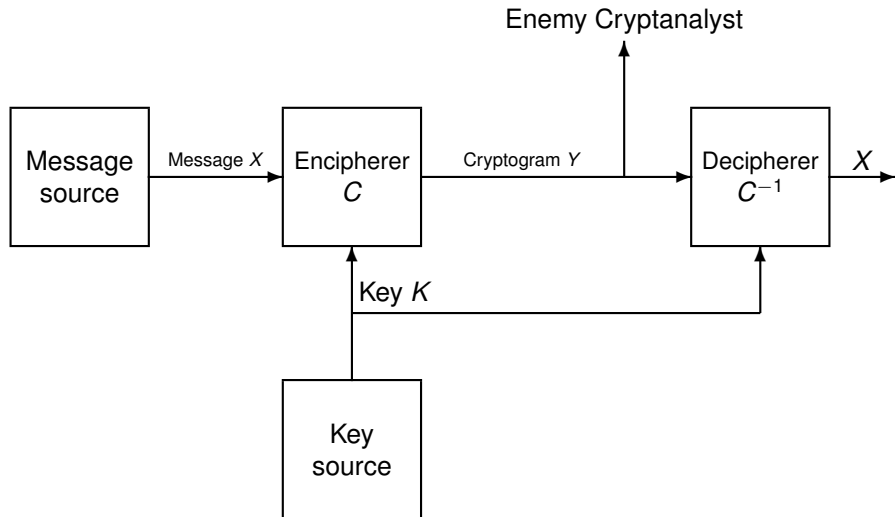
- Applying this to  $t_x = 1 / \Pr[X = x]$  yields

$$H(X) \leq \log_2 n$$

with equality iff all nonzero  $\Pr[X = x]$  are equal



# The Shannon Encryption Model



# The Shannon Encryption Model

- message is a random variable with a given a priori distribution  
for later: with *any* a priori distribution
- key is a random variable with specified distribution, independent from the message
- correctness property:  $\Pr[C_K^{-1}(C_K(X)) = X] = 1$
- adversary gets the random variable  $Y = C_K(X)$  only  
for other security models to be seen: other assumptions

# Perfect Secrecy — i

## Definition

**Perfect secrecy** means that the a posteriori distribution of the plaintext  $X$  after we know the ciphertext  $Y$  is equal to the a priori distribution of the plaintext:

$$\forall x, y \quad \Pr[Y = y] \neq 0 \implies \Pr[X = x | Y = y] = \Pr[X = x].$$

The adversary learns nothing about  $X$  by intercepting  $Y$ .  
(Remark: this definition is relative to the distribution of  $X$ .)

# Perfect Secrecy — ii

## Proposition

Perfect secrecy is equivalent to the statistic independence of  $X$  and  $Y$ .

### Proof.

Independence

$$\iff \forall x, y \quad \Pr[X = x, Y = y] = \Pr[X = x] \Pr[Y = y].$$

Since  $\Pr[X = x | Y = y] = \frac{\Pr[X=x, Y=y]}{\Pr[Y=y]}$  by definition, the result is trivial! □



# Perfect Secrecy — iii

## Proposition

Perfect secrecy is equivalent to  $H(X|Y) = H(X)$ .

### Proof.

Perfect secrecy is equivalent to statistical independence of  $X$  and  $Y$ .

Statistical independence of  $X$  and  $Y$  is equivalent to

$$H(X, Y) = H(X) + H(Y).$$

Since  $H(X|Y) = H(X, Y) - H(Y)$  the result is trivial. □

# Vernam Cipher Provides Perfect Secrecy

## Theorem

*For any distribution of the plaintext, the generalized Vernam cipher provides perfect secrecy.*

# Influence of the Plaintext Distribution

## Theorem

Let  $C_K$  be a cipher with  $K$  following a given distribution. Let  $p$  and  $p'$  be two distributions for  $X$  such that  $\text{support}(p') \subseteq \text{support}(p)$ .  $C_K$  provides perfect secrecy with  $p$  implies that  $C_K$  provides perfect secrecy with  $p'$ .

**Proof.** If  $\Pr_{p'}[Y = y] \neq 0$ , there exists  $k$  and  $x_0$  such that  $C_k(x_0) = y$ ,  $\Pr[K = k] \neq 0$ , and  $p'(x_0) \neq 0$ . Since  $\text{support}(p') \subseteq \text{support}(p)$ , we have  $p(x_0) \neq 0$  so  $\Pr_p[Y = y] \neq 0$ . Due to perfect secrecy,

$$\Pr_p[Y = y] = \Pr_p[Y = y|X = x] = \Pr[C_K(x) = y] = \Pr_{p'}[Y = y|X = x]$$

then

$$\begin{aligned}\Pr_{p'}[Y = y] &= \sum_x \Pr_{p'}[Y = y|X = x]p'(x) = \sum_x \Pr_p[Y = y|X = x]p'(x) \\ &= \Pr_p[Y = y] \sum_x p'(x) = \Pr_p[Y = y] = \Pr_{p'}[Y = y|X = x]\end{aligned}$$



# Shannon Theorem

## Theorem (Shannon 1949)

*Perfect secrecy implies  $H(K) \geq H(X)$ .*

### Proof.

- we have  $H(Y) \geq H(Y|K)$
- knowledge of  $K$  makes  $X \leftrightarrow Y$ , thus  $H(Y|K) = H(X|K)$
- since  $X$  and  $K$  are independent, we obtain  $H(Y|K) = H(X)$   
we thus have  $H(Y) \geq H(X)$
- knowledge of  $X$  makes  $K \rightarrow Y$ , thus  $H(Y, K|X) = H(K|X)$
- since  $X$  and  $K$  are independent,  $H(K|X) = H(K)$ , so  
 $H(Y, K|X) = H(K)$
- we have  $H(Y, K|X) \geq H(Y|X)$ , thus  $H(K) \geq H(Y|X)$
- if we have perfect secrecy, we have  
 $H(Y|X) = H(X|Y) + H(Y) - H(X) = H(Y)$   
thus, we have  $H(K) \geq H(Y) \geq H(X)$



# Other Form of the Shannon Theorem

## Theorem (Shannon 1949)

*Perfect secrecy implies that the support of  $K$  is at least as large as the support of  $X$ .*

**Proof.** Let  $y$  be such that  $\Pr[Y = y] \neq 0$ .

- since  $X$  and  $K$  must be independent

$$\Pr[X = x, Y = y] = \Pr[X = x, C_K(x) = y] = \Pr[X = x] \Pr[C_K(x) = y]$$

- perfect secrecy implies

$$\Pr[C_K(x) = y] = \Pr[Y = y | X = x] = \Pr[Y = y] \text{ for all } x \text{ such that } \Pr[X = x] \neq 0$$

- consequently, for all  $x$  in the support of  $X$  we have  $\Pr[C_K(x) = y] \neq 0$  so there exists one  $k$  in the support of  $K$  such that  $C_k(x) = y$ . Let write it  $k = f(x)$ .
- for any  $x$  in the support of  $X$  we have  $C_{f(x)}^{-1}(y) = x$ .

Clearly,  $f(x) = f(x')$  implies  $x = x'$ .

Consequently, we have an injection from the support of  $X$  to the support of  $K$ . □

# The Negative Side of Shannon Theorem

## Corollary

*If we want to achieve perfect secrecy the number of possible keys must be at least as large of the number of possible plaintexts.*

Conclusion: we cannot do better than the Vernam cipher

# Other Consequences

## Theorem

*Perfect secrecy implies that  $X$  has a finite support.*

### Proof.

- let  $y$  s.t.  $p = \Pr[Y = y] \neq 0$
- due to perfect secrecy we have  $\Pr[Y = y] = \Pr[C_K(x) = y]$  for all  $x$  in the support
- since  $[C_K^{-1}(y) = x] \iff [C_K(x) = y]$ , we have  $\Pr[C_K^{-1}(y) = x] \geq \Pr[C_K(x) = y] = p$  for all  $x$  in the support thus

$$1 \geq \sum_{x \in \text{support}} \Pr[C_K^{-1}(y) = x] \geq p \cdot \#\text{support}$$

- $\#\text{Support}(X) \leq \frac{1}{p}$



# Summary on the Shannon Results

- we have mathematically formalized the notion of perfect secrecy
- Vernam Cipher achieves perfect secrecy
- despite Vernam Cipher is expensive, there is no cheaper alternative

Q: Can the theory of cryptography stop here?

A: Abg lrg: jung zvfrrf vf gur abgvba bs  
pbzcyrkvgl

▶ ROT13



# Information Theory vs Complexity Theory

## Information Theory

## Complexity Theory

Is information there or not?

How much does it cost to recover information?

Is it *possible* to recover information?

Is it *doable* to recover information?

security shall rather be based on lower bounding the complexity of breaking the system

# The Early Days of Computer Science

Alan Turing



# Conclusion

- in prehistory: security by obscurity
- now a need for standard solutions
- perfect security requires an unreasonable cost
- conclusion: we must trade security against cost

# References

- **Singh.** *The Code Book*. Fourth Estate. 2000.  
Easy reading stories
- **Kahn.** *The Codebreakers*. Smith & Daniel. 1997.  
Textbook about (pre)history of cryptography
- **Levy.** *Crypto*. Penguin. 2001.  
Easy reading story about the begining of public-key cryptography
- **Hinsley-Stripp.** *The Inside Story of Bletchley Park*. Oxford University Press. 1993.
- **Naor-Shamir.** Visual Cryptography. In *EUROCRYPT 1994*, LNCS 950.
- **Shannon.** Communication Theory of Secrecy Systems. 1949.  
Re-edited by Sloane-Wyner Eds in *Claude Elwood Shannon collected papers*. IEEE Press. 1993.

# Must be Known

- Kerckhoffs principle
- the ACI trilogy (Authentication, Confidentiality, Integrity)
- Vernam cipher
- Shannon model of encryption
- perfect secrecy
- Shannon Theorem

# Train Yourself

- Vigenère: final exam 2009–10 ex1
- Vernam:
  - midterm exam 2010–11 ex3
  - midterm exam 2015–16 ex1
- entropy: final exam 2012–13 ex3







- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography**
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

# Roadmap

- reminders on arithmetics, groups,  $\mathbf{Z}_n$
- Diffie-Hellman key exchange over a group
- reminders on rings, fields,  $\mathbf{Z}_p^*$
- Diffie-Hellman key exchange, concretely
- ElGamal cryptosystem

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# Prime Numbers

## Definition

A prime number is a positive integer which has exactly two positive factors: 1 and itself.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...

# Unique Factorization

## Theorem

*Each integer  $n$  can be uniquely written*

$$n = u \times p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$$

*where  $p_1 < \cdots < p_r$  are prime,  $u = \pm 1$ , and  $\alpha_1, \dots, \alpha_r$  are non-negative integers.*

# Modulo $n$

Operation  $x \bmod n$ : remainder in the Euclidean division of  $x$  by  $n$

$$\begin{array}{r|l} x = 8273 & 143 = n \\ -715 & 57 = \lfloor x/n \rfloor \\ \hline 1123 & \\ -1001 & \\ \hline x \bmod n = 122 & \end{array}$$

$$8273 \bmod 143 = 122$$

$$8273 = 122 + 143 \times 57$$

# Euclidean Division



## Theorem (Euclidean Division)

For any  $a \in \mathbf{Z}$  and any  $n > 0$  there exists a unique pair  $(q, r) \in \mathbf{Z}^2$  such that  $a = qn + r$  and  $0 \leq r < n$ .

We denote  $r = a \bmod n$  and have  $q = \lfloor \frac{a}{n} \rfloor$ .

# Two Notations for “mod”

- **without parentheses:**  $x \bmod n$ 
  - a two-adic operator
  - = remainder in the Euclidean division of  $x$  by  $n$
- **with parentheses:**  $a \equiv b \pmod{n}$ 
  - an attribute to an equivalence relation (here:  $\equiv$ )
  - means that  $b - a$  is divisible by  $n$
  - or equivalently:  $a \bmod n = b \bmod n$
- do not mix up

$$\begin{array}{ccc} a = b \bmod n & \text{and} & a \equiv b \pmod{n} \\ \uparrow & & \uparrow \\ a \text{ set to } (b \bmod n) & & a \text{ and } b \text{ are (equal modulo } n \text{ )} \end{array}$$



# $\mathbf{Z}_n$ for Dummies

- $\mathbf{Z}_n = \{0, 1, \dots, n - 1\}$
- addition in  $\mathbf{Z}_n$ :  $a \boxplus b = (a + b) \bmod n$
- multiplication in  $\mathbf{Z}_n$ :  $a \boxtimes b = (a \times b) \bmod n$
- useful lemma:  $(a + (b \bmod n)) \bmod n = (a + b) \bmod n$
- useful lemma:  $(a \times (b \bmod n)) \bmod n = (ab) \bmod n$
- $\boxplus$  and  $\boxtimes$  closure: comes from  $x \bmod n \in \mathbf{Z}_n$  for any  $x \in \mathbf{Z}$
- $\boxplus$  associativity: comes from the lemma:

$$a \boxplus (b \boxplus c) = (a + ((b + c) \bmod n)) \bmod n = (a + b + c) \bmod n \dots$$

- $\boxtimes$  associativity: comes from the lemma:

$$a \boxtimes (b \boxtimes c) = (a \times ((bc) \bmod n)) \bmod n = (abc) \bmod n \dots$$

- neutral elements: 0 for  $\boxplus$  and 1 for  $\boxtimes$
- invertibility for  $\boxplus$ :  $(-a) \bmod n$ , comes from the lemma:

$$a \boxplus ((-a) \bmod n) = (a + ((-a) \bmod n)) \bmod n = (a - a) \bmod n = 0$$

- distributivity: comes from the lemma:

$$a \boxtimes ((b + c) \bmod n) = (a \times (b + c)) \bmod n = (ab + ac) \bmod n \dots$$

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# Definition of a Group

## Definition

A **group** is a set  $G$  together with a mapping from  $G \times G$  to  $G$  which maps  $(a, b)$  to an element denoted  $a \odot b$  and such that

1. [closure] for any  $a, b \in G$ , we have  $a \odot b \in G$
2. [associativity] for any  $a, b, c$ , we have  $(a \odot b) \odot c = a \odot (b \odot c)$
3. [neutral element] there exists an element  $e$  s.t. for any  $a$ ,  
 $a \odot e = e \odot a = a$
4. [invertibility] for any  $a$  there exists  $b$  s.t.  $a \odot b = b \odot a = e$

## Definition

An **Abelian group** is a set  $G$  together with a mapping from  $G \times G$  to  $G$  which maps  $(a, b)$  to an element denoted  $a \odot b$  and such that

- 1–4. [group] it is a group
5. [commutativity] for any  $a, b$  we have  $a \odot b = b \odot a$

# Additive vs Multiplicative Notations for Groups

	additive notations	multiplicative notations
group	$(G, +)$	$(G, \times)$
operation	$a + b$	$ab$
neutral element	$0$	$1$
inverse	$-a$	$a^{-1}$
exponential	$n.a$	$a^n$

( $a$  and  $b$  are group elements;  $n$  is an integer)

# Group Homomorphism

**Homomorphism:** given two groups  $(G_1, \times_1)$  and  $(G_2, \times_2)$ , a mapping  $f$  from  $G_1$  to  $G_2$  is a group homomorphism if for any  $a, b \in G_1$

$$f(a \times_1 b) = f(a) \times_2 f(b)$$

**Example:** If  $g \in G$ , the mapping  $\varphi : \mathbf{Z} \rightarrow G$  defined by  $\varphi(a) = g^a$  is a group homomorphism.  
 $\forall a, b \in \mathbf{Z} \quad \varphi(a + b) = \varphi(a)\varphi(b)$

**Isomorphism:** a group homomorphism which is bijective is called an isomorphism

isomorphism = change of notation

**Property:** A group homomorphism is injective iff  
 $\forall a \in G_1 \quad f(a) = 1 \implies a$  neutral in  $G_1$

# Group Constructions: Subgroups

**Subgroups:** given  $(G, \times)$ , and given  $H \subseteq G$  which is nonempty and stable by  $\times$  and inversion, consider  $(H, \times)$

Example:

- $5\mathbf{Z} = \{\dots, -15, -10, -5, 0, 5, 10, 15, \dots\}$  is a subgroup of  $\mathbf{Z}$

# Subgroups of $\mathbf{Z}$

## Theorem

If  $H$  is a subgroup of  $\mathbf{Z}$  not reduced to  $\{0\}$ , then  $H = n\mathbf{Z}$  where  $n$  is the smallest positive element of  $H$ .

## Proof.

- let  $a \in H$  and write  $a = qn + r$  with  $q, r \in \mathbf{Z}$  and  $0 \leq r < n$  (Euclidean division)
- since  $H$  is a group and  $a, n \in H$  we have  $r = a - qn \in H$
- since  $0 \leq r < n$  and  $n$  is the smallest positive element of  $H$  we must have  $r = 0$ , thus  $a = qn \in n\mathbf{Z}$
- therefore,  $H \subseteq n\mathbf{Z}$
- conversely,  $rn$  must be in  $H$  for all  $r \in \mathbf{Z}$ , therefore  $H = n\mathbf{Z}$  □

# Generators

- Given a group  $(G, \cdot)$ , an element  $g$  **generates/spans** a subgroup

$$\langle g \rangle = \{\dots, g^{-2}, g^{-1}, g^0, g^1, g^2, \dots\}$$

- If  $\langle g \rangle$  is finite, of cardinality  $n$ , then  $g^n = 1$  and

$$\langle g \rangle = \{g^0, g^1, \dots, g^{n-1}\}$$

(see next slide)

- if  $x \in \langle g \rangle$ ,  $\log_g x$  is uniquely determined up to some multiple of  $n$ :
  - $\log_g x$  is an element of  $\mathbf{Z}_n$
  - $i \mapsto g^i$  is a group isomorphism between  $\mathbf{Z}_n$  and  $\langle g \rangle$



# Finite Groups and Orders

## Definition

If  $(G, \cdot)$  is a group and if  $G$  is a finite set, then the cardinality of  $G$  is called the group **order**.

If  $g$  generates a subgroup of order  $n$ , then  $n$  is called the **order** of  $g$ .

## Proposition

The order of  $g$  is the smallest  $i > 0$  s.t.  $g^i = 1$ .

## Proof.

- the set of all  $i \in \mathbf{Z}$  such that  $g^i = 1$  is a subgroup of  $\mathbf{Z}$   
(preimage of subgroup  $\{1\}$  by group homomorphism  $i \mapsto g^i \dots$ )
- it must be of form  $n\mathbf{Z}$  where  $n$  is the smallest among all  $i > 0$
- $\{1, g, g^2, \dots, g^{n-1}\}$  is a non-repeating exhaustive list of all  $\langle g \rangle$  elements



# Consequence

if  $g$  is of order  $n$ ...

- then  $\langle g \rangle = \{1, g, g^2, \dots, g^{n-1}\}$
- $\forall i \quad g^i = 1 \iff n|i$
- $\forall i, j \quad g^i = g^j \iff i \equiv j \pmod{n}$

# Group Constructions: Product Groups

**Product groups:** given  $(G_1, \times_1)$  and  $(G_2, \times_2)$ , consider  $G = G_1 \times G_2$   
and  $(a_1, a_2) \times (b_1, b_2) = (a_1 \times_1 b_1, a_2 \times_2 b_2)$

**Power groups:** given  $(G, \cdot)$  and  $I$ , consider  $G^I$  and  
 $(a_i)_{i \in I} \times (b_i)_{i \in I} = (a_i \cdot b_i)_{i \in I}$

Example:

- $\mathbf{C}^* \times \{-1, +1\} = \{(z, s); z \in \mathbf{C}^*, s = \pm 1\}$  with  
 $(z, s) \times (z', s') = (zz', ss')$
- $\mathbf{Z}^{\{a,b,c\}}$  is the set of mappings from  $D = \{a, b, c\}$  to  $\mathbf{Z}$  with  $f + g$   
defined by  $(f + g)(x) = f(x) + g(x)$

# Functional vs Family Notations for Power Sets

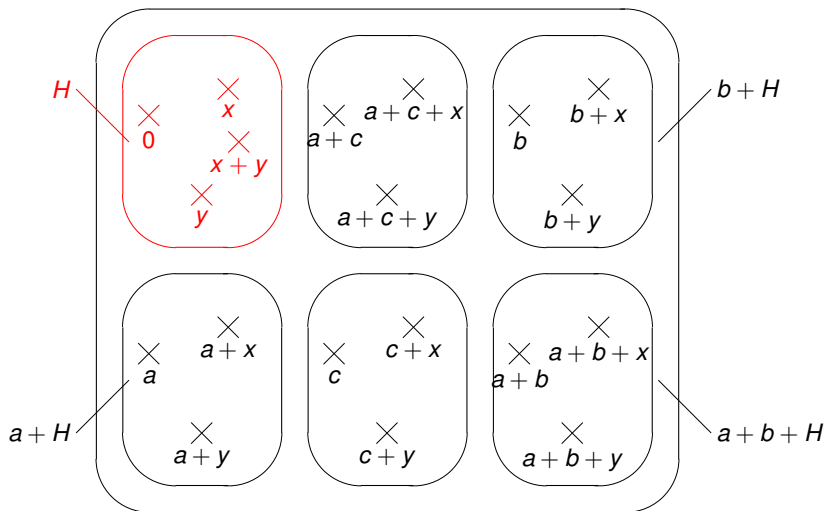
	functional notations	family notations
	function domain $D$	index set $I$
	function range $R$	set $S$
finite domain	$f : \{1, \dots, n\} \rightarrow R$	$(x_1, \dots, x_n)$
infinite domain	$f : D \rightarrow R$	$(x_i)_{i \in I}$
input	$x \in D$	$i \in I$
image	$f(x) \in R$	$x_i \in S$
set	$R^D$	$S^I$ or $S^n$

# Group Constructions: Quotient Groups

**Quotient groups:** given a commutative group  $G$  and a subgroup  $H$ , consider the set  $G/H$  of classes for congruence modulo  $H$  with the law induced by  $+$

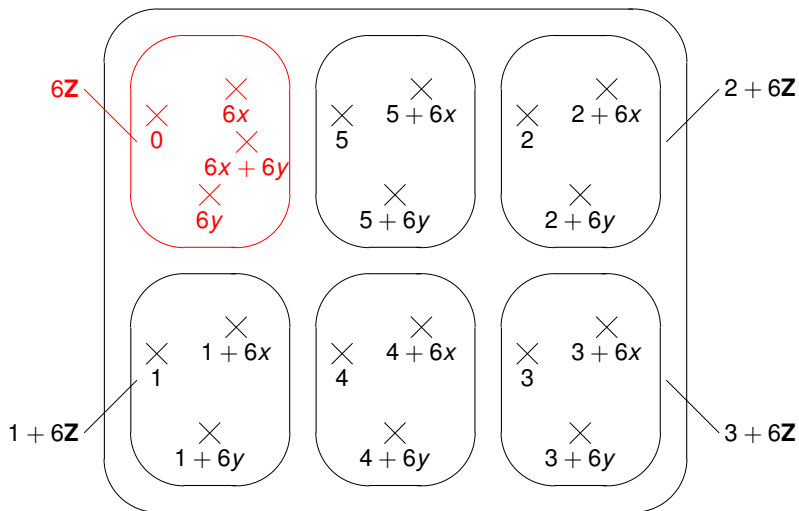
- $a$  and  $b$  in  $G$  are said to be *congruent modulo  $H$*  if  $b - a \in H$   
notation:  $a \equiv b \pmod{H}$
- the relation “...is congruent to ... modulo  $H$ ” is an equivalence relation (reflexive, symmetric, transitive)
- notation: for  $a \in G$ ,  $a + H$  is the set of all  $G$  elements which can be written  $a + h$  for some  $h \in H$  (elements congruent to  $a$ )
- every class of equivalence can be written  $a + H$  for some  $a \in G$   
 $a$  is called a *representative* for the class

# Quotient of an Abelian Group by a Subgroup



$$(a + H) + (b + H) = (a + b) + H$$

# Quotient Example: $\mathbb{Z}/6\mathbb{Z}$



$$\mathbb{Z}/H = \{H, 1 + H, 2 + H, 3 + H, 4 + H, 5 + H\}$$

# Lagrange Theorem

## Theorem (Lagrange)

*In any finite group, the order of any element is a factor of the order of the group.*

### Proof.

in  $G/\langle g \rangle$ , all  $a + \langle g \rangle$  have same number of elements so  $\#G$  (the order of  $G$ ) is divisible by  $\#\langle g \rangle$  (the order of  $g$ )  $\square$

## Consequence

$$\forall g \in G \quad g^{\#G} = 1$$



# Application: Generators in a Group of Prime Order

## Theorem

*if  $G$  has prime order, all elements (except 1) are generators*

### Proof.

- let  $p$  be the order of  $G$
- an element  $x \in G$  such that  $x \neq 1$  has an order  $n > 1$
- due to the Lagrange Theorem,  $n|p$ , so  $n = p$  since  $p$  is prime
- $g^0, \dots, g^{n-1}$  must be pairwise different, so  $n \leq p$
- so  $n = p$ :  $g$  must generate  $G$  □

# The Diffie-Hellman Key Agreement Protocol

Assume a group generated by some  $g$  ( $g$  is public)

**Alice**

**Bob**

pick  $x$  at random

$$X \leftarrow g^x$$

$$\xrightarrow{x}$$

$$\xleftarrow{y}$$

$$K \leftarrow Y^x$$

pick  $y$  at random

$$Y \leftarrow g^y$$

$$K \leftarrow X^y$$

$$(K = g^{xy})$$

security requirement: given  $(g, g^x, g^y)$ , it must be hard to compute  $g^{xy}$  (**Computational Diffie-Hellman Problem**)

# Using the Diffie-Hellman Key Agreement Protocol

- allows to set up a secret key over a public channel (assuming authentication)
- no further need to set up pre-shared keys: sets up keys when needed
  - public-key cryptography

Example of Diffie-Hellman groups:

- $\mathbf{Z}_p^*$  (compute  $g^x \bmod p$ )
- elliptic curves

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- **Algorithms for Big Numbers**
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem



# Addition in Binary

$$1 + 1 = 10$$

$$\begin{array}{r} \phantom{+} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \\ \phantom{+} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \\ \phantom{+} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \\ + \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \\ \hline = \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \\ \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \phantom{=} \end{array}$$

**Input:**  $a$  and  $b$ , two integers of at most  $\ell$  bits

**Output:**  $c$ , an integer of at most  $\ell+1$  bits representing  $a + b$

**Complexity:**  $\mathcal{O}(\ell)$

- 1:  $r \leftarrow 0$
- 2: **for**  $i = 0$  to  $\ell - 1$  **do**
- 3:    $d \leftarrow a_i + b_i + r$
- 4:   set  $c_i$  and  $r$  to bits such that  $d = 2r + c_i$
- 5: **end for**
- 6:  $c_\ell \leftarrow r$

# Addition (Binary/Hexadecimal/Decimal)

$$\begin{array}{rcccccccc} & & & 1 & 0 & 1 & 0 & 1 & 0 & 0 & & 0x54 & (84) \\ + & & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & 0x92 & (146) \\ \hline = & & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & & 0xe6 & (230) \end{array}$$

hexadecimal = compact way to represent bistrings  
(bits grouped into “nibbles” = packets of 4 bits)

# Definition of a Monoid

## Definition

A **monoid** is a set  $G$  together with a mapping from  $G \times G$  to  $G$  which maps  $(a, b)$  to an element denoted  $a + b$  and such that

1. [closure] for any  $a, b \in G$ , we have  $a + b \in G$
2. [associativity] for any  $a, b, c$ , we have  $(a + b) + c = a + (b + c)$
3. [neutral element] there exists an element  $0$  s.t. for any  $a$ ,  
 $a + 0 = 0 + a = a$

multiplication of a positive integer  $n$  by a monoid element  $a$ :

$$n.a = \underbrace{a + a + \cdots + a}_{n \text{ times}}$$



# Multiplication

we want to multiply a monoid element ( $a = 12$ ) by an integer ( $n = 100101$  in binary):

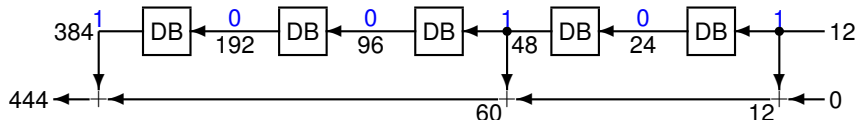
$$\begin{aligned} & 12 \times 100101 \\ = & 12 \times (1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1) \\ = & 12 \times (2^5 + 2^2 + 1) \\ = & 12 \times 2^5 + 12 \times 2^2 + 12 \times 1 \end{aligned}$$

multiplication by 2 consists of adding to itself  
(= a shift left for addition over the integers in binary)  
multiplication by  $2^i$  consists of multiplying  $i$  times by 2

# Multiplication Algorithm

$$12 \times 100101 = 444$$

				1	1	0	0			0x00c	(12)	
×			1	0	0	1	0	1		0x025	(37)	
						1	1	0	0	0x00c	(12)	
+						0	0	0	0	0x000	(0)	
+			1	1	0	0				0x030	(48)	
+			0	0	0	0				0x000	(0)	
+			0	0	0	0				0x000	(0)	
+	1	1	0	0						0x180	(384)	
=	1	1	0	1	1	1	1	0	0	0x1bc	(444)	



# Double-and-Add From Right to Left

**Input:**  $a$  in monoid,  $n$  integer of at most  $\ell$  bits  
( $n$  in binary)

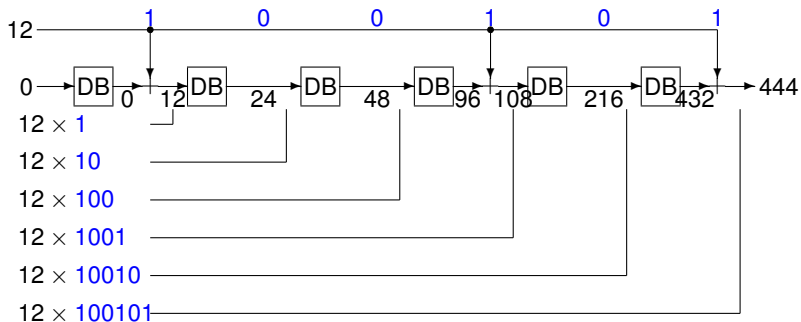
**Output:**  $c = a \times n$

**Complexity:**  $\mathcal{O}(\ell)$  monoid additions

- 1:  $x \leftarrow 0$
- 2:  $y \leftarrow a$
- 3: **for**  $i = 0$  to  $\ell - 1$  **do**
- 4:   **if**  $n_i = 1$  **then**
- 5:      $x \leftarrow x + y$
- 6:   **end if**
- 7:    $y \leftarrow y + y$
- 8: **end for**
- 9:  $c \leftarrow x$

# From Left to Right

$$12 \times 100101 = 444$$



# Double-and-Add From Left to Right

**Input:**  $a$  in monoid,  $n$  integer of at most  $\ell$  bits  
( $n$  in binary)

**Output:**  $c = a \times n$

**Complexity:**  $\mathcal{O}(\ell)$  monoid additions

```
1:  $x \leftarrow 0$ 
2: for  $i = \ell - 1$  to 0 do
3:    $x \leftarrow x + x$ 
4:   if  $n_i = 1$  then
5:      $x \leftarrow x + a$ 
6:   end if
7: end for
8:  $c \leftarrow x$ 
```

# From Double-and-Add to Square-and-Multiply

- if we can compute a monoid law  $a + b$  in  $\mathcal{O}(T)$  then we can compute  $n.a$  for  $n \in \mathbf{N}$  in  $\mathcal{O}(T \log n)$  instead of  $\mathcal{O}(Tn)$  by trivial algorithm

Example:

- monoid  $(\mathbf{Z}, +)$ : a positive integer multiplied by a  $\mathbf{Z}$  element
- monoid  $(\text{EC}, +)$ : an integer multiplied by a point
- monoid  $(\mathbf{Z}_m, \times)$ : a  $\mathbf{Z}_m$  element raised to some integral power

Same with multiplicative notation:

- if we can compute a monoid law  $ab$  in  $\mathcal{O}(T)$  then we can compute  $a^n$  for  $n \in \mathbf{N}$  in  $\mathcal{O}(T \log n)$

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# Definition of a Ring

## Definition

A **ring** is an Abelian group  $(R, +)$  together with a mapping from  $R \times R$  to  $R$  which maps  $(a, b)$  to an element denoted  $ab$  and such that

- 1-4. [group]  $R$  with  $+$  is a group
5. [Abelian] for any  $a, b$ , we have  $a + b = b + a$
6. [closure] for any  $a, b \in R$ , we have  $ab \in R$
7. [associativity] for any  $a, b, c$ , we have  $(ab)c = a(bc)$
8. [neutral element] there exists  $1$  s.t. for any  $a$ ,  $a1 = 1a = a$
9. [distributivity] for any  $a, b, c$ , we have  $a(b + c) = ab + ac$  and  $(a + b)c = ac + bc$

## Definition

A **commutative ring** is a ring  $R$  such that

- 1-9. [ring] it is a ring
10. [commutativity] for any  $a, b$  we have  $ab = ba$



# Group of Units

- not every element  $x$  in a ring  $R$  has an inverse for the multiplication
- we denote by  $R^*$  the set of elements having a multiplicative inverse  
those elements are called **units**
- $R^*$  with the multiplication is a group  
this is the **group of units** of the ring  $R$

common mistake:  ~~$R^* = R \setminus \{0\}$~~

# Group and Ring Constructors

- **sub-structure** (sub-group, ideal)  
**subgroup**: subset of a group stable by group law and inversion  
**ideal**: subgroup of a ring stable by multiplication by any ring element
- **spanned structure**  
set of all values generated by structure operations
- **product structure**  
set of pairs with inherited structure operations
- **power structure**  
set of tuples / set of functions of given domain with range in structure
- **quotient** (Abelian group by a subgroup, ring by an ideal)  
structure induced by grouping “equivalent” elements

## Example: $\mathbf{Z}$ with addition

$$\mathbf{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

1.  $\mathbf{Z}$  is closed for the addition
2. the addition is associative in  $\mathbf{Z}$
3. 0 is neutral for the addition
4. for any  $a \in \mathbf{Z}$  we have  $-a \in \mathbf{Z}$  which is the inverse of  $a$  for addition
5. the addition is commutative in  $\mathbf{Z}$
6.  $\mathbf{Z}$  is closed for the multiplication
7. the multiplication is associative in  $\mathbf{Z}$
8. 1 is neutral for the multiplication
9. addition is distributive for multiplication
10. the multiplication is commutative in  $\mathbf{Z}$

$\mathbf{Z}$  is a commutative ring of infinite size

## Example: $\mathbf{Z}[X]$

$\mathbf{Z}[X]$  = set of polynomials with coefficients in  $\mathbf{Z}$

example:  $(5X^3 - 3X^2 + X - 4) + (X^2 - 2X + 1) = 5X^3 - 2X^2 - X - 3$

1-5.  $\mathbf{Z}[X]$  with the addition is an Abelian group (isomorphic to  $\mathbf{Z}^{(\mathbf{N})}$ )

6.  $\mathbf{Z}[X]$  is closed under multiplication

7. multiplication is associative in  $\mathbf{Z}[X]$

8. the constant polynomial 1 is neutral for the multiplication

9. distributivity: we have

$$A(X)(B(X) + C(X)) = A(X)B(X) + A(X)C(X) \text{ for all } A(X), B(X), C(X) \in \mathbf{Z}[X]$$

10. multiplication is commutative in  $\mathbf{Z}[X]$

$\mathbf{Z}[X]$  is a commutative ring of infinite size

(same for  $R[X]$  for any commutative ring  $R$ )

# Example: Modulo 9 Reduction of Large Numbers

296 527 mod 9

$$\begin{aligned} &= (200\,000 + 90\,000 + 6\,000 + 500 + 20 + 7) \bmod 9 \\ &= (2 \times 100\,000 + 9 \times 10\,000 + 6 \times 1\,000 + 5 \times 100 + 2 \times 10 + 7) \bmod 9 \\ &= (2 \times 10^5 + 9 \times 10^4 + 6 \times 10^3 + 5 \times 10^2 + 2 \times 10 + 7) \bmod 9 \\ &= (2 \times (10 \bmod 9)^5 + 9 \times (10 \bmod 9)^4 + 6 \times (10 \bmod 9)^3 + \\ &\quad + 5 \times (10 \bmod 9)^2 + 2 \times (10 \bmod 9) + 7) \bmod 9 \\ &= (2 \times 1^5 + 9 \times 1^4 + 6 \times 1^3 + 5 \times 1^2 + 2 \times 1 + 7) \bmod 9 \\ &= (2 + 9 + 6 + 5 + 2 + 7) \bmod 9 \\ &= 31 \bmod 9 \\ &= (3 + 1) \bmod 9 \\ &= 4 \bmod 9 \\ &= 4 \end{aligned}$$



## Example: the Ring of Residues Modulo $n$

$$\mathbf{Z}_n = \{0, 1, 2, 3, \dots, n - 1\}$$

1.  $\mathbf{Z}_n$  is closed for the **addition modulo  $n$**
2. the **addition modulo  $n$**  is associative in  $\mathbf{Z}_n$  (next slides)
3. 0 is neutral for the addition
4. for any nonzero  $a \in \mathbf{Z}_n$  we have  $n - a \in \mathbf{Z}_n$  which is the inverse of  $a$  for **addition modulo  $n$**  (0 is self-inverse)
5. the **addition modulo  $n$**  is commutative in  $\mathbf{Z}_n$
6.  $\mathbf{Z}_n$  is closed for the **multiplication modulo  $n$**
7. the **multiplication modulo  $n$**  is associative in  $\mathbf{Z}_n$
8. 1 is neutral for the multiplication
9. addition modulo  $n$  is distributed over multiplication modulo  $n$  (next slides)
10. the **multiplication modulo  $n$**  is commutative in  $\mathbf{Z}_n$

$\mathbf{Z}_n$  is a commutative ring of  $n$  elements

# Cerebral $\mathbf{Z}_n$

- $n\mathbf{Z}$  is an ideal of  $\mathbf{Z}$  (with laws  $+$  and  $\times$ ) (ideal generated by  $n$ )
- we can do the quotient  $\mathbf{Z}/n\mathbf{Z}$  of  $\mathbf{Z}$  by  $n\mathbf{Z}$
- congruence modulo  $n\mathbf{Z}$  is written

$$a \equiv b \pmod{n} \iff a - b \in n\mathbf{Z} \iff a \bmod n = b \bmod n$$

- an exhaustive list of equivalence classes is

$$0 + n\mathbf{Z} , 1 + n\mathbf{Z} , 2 + n\mathbf{Z} , \dots , (n - 1) + n\mathbf{Z}$$

- note that  $(a + n\mathbf{Z}) + (b + n\mathbf{Z}) = ((a + b) \bmod n) + n\mathbf{Z}$
- note that  $(a + n\mathbf{Z}) \times (b + n\mathbf{Z}) = ((a \times b) \bmod n) + n\mathbf{Z}$
- we simply write  $a$  (the representative in  $[0, n - 1]$ ) instead of  $a + n\mathbf{Z}$



# $\mathbf{Z}_n$ Tips

- for any polynomial  $P(x) \in \mathbf{Z}[x]$  and any  $a, n \in \mathbf{Z}$  we have

$$P(a) \bmod n = P(a \bmod n) \bmod n$$

can put “ $\bmod \underline{n}$ ” reductions in the ground floor

- if  $x$  has order  $m$  in  $\mathbf{Z}_n^*$  then for any  $i \in \mathbf{Z}$

$$x^i \bmod n = x^{i \bmod m} \bmod n$$

can put “ $\bmod \underline{m}$ ” reductions in the upper floor

# Exercise

$\mathbf{Z}_{15}$  has order 15

- We have  $\langle 5 \rangle = \{0, 5, 10\}$ .  
This is a subgroup of order 3  
5 has order 3 in  $\mathbf{Z}_{15}$
- in  $\mathbf{Z}_{15}$ :  $\langle 2 \rangle = \{0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13\}$ .  
in  $\mathbf{Z}_{15}$ , 2 has order 15 (so, 2 is a generator)
- We have  $\langle 1 \rangle = \mathbf{Z}_{15}$   
1 is a generator
- $\mathbf{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$
- in  $\mathbf{Z}_{15}^*$ , 2 has the order 4:  $\langle 2 \rangle = \{1, 2, 4, 8\}$

# $\mathbb{Z}_n$ Computations

Efficiently computable operations:

- addition:  $(a + b) \bmod n$
- multiplication:  $(a \times b) \bmod n$  (double-and-add)
- modulo:  $a \bmod n$  (Euclidean division)
- inverse:  $a^{-1} \bmod n$  (when  $\gcd(a, n) = 1$ ) (extended Euclid algorithm)
- power:  $a^e \bmod n$  (for  $e$  integer only) (square-and-multiply)

Remaining problem: extracting roots:  $\sqrt[e]{a} \bmod n$  (or  $a^r \bmod n$  for  $r$  rational)

# Addition in $\mathbb{Z}_n$

**Input:** an integer  $n$  of  $\ell$  bits, two integers  $a$  and  $b$  less than  $n$

**Output:**  $c$ , an integer which represents  $a + b \bmod n$

**Complexity:**  $\mathcal{O}(\ell)$

- 1: add  $a$  and  $b$  in  $c$
- 2: compare  $c$  and  $n$
- 3: **if**  $c \geq n$  **then**
- 4:     subtract  $n$  from  $c$
- 5: **end if**

remark: comparison and subtraction take  $\mathcal{O}(\ell)$  time as well

# Multiplication in $\mathbf{Z}_n$ From Left to Right

**Input:** an integer  $n$  of  $\ell$  bits,  $a, b \in \mathbf{Z}_n$   
( $b$  in binary)

**Output:**  $c = a \times b \bmod n$

**Complexity:**  $\mathcal{O}(\ell^2)$

- 1:  $x \leftarrow 0$
- 2: **for**  $i = \ell - 1$  to 0 **do**
- 3:    $x \leftarrow x + x \bmod n$
- 4:   **if**  $b_i = 1$  **then**
- 5:      $x \leftarrow x + a \bmod n$
- 6:   **end if**
- 7: **end for**
- 8:  $c \leftarrow x$

# Exponentiation From Left to Right

## Square-and-Multiply

**Input:**  $a$  and  $n$ , two integers of at most  $\ell$  bits, an integer  $e$  ( $e$  in binary)

**Output:**  $x = a^e \bmod n$

**Complexity:**  $O(\ell^2 \log e)$

- 1:  $x \leftarrow 1$
- 2: **for**  $i = \log e - 1$  to 0 **do**
- 3:    $x \leftarrow x \times x \bmod n$
- 4:   **if**  $e_i = 1$  **then**
- 5:      $x \leftarrow x \times a \bmod n$
- 6:   **end if**
- 7: **end for**

# Euclidean Division

we can just adapt the algorithm we have learnt at school  
(not trivial to implement!)

- for any  $a \in \mathbf{Z}$  and  $n > 0$  there exists a unique pair  $(q, r) \in \mathbf{Z}^2$  such that  $a = qn + r$  and  $0 \leq r < n$   
 $q = \lfloor \frac{a}{n} \rfloor$  and  $r = a \bmod n$
- algorithm runs in  $\mathcal{O}(\ell^2)$

# Modular Inversion

## Theorem

$x \in \mathbf{Z}_n$  is invertible if and only if  $\gcd(x, n) = 1$ .

### Proof.

$\implies$  if  $\gcd(x, n) = d > 1$  then  $d$  divides  $(x \cdot y) \bmod n$  for any  $y$  so  $(x \cdot y) \bmod n \neq 1$  and  $x$  is non invertible.

$\longleftarrow$  to be seen later



# Euclid Algorithm

**Input:**  $a$  and  $b$ , two integers of at most  $\ell$  bits

**Output:**  $d = \gcd(a, b)$

**Complexity:**  $O(\ell^2)$

1:  $x \leftarrow a, y \leftarrow b$

2: **while**  $y > 0$  **do**

3:   make an Euclidean division  $x = qy + r$

4:   do simultaneously  $x \leftarrow y$  and  $y \leftarrow x - qy$

5: **end while**

6:  $d \leftarrow x$

## Example

We run the algorithm with  $a = 22$  and  $b = 35$ . We obtain the following sequence.

iteration	$x$	$y$	$q$
0	22	$-35 \times 0$	
		↙	
1	35	$-22 \times 1$	
		↙	
2	22	$-13 \times 1$	
		↙	
3	13	$-9 \times 1$	
		↙	
4	9	$-4 \times 2$	
		↙	
5	4	$-1 \times 4$	
		↙	
6	1	0	

Thus  $\gcd(22, 35) = 1$ .

# Why does it Work?

- it eventually stops ( $y$  strictly decreases and  $y \geq 0$ )
- a divisor of  $x$  and  $y$  is a divisor of  $x - qy$  for all  $q$
- $x = (x - qy) - (-q)y$
- $d$  divides  $x$  and  $y \iff d$  divides  $y$  and  $x - qy$
- for any  $q$ ,  $\gcd(x, y) = \gcd(y, x - qy)$
- $\gcd(x, 0) = x$
- conclusion: the algorithm terminates with  $\gcd(a, b)$
- to be discussed (in another course): running time (complexity) is quadratic

# Extended Euclid Algorithm

**Input:**  $a$  and  $b$ , two integers of at most  $\ell$  bits

**Output:**  $d, u, v$  such that  $d = au + bv = \gcd(a, b)$

**Complexity:**  $\mathcal{O}(\ell^2)$

1:  $\vec{x} \leftarrow (a, 1, 0), \vec{y} \leftarrow (b, 0, 1)$

2: **while**  $y_1 > 0$  **do**

3:   make an Euclidean division  $x_1 = qy_1 + r$

4:   do simultaneously  $\vec{x} \leftarrow \vec{y}$  and  $\vec{y} \leftarrow \vec{x} - q\vec{y}$

5: **end while**

6:  $(d, u, v) \leftarrow \vec{x}$

$$\vec{x}, \vec{y} \in \{(\alpha, \beta, \gamma); \alpha = a \cdot \beta + b \cdot \gamma\}$$

## Example

We run the algorithm with  $a = 22$  and  $b = 35$ . We obtain the following sequence of vectors.

iteration	$\vec{x}$		$\vec{y}$	$q$
0	$(22, 1, 0)$	–	$(35, 0, 1)$	$\times 0$
1	$(35, 0, 1)$	$\swarrow$	$(22, 1, 0)$	$\times 1$
2	$(22, 1, 0)$	$\swarrow$	$(13, -1, 1)$	$\times 1$
3	$(13, -1, 1)$	$\swarrow$	$(9, 2, -1)$	$\times 1$
4	$(9, 2, -1)$	$\swarrow$	$(4, -3, 2)$	$\times 2$
5	$(4, -3, 2)$	$\swarrow$	$(1, 8, -5)$	$\times 4$
6	$(1, 8, -5)$	$\swarrow$	$(0, -35, 22)$	

Thus  $1 = 22 \times 8 - 35 \times 5$ .

# Modular Inversion

to compute the inverse of  $x$  modulo  $n$ :

- 1 run the Extended Euclid algorithm with input  $(x, n)$  and get  $u, v$  such that  $ux + vn = d = \gcd(x, n)$
- 2 if  $d \neq 1$ ,  $x$  is not invertible: error!
- 3 output  $u$ : it is such that  $ux \bmod n = 1$

# Modular Inversion

## Theorem

$x \in \mathbf{Z}_n$  is invertible if and only if  $\gcd(x, n) = 1$ .

### Proof:

- $\Rightarrow$ : already seen ([slide 168](#))
- $\Leftarrow$ : if  $\gcd(x, n) = 1$ , run the Extended Euclid algorithm and get an equation  $ux + vn = 1$  then deduce  $ux \bmod n = 1$  □

Conclusion: the Extended Euclid algorithm is an inversion algorithm with complexity  $\mathcal{O}(\ell^2)$

# Arithmetics with Big Numbers

- addition ( $\mathcal{O}(\ell)$ ):  $x, y \mapsto x + y$
- multiplication ( $\mathcal{O}(\ell^2)$ ):  $x, y \mapsto x \times y$
- Euclidean division ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto x \bmod n$
  
- Euclid Algorithm ( $\mathcal{O}(\ell^2)$ ):  $x, y \mapsto u, v$  s.t.  $ux + vy = \gcd(x, y)$



# Modular Arithmetic

- addition ( $\mathcal{O}(\ell)$ ):  $x, y, n \mapsto (x + y) \bmod n$
- multiplication ( $\mathcal{O}(\ell^2)$ ):  $x, y, n \mapsto (x \times y) \bmod n$
- modulo ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto x \bmod n$
  
- fast exponential ( $\mathcal{O}(\ell^2 \log e)$ ):  $x, e, n \mapsto x^e \bmod n$
- inversion in  $\mathbf{Z}_n$  ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto y$  s.t.  $xy \bmod n = 1$  (when feasible)

# FFT-based Multiplication

- we could have better complexities with a better multiplication algorithm
- in this lecture, we limit to the values from the school-book algorithm
- in practice, this algorithm is sufficient for the lengths we use

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- **The  $\mathbf{Z}_p$  Field**
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# Definition of a Field

## Definition

A **field** is a commutative ring  $(K, +, \times)$  such that

**1-9.** [ring]  $K$  is a ring with  $+$  and  $\times$

**10.** [commutativity] for any  $a, b$ , we have  $ab = ba$

**11.** [invertibility] for any  $a \neq 0$  there exists  $b = a^{-1}$  s.t.  $ab = ba = 1$

example:

- **Q, R, C**
- $\mathbf{Z}_p$  for  $p$  prime (next slide)
- $\text{GF}(2^n)$  (in Chapter 4)

# $\mathbf{Z}_p$ Properties

## Theorem ( $\mathbf{Z}_p$ structure)

Let  $p$  be a prime number.

- 1  $\mathbf{Z}_p^* = \{1, \dots, p - 1\}$
- 2 (Little Fermat Theorem) for any  $x \in \mathbf{Z}_p^*$ , we have  $x^{p-1} \equiv 1 \pmod{p}$
- 3  $\mathbf{Z}_p^*$  is a cyclic group. So, there exist  $g$  such that

$$\mathbf{Z}_p^* = \{g^0, g^1, g^2 \pmod{p}, \dots, g^{p-2} \pmod{p}\}$$

# Proof

- 1 if  $1 \leq x \leq p - 1$ , since  $p$  is prime, we must have  $\gcd(x, p) = 1$   
thus  $x \in \mathbf{Z}_p^*$
- 2 due to the Lagrange Theorem, for any  $x \in \mathbf{Z}_p^*$ , we have  $x^{p-1} \equiv 1 \pmod{p}$
- 3 (hard)



# To Be Seen Later

- we can generate large prime numbers
- we can verify the primality of a number
- we can find generators in  $\mathbf{Z}_p^*$
- we can find  $(p, q, g)$  such that  $p$  and  $q$  are prime,  $q$  divides  $p - 1$ , and  $g$  has order  $q$  in  $\mathbf{Z}_p^*$

# The Discrete Logarithm Problem

(implicit: a parameter generator)

## Discrete Logarithm (DL) Problem

**Parameters:**  $G$ , a group,  $g \in G$  and  $n$ , the order of  $g$

**Instance:**  $y$ , power of  $g$

**Problem:** find  $x$  such that  $y = g^x$

Examples:

- $\mathbf{Z}_n$ : easy (use the Extended Euclid algorithm)
- $\mathbf{Z}_p^*$ : (maybe) hard
- over an elliptic curve: (maybe) hard



# Some Facts About The Discrete Logarithm Problem

in a group of order  $n$ :

- easy on a quantum computer:
  - Shor algorithm
- easy if  $n$  has only small prime factors:
  - Pohlig-Hellman algorithm
- best algorithm for a subgroup of  $\mathbf{Z}_p^*$  with  $n$  and  $p$  prime:
  - General Number Field Sieve (GNFS) with complexity

$$e^{\left(\sqrt[3]{\frac{64}{9}+o(1)}\right)(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}}$$

this is mostly precomputation (without  $y$ )

the computation from  $y$  takes  $e^{\left(\sqrt[3]{3+o(1)}\right)(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}}$

- generic algorithms in  $\mathcal{O}(\sqrt{n})$ :
  - baby-step giant-step algorithm
  - Pollard  $\rho$  algorithm

# Attacks based on Precomputation

over  $\mathbf{Z}_p^*$ , the discrete logarithm can be solved in

$\rho$ length (bits)	precomputation (core-time)	attack (core-time)
512	10.2 years	10 minutes
768	36 500 years	2 days
1 024	45 000 000 years	30 days

remember SSH2 uses a fixed  $p$  of 1 024 bits...

## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# The Diffie-Hellman Key Agreement Protocol (again)

Assume a group generated by some  $g$

**Alice**

**Bob**

pick  $x$  at random

$$X \leftarrow g^x$$

$$\xrightarrow{x}$$

$$\xleftarrow{y}$$

$$K \leftarrow Y^x$$

pick  $y$  at random

$$Y \leftarrow g^y$$

$$K \leftarrow X^y$$

$$(K = g^{xy})$$

security requirement: given  $(g, g^x, g^y)$ , it must be hard to compute  $g^{xy}$  (**Computational Diffie-Hellman Problem**)

# Passive Adversaries

- **passive adversary:** just listens to communications and tries to decrypt communications (e.g. by recovering the key)
- the Diffie-Hellman shall resist to passive attacks: given only  $g$ ,  $X$ , and  $Y$ , it must be hard to compute  $K$

# The Computational Diffie-Hellman Problem

(implicit: a parameter generator)

## Computational Diffie-Hellman (CDH) Problem

**Parameters:**  $G$ , a group,  $g \in G$  and  $n$ , the order of  $g$

**Instance:**  $X, Y \in \langle g \rangle$

**Problem:** find  $K = g^{xy}$  where  $X = g^x$  and  $Y = g^y$

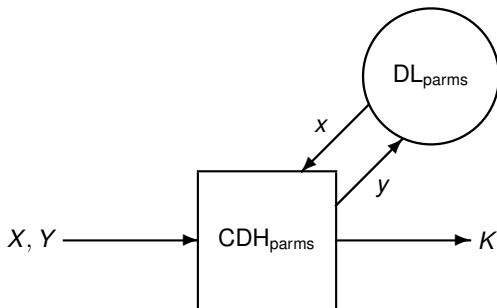
hardness requires the Discrete Logarithm Problem to be hard (see next slide)

Examples:

- a subgroup of  $\mathbf{Z}_p^*$  of prime order  $q$
- an elliptic curve

# DL $\implies$ CDH

## The CDH Problem Reduces to the DL Problem



parms  $\rightarrow (g, n)$

- set  $y = X$
- submit  $y$
- get  $x$
- compute  $K = Y^x$

# Problems with the Original Diffie-Hellman Protocol

- problems with subgroups of  $\langle g \rangle$ 
  - subgroup  $\{1\}$  (unavoidable): if either  $X$  or  $Y$  is 1, then  $K = 1$  for sure
  - other subgroups (avoidable): the discrete logarithm problem may become easy in subgroups
- problem with  $g^{xy}$  having a bad distribution (elements in  $\langle g \rangle$  may be sparse, so there is a structured information in  $g^{xy}$ )



# Correct Diffie-Hellman Key Exchange

Assume a group  $\langle g \rangle$  generated by some  $g$  of prime order  $q$

**Alice**

**Bob**

pick $x \in \mathbf{Z}_q^*$ , $X \leftarrow g^x$	$\xrightarrow{x}$	if $X \notin \langle g \rangle - \{1\}$ , abort
if $Y \notin \langle g \rangle - \{1\}$ , abort	$\xleftarrow{y}$	pick $y \in \mathbf{Z}_q^*$ , $Y \leftarrow g^y$
$K \leftarrow \text{KDF}(Y^x)$		$K \leftarrow \text{KDF}(X^y)$
	$(K = \text{KDF}(g^{xy}))$	

KDF: a Key Derivation Function

since  $\mathbf{Z}_q^*$  is cyclic,

- if Bob is honest, his  $X^y$  is uniformly distributed in  $\langle g \rangle - \{1\}$

# RFC 2631

## Diffie-Hellman Key Agreement Method

- group parameters  $(p, q, g)$  (meant to be defined by a CA):  $p$  prime,  $q$  prime,  $q$  divides  $p - 1$ ,  $g = h^{\frac{p-1}{q}} \bmod p$ ,  $h$  is random such that  $1 < h < p - 1$  and  $g > 1$
- secret keys:  $x_A, x_B$  between 1 and  $q - 1$
- public keys:  $y_A = g^{x_A} \bmod p$ ,  $y_B = g^{x_B} \bmod p$
- 3 modes:
  - ephemeral-ephemeral mode: both keys are fresh
  - ephemeral-static mode: recipient uses a static public key
  - static-static mode: both participants use a static public key
- shared secret:  $ZZ = g^{x_A x_B} \bmod p$   
(ZZ is the notation from the RFC, sorry!)

# Exercise

- group parameters  $(p, q, g)$  (meant to be defined by a CA):  $p$  prime,  $q$  prime,  $q$  divides  $p - 1$ ,  $g = h^{\frac{p-1}{q}} \bmod p$ ,  $h$  is random such that  $1 < h < p - 1$  and  $g > 1$

Show that  $g$  generates a subgroup of  $\mathbf{Z}_p^*$  of order  $q$ .

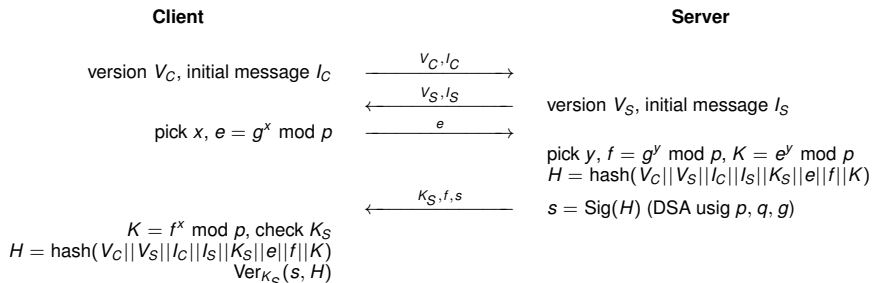
# Key Wrapping in RFC 2631

Objective: make a key transfer protocol based on Diffie-Hellman  
Alice wants to send a content-encryption key CEK to Bob

- keying material:  $KM = \text{SHA1}(ZZ \parallel \text{OtherInfo})$   
OtherInfo includes algorithm, counter (we can generate many KM blocks from the same ZZ), some ad-hoc string and the length of the KEK to generate
- key-encryption key:  $\text{KEK} = \text{trunc}(KM_1 \parallel KM_2 \parallel \dots)$
- to transfer CEK: send  $\text{Enc}_{\text{KEK}}(\text{CEK})$  (**key wrap**)

# Example: Semi-Authenticated Key Exchange in SSH2

- $I_C$  and  $I_S$ : negotiation of crypto algorithms
- $K_S$ : public key of the server (may come with a certificate)
- for diffie-hellman-group1-sha1 key exchange:  
 $p = 2^{1024} - 2^{960} - 1 + 2^{64} \lfloor 2^{894} \pi + 129093 \rfloor$ ,  $g = 2$ ,  $q = \frac{p-1}{2}$



# Parameter Validation in RFC 2631

- $p$  and  $q$  are prime,  $g^q \bmod p = 1$
- group parameters validation:  $q$  divides  $p - 1$ , and (optional)  $p$  and  $q$  follow parameter generation algorithm from seed and counter
- public key validation:  $2 \leq y \leq p - 1$ ,  $y^q \bmod p = 1$

# An Interesting Result

$\langle g \rangle$  is the *unique* subgroup of  $\mathbf{Z}_p^*$  of order  $q$ )

## Theorem

Let  $p, q, g$  be integers such that  $p$  and  $q$  are prime,  $q$  divides  $p - 1$ ,  $g \bmod p \neq 1$ , and  $g^q \bmod p = 1$ . Then

- $\langle g \rangle$  is a subgroup of  $\mathbf{Z}_p^*$  of order  $q$
- $\langle g \rangle = \{y \in \mathbf{Z}_p^*; y^q \bmod p = 1\}$

Application to RFC 2631: we can check that  $y$  is in the group generated by  $g$  by checking  $y^q \bmod p = 1$

# Proof

- $\langle g \rangle$  is a subgroup of  $\mathbf{Z}_p^*$  of order  $q$ : clear
- $\langle g \rangle \subseteq \{y \in \mathbf{Z}_p^*; y^q \bmod p = 1\}$ : clear
- $\langle g \rangle \supseteq \{y \in \mathbf{Z}_p^*; y^q \bmod p = 1\}$ :  
let  $y \in \mathbf{Z}_p^*$  be such that  $y^q \bmod p = 1$

- let  $\theta \in \mathbf{Z}_p^*$  be a generator of  $\mathbf{Z}_p^*$ , write  $g = \theta^a \bmod p$ ,  $y = \theta^b \bmod p$
- since  $g^q \equiv y^q \equiv 1 \pmod{p}$ , we have  $qa \equiv qb \equiv 0 \pmod{p-1}$
- so, we can write  $a = \frac{p-1}{q} a'$  and  $b = \frac{p-1}{q} b'$  with  $a', b' \leq q$
- since  $g \bmod p \neq 1$ , we have  $1 \leq a' < q$
- since  $q$  is prime, there exists  $c$  such that  $a'c \bmod q = 1$
- we have

$$g^{b'c} \equiv \theta^{ab'c} \equiv \theta^{a'bc} \equiv y^{a'c} \equiv y^{1+kq} \equiv y \pmod{p}$$

so,  $y \in \langle g \rangle$

□



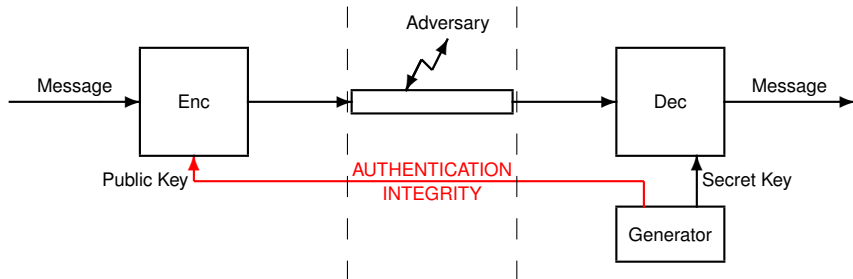
# Group Parameter Generation in RFC 2631

- 1:  $m =$  required length for  $q$ ,  $m' = \lceil \frac{m}{160} \rceil$
- 2: **repeat**
- 3: pick a random seed
- 4:  $U = \sum_{i=0}^{m'-1} 2^{160i} (\text{SHA1}(\text{seed} + i) \oplus \text{SHA1}(\text{seed} + m' + i))$
- 5:  $q = U \text{ OR } 1 \text{ OR } 2^m$
- 6: **until**  $q$  is prime
- 7:  $L =$  required length for  $p$ ,  $L' = \lceil \frac{L}{160} \rceil$
- 8: counter = 0
- 9: **repeat**
- 10:  $R = \text{seed} + 2m' + (L' * \text{counter})$
- 11:  $W = \left( \sum_{i=0}^{L'} 2^{160i} \text{SHA1}(R + i) \right) \bmod 2^L$
- 12:  $X = W \text{ OR } 2^{L-1}$
- 13:  $p = X - (X \bmod (2q)) + 1$
- 14: counter  $\leftarrow$  counter + 1
- 15: if counter  $\geq 4096N$  then abort (fail)
- 16: **until**  $p > 2^{L-1}$  and  $p$  is prime
- 17: output  $p$ ,  $q$ , seed, counter

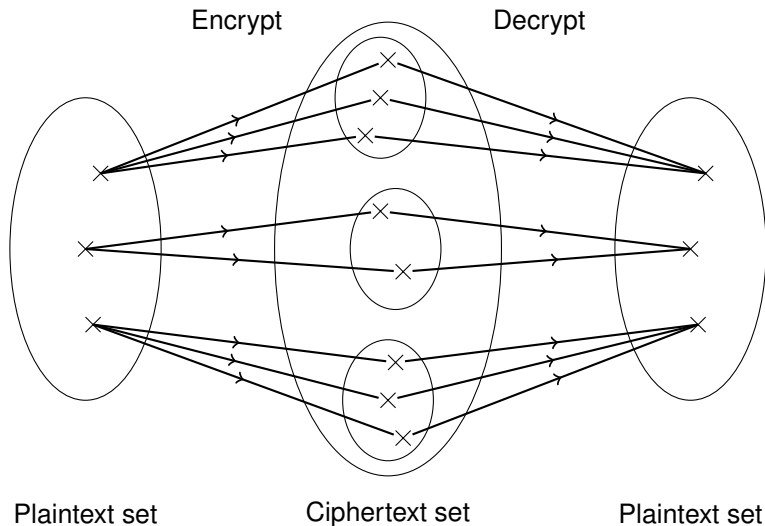
## Diffie-Hellman Cryptography

- Arithmetics and  $\mathbf{Z}_n$
- Some Notions of Groups Theory
- Algorithms for Big Numbers
- $\mathbf{Z}_n$ : The Ring of Residues Modulo  $n$
- The  $\mathbf{Z}_p$  Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

# Public-Key Cryptosystem



# Non-Deterministic Encryption



# Semi-Static-DH to Public-Key Encryption

## Towards ElGamal Encryption

**Alice**  
input:  $m$

**Bob**  
secret key:  $y$   
public key:  $Y = g^y$

$\longleftarrow Y$

pick  $x$  at random

$$X = g^x$$

$$K = \text{KDF}(Y^x)$$

$$c = \text{symEnc}_K(m)$$

$\xrightarrow{X}$

$$K = \text{KDF}(X^y)$$

$\xrightarrow{c}$

$$m = \text{symDec}_K(c)$$

**output:  $m$**

# The Plain ElGamal Encryption Case

- no KDF
- symEnc is one-time-pad, adapted in the DH group

# ElGamal Cryptosystem

**Public parameters:**  $(g, n)$ , a group  $\langle g \rangle$  of order  $n$  generated by some  $g$

**Set up:** generate a random  $x \in \mathbf{Z}_n$ , and compute  $y = g^x$

**Message:** an element  $m \in \langle g \rangle$

**Public key:**  $K_p = y$

**Secret key:**  $K_s = x$

**Encryption:** pick a random  $r \in \mathbf{Z}_n$ , compute  $u = g^r$ , and  $v = my^r$   
The ciphertext is  $(u, v)$

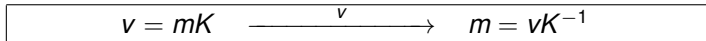
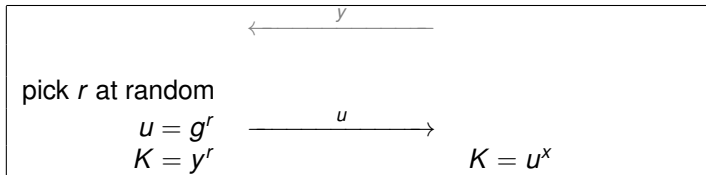
**Decryption:** extract the  $u$  and  $v$  parts of the ciphertext and compute  
 $m = vu^{-x}$

# ElGamal Cryptosystem

Semi-Static DH + Vernam Generalized

**Alice**  
input:  $m$

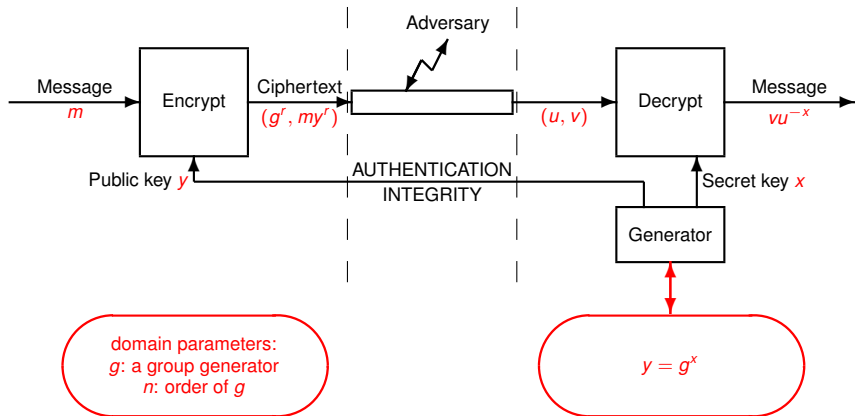
**Bob**  
secret key:  $x$   
public key:  $y = g^x$



**output:  $m$**



# Plain ElGamal Encryption



(assume  $m \in \langle g \rangle$ )

# ElGamal Encryption Complexity

in subgroups of  $\mathbf{Z}_p^*$  with  $p$  of length  $\ell$ :

- Domain parameter selection:  $\mathcal{O}(\ell^4)$   
(prime numbers generation to be seen in next chapter)
- Generator:  $\mathcal{O}(\ell^3)$
- Encryption:  $\mathcal{O}(\ell^3)$
- Decryption:  $\mathcal{O}(\ell^3)$

# EIGamal Security: EIGamal Problems

(implicit: a parameter generator)

## EIGamal Decryption (EGD) Problem

**Parameters:**  $G$ , a group,  $g \in G$  and  $n$ , the order of  $g$

**Input:**  $(y, u, v)$  such that  $y, u, v \in \langle g \rangle$ .

**Problem:** compute  $m$  such that there exists  $r$  such that  $u = g^r$  and  $v = my^r$ .

## EIGamal Key Recovery (EGKR) Problem

**Parameters:**  $G$ , a group,  $g \in G$  and  $n$ , the order of  $g$

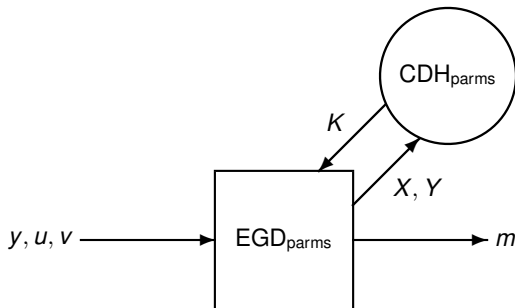
**Input:**  $y$  such that  $y \in \langle g \rangle$ .

**Problem:** compute  $x$  such that  $y = g^x$ .

decryption problem       $\iff$       Diffie-Hellman problem  
key recovery problem      =      discrete logarithm problem

# CDH $\implies$ EGD

## The EGD Problem Reduces to the CDH Problem

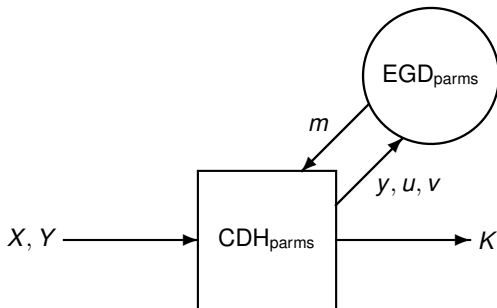


parms  $\rightarrow (g, n)$

- set  $X = u$  and  $Y = y$
- submit  $(X, Y)$
- get  $K$  (this should be  $g^{xr}$ )
- compute  $m = v/K$

# EGD $\implies$ CDH

The CDH Problem Reduces to the EGD Problem



parms  $\rightarrow (g, n)$

- set  $u = X, y = Y$ , pick a random  $v \in \langle g \rangle$
- submit  $(y, u, v)$
- get  $m$
- compute  $K = v/m$

# ElGamal Encryption Security

- key recovery is equivalent to the discrete logarithm problem
- decryption is equivalent to the Diffie-Hellman problem
- some tricky things about the selection of groups  
(left for another course)

# Conclusion

- **$\mathbf{Z}_n$  ring,  $\mathbf{Z}_p$  field**: a nice playground for cryptography
- **algorithmic number theory**: easy to add multiply, invert, compute exponentials in  $\mathbf{Z}_n$  and  $\mathbf{Z}_p$
- **DL and CDH problems**: some cryptosystems based on their hardness
- **Diffie-Hellman key exchange**: can set up a symmetric key over a public channel, resist to passive adversaries
- **ElGamal encryption**: an example of probabilistic cryptosystem

# References

- **Shoup**. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press. 2005.  
<http://shoup.net/ntb>  
Textbook on algebra for cryptographers and applications.
- **Menezes-van Oorschot-Vanstone**. *Handbook of Applied Cryptography*. CRC. 1997.  
<http://www.cacr.math.uwaterloo.ca/hac/>  
Reference book
- **Vaudenay**. *A Classical Introduction to Cryptography — Applications for Communications Security*. Springer. 2005.  
<http://www.vaudenay.ch/crypto/>  
Textbook on cryptography
- **Diffie-Hellman**. New Directions in Cryptography. *IEEE Transactions on Information Theory* vol. 22, 1976.



# Must be Known

- **groups, rings, fields:**
  - orders
  - Lagrange Theorem
- **$Z_n$  ring:** invertibility
- **$Z_p$  field:** the multiplicative group is cyclic
- **algorithmic number theory:**
  - square-and-multiply
  - extended Euclid algorithm
- **Diffie-Hellman key exchange:**
  - resist to passive adversaries
  - man-in-the-middle active adversary
  - ephemeral or static mode
  - better on a group of prime order
  - requires the hardness of DL
- **ElGamal encryption:**
  - requires the hardness of CDH
  - encrypt group elements
  - better on a group of prime order

















- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography**
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

# Roadmap

- more on number theory
- prime number generation
- RSA cryptosystem
- square roots
- factoring problem

## RSA Cryptography

- Euler and Other Chinese
- Orders in a Group
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

# Euler Totient Function



$\varphi(n)$  is the order of  $\mathbf{Z}_n^*$

## Theorem

Given an integer  $n$ , we have the following results.

- For all  $x \in \mathbf{Z}_n$  we have  $x \in \mathbf{Z}_n^* \iff \gcd(x, n) = 1$ .
- $\mathbf{Z}_n$  is a field  $\iff \mathbf{Z}_n^* = \mathbf{Z}_n \setminus \{0\} \iff \varphi(n) = n - 1 \iff n$  is prime
- For all  $x \in \mathbf{Z}_n^*$  we have  $x^{\varphi(n)} \equiv 1 \pmod{n}$ .
- if  $e$  is such that  $\gcd(e, \varphi(n)) = 1$ , we let  $d = e^{-1} \pmod{\varphi(n)}$ . For all  $x \in \mathbf{Z}_n^*$ ,  $x^d \pmod{n}$  is the only  $e$ th root of  $x$  modulo  $n$

# Proof — i

For all  $x \in \mathbf{Z}_n$  we have  $x \in \mathbf{Z}_n^* \iff \gcd(x, n) = 1$ .

## Proof.

$\implies$ : if  $\gcd(x, n) = d > 1$ , then  $d$  divides  $(x \cdot y) \bmod n$  for any  $y$  so  $(xy) \bmod n$  cannot be equal to 1.

$\impliedby$ : if  $\gcd(x, n) = 1$ , the extended Euclid algorithm constructs an inverse of  $x$  (see [slide 175](#)) □

## Proof — ii

$\mathbf{Z}_n$  is a field  $\iff \mathbf{Z}_n^* = \mathbf{Z}_n \setminus \{0\} \iff \varphi(n) = n - 1 \iff n$  is prime

**Proof.** By definition,  $\mathbf{Z}_n$  is a field  $\iff \mathbf{Z}_n^* = \mathbf{Z}_n \setminus \{0\}$ .

Since  $\mathbf{Z}_n^* \subseteq \mathbf{Z}_n \setminus \{0\}$ ,  $\mathbf{Z}_n^*$  and  $\mathbf{Z}_n \setminus \{0\}$  are equal iff they have the same cardinality.

We have  $\#\mathbf{Z}_n^* = \varphi(n)$  and  $\#\mathbf{Z}_n \setminus \{0\} = n - 1$ , so we deduce  $\mathbf{Z}_n^* = \mathbf{Z}_n \setminus \{0\} \iff \varphi(n) = n - 1$ .

$$\begin{aligned} \mathbf{Z}_n^* = \mathbf{Z}_n \setminus \{0\} &\iff \forall x \in \{1, \dots, n - 1\} \quad \gcd(x, n) = 1 \\ &\iff n \text{ is prime} \end{aligned}$$

( $\mathbf{Z}_n$  field  $\iff n$  prime was seen on [slide 181](#))



## Proof — iii

For all  $x \in \mathbf{Z}_n^*$  we have  $x^{\varphi(n)} \equiv 1 \pmod{n}$ .

**Proof.** Due to the Lagrange Theorem, the order  $k$  of  $x$  divides the order  $\varphi(n)$  of  $\mathbf{Z}_n^*$ .

Let  $\varphi(n) = k \cdot r$ . We have  $x^{\varphi(n)} \equiv x^{k \cdot r} \equiv (x^k)^r \equiv 1^r \equiv 1$ . □

## Proof — iv

If  $e$  is such that  $\gcd(e, \varphi(n)) = 1$ , we let  $d = e^{-1} \bmod \varphi(n)$ . For all  $x \in \mathbf{Z}_n^*$ ,  $x^d \bmod n$  is the only  $e$ th root of  $x$  modulo  $n$

**Proof.** We have  $e \cdot d = 1 + k \cdot \varphi(n)$  for some  $k$ .

$y \equiv x^d \implies y^e \equiv x^{1+k \cdot \varphi(n)} \equiv x$  so  $y = x^d$  is a  $e$ th root of  $x$ .

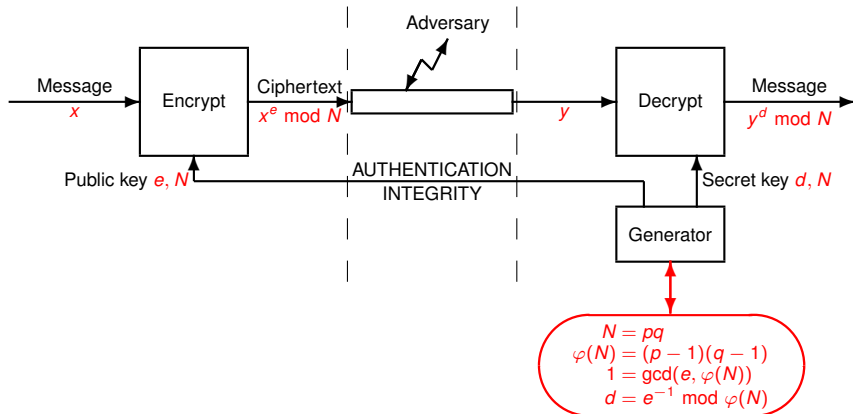
If  $x \equiv y^e$ , we have  $y \in \mathbf{Z}_n^*$  because

$$(x^{-1}y^{e-1})y \equiv 1$$

we have  $x \equiv y^e \implies x^d \equiv y^{1+k \cdot \varphi(n)} \equiv y$  so a  $e$ th root of  $x$  must be unique. □



# Application: RSA Cryptosystem



# Chinese Remainder Theorem

## Theorem (Chinese Remainder Theorem)

Let  $m$  and  $n$  be two integers such that  $\gcd(m, n) = 1$ . For any  $a, b \in \mathbf{Z}$ , there exists  $x \in \mathbf{Z}$  such that

$$x \equiv a \pmod{m}$$

$$x \equiv b \pmod{n}$$

Furthermore, for all such solution,  $x \bmod (mn)$  is unique.

Example: ( $m = 5$ ,  $n = 7$ ,  $mn = 35$ ,  $a = 3$ ,  $b = 4$ )

We find that  $x = 18$  is a solution and for all solution,  $x \bmod (mn) = 18$

# Chinese Remainder Theorem

## Theorem (Chinese Remainder Theorem)

Let  $m$  and  $n$  be two integers such that  $\gcd(m, n) = 1$ . We have

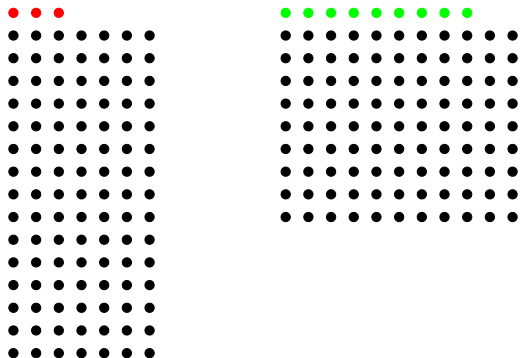
- $f : \mathbf{Z}_{mn} \rightarrow \mathbf{Z}_m \times \mathbf{Z}_n$  defined by  $f(x) = (x \bmod m, x \bmod n)$  is a ring isomorphism
- $f^{-1}(a, b) \equiv an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \pmod{mn}$

Example: ( $m = 5, n = 7, mn = 35$ )

$$\begin{aligned} f^{-1}(3, 4) &= (3 \times 7 \times (7^{-1} \bmod 5) + 4 \times 5 \times (5^{-1} \bmod 7)) \bmod 35 \\ &= \dots = 18 \end{aligned}$$

Application:  $\varphi(pq) = (p - 1)(q - 1)$  when  $p$  and  $q$  are two different primes

# Application 1: Count Soldiers



$$\begin{aligned}x &\equiv 3 \cdot 11 \cdot (11^{-1} \bmod 7) + 9 \cdot 7 \cdot (7^{-1} \bmod 11) \pmod{77} \\ &\equiv 3 \times 22 + 9 \times 56 \pmod{77} \\ &\equiv 31 \pmod{77}\end{aligned}$$

... there must be 108 soldiers

## Application 2: Equality Modulo Composite Numbers

### Theorem

For any  $a, b, m, n \in \mathbf{Z}$  such that  $\gcd(m, n) = 1$ , then

$$\left. \begin{array}{l} a \equiv b \pmod{m} \\ a \equiv b \pmod{n} \end{array} \right\} \iff a \equiv b \pmod{mn}.$$

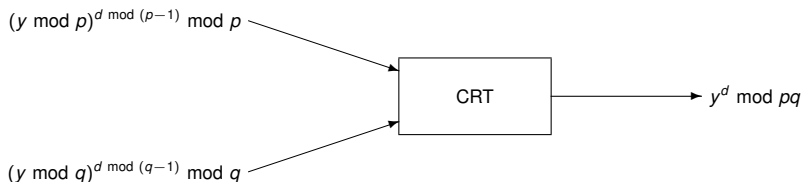
Indeed,  $f(a \bmod (mn)) = f(b \bmod (mn))$  hence  
 $a \bmod (mn) = b \bmod (mn)$

## Application 3: Correctness of RSA

- let  $N = pq$  be the product of two different prime numbers  $p$  and  $q$
- for any  $x \in \mathbf{Z}$  such that  $x \bmod p \neq 0$  we have  
 $(x^e \bmod N)^d \bmod N \equiv x \pmod{p}$   
(comes from  $p - 1$  divides  $\varphi(N)$  thus  $ed \bmod (p - 1) = 1$ )
- this also holds when  $x \bmod p = 0$
- similarly: for any  $x \in \mathbf{Z}$  we have  $(x^e \bmod N)^d \bmod N \equiv x \pmod{q}$
- from CRT (Application 2): for any  $x \in \mathbf{Z}$  we have  
 $(x^e \bmod N)^d \bmod N \equiv x \pmod{N}$
- for any  $x \in \mathbf{Z}_N$  we have  $(x^e \bmod N)^d \bmod N = x$

# Application 4: Exponentiation Acceleration

$$\log_2 p \approx \log_2 q \approx \frac{\ell}{2}$$



$$2 \times \mathcal{O}\left(\left(\frac{\ell}{2}\right)^3\right)$$

$$\mathcal{O}(\ell^3)$$

# Proof of CRT — i

**Fact 1:**  $f$  is a ring homomorphism from  $\mathbf{Z}_{mn}$  to  $\mathbf{Z}_m \times \mathbf{Z}_n$

- $f(x +_{\mathbf{Z}_{mn}} y) = f(x) +_{\mathbf{Z}_m \times \mathbf{Z}_n} f(y)$

indeed:

$$((x + y) \bmod (mn)) \bmod m = ((x \bmod m) + (y \bmod m)) \bmod m$$

$$((x + y) \bmod (mn)) \bmod n = ((x \bmod n) + (y \bmod n)) \bmod n$$

- $f(x \times_{\mathbf{Z}_{mn}} y) = f(x) \times_{\mathbf{Z}_m \times \mathbf{Z}_n} f(y)$

(same)

- $f(1) = (1, 1)$



## Proof of CRT — ii

**Fact 2:**  $f$  is an isomorphism

- $f(x) = (0, 0)$  implies  $m$  and  $n$  divide  $x$   
since  $\gcd(m, n) = 1$ ,  $mn$  divides  $x$  (see next slide)  
thus  $x \bmod (mn) = 0$
- $f$  is injective: for all  $x, y \in \mathbf{Z}_{mn}$ , if  $f(x) = f(y)$  then  
 $f(x - y) = (0, 0)$  thus  $x - y \bmod (mn) = 0$  hence  $x = y$
- $f$  is an isomorphism:  $\mathbf{Z}_{mn}$  and  $\mathbf{Z}_m \times \mathbf{Z}_n$  have the same cardinality  
and  $f$  is injective thus  $f$  is a bijection  
since  $f$  is further a homomorphism,  $f$  is an isomorphism

# Euclid Lemma



## Lemma

*If  $p$  is prime and  $p$  divides  $ab$ , then  $p$  divides  $a$  or  $p$  divides  $b$ .*

(Proof with a big hammer:  $\mathbf{Z}_p$  is a field!)

## Lemma (Generalization)

*If  $n$  divides  $ab$  and  $\gcd(n, a) = 1$ , then  $n$  divides  $b$ .*

Consequence: if  $n$  and  $a$  divide  $x$  and  $\gcd(n, a) = 1$ , then  $na$  divides  $x$ .

(take  $b = x/a$ )

## Proof of CRT — iii

**Fact 3:**  $f(an(n^{-1} \bmod m) + bm(m^{-1} \bmod n)) = (a, b)$

$$an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \equiv a \pmod{m}$$

$$an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \equiv b \pmod{n}$$

thus  $f$  of the left hand side is  $(a, b)$

# CRT Backward: Another Approach

## Theorem (CRT Backward)

Let  $m$  and  $n$  be two integers such that  $\gcd(m, n) = 1$ . Let  $u = n(n^{-1} \bmod m)$  and  $v = m(m^{-1} \bmod n)$ . The function

$$g : \mathbf{Z}_m \times \mathbf{Z}_n \longrightarrow \mathbf{Z}_{mn} \\ (a, b) \longmapsto au + bv \bmod (mn)$$

is well defined and is a ring isomorphism.

Note:  $g$  is well defined because

$$g : \mathbf{Z} \times \mathbf{Z} \longrightarrow \mathbf{Z}_{mn} \\ (a, b) \longmapsto (a + im)u + (b + jn)v \bmod (mn)$$

does not depend on  $i$  or  $j$

Remark:  $(u + v) \bmod (mn) = g(1, 1) = 1$

# Proof

$$\begin{aligned} g : \mathbf{Z}_m \times \mathbf{Z}_n &\longrightarrow \mathbf{Z}_{mn} \\ (a, b) &\longmapsto au + bv \pmod{mn} \end{aligned}$$

## Proof.

- $g(a, b) + g(a', b') \equiv g(a + a', b + b') \pmod{mn}$  so  $g$  is a group homomorphism
- $g(a, b) = 0$  implies  $a \pmod{m} = 0$  and  $b \pmod{n} = 0$  so  $g$  is injective
- due to cardinality,  $g$  is bijective: so, a group isomorphism
- $g^{-1}(x) = (x \pmod{m}, x \pmod{n})$  is homomorphic for  $\times$  so we have a ring isomorphism □

# Euler Totient Function

## Corollary

*Let  $m$  and  $n$  be two integers such that  $\gcd(m, n) = 1$ . We have  $\varphi(mn) = \varphi(m)\varphi(n)$ .*

# Proof

**Fact:**  $f$  is a bijection from  $\mathbf{Z}_{mn}^*$  to  $\mathbf{Z}_m^* \times \mathbf{Z}_n^*$  (thus  $\varphi(mn) = \varphi(m)\varphi(n)$ ):

- if  $x \in \mathbf{Z}_{mn}^*$  then  $f(x).f(x^{-1}) = f(1) = (1, 1)$  so both components of  $f(x)$  are invertible:  $f(x) \in \mathbf{Z}_m^* \times \mathbf{Z}_n^*$
- conversely, if  $(a, b) \in \mathbf{Z}_m^* \times \mathbf{Z}_n^*$ , let  $x = f^{-1}(a, b)$  and  $y = f^{-1}(a^{-1}, b^{-1})$   
we have  $f(xy) = f(x).f(y) = (a, b).(a^{-1}, b^{-1}) = (1, 1) = f(1)$  so  $xy = 1$  so  $x \in \mathbf{Z}_{mn}^*$
- $f$  is a bijection from  $\mathbf{Z}_{mn}$  to  $\mathbf{Z}_m \times \mathbf{Z}_n$ , so a bijection from  $\mathbf{Z}_{mn}^*$  to  $\mathbf{Z}_m^* \times \mathbf{Z}_n^*$

actually,  $\mathbf{Z}_{mn}^*$  and  $\mathbf{Z}_m^* \times \mathbf{Z}_n^*$  are isomorphic groups (and  $f$  is such isomorphism)

# Computation of Euler Totient Function

- $\varphi(p) = p - 1$  for  $p$  prime
- $\varphi(mn) = \varphi(m) \times \varphi(n)$  when  $\gcd(m, n) = 1$
- $\varphi(p^a) = (p - 1)p^{a-1}$  for  $p$  prime

$$\begin{aligned}\varphi(p_1^{a_1} \times \cdots \times p_r^{a_r}) &= (p_1 - 1)p_1^{a_1-1} \times \cdots \times (p_r - 1)p_r^{a_r-1} \\ &= p_1^{a_1} \times \cdots \times p_r^{a_r} \frac{(p_1 - 1) \times \cdots \times (p_r - 1)}{p_1 \times \cdots \times p_r}\end{aligned}$$

for pairwise different prime numbers  $p_1, \dots, p_r$



## RSA Cryptography

- Euler and Other Chinese
- Orders in a Group
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

# Structure Property of $\mathbf{Z}$ (Reminder)

(already seen, see [slide 127](#))

## Theorem

*For all proper subgroup  $I$  of  $\mathbf{Z}$  there exists  $n$  such that*

$$I = n\mathbf{Z} = \{\dots, -3n, -2n, -n, 0, n, 2n, 3n, \dots\}$$

# Element Order

Given  $x$  in a group  $G$ :

- $\{i \in \mathbf{Z}; x^i = 1\}$  is a subgroup of  $\mathbf{Z}$
- so,  $\{i \in \mathbf{Z}; x^i = 1\} = n\mathbf{Z}$  for some  $n$  which is the smallest positive  $n$  such that  $x^n = 1$   
 $n$  is called the **order** of  $x$  in  $G$ .
- $n$  is such that

$$x^i = 1 \iff (n \text{ divides } i)$$

see [slide 128](#)

# Group Exponent

Given a group  $G$ :

- $\{i \in \mathbf{Z}; \forall x \in G x^i = 1\}$  is a subgroup of  $\mathbf{Z}$
- so,  $\{i \in \mathbf{Z}; \forall x \in G x^i = 1\} = \lambda\mathbf{Z}$  for some  $\lambda$  which is the smallest positive  $\lambda$  such that  $\forall x \in G, x^\lambda = 1$   
 $\lambda$  is called the **exponent** of  $G$ .
- $\lambda$  is such that

$$\left(\forall x \in G x^i = 1\right) \iff (\lambda \text{ divides } i)$$

- note that for all  $x$ ,  $\lambda \in \{i \in \mathbf{Z}; x^i = 1\} = n\mathbf{Z}$  so  $\lambda$  is a multiple of  $n$ , the order of  $x$
- note that  $\#G \in \{i \in \mathbf{Z}; \forall x \in G x^i = 1\} = \lambda\mathbf{Z}$  so  $\lambda$  is a factor of  $\#G$   
so,  $\forall x \in G \text{ order}(x) | \lambda | \#G$
- $\lambda$  is the lcm of all  $\text{order}(x)$ ,  $x \in G$

# Orders in $\mathbf{Z}_m^*$

- $\mathbf{Z}_m^*$  is of order  $\varphi(m)$  (example:  $\mathbf{Z}_{35}^*$  is of order 24)  
 $\mathbf{Z}_m^*$  is of exponent  $\lambda(m)$  (example:  $\mathbf{Z}_{35}^*$  is of exponent 12)  
for  $m = p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$  with pairwise different prime numbers  $p_1, \dots, p_r$ , we have

$$\begin{aligned}\varphi(m) &= (p_1 - 1)p_1^{\alpha_1 - 1} \times \cdots \times (p_r - 1)p_r^{\alpha_r - 1} \\ \lambda(m) &= \text{lcm}(\lambda(p_1^{\alpha_1}), \dots, \lambda(p_r^{\alpha_r}))\end{aligned}$$

we have  $\lambda(p^\alpha) = \varphi(p^\alpha)$ , except for  $p = 2$  and  $\alpha \geq 3$  for which  $\lambda(p^\alpha) = \frac{1}{2}\varphi(p^\alpha)$

- for any  $x \in \mathbf{Z}_m^*$ ,  $\text{order}(x) \mid \lambda(m) \mid \varphi(m)$

# Checking a Generator of a Group with Known Order Factorization

**Input:** an element  $g$  in an Abelian cyclic group of order with known factorization  $n = p_1^{\alpha_1} \times \dots \times p_r^{\alpha_r}$

**Output:** say if  $g$  is a generator

**Complexity:**  $\mathcal{O}(r)$  exponentials

- 1: **for**  $i = 1$  to  $r$  **do**
- 2:    $y \leftarrow g^{n/p_i}$
- 3:   **if**  $y = 1$  **then**
- 4:     abort:  $g$  is not a generator
- 5:   **end if**
- 6: **end for**
- 7:  $g$  is a generator

**Proof.** The order of  $g$  is a factor of  $n$ . If it is no factor of any  $n/p_i$  then it must be  $n$ . □

# Discussion

- for  $g$  arbitrary, we need the factorization of  $n$
- if  $g$  is randomly selected, we only need the small factors of  $n$
- if  $n$  is hard to factor, we can still find generators

# Picking a Generator in a Cyclic Group with Known Order

**Input:** the order  $n$  of an Abelian cyclic group, a bound  $B$

**Output:** a generator  $g$  of the group

- 1: find the list  $p_1, \dots, p_r$  of all prime factors of  $n$  which are less than  $B$
- 2: **repeat**
- 3:   pick a random  $g$  in the group
- 4:    $b \leftarrow \text{true}$
- 5:   **for**  $i = 1$  to  $r$  **do**
- 6:      $y \leftarrow g^{n/p_i}$
- 7:     **if**  $y = 1$  **then**
- 8:        $b \leftarrow \text{false}$
- 9:     **end if**
- 10:   **end for**
- 11: **until**  $b$

$$\Pr[\text{output } g \text{ not a generator}] \leq \frac{1}{B \log B} \log n$$



# Application

generate a generator of  $\mathbf{Z}_p^*$  for a prime  $p$

# Generating a Generator — i

We consider a cyclic group  $G$  of order  $n$  and we let  $n = \prod_{i=1}^r p_i^{\alpha_i}$  with pairwise different primes  $p_i$

- $g$  is a generator of  $G$  iff  $g^{\frac{n}{p_i}} \neq 1$  for  $i = 1, \dots, r$
- given a random  $g \in_U G$ , events  $g^{\frac{n}{p_i}} = 1$  are independent:

$g \in_U G$  is equivalent to its logarithm  $a \in_U \mathbf{Z}_n$

this is equivalent to  $(a_i)_{1 \leq i \leq r} \in_U \mathbf{Z}_{p_1^{\alpha_1}} \times \dots \times \mathbf{Z}_{p_r^{\alpha_r}}$

$g^{\frac{n}{p_i}} = 1$  is equivalent to  $\frac{n}{p_i} a \bmod (n) = 0$

this is equivalent to  $(0, \dots, 0, q_i p_i^{\alpha_i - 1} a_i \bmod p_i^{\alpha_i}, 0, \dots, 0) = 0$  for some invertible  $q_i$  modulo  $p_i^{\alpha_i}$

so,  $g^{\frac{n}{p_i}} = 1$  is equivalent to  $a_i \bmod p_i = 0$  (independent, with probability  $\frac{1}{p_i}$ )

## Generating a Generator — ii

- $\Pr_{g \in_U G} \left[ g^{\frac{n}{p_i}} = 1 \right] = \frac{1}{p_i}$  and these events are independent
- we can just simply work with an incomplete factorization: we let  $n = q \prod_{i=1}^s p_i^{\alpha_i}$  which includes all small factors  $p_i \leq B$  (i.e.  $p_i > B$  for all  $i > s$ )  
we say that  $g$  passes the test if  $g^{\frac{n}{p_i}} \neq 1$  for  $i = 1, \dots, s$

$$\begin{aligned} \Pr[\text{not generator} | \text{passed}] &= \Pr \left[ \exists i > s \ g^{\frac{n}{p_i}} = 1 \mid \forall i \leq s \ g^{\frac{n}{p_i}} \neq 1 \right] \\ &\leq \frac{1}{B} (r - s) \\ &\leq \frac{\log q}{B \log B} \\ &\leq \frac{\log n}{B \log B} \end{aligned}$$

example:  $n$  of 1 024 bits and  $B = 2^{32}$ ;  
 $\Pr[\text{not generator} | \text{passed}] \leq 2^{-27}$

- 3 **RSA Cryptography**
  - Euler and Other Chinese
  - Orders in a Group
  - **Primality Testing**
  - RSA Basics
  - Quadratic Residuosity
  - The Factoring Problem

# Trial Division Algorithm

**Input:** an integer  $n$

**Output:** a list of prime numbers whose product is  $n$

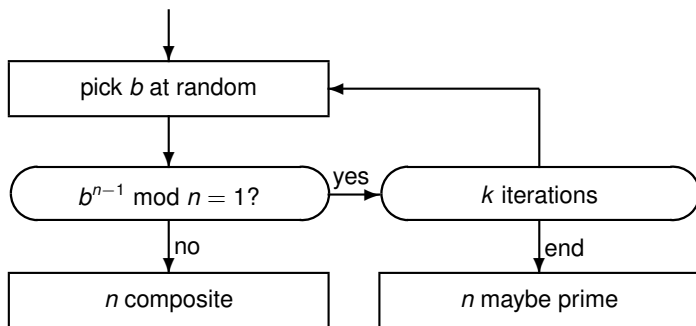
**Complexity:**  $\mathcal{O}(\sqrt{n})$  arithmetic operations

- 1:  $b \leftarrow \lfloor \sqrt{n} \rfloor$ ,  $x \leftarrow n$ ,  $i \leftarrow 2$
- 2: **while**  $x > 1$  and  $i \leq b$  **do**
- 3:     **while**  $i$  divides  $x$  **do**
- 4:         print  $i$
- 5:          $x \leftarrow x/i$
- 6:          $b \leftarrow \lfloor \sqrt{x} \rfloor$
- 7:     **end while**
- 8:      $i \leftarrow i + 1$
- 9: **end while**
- 10: if  $x > 1$  then print  $x$

# Fermat Test

## Theorem (Little Fermat Theorem)

If  $n$  is prime, for any  $b \in \{1, \dots, n-1\}$ ,  $b^{n-1} \bmod n = 1$ .



# Fermat Test

**Parameter:**  $k$ , an integer

**Input:**  $n$ , an integer of  $\ell$  bits

**Output:** notification of non-primality or pseudo-primality

**Complexity:**  $\mathcal{O}(k\ell^3)$

1: **repeat**

2:   pick a random  $b$  such that  $0 < b < n$

3:    $x \leftarrow b^{n-1} \bmod n$

4:   **if**  $x \neq 1$  **then**

5:     output “composite” and stop

6:   **end if**

7: **until**  $k$  iterations are made

8: output “maybe prime” and stop

# Significance of the Fermat Test

- False Negative:  $\Pr[\text{output} : \text{composite} | n \text{ prime}] = 0$
- False Positive: there exist pathologic numbers  $n$  which are not prime such that  $\Pr[\text{output} : \text{maybe prime} | n]$  is high.  
Carmichael Numbers  $n$  are composite such that for any  $b$ ,  
 $b \in \mathbf{Z}_n^* \iff b^{n-1} \bmod n = 1$ . Hence  
 $\Pr[\text{output} : \text{maybe prime} | n] = \left(\frac{\varphi(n)}{n-1}\right)^k$ .



# Carmichael Numbers

## Definition

We call Carmichael number any integer  $n$  which is a product of (at least 2) pairwise different prime numbers  $p_i$  such that  $p_i - 1$  is a factor of  $n - 1$ .

## Theorem

*An integer  $n$  is a Carmichael number if and only if it is composite and for any  $b$  s.t.  $\gcd(b, n) = 1$ , we have  $b^{n-1} \equiv 1 \pmod{n}$ .*

Example:  $n = 561 = 3 \cdot 11 \cdot 17$  is such that for all  $b$  s.t.  $\gcd(b, n) = 1$ , we have  $b^{n-1} \equiv 1 \pmod{n}$ .

# Carmichael Numbers: the 561 Case

Example:  $n = 561 = 3 \cdot 11 \cdot 17$  is such that for all  $b$  s.t.  $\gcd(b, n) = 1$ , we have  $b^{n-1} \equiv 1 \pmod{n}$ .

**Proof** (of  $\Rightarrow$  in the 561 case). We notice that  $n - 1 = 560 = 2^4 \cdot 5 \cdot 7$  which is a multiple of  $3 - 1$ ,  $11 - 1$ , and  $17 - 1$ . Therefore, if  $b$  is prime with 3, we have  $b^{n-1} \equiv 1 \pmod{3}$  and the same for 11 and 17. Hence, from the Chinese Remainder Theorem we obtain that if  $b$  is prime with  $n$  we have  $b^{n-1} \equiv 1 \pmod{n}$ .

The test may be wrong with probability

$$\left(\frac{\varphi(n)}{n-1}\right)^k = \left(\frac{2 \times 10 \times 16}{560}\right)^k = \left(\frac{4}{7}\right)^k$$

# Carmichael Numbers: the 949 631 589 089 Case

$$949\,631\,589\,089 = 6917 \times 10193 \times 13469$$

$$949\,631\,589\,088 = 2^5 \times 7^3 \times 13 \times 19 \times 37 \times 9467$$

- 6917 is prime,  $6916 = 2^2 \times 7 \times 13 \times 19$
- 10193 is prime,  $10192 = 2^4 \times 7^2 \times 13$
- 13469 is prime,  $13468 = 2^2 \times 7 \times 13 \times 37$
- the test may be wrong with probability

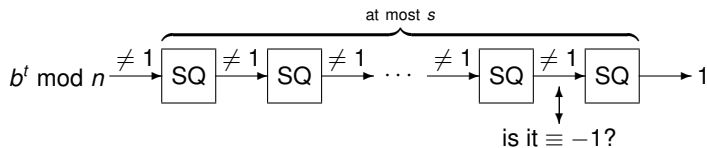
$$\left(\frac{\varphi(n)}{n-1}\right)^k = \left(\frac{9464}{9467}\right)^k \approx (1 - 0.000317)^k$$

example: for  $k = 20$  the error probability is approximately  
 $1 - 0.00631$

# Towards The Miller-Rabin Test

- We write  $n - 1 = 2^s t$  with  $t$  odd
- If  $n$  is prime, we have
$$b^{n-1} \bmod n = \left( \dots ((b^t)^2) \dots \right)^2 \bmod n = 1$$
- If  $n$  is prime,  $+1$  and  $-1$  are the only possible square roots of 1

# The Miller-Rabin Test



Miller-Rabin test: check that the sequence  $(b^t, b^{2t}, \dots, b^{2^s t})$  is of form either  $(1, 1, \dots, 1)$  or  $(*, \dots, *, -1, 1, \dots, 1)$

# The Miller-Rabin Primality Test

**Parameter:**  $k$ , an integer

**Input:**  $n$ , an integer of  $\ell$  bits

**Output:** notification of non-primality or pseudo-primality

**Complexity:**  $O(k\ell^3)$

1: **if**  $n = 2$  **then**

2:   output “prime” and stop

3: **end if**

4: **if**  $n$  is even **then**

5:   output “composite” and stop

6: **end if**

7: write  $n = 2^s t + 1$  with  $t$  odd

8: **repeat**

9:   pick  $b \in \{1, \dots, n - 1\}$

10:    $x \leftarrow b^t \bmod n, i \leftarrow 0$

11:   **if**  $x \neq 1$  **then**

12:     **while**  $x \neq n - 1$  **do**

13:        $x \leftarrow x^2 \bmod n, i \leftarrow i + 1$

14:       **if**  $i = s$  or  $x = 1$  **then**

15:         output “composite”  
          and stop

16:       **end if**

17:     **end while**

18:   **end if**

19: **until**  $k$  iterations are made

20: output “maybe prime” and stop

# Miller-Rabin Criterion

## Theorem

*An integer  $n$  is prime if and only if it passes the Miller-Rabin test for all  $b \in \mathbf{Z}_n^*$ .*

## Proof (Sketch).

- $\Rightarrow$  trivial
- $\Leftarrow$  observe that passing Miller-Rabin implies passing Fermat  
 $\rightarrow$  just prove that Carmichael numbers do not pass □

# Bounding Errors

## Theorem (Miller-Rabin)

*If more than a quarter of  $b \in \mathbf{Z}_n^*$  pass the Miller-Rabin test, then all  $b \in \mathbf{Z}_n^*$  do so.*

Consequence: false positives are negligible:

$$\Pr[\text{output maybe prime} | n \text{ composite}] \leq 4^{-k}$$



# Prime Number Generation

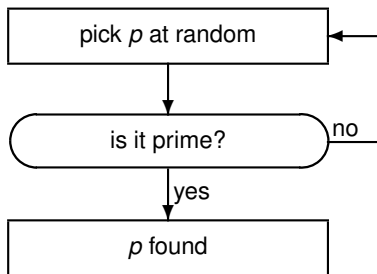
## Theorem (Prime Number Theorem)

Let  $p(N)$  denote the number of prime numbers in  $\{2, 3, \dots, N\}$ . We have  $p(N) \sim \frac{N}{\ln N}$  when  $N$  increases toward the infinity.

→ the probability that a random  $\ell$ -bit number is prime is  $\approx \frac{1}{\ell \ln 2}$

Example: a 512-bit random integer is prime with probability  $\approx \frac{1}{355}$

→ generating a random  $\ell$ -bit prime number takes  $\mathcal{O}(\ell^4)$



# Implementation

**Input:**  $\ell$

**Output:** a random prime number between  $2^{\ell-1}$   
and  $2^\ell$

**Complexity:**  $O(\ell^4)$  arithmetic operations

- 1: **repeat**
- 2:   pick a random number  $n$  of  $\ell$  bits
- 3: **until** a primality test with  $k$  iterations accepts  
     $n$  as a prime number
- 4: output  $n$

With  $k = \frac{1}{2}(\log_2 \ell - \log_2 \varepsilon)$  the probability that this algorithm outputs a composite number is less than  $\varepsilon$ .

## **RSA Cryptography**

- Euler and Other Chinese
- Orders in a Group
- Primality Testing
- **RSA Basics**
- Quadratic Residuosity
- The Factoring Problem

# Plain RSA Cryptosystem

**Public parameter:** an integer  $\ell$ .

**Set up:** find two random different prime numbers  $p$  and  $q$  of size  $\frac{\ell}{2}$  bits. Set  $N = pq$ . Pick a random  $e$  until  $\gcd(e, (p-1)(q-1)) = 1$ . (Sometimes we pick special constant  $e$  like  $e = 17$  or  $e = 2^{16} + 1$ .) Set  $d = e^{-1} \bmod ((p-1)(q-1))$ .

**Message:** an element  $x \in \mathbf{Z}_N$ .

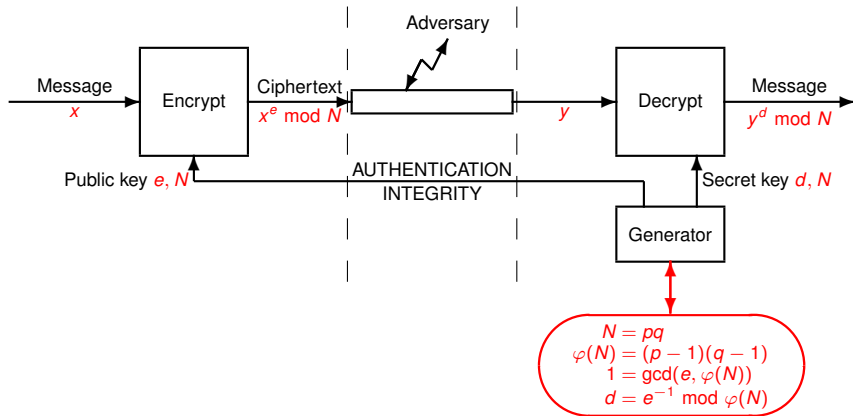
**Public key:**  $K_p = (e, N)$ .

**Secret key:**  $K_s = (d, N)$ .

**Encryption:**  $y = x^e \bmod N$ .

**Decryption:**  $x = y^d \bmod N$ .

# Plain RSA



# RSA Completeness

## Theorem (Euler)

Let  $p, q$  be two different primes and  $N = p \times q$ .

For any  $x \in \{0, \dots, N - 1\}$  and any  $k$ , we have  $x^{k\varphi(N)+1} \bmod N = x$ .

Consequence: RSA decryption works!

**Proof.** from CRT...

# RSA Complexity

RSA with a modulus of  $\ell$  bits and a random  $e$ .

- Generator:  $\mathcal{O}(\ell^4)$  (prime numbers generation)
- Encryption:  $\mathcal{O}(\ell^3)$
- Decryption:  $\mathcal{O}(\ell^3)$

RSA with a modulus of  $\ell$  bits and a constant  $e$  (e.g.  $e = 2^{16} + 1$ ).

- Generator:  $\mathcal{O}(\ell^4)$  (prime numbers generation)
- Encryption:  $\mathcal{O}(\ell^2)$
- Decryption:  $\mathcal{O}(\ell^3)$

# ElGamal vs RSA

- Complexity of Gen is much lower for ElGamal
- Problem: ElGamal encryption is length-increasing
- Can be easily adapted to other groups (e.g. elliptic curves)



## 3

## RSA Cryptography

- Euler and Other Chinese
- Orders in a Group
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

# Square Roots in Finite Fields

## Lemma

Let  $\mathbf{K}$  be a finite field. For any  $x \in \mathbf{K}$  we have

$$x^2 = 1 \implies \begin{cases} x = 1 \\ \text{or} \\ x = -1 \end{cases}$$

**Proof.** Assume that  $x^2 = 1$ . We know that  $x^2 - 1 = (x - 1)(x + 1)$ .

- Case 1:  $x - 1 = 0$  thus  $x = 1$ .
- Case 2:  $x - 1 \neq 0$  so we can divide  $0 = x^2 - 1$  by  $x - 1$  and obtain  $x + 1 = 0$  thus  $x = -1$ .



Consequence:  $x^2 = a$  has at most 2 roots in a finite field

# Existence of Square Roots in $\mathbf{Z}_p$

## Theorem

Let  $p$  be an odd prime number.

$b \in \mathbf{Z}_p^*$  has a square root if and only if  $b^{\frac{p-1}{2}} \pmod p = 1$ .

In that case, we say that  $b$  is a **quadratic residue**.

## Proof:

- $\Rightarrow$  if  $c^2 \equiv b$  then  $b^{\frac{p-1}{2}} \equiv c^{p-1} = 1$
- $\Leftarrow$  since  $\mathbf{Z}_p^*$  is cyclic, let  $g$  be a generator and write  $b \equiv g^e$   
we have  $b^{\frac{p-1}{2}} \equiv 1$  so  $\frac{p-1}{2}e$  is multiple of  $p-1$   
thus  $e$  is even, let  $e = 2e'$  and we have  $b \equiv g^{2e'} \equiv (g^{e'})^2$  so  $b$   
has a square root  $g^{e'}$



# Computing Square Roots in $\mathbf{Z}_p$ , $p \equiv 3 \pmod{4}$

## Lemma

Let  $p$  be a prime number such that  $p \equiv 3 \pmod{4}$ . For any  $x \in \mathbf{Z}_p$  we have

$$y^2 \equiv x \pmod{p} \implies \begin{cases} y \equiv x^{\frac{p+1}{4}} \pmod{p} \\ \text{or} \\ y \equiv -x^{\frac{p+1}{4}} \pmod{p} \end{cases}$$

## Proof.

In  $\mathbf{Z}_p$ , we have

$$\left(x^{\frac{p+1}{4}}\right)^2 = x^{\frac{p+1}{2}} = y^{p+1} = y^{p-1} \times y^2 = y^2 = x$$

so  $x^{\frac{p+1}{4}} = \pm y$ .



# Example

square root of 5 in  $\mathbf{Z}_{11}$

- remark that  $11 \bmod 4 = 3$
- remark that  $5^{\frac{11-1}{2}} \bmod 11 = 5 \times (5^2)^2 \bmod 11 = 1$  so 5 has a square root modulo 11
- compute  $5^{\frac{11+1}{4}} \bmod 11 = 5 \times 5^2 \bmod 11 = 4$
- remark that  $4^2 \bmod 11 = 5$  so 4 is a square root of 5
- other square root is  $-4 \bmod 11 = 7$

# Tonelli Algorithm

**Input:** a quadratic residue  $a \in \mathbf{Z}_p^*$  where  $p \geq 3$   
is prime

**Output:**  $b$  such that  $b^2 \equiv a \pmod{p}$

**Complexity:**  $\mathcal{O}((\log p)^3)$

- 1: **repeat**
- 2:   choose  $g \in \mathbf{Z}_p^*$  at random
- 3: **until**  $g$  is not a quadratic residue
- 4: let  $p - 1 = 2^s t$  with  $t$  odd
- 5:  $e \leftarrow 0$
- 6: **for**  $i = 2$  to  $s$  **do**
- 7:   **if**  $(ag^{-e})^{\frac{p-1}{2^i}} \pmod{p} \neq 1$  **then**
- 8:      $e \leftarrow 2^{i-1} + e$
- 9:   **end if**
- 10: **end for**
- 11:  $b \leftarrow g^{-t\frac{e}{2}} a^{\frac{t+1}{2}} \pmod{p}$

# Square Roots in $\mathbf{Z}_n$ , $n = pq$

## Lemma

Let  $p, q$  be two different prime numbers and  $n = pq$ . Let  $x \in \mathbf{Z}_n$ , and  $a$  and  $b$  such that

$$x \equiv a^2 \pmod{p}$$

$$x \equiv b^2 \pmod{q}$$

We have

$$x \equiv y^2 \pmod{n} \iff \begin{cases} y \equiv \pm a \pmod{p} \\ y \equiv \pm b \pmod{q} \end{cases}$$

Consequence:  $x$  has 4 square roots in  $\mathbf{Z}_n$ .

**Proof.** Thanks to the CRT  $x \equiv y^2 \pmod{n}$  is equivalent to

$$\left. \begin{array}{l} x \equiv y^2 \pmod{p} \\ x \equiv y^2 \pmod{q} \end{array} \right\} \iff \left\{ \begin{array}{l} a^2 \equiv y^2 \pmod{p} \\ b^2 \equiv y^2 \pmod{q} \end{array} \right\} \iff \left\{ \begin{array}{l} y \equiv \pm a \pmod{p} \\ y \equiv \pm b \pmod{q} \end{array} \right.$$

□

# Legendre and Jacobi Symbols

- Legendre Symbol: for  $p$  an odd prime

$$\left(\frac{b}{p}\right) = \begin{cases} 0 & \text{if } b \bmod p = 0 \\ 1 & \text{if } b \text{ is a quadratic residue in } \mathbf{Z}_p^* \\ -1 & \text{if } b \text{ is not a quadratic residue in } \mathbf{Z}_p^*. \end{cases}$$

- Jacobi Symbol: for  $n$  odd

$$\left(\frac{b}{n}\right) = \left(\frac{b}{p_1}\right)^{\alpha_1} \times \dots \times \left(\frac{b}{p_r}\right)^{\alpha_r}$$

where  $n = p_1^{\alpha_1} \times \dots \times p_r^{\alpha_r}$  is the factorization of  $n$  into prime numbers

(remark: for  $n = 1$  the empty product leads us to  $(b/n) = +1$ )



# Application to Quadratic Residuosity

- for  $b \in \mathbf{Z}_p^*$ :  
 $b$  is a quadratic residue in  $\mathbf{Z}_p^*$   $\iff (b/p) = +1$   
( $p$  is prime)
- for  $b \in \mathbf{Z}_n^*$ :  
 $b$  is a quadratic residue in  $\mathbf{Z}_n^*$   $\implies (b/n) = +1$   
**BUT  $\Leftarrow$  IS WRONG!**  
( $n$  is composite)

# Computing the Legendre Symbol

Let  $p$  be an odd prime

$$\left(\frac{b}{p}\right) = \begin{cases} 0 & \text{if } b^{\frac{p-1}{2}} \bmod p = 0 \\ 1 & \text{if } b^{\frac{p-1}{2}} \bmod p = 1 \\ -1 & \text{if } b^{\frac{p-1}{2}} \bmod p = p-1 \end{cases}$$

so  $(b/p)$  is the modulo  $p$  representative of  $b^{\frac{p-1}{2}}$  in  $\{-1, 0, +1\}$

Note that  $x \mapsto (x/p)$  is a group homomorphism from  $\mathbf{Z}_p^*$  to  $\{-1, +1\}$

# Computing the Jacobi Symbol

- $\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$  for  $b$  odd,
- $\left(\frac{ab}{c}\right) = \left(\frac{a}{c}\right) \left(\frac{b}{c}\right)$  for  $c$  odd,
- $\left(\frac{2}{a}\right) = 1$  if  $a \equiv \pm 1 \pmod{8}$  and  $\left(\frac{2}{a}\right) = -1$  if  $a \equiv \pm 3 \pmod{8}$  for  $a$  odd,
- $\left(\frac{a}{b}\right) = -\left(\frac{b}{a}\right)$  if  $a \equiv b \equiv 3 \pmod{4}$  and  $\left(\frac{a}{b}\right) = \left(\frac{b}{a}\right)$  otherwise for  $a$  and  $b$  odd.

# Example

$$\begin{aligned} \left(\frac{b}{n}\right) &= \left(\frac{362}{561}\right) && \text{(factor 2 isolation)} &= \left(\frac{2 \times 9}{181}\right) \\ \text{(factor 2 isolation)} &= \left(\frac{2 \times 181}{561}\right) && \text{(multiplicativity)} &= \left(\frac{2}{181}\right) \times \left(\frac{9}{181}\right) \\ \text{(multiplicativity)} &= \left(\frac{2}{561}\right) \times \left(\frac{181}{561}\right) && (181 \equiv 5 \pmod{8}) &= -\left(\frac{9}{181}\right) \\ (561 \equiv 1 \pmod{8}) &= \left(\frac{181}{561}\right) && \text{(quadratic reciprocity)} &= -\left(\frac{181}{9}\right) \\ \text{(quadratic reciprocity)} &= \left(\frac{561}{181}\right) && \text{(modular reduction)} &= -\left(\frac{1}{9}\right) \\ \text{(modular reduction)} &= \left(\frac{18}{181}\right) && &= -1 \end{aligned}$$

# The Group of Quadratic Residues

the Jacobi symbol is homomorphic:

$$\left(\frac{x}{n}\right) \left(\frac{y}{n}\right) = \left(\frac{xy}{n}\right)$$

let  $QR_n$  be the subgroup of  $\mathbf{Z}_n^*$  of all quadratic residues  
we have the following properties:

- $QR_n$  is included in the subgroup of  $\mathbf{Z}_n^*$  of all  $x$  such that  $\left(\frac{x}{n}\right) = +1$ . They match if  $n$  is prime.
- $x \in QR_n$  and  $y \in QR_n$  implies  $xy \in QR_n$
- $x \in QR_n$  and  $y \in \mathbf{Z}_n^* - QR_n$  implies  $xy \in \mathbf{Z}_n^* - QR_n$
- for  $p$  prime,  $x \in \mathbf{Z}_p^* - QR_p$  and  $y \in \mathbf{Z}_p^* - QR_p$  implies  $xy \in QR_p$
- this does not extend to composite  $n$ :  
 $3 \in \mathbf{Z}_{35}^* - QR_{35}$  and  $2 \in \mathbf{Z}_{35}^* - QR_{35}$  but  $6 \notin QR_{35}$

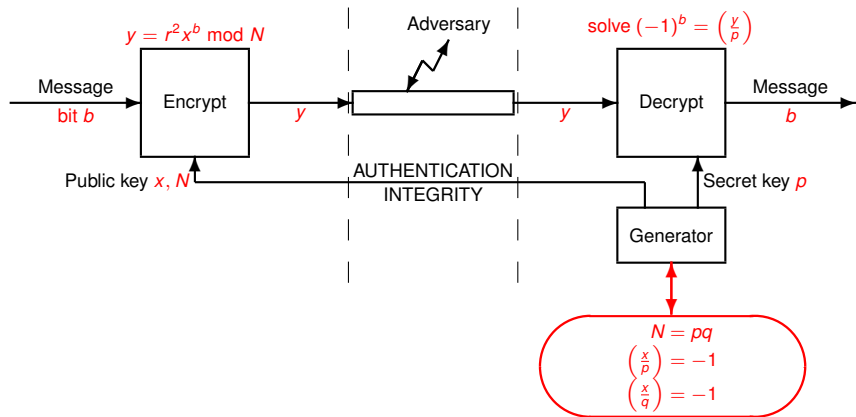
# Conclusion

- algorithm to compute  $(b/n)$  in  $\mathcal{O}(\ell^2)$
- can be used to check quadratic residuosity if  $n$  is prime

# Use of Quadratic Residuosity

- Goldwasser-Micali cryptosystem
- Solovay-Strassen primality testing  $b^{\frac{p-1}{2}} \equiv \left(\frac{b}{p}\right) \pmod{p}$
- breaking the DDH assumption in  $\mathbf{Z}_p^*$   
note: the ElGamal cryptosystem is IND-CPA secure iff the DDH assumption on the group is hard  
so, it is unsafe to use the ElGamal cryptosystem in  $\mathbf{Z}_p^*$
- mapping  $\{1, \dots, q\}$  to  $\text{QR}_p$  for  $p = 2q + 1$  and use the ElGamal cryptosystem in  $\text{QR}_p$  ( $p$  and  $q$  prime)

# Goldwasser-Micali Encryption





# Solovay-Strassen Test

## Theorem

Let  $n$  be an odd number.  
 $n$  is prime if and only if

$$\Pr \left[ b^{\frac{n-1}{2}} \equiv \left( \frac{b}{n} \right) \pmod{n} \right] \geq \frac{1}{2}$$

for  $b \in \mathbf{Z}_n^*$  with uniform distribution.

Note: the square of this equation is  $b^{n-1} \equiv 1 \pmod{n}$  but we do not have a so strong result with the Fermat test.

# Breaking the Decisional Diffie-Hellmann Assumption in $\mathbf{Z}_p^*$

- Let  $p$  be an odd prime and  $g$  be a generator of  $\mathbf{Z}_p^*$
- We consider the following algorithm:

Algorithm  $\mathcal{A}(g, X, Y, K)$

- 1: set  $a = \mathbf{1}_{(K/p)=-1}$
- 2: set  $b = \mathbf{1}_{(X/p)=(Y/p)=-1}$
- 3: output  $\mathbf{1}_{a=b}$

- Let  $x, y, k \in \mathbf{Z}_{p-1}$  be uniform and independent
- For  $X = g^x, Y = g^y, K = g^k$ , we have  $\Pr[\mathcal{A}(g, X, Y, K) = 1] = \frac{1}{2}$
- For  $X = g^x, Y = g^y, K = g^{xy}$ , we have  $\Pr[\mathcal{A}(g, X, Y, K) = 1] = 1$
- so,  $\mathcal{A}(g, X, Y, K)$  can distinguish if  $K$  is random or the solution to the Diffie-Hellman problem with  $(g, X, Y)$ .

# Mapping a Number to an ElGamal Group Element

- Let  $p = 2q + 1$  with  $p$  and  $q$  prime, we have  $(-1/p) = -1$  so  $-1$  is not a quadratic residue
- the group of quadratic residues  $QR_p$  is cyclic and of order  $q$ , not containing  $-1$
- for all  $x \in \mathbf{Z}_p^*$ , either  $x \in QR_p$  or  $-x \in QR_p$  but not both
- so,  $\text{map}(x) = x \times \left(\frac{x}{p}\right) \bmod p$  maps  $\{1, \dots, q\}$  onto  $QR_p$
- we can define the ElGamal cryptosystem on  $QR_p$  and use map to represent messages

## 3

## RSA Cryptography

- Euler and Other Chinese
- Orders in a Group
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

# Factoring Problem

## Factoring Problem

**Parameters:** Gen, a pseudorandom generator

**Instance:**  $n$ , an integer produced by Gen

**Problem:** factor  $n$

Examples:

- Gen generates an RSA modulus
- Gen generates Mersenne numbers

# Record using the Number Field Sieve Algorithm

Complexity:  $e^{\mathcal{O}\left((\ln n)^{\frac{1}{3}} (\ln \ln n)^{\frac{2}{3}}\right)}$

RSA768

= 1230186684530117755130494958384962720772853569595334792197322452151726400507  
2636575187452021997864693899564749427740638459251925573263034537315482685079  
1702612214291346167042921431160222124047927473779408066535141959745985690214  
3413

= 3347807169895689878604416984821269081770479498371376856891243138898288379387  
8002287614711652531743087737814467999489

×

3674604366679959042824463379962795263227915816434308764267603228381573966651  
1279233373417143396810270092798736308917

factored in 2009 by an equivalent of 1500 years of computation on  
one core 2.2GHz Opteron.

# Record using the Number Field Sieve Algorithm

$$2^{1039} - 1$$

$$= 5080711$$

×

(306 digits)

$$= 5080711$$

×

5585366661993629126074920465831594496864652701848863764801005234631985328837  
4753

×

2075818194644238276457048137035946951629397080073952098812083870379272909032  
4679382343143884144834882534053344769112223028158327696525376091410189105241  
993899334109711624358962065972167481161749004803659735573409253205425523689

factored in 2007 by an equivalent of 100 years of computation on a  
PC 2.2GHz (Opteron).

# Factorization Tomorrow

Factorization of  $n$  with complexity  $\mathcal{O}((\ln n)^2 \ln \ln n \ln \ln \ln n)$  by using Shor's algorithm

It only works on a quantum computer (if exists)



# Factoring Algorithms on Classical Computers

- GNFS: factor  $n$

$$\text{complexity} = e^{\sqrt{\frac{64}{9} + o(1)}(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}}$$

best algorithm for RSA moduli

- ECM: finds a factor  $p$

$$\text{complexity} = e^{\sqrt{2+o(1)}(\ln p)^{\frac{1}{2}}(\ln \ln p)^{\frac{1}{2}}}$$

useful for numbers with a small prime factor

# Square Roots in $\mathbf{Z}_{pq}$

Gen: generates integers of form  $n = pq$  with  $p \neq q$  both prime

## Factoring $n$

**Params.:** generator Gen

**Instance:**  $n$  generated by Gen

**Problem:** factor  $n$



## Square roots in $\mathbf{Z}_n$

**Params.:** generator Gen

**Instance:**  $n$  generated by Gen and a quadratic residue  $x \in \mathbf{Z}_n$

**Problem:** find  $y$  s.t.  $y^2 \bmod n = x$

# Factoring $n \implies$ Computing Square Roots in $\mathbb{Z}_n$

**Input:** factorization  $n = pq$  and  $x$

**Output:**  $y$  such that  $y^2 \bmod n = x$

**Complexity:**  $\mathcal{O}((\log n)^3)$

- 1: find  $y_p$ , a square roots of  $x$  modulo  $p$  by using efficient algorithms  
(e.g. for  $p \bmod 4 = 3$  compute  $x^{\frac{p+1}{4}} \bmod p$ )
- 2: find  $y_q$ , a square roots of  $x$  modulo  $q$
- 3:  $y = \text{CRT}_{p,q}(y_p, y_q)$

# Computing Square Roots in $\mathbb{Z}_n \implies$ Factoring $n$

**Input:**  $n$ , access to a square root oracle SQRT

**Output:**  $p, q$  prime such that  $n = pq$

**Complexity:**  $\mathcal{O}((\log n)^2 + |\text{SQRT}|)$

1: **repeat**

2: pick  $y_0 \in \{1, \dots, n-1\}$

3:  $x = y_0^2 \bmod n$

4:  $y = \text{SQRT}(n, x)$

5: **until**  $y \neq y_0$  and  $y \neq -y_0 \bmod n$

6:  $p = \gcd(y - y_0, n)$

7:  $q = n/p$

- since there are 4 square roots, we have  $\Pr[y = y_0 \text{ or } y = -y_0 \bmod n] = \frac{1}{2}$
- in other cases,  $y - y_0$  is zero modulo one of the two factors but not modulo the other:  $\gcd(y - y_0, n)$  is the former factor

# Computing Element Orders in $\mathbf{Z}_n^*$

Gen: generates integers of form  $n = pq$  with  $p \neq q$  both prime

## Factoring $\lambda(n)$

**Params.:** generator Gen

**Instance:** Gen  $\rightarrow n$

**Problem:** factor  $\lambda(n)$



## Computing orders in $\mathbf{Z}_n^*$

**Params.:** generator Gen

**Instance:** Gen  $\rightarrow n, x \in \mathbf{Z}_n^*$

**Problem:** order of  $x$



## Factoring $n$

**Params.:** generator Gen

**Instance:** Gen  $\rightarrow n$

**Problem:** factor  $n$



## Computing $\lambda(n)$

**Params.:** generator Gen

**Instance:** Gen  $\rightarrow n$

**Problem:** compute  $\lambda(n)$

# Factoring $\lambda(n) \implies$ Computing Element Orders in $\mathbf{Z}_n^*$

**Input:** factorization

$$\lambda(n) = p_1^{\alpha_1} \cdots p_r^{\alpha_r}, x \in \mathbf{Z}_n^*$$

**Output:** the order  $u$  of  $x$

**Complexity:**  $\mathcal{O}(r)$  exponentials

```
1:  $u \leftarrow 1$ 
2: for  $i = 1$  to  $r$  do
3:    $y \leftarrow x^{\lambda(n)/p_i^{\alpha_i}} \bmod n$ 
4:   while  $y \neq 1$  do
5:      $y \leftarrow y^{p_i} \bmod n$ 
6:      $u \leftarrow u \times p_i$ 
7:   end while
8: end for
```

**Fact.** If the order of  $x$  is  $p_1^{\beta_1} \cdots p_r^{\beta_r}$  then, for all  $i$ ,

- $\beta_i \leq \alpha_i$
- $x^{\lambda(n)p_i^{\beta_i - \alpha_i}} \bmod n = 1$
- $x^{\lambda(n)p_i^{\beta_i - \alpha_i - 1}} \bmod n \neq 1$

# Computing Element Orders in $\mathbf{Z}_n^*$ $\implies$ Knowing $\lambda(n)$

**Input:** an element order oracle in  $\mathbf{Z}_n^*$

**Output:**  $\lambda(n)$

1:  $\lambda \leftarrow 1$

2: **repeat**

3:   pick a random  $x$  in  $\mathbf{Z}_n^*$

4:   compute the order  $u$  of  $x$

5:    $\lambda \leftarrow \text{lcm}(\lambda, u)$

6: **until**  $\lambda$  has not changed for a while

**Fact.** With the same notations: for all  $i$ ,  $\Pr[\beta_i < \alpha_i] \leq 1/p_i$

Thus, the number of iterations is likely to be very small

# Knowing $\lambda(n) \implies$ Factoring $n$

**Input:**  $\lambda(n)$  ( $n$  odd)

**Output:** a non trivial factor of  $n$

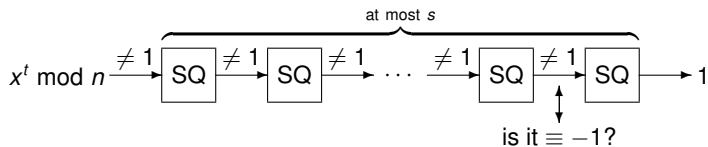
- 1: write  $\lambda(n) = 2^s t$  with  $t$  odd
- 2: **repeat**
- 3:   pick a random  $x$  in  $\mathbf{Z}_n^*$
- 4:    $x \leftarrow x^t \bmod n$
- 5:    $y \leftarrow \perp$
- 6:   **while**  $x \neq 1$  **do**
- 7:      $y \leftarrow x$
- 8:      $x \leftarrow x^2 \bmod n$
- 9:   **end while**
- 10: **until**  $y \neq \perp$  and  $y \not\equiv -1 \pmod{n}$
- 11: output  $\gcd(y - 1, n)$

**Fact.** For  $x \in \mathbf{Z}_n$ , if  $x^2 \bmod n = 1$ ,  $x \neq 1$ ,  $x \neq n-1$  then  $1 < \gcd(n, x-1) < n$  which is a non-trivial factor of  $n$ :

- $n$  divides  $(x-1)(x+1)$
- if  $\gcd(n, x-1) = n$  then  $n$  divides  $x-1$  thus  $x = 1$  which is wrong
- if  $\gcd(n, x-1) = 1$  then  $n$  divides  $x+1$  thus  $x = n-1$  which is wrong



# Factorization using $\lambda(n)$



# Knowing $\lambda(n) \iff$ Factoring $n$

- $\implies$ : previous slide
- $\impliedby$ :  $\lambda(p_1^{\alpha_1} \cdots p_r^{\alpha_r})$  is computed by

$$\text{lcm}((p_1 - 1)p_1^{\alpha_1 - 1}, \dots, (p_r - 1)p_r^{\alpha_r - 1})$$

- NB: knowing a multiple of  $\lambda(n) \iff$  Factoring  $n$   
(same proof)
- example: knowing  $\varphi(n) \iff$  Factoring  $n$

Conclusion: computing  $\varphi(n)$  is hard, computing orders in  $\mathbf{Z}_n^*$  is hard

# Consequence

- knowing  $K_p$  and  $K_s$  in RSA implies factoring  $N$
- it is insecure to use common prime numbers between two RSA keys

# Conclusion

- **Euler  $\varphi$  function**: to compute the order of  $\mathbf{Z}_n^*$
- **Chinese Remainder Theorem**: parallel  $\mathbf{Z}_m$  and  $\mathbf{Z}_n$
- **primality testing**: efficient, used to generate prime numbers
- **RSA cryptosystem**: public-key cryptosystem
- **factoring problem**: believed to be hard

# Computational Problems

## easy

- gcd
- inverse modulo  $n$
- exponential
- square root mod  $n$  when factorization of  $n$  is known
- Legendre/Jacobi symbol
- checking primality
- finding a generator when group order is known
- computing order when factorization of group order is known

## hard

- factoring
- discrete logarithm (sometimes)
- square root mod  $n$
- computing  $\varphi(n)$ ,  $\lambda(n)$
- checking quadratic residuosity
- computing order in group

# References

- **Shoup.** *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press. 2005.  
<http://shoup.net/ntb>  
Textbook on algebra for cryptographers and applications.
- **Menezes-van Oorschot-Vanstone.** *Handbook of Applied Cryptography.* CRC. 1997.  
<http://www.cacr.math.uwaterloo.ca/hac/>  
Reference book
- **Vaudenay.** *A Classical Introduction to Cryptography — Applications for Communications Security.* Springer. 2005.  
<http://www.vaudenay.ch/crypto/>  
Textbook on cryptography
- **Rivest-Shamir-Adleman.** A Method for Obtaining Digital Signatures and Public-key Cryptosystem. *Communications of the ACM* vol. 21, 1978.

# Must be Known

- **Euler  $\phi$  function**: formula, properties
- **Chinese Remainder Theorem**: how to use it
- **orders**: tricks to check/pick a generator
- **primality testing**: properties, how to use to generate prime numbers
- **RSA**: why it works, complexity
- **quadratic residuosity**: how to check, when it is easy to extract square roots
- **factoring problem**: some reductions to other problems

# Train Yourself

- Chinese Remainder Theorem:  
midterm exam 2013–14 ex1  
final exam 2012–13 ex1  
midterm exam 2012–13 ex2  
midterm exam 2011–12 ex2  
midterm exam 2010–11 ex1  
midterm exam 2010–11 ex2  
midterm exam 2009–10 ex2  
midterm exam 2008–09 ex1
- square roots, cubic roots:  
midterm exam 2013–14 ex2  
midterm exam 2009–10 ex1
- quadratic residuosity:  
midterm exam 2012–13 ex1
- prime number generation:  
midterm exam 2014–15 ex1
- RSA variant:  
final exam 2015–16 ex2



- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography**
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

# Roadmap

- Galois fields
- elliptic curves over  $\mathbf{Z}_p$
- elliptic curves over  $\text{GF}(2^k)$
- using standard curves
- Diffie-Hellman over elliptic curves
- ElGamal over elliptic curves
- pairing-based cryptography

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curves over a Binary Field
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

# $\text{GF}(p^k)$ for Dummies

$p$ : a prime number.

- Euclidean division in  $\mathbf{Z}_p[x]$ : for any polynomials  $A(x)$  and  $P(x)$  such that  $P \neq 0$ , there exists polynomials  $R(x)$  and  $B(x)$  such that  $A(x) = R(x) + P(x) \cdot B(x)$  and  $\deg(R) < \deg(P)$ . We call  $R(x) = A(x) \bmod P(x)$  the remainder of  $A(x)$  modulo  $P(x)$ .
- Select a monic (i.e. with leading coefficient 1) irreducible (i.e. who cannot be expressed as a product of polynomials with smaller degree) polynomial  $P(x)$  of degree  $k$  in  $\mathbf{Z}_p[x]$ .
- Let  $\text{GF}(p^k)$  be the set of all polynomials in  $\mathbf{Z}_p[x]$  of degree at most  $k - 1$ .
- Addition: regular polynomial addition modulo  $p$ .
- Multiplication: regular multiplication in  $\mathbf{Z}_p[x]$  reduced modulo  $P(x)$ .
- We can prove this constructs a field.

## Example: GF(8)

In order to construct  $GF(2^3)$ :

- consider the ring  $\mathbf{Z}_2[x]$  of polynomials
- take the monic irreducible polynomial  $P(x) = x^3 + x + 1$  of degree 3
- construct

$$GF(2^3) = \{0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1\}$$

Example:  $(x + 1) + (x^2 + 1) = x^2 + x$  in  $GF(2^3)$ .

Example:  $(x + 1) \times (x^2 + 1) = x^3 + x^2 + x + 1 = x^2$  in  $GF(2^3)$ .

# Cerebral $\text{GF}(p^k)$

$p$ : a prime number.

- $\mathbf{Z}_p[x]$  is a Euclidean ring.
- Select a monic irreducible polynomial  $P(x)$  of degree  $k$  in  $\mathbf{Z}_p[x]$ .
- $P(x)$  spans a maximal ideal  $(P(x))$
- Let  $\text{GF}(p^k) = \mathbf{Z}_p[x]/(P(x))$  be the quotient of ring  $\mathbf{Z}_p[x]$  by ideal  $(P(x))$ .
- We obtain a field who inherits the addition and multiplication from the ring structure of  $\mathbf{Z}_p[x]$ .

# Galois Fields

## Theorem

We have the following results.

- The cardinality of any finite field is a prime power  $p^k$ .
- For any prime power  $p^k$ , there exists a finite field of cardinality  $p^k$ .  $p$  is called the **characteristic** of the field.
- Two finite fields of same cardinality are isomorphic, so the finite field of cardinality  $p^k$  is essentially unique. We denote it  $\text{GF}(p^k)$  as **Galois field** of cardinality  $p^k$ .
- $\text{GF}(p^k)$  is isomorphic to a subfield of  $\text{GF}(p^{k \times \ell})$ .
- $\text{GF}(p^k)$  can be defined as the quotient of ring of polynomials with coefficients in  $\mathbf{Z}_p$  by a principal ideal spanned by an irreducible polynomial of degree  $k$ :  $\mathbf{Z}_p[x]/(P(x))$ .

## Example: GF(5)

$$\text{GF}(5) = \mathbf{Z}_5 = \{0, 1, 2, 3, 4\}$$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$$(\text{GF}(5), +) \approx (\mathbf{Z}_5, +) \quad (\text{GF}(5)^*, \times) \approx (\mathbf{Z}_4, +)$$



## Example: GF(4)

$$\text{GF}(4) = \{0, 1, x, x + 1\} \neq \mathbf{Z}_4$$

+	0	1	x	x + 1
0	0	1	x	x + 1
1	1	0	x + 1	x
x	x	x + 1	0	1
x + 1	x + 1	x	1	0

×	0	1	x	x + 1
0	0	0	0	0
1	0	1	x	x + 1
x	0	x	x + 1	1
x + 1	0	x + 1	1	x

$$(\text{GF}(4), +) \approx (\mathbf{Z}_2 \times \mathbf{Z}_2, +) \quad (\text{GF}(4)^*, \times) \approx (\mathbf{Z}_3, +)$$

$$P(x) = x^2 + x + 1 \text{ irreducible in } \mathbf{Z}_2[x], \text{GF}(4) = \mathbf{Z}_2[x]/(P(x))$$

# Example: GF(2<sup>8</sup>)

## Arithmetics in AES

A byte  $a = a_7 \dots a_1 a_0$  represents an element of the finite field GF(2<sup>8</sup>) as a polynomial  $a_0 + a_1.x + \dots + a_7.x^7$  modulo  $x^8 + x^4 + x^3 + x + 1$  and modulo 2

byte	polynomial
0x00	0
0x01	1
0x02	$x$
0x03	$x + 1$
0x1b	$x^4 + x^3 + x + 1$

Addition: bitwise XOR

Multiplication by 0x02: shift and XOR with 0x1b if carry

# Most Important Finite Fields

- “prime field”:  $\mathbf{Z}_p$  for a large prime  $p$
- “binary field”:  $\text{GF}(2^k)$

	$\mathbf{Z}_p$	$\text{GF}(2^k)$
representation	integers from 0 to $p - 1$	polynomials in $x$ of degree at most $k - 1$ with binary coefficients ( $k$ -bit strings) requires the choice of an irreducible polynomial $P(x)$ of degree $k$
addition	addition modulo $p$	bitwise XOR
multiplication	multiplication modulo $p$	ad-hoc algorithms multiplication by $0_{\text{x}2}$ : shift to the left and XOR to a constant if carry

# Characteristic 2 Tips

In  $\text{GF}(2^k)$ :

- $1 + 1 = 0$
- minus = plus:  $-a = a$
- square is linear:  $(a + b)^2 = a^2 + b^2$
- power  $2^i$  is linear
- for  $k > 1$ ,  $a^{2^{k-1}}$  is the unique square root of  $a$
- trace function:  $\text{Tr}(a) = a + a^2 + a^{2^2} + \dots + a^{2^{k-1}} \in \{0, 1\}$   
(traces are roots of  $z^2 = z$ )  
Fact: Tr is linear:  $\text{Tr}(a + b) = \text{Tr}(a) + \text{Tr}(b)$   
Fact: for all  $a$  in  $\text{GF}(2^k)$  we have  $\text{Tr}(a^2) = \text{Tr}(a)$

# Exercise

- 1 show that if  $z$  is a root of  $a = z^2 + z$  then  $\text{Tr}(a) = 0$   
 $\text{Tr}(a) = \text{Tr}(z^2 + z) = \text{Tr}(z^2) + \text{Tr}(z) = \text{Tr}(z) + \text{Tr}(z) = 0$
- 2 show that if  $\text{Tr}(a) = 0$  then  $a = z^2 + z$  has exactly two roots  $\theta$  and  $\theta + 1$

we have  $(z + 1)^2 + (z + 1) = z^2 + 1 + z + 1 = z^2 + z$  so the mapping  $z \mapsto z^2 + z$  has at most  $\frac{1}{2}2^k$  images

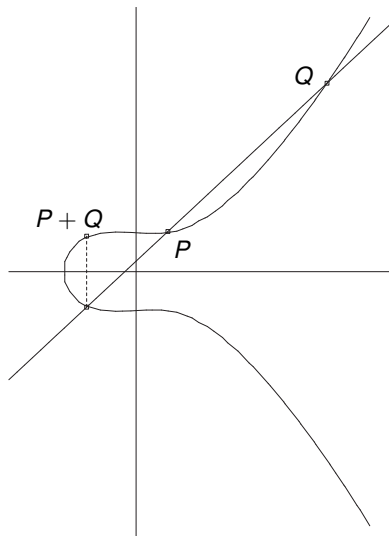
$z^2 + z = a$  cannot have more than two roots, to the mapping  $z \mapsto z^2 + z$  has exactly  $\frac{1}{2}2^k$  images and each image is reached exactly twice, by some  $\{\theta, \theta + 1\}$  pair

thanks to the first question, images are in the set of the  $\frac{1}{2}2^k$  field elements with trace zero

## Elliptic Curve Cryptography

- Galois Fields
- **Elliptic Curves**
- Elliptic Curves over a Prime Field
- Elliptic Curves over a Binary Field
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

# Elliptic Curves



# Addition in Elliptic Curves

## Chord and Tangent Formula

$$E_{a,b} = \{\mathcal{O}\} \cup \{(x, y); y^2 = x^3 + ax + b\}$$

- we assume that  $E_{a,b}(\mathbf{K})$  is **non-singular**:  
when a point is non-singular we can define the tangent to this point  
singular point  $\iff$  differential of  $y^2 - (x^3 + ax + b)$  vanishes  
 $\iff y = 0$  and  $x^3 + ax + b = 0$  multiple root  
curve non-singular  $\iff 4a^3 + 27b^2 \neq 0$
- $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$  is the chord slope
- $\lambda = \frac{3x_P^2 + a}{2y_P}$  is the tangent slope  
( $\lambda = \infty \iff y_P = 0 \iff P + P = \mathcal{O}$ )
- the sum of the 3 roots  $x$  of the intersection between  $E_{a,b}(\mathbf{K})$  and the straight line  $y = \lambda x + \mu$  is  $\lambda^2 = x_P + x_Q + x_R$



# Group Structure

$$E_{a,b} = \{\mathcal{O}\} \cup \{(x, y); y^2 = x^3 + ax + b\}$$

- Given  $P = (x_P, y_P)$ , we define  $-P = (x_P, -y_P)$  and  $-\mathcal{O} = \mathcal{O}$ .
- Given  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q = -P$ , we define  $P + Q = \mathcal{O}$ .
- Given  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q \neq -P$ , we let

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } x_P \neq x_Q \\ \frac{3x_P^2 + a}{2y_P} & \text{if } x_P = x_Q \end{cases}$$

$$x_R = \lambda^2 - x_P - x_Q$$

$$y_R = (x_P - x_R)\lambda - y_P$$

$R = (x_R, y_R)$  and  $P + Q = R$ .

- In addition,  $P + \mathcal{O} = \mathcal{O} + P = P$  and  $\mathcal{O} + \mathcal{O} = \mathcal{O}$ .

# Elliptic Curves are Abelian Groups

by restricting to  $x, y \in \mathbf{K}$  where  $\mathbf{K}$  is a field (example:  $\mathbf{Q}$ ,  $\mathbf{R}$ ,  $\mathbf{C}$ ,  $\text{GF}(p^k)$ )

1.  $E_{a,b}(\mathbf{K})$  is closed for the addition
2. the addition is associative in  $E_{a,b}(\mathbf{K})$   
HARD (from the chord and tangent formula)
3.  $\mathcal{O}$  is neutral for the addition
4. for any  $P \in E_{a,b}(\mathbf{K})$  we have  $-P \in E_{a,b}(\mathbf{K})$  which is the inverse of  $P$  for addition
5. the addition is commutative

$E_{a,b}(\mathbf{K})$  is an Abelian group

## Remark on Points of Order 2 (Characteristic $> 2$ )

$$\begin{aligned} P = (x, y) \text{ has order 2} &\iff P = -P \text{ and } P \neq \mathcal{O} \\ &\iff y = 0 \text{ and } x^3 + ax + b = 0 \end{aligned}$$

So, the number of points of order 2 is the number of roots of  $x^3 + ax + b$  in  $\mathbf{K}$

(If we have more than 1 root, the group cannot be cyclic!)

# Recap

(for characteristic  $> 3$ )

- EC are **curves** (set of points whose coordinate satisfy an equation)
- the curve must be **non-singular** ( $\Delta \neq 0$  for some parameter  $\Delta$ )
- EC can (depending on the field) be defined by the equation  $y^2 = x^3 + ax + b$  (need to add a point  $\mathcal{O}$ )
- EC have an addition rule, making a **group** structure
  - can multiply a point by an integer
  - some curves can be isomorphic
  - contrarily to  $\mathbf{Z}_p^*$ , EC are not always cyclic (but we can work on a cyclic subgroup)

## 4

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- **Elliptic Curves over a Prime Field**
- Elliptic Curves over a Binary Field
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

# Roadmap

- same formulas, but over  $\mathbf{Z}_p$
- notion of twist: elliptic curves come in pairs
- notion of  $j$ -invariant: an invariant value by isomorphism
- cardinality close to  $p$

# Addition over an Elliptic Curve (Characteristic $p > 3$ )

(Field  $\mathbf{K}$  of characteristic  $p > 3$ )

$$E_{a,b}(\mathbf{K}) = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{K}^2; y^2 = x^3 + ax + b\}$$

Hypothesis: (**discriminant**)  $\Delta = -16(4a^3 + 27b^2) \neq 0$

- for  $P = (x_P, y_P)$ , we let  $-P = (x_P, -y_P)$  and  $-\mathcal{O} = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q = -P$  we let  $P + Q = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q \neq -P$  we let

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } x_P \neq x_Q \\ \frac{3x_P^2 + a}{2y_P} & \text{if } x_P = x_Q \end{cases}$$

$$x_R = \lambda^2 - x_P - x_Q$$

$$y_R = (x_P - x_R)\lambda - y_P$$

$R = (x_R, y_R)$  and  $P + Q = R$ .

- addition to  $\mathcal{O}$ :  $P + \mathcal{O} = \mathcal{O} + P = P$  and  $\mathcal{O} + \mathcal{O} = \mathcal{O}$ .

# Maybe Useful to Know $p > 3$

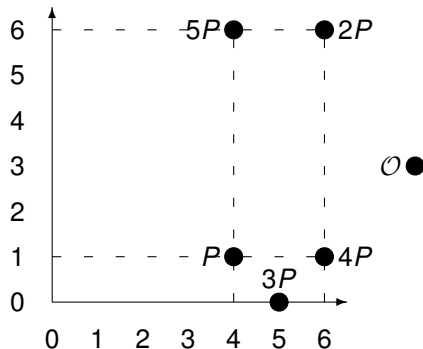
Hypothesis: field  $\mathbf{K}$  of characteristic  $p > 3$  and  $\Delta \neq 0$

- $E_{a,b}$  and  $E_{u^4a, u^6b}$  are isomorphic (by  $(x, y) \mapsto (u^2x, u^3y)$ )
- $E_{a,b}$  and  $E_{v^2a, v^3b}$  are **twist** of each other if  $v$  is not a square  
NB: they become isomorphic in an extension of  $\mathbf{K}$  where  $v$  becomes a square
- **$j$ -invariant:**  $j = 1728 \frac{4a^3}{4a^3 + 27b^2}$
- $\#E_{a,b}$  is between  $q + 1 - 2\sqrt{q}$  and  $q + 1 + 2\sqrt{q}$  where  $q$  is the cardinality of  $\mathbf{K}$  (Hasse Theorem)  
NB: for two twists, the average of  $\#E_{a,b}$  is  $q + 1$



# Other Example

$E_{1,3}$  over  $\text{GF}(7) = \mathbf{Z}_7$  is isomorphic to  $\mathbf{Z}_6$   
 $y^2 = x^3 + x + 3$



# Recap

- EC can be defined by the equation  $y^2 = x^3 + ax + b$  (plus a point  $\mathcal{O}$ )
- **twist**: pair of non-isomorphic curves which become isomorphic when defined over a larger field
- **$j$ -invariant**: parameter which is always the same for isomorphic curves and for twists
- the order of a curve is close to the cardinality of the field

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- Elliptic Curves over a Prime Field
- **Elliptic Curves over a Binary Field**
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

# Roadmap

- similar, but with different formulas
- again: notions of twist,  $j$ -invariant
- a special case: “supersingular curves”
- recent results on DL raise big concerns on their security

▶ skip binary curves

# Addition over an Elliptic Curve (Characteristic

$p = 2)$

(Field  $\mathbf{K}$  of characteristic  $p = 2$ )

case of curve **non supersingular** (= **ordinary** curve)

$$E_{a_2, a_6}(\mathbf{K}) = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{K}^2; y^2 + xy = x^3 + a_2x^2 + a_6\}$$

hypothesis: (**discriminant**)  $\Delta = a_6 \neq 0$

- for  $P = (x_P, y_P)$ , we let  $-P = (x_P, x_P + y_P)$  and  $-\mathcal{O} = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q = -P$ , we let  $P + Q = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q \neq -P$ , we let

$$\lambda = \begin{cases} \frac{y_Q + y_P}{x_Q + x_P} & \text{if } x_P \neq x_Q \\ \frac{x_P^2 + y_P}{x_P} & \text{if } x_P = x_Q \end{cases}$$

$$x_R = \lambda^2 + \lambda + a_2 + x_P + x_Q$$

$$y_R = (x_P + x_R)\lambda + y_P + x_R$$

$R = (x_R, y_R)$  and  $P + Q = R$ .

- addition to  $\mathcal{O}$ :  $P + \mathcal{O} = \mathcal{O} + P = P$  and  $\mathcal{O} + \mathcal{O} = \mathcal{O}$ .

# Maybe Useful to Know (Non-supersingular, Binary)

Hypothesis: field  $\mathbf{K} = \text{GF}(q)$  of characteristic 2 and  $\Delta \neq 0$ , non-supersingular elliptic curve

- $E_{a_2, a_6}(\mathbf{K})$  is **non-singular**:  
a point is singular  $\iff$  the differential of  $(y^2 + xy) - (x^3 + a_2x^2 + a_6)$  vanishes  $\iff x = y = a_6 = 0$   
existence  $\iff a_6 = 0$
- $E_{a_2, a_6}$  and  $E_{a_2+u^2+u, a_6}$  are isomorphic (by  $(x, y) \mapsto (x, ux + y)$ )
- $E_{a_2, a_6}$  and  $E_{a_2+v, a_6}$  are **twist** of each other if  $\text{tr}_2(v) = \sum_{2^i < q} v^{2^i} = 1$  (they become isomorphic in an extension of  $\mathbf{K}$  in which  $\text{tr}_2(v)$  vanishes)
- **$j$ -invariant**:  $j = 1/\Delta$
- $\#E_{a_2, a_6}$  is between  $q + 1 - 2\sqrt{q}$  and  $q + 1 + 2\sqrt{q}$  where  $q$  is the cardinality of  $\mathbf{K}$  (Hasse Theorem)  
NB: for two twists, the average of  $\#E_{a_2, a_6}$  is  $q + 1$

# The Supersingular Case

(Field  $\mathbf{K}$  of characteristic  $p = 2$ )  
case of curve **supersingular**

$$E_{a_3, a_4, a_6}(\mathbf{K}) = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{K}^2; y^2 + a_3y = x^3 + a_4x + a_6\}$$

hypothesis: (**discriminant**)  $\Delta = a_3^4 \neq 0$

- for  $P = (x_P, y_P)$ , we let  $-P = (x_P, y_P + a_3)$  and  $-\mathcal{O} = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q = -P$ , we let  $P + Q = \mathcal{O}$ .
- for  $P = (x_P, y_P)$  and  $Q = (x_Q, y_Q)$ , if  $Q \neq -P$ , we let

$$\lambda = \begin{cases} \frac{y_Q + y_P}{x_Q + x_P} & \text{if } x_P \neq x_Q \\ \frac{x_P^2 + a_4}{a_3} & \text{if } x_P = x_Q \end{cases}$$

$$x_R = \lambda^2 + x_P + x_Q$$

$$y_R = (x_P + x_R)\lambda + y_P + a_3$$

$R = (x_R, y_R)$  and  $P + Q = R$ .

- addition to  $\mathcal{O}$ :  $P + \mathcal{O} = \mathcal{O} + P = P$  and  $\mathcal{O} + \mathcal{O} = \mathcal{O}$ .

# Maybe Useful to Know (Supersingular, Binary)

Hypothesis: field  $\mathbf{K} = \text{GF}(q)$  of characteristic 2 and  $\Delta \neq 0$ ,  
supersingular curve

- $E_{a_3, a_4, a_6}(\mathbf{K})$  is **non-singular** iff  $a_3 \neq 0$ :  
a point is singular  $\iff$  the differential of  
 $(y^2 + a_3y) - (x^3 + a_4x + a_6)$  vanishes  $\iff x^2 = a_4, y^2 = a_6,$   
 $a_3 = 0$   
existence  $\iff a_3 = 0$
- the  **$j$ -invariant** vanishes ( $j = 0$ )



## 4

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curves over a Binary Field
- **Using Elliptic Curves**
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

# Hardness of the Discrete Logarithm

- DL is easy in *anomalous* curves over  $\mathbf{Z}_p$
- binary curves may be exposed to recent attacks
- there are other families of weak curves
- in a group of order  $n$ , Pollard Rho algorithm solves DL in  $\mathcal{O}(\sqrt{n})$
- we can consider tradeoffs:  
run precomputation of  $\mathcal{O}(n^{\frac{2}{3}})$  then compute any DL in  $\mathcal{O}(n^{\frac{2}{3}})$   
(people tend to use the very same curves...)

Note: curves which are bad for DL may be good for other things...  
(e.g. pairing-based cryptography)

# Using Point Compression (Prime Field Case)

Elliptic curve equation:

$$y^2 = x^3 + ax + b$$

A single  $x$  leads to two  $y$  which are opposite from each other.

→ *we can get  $y$  from*

- $x$
- *the parity of  $y$  ( $y$  and  $p - y$  have different parity)*

Format “ $hh$  hexstring”

- $hh = 00$  point  $\mathcal{O}$  (following: nothing)
- $hh = 02$  point compression with  $y$  even (following:  $x$ )
- $hh = 03$  point compression with  $y$  odd (following:  $x$ )
- $hh = 04$  no compression (following:  $x$  and  $y$ )

# Using Point Compression (Binary Field Case)

Elliptic curve equation:

$$\left(\frac{y}{x}\right)^2 + \frac{y}{x} = x + a_2 + \frac{a_6}{x^2}$$

A single  $x$  leads to two  $y$  such that  $\frac{y}{x} = \theta$  or  $\theta + 1$  for some  $\theta$

→ *we can get  $y$  from*

- $x$
- *the constant term of  $y/x$  as a polynomial (the two roots  $y/x$  have sum 1 thus only differ in their constant term)*

Format “ $hh$  hexstring”

- $hh = 00$  point  $\mathcal{O}$  (following: nothing)
- $hh = 02$  point compression with  $y/x$  even (following:  $x$ )
- $hh = 03$  point compression with  $y/x$  odd (following:  $x$ )
- $hh = 04$  no compression (following:  $x$  and  $y$ )

# Manipulating Elliptic Curves in Practice

A representation problem:

- bit strings
- byte strings
- integers
- polynomials
- field elements
- elliptic curve points

see <http://www.secg.org/sec1-v2.pdf> for an example of representation standard

# Domain Parameters

- a field
  - either a prime number  $p$
  - or a power  $q$  of 2 together with an irreducible polynomial over  $\text{GF}(2)$  of degree  $\log_2 q$
- field elements defining an elliptic curve  $E$
- a point  $G$  in  $E$
- the order  $n$  of  $G$  in  $E$  (may be smaller than the order of  $E$ )
- (for pseudorandom curves) a seed  $s$  (to generate a  $j$ -invariant)

# Standard Curves

- pseudorandom curves over  $\mathbf{Z}_p$ 
  - $y^2 = x^3 + ax + b$
  - provide seed to generate  $j$

→ Discrete Log is assumed to be hard
- ordinary curves over a binary field
  - $y^2 + xy = x^3 + a_2x^2 + a_6$
  - for pseudorandom curves: provide seed to generate  $j$
  - for special curves (Koblitz curves):  $a_6 = 1, a_2 \in \{0, 1\}$

# NIST Standard Curves (2013)

NIST Recommended Elliptic Curves for Federal Government Use  
Appendix D of FIPS186-4

<http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>

- pseudorandom over  $\mathbf{Z}_p$ : P192, P224, P256, P384, P521
- ordinary curves over binary fields:
  - pseudorandom: B163, B233, B283, B409, B571
  - special: K163, K233, K283, K409, K571  
(called **Koblitz curves** or **anomalous binary curves (ABC)**)



# SECG Standard Curves (2000)

## SEC2: Recommended Elliptic Curve Domain Parameters

<http://www.secg.org/sec2-v2.pdf>

- pseudorandom over  $\mathbf{Z}_p$ : secp112r1, secp112r2, secp128r1, secp128r2, secp160r1, secp160r2, secp192r1, secp224r1, secp256r1, secp384r1, secp521r1
- special over  $\mathbf{Z}_p$ : secp160k1, secp192k1, secp224k1, secp256k1 (called **generalized Koblitz curves**)
- pseudorandom over binary fields: sect113r1, sect113r2, sect131r1, sect131r2, sect163r1, sect163r2, sect193r1, sect193r2, sect233r1, sect283r1, sect409r1, sect571r1
- special over binary fields: sect163k1, sect233k1, sect239k1, sect283k1, sect409k1, sect571k1 (called **Koblitz curves** or **anomalous binary curves (ABC)**)

# Other Standards

- ANSI X9.62
- IEEE P1363

## Example: secp192r1 = P192

$$\text{secp192r1} = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{Z}_p; y^2 = x^3 + ax + b\}$$

$p$  = 6277101735386680763835789423207666416083908700390324961279  
 $a$  =  $p - 3$   
= 6277101735386680763835789423207666416083908700390324961276  
 $b$  = 2455155546008943817740293915197451784769108058161191238065  
 $n$  = 6277101735386680763835789423176059013767194773182842284081  
 $G$  = 03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012  
= 03 : 602046282375688656758213480587526111916698976636884684818  
seed = 3045ae6f c8422f64 ed579528 d38120ea e12196d5

note that  $p = 2^{192} - 2^{64} - 1$ ,  $2^{192} - 2^{95} < n < 2^{192}$ , and  $n$  is prime

## Example: sect163r2 = B163

$$\text{sect163r2} = \{\mathcal{O}\} \cup \{(x, y) \in \text{GF}(q); y^2 + xy = x^3 + a_2x^2 + a_6\}$$

$$\begin{aligned}q &= 2^{163} \\f(x) &= x^{163} + x^7 + x^6 + x^3 + 1 \\a_2 &= 1 \\a_6 &= 02\ 0a601907\ b8c953ca\ 1481eb10\ 512f7874\ 4a3205fd \\n &= 04\ 00000000\ 00000000\ 000292fe\ 77e70c12\ a4234c33 \\&= 5846006549323611672814742442876390689256843201587 \\G &= 03\ 03\ f0eba162\ 86a2d57e\ a0991168\ d4994637\ e8343e36 \\seed &= 85e25bfe\ 5c86226c\ db12016f\ 7553f9d0\ e693a268\end{aligned}$$

note that  $2^{162} < n < 2^{162} + 2^{82}$  and  $n$  is prime

# Elliptic Curves are Real

secp256r1 = P256

used for digital signature in Swiss biometric passports

## Example: Curve25519

$$\text{Curve25519} = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{Z}_p; y^2 = x^3 + 486\,662x^2 + x\}$$

$$\begin{aligned} p &= 2^{255} - 19 \\ x_G &= 9 \\ \text{order}(G) &= 2^{252} + 27742317777372353535851937790883648493 \end{aligned}$$

Some X25519 function comes with it for ECDH

- equation different than previous ones!
- optimized implementations
- made by no company or government agency
- used in SSH, Tor, Signal, Bitcoin, ...

## 4

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curves over a Binary Field
- Using Elliptic Curves
- **Elliptic Curve Cryptography**
- Pairing-Based Cryptography

# ECDH: Elliptic Curve Diffie-Hellman

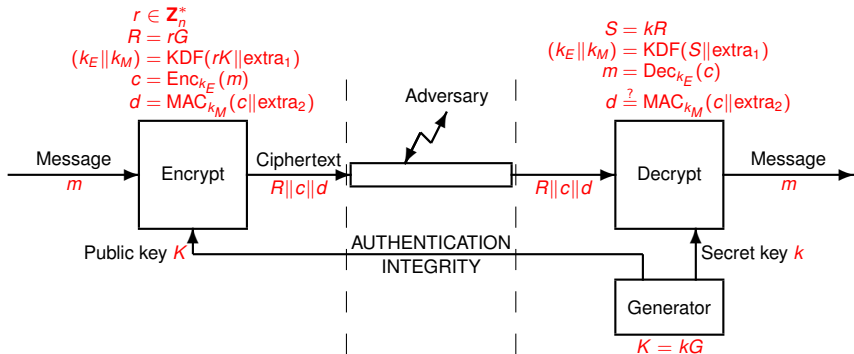
- specified in SEC1 (<http://www.secg.org/sec1-v2.pdf>) and IEEE1363
- used in Bluetooth 2.1
- used in EAC for epassports



Participants:  $U$  and  $V$

- $U$  and  $V$  agree on domain parameters  $T = (p, a, b, G, n, h)$  or  $T = (m, f(x), a, b, G, n, h)$   
( $h$  is the cofactor  $\frac{1}{n} \# E(\text{GF}(q))$  with  $q = p$  or ( $q = 2^m$ ))
- $U$  resp.  $V$  selects his secret key  $d_U$  resp.  $d_V \in \mathbf{Z}_n^*$  and compute his public key  $Q_U = d_U \cdot G$  resp.  $Q_V = d_V \cdot G$
- $U$  and  $V$  exchange their public keys
- both check  $Q \in E(\text{GF}(p))$ ,  $Q \neq \mathcal{O}$ ,  $n \cdot Q = \mathcal{O}$
- both compute  $P = d_U \cdot Q_V = d_V \cdot Q_U$
- set  $z = x_P$
- convert the field element  $z$  into a byte string  $Z$
- use a KDF as agreed to derive a key  $K$

# ECIES (EC Integrated Encryption Scheme)



select field, elliptic curve  
G point of order  $n$   
 $n$  prime

# Principles of ECIES

- use Diffie-Hellman to exchange a symmetric  $k_E || k_M$
- use  $k_E$  to encrypt
- use  $k_M$  for integrity protection

this is a **hybrid encryption**:

we use public-key cryptosystem to exchange a symmetric key and symmetric cryptography to transport the message securely

# Exercise

identify the algebraic structure (group/ring/field), the corresponding law(s) and neutral element(s)

- $\mathbf{Z}_{26}$ ...
- the set of permutations over the alphabet...
- secp192r1...
- $\text{GF}(2^{128})$ ...

## Elliptic Curve Cryptography

- Galois Fields
- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curves over a Binary Field
- Using Elliptic Curves
- Elliptic Curve Cryptography
- **Pairing-Based Cryptography**

# Pairing of Elliptic Curves

for some pairs of elliptic curves  $G_1$  and  $G_2$  we can construct a function

$$e : G_1 \times G_2 \rightarrow G_T$$

to a group  $G_T$  (with multiplicative notations) such that

- $e$  is **bilinear**:  $e(aP, bQ) = e(P, Q)^{ab}$  for  $a, b \in \mathbf{Z}$ ,  $P \in G_1$ ,  $Q \in G_2$
- $e$  is **non-degenerate**:  $e(P, Q) \neq 1$  for some  $P \in G_1$  and  $Q \in G_2$

(we use supersingular curves)

consequences:

- this may be bad for EC-security in  $G_1 = G_2$  as we can distinguish  $(P, xP, yP, xyP)$  from  $(P, xP, yP, zP)$  by checking  $e(xP, yP) = e(P, xyP)$   
we call  $G_1 = G_2$  a **gap group** because the *computational Diffie-Hellman problem* may remain hard even though the *decisional Diffie-Hellman problem* is easy
- this may create new cryptographic primitives

# 3-Party Diffie-Hellman Key Agreement in a Single Round

let  $G$  generate a subgroup of order  $p$  of  $G_1 = G_2$  such that  $e(G, G) \neq 1$

- Alice picks  $a \in \mathbf{Z}_p$  and broadcasts  $A = aG$
- Bob picks  $b \in \mathbf{Z}_p$  and broadcasts  $B = bG$
- Charly picks  $c \in \mathbf{Z}_p$  and broadcasts  $C = cG$
- all compute  $K = e(G, G)^{abc}$ 
  - Alice computes  $e(B, C)^a = K$
  - Bob computes  $e(C, A)^b = K$
  - Charly computes  $e(A, B)^c = K$

# Popular Cryptographic Constructions based on Pairings

- Joux 2000: 3-party Diffie-Hellman key agreement in one round
- Boneh-Franklin 2001: identity-based encryption
- Boneh-Lynn-Shacham 2003: a signature scheme (short)
- Sahai-Water 2004: attribute-based encryption



# Conclusion

- elliptic curves are groups which can be used in cryptography
- advantage: smaller parameters for the same security
- better complexity than RSA
- many standards are using elliptic curves

# References

- **Shoup.** *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press. 2005.  
<http://shoup.net/ntb>
- <http://www.secg.org/sec1-v2.pdf>
- <http://www.secg.org/sec2-v2.pdf>

# Must be Known

- understand how to add points with the help of the formulas (don't learn them!)
- understand how to manipulate objects (field elements, points, integers)
- understand point compression
- understand the standards

# Train Yourself

- finite fields: midterm 2008–09 ex3
- projective coordinates: midterm 2013–14 ex3
- discrete logarithm: final exam 2013–14 ex3
- mapping a message to a point: midterm exam 2014–15 ex2
- elliptic curve factoring method: midterm exam 2015–16 ex2











- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption**
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

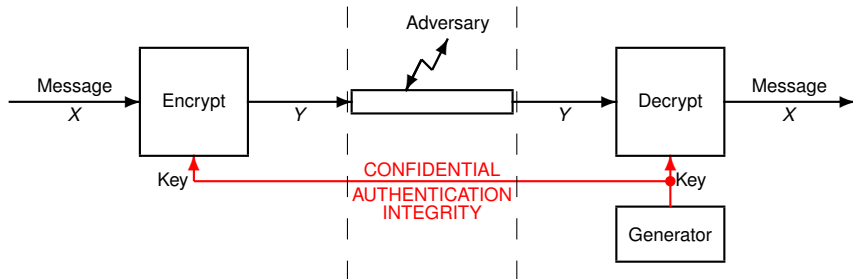
# Roadmap

- block ciphers: DES, triple-DES, AES
- modes of operations: ECB, CBC, OFB, CFB, CTR, XTS
- stream ciphers: RC4, A5/1
- exhaustive search and tradeoffs
- meet-in-the-middle attack

## Symmetric Encryption

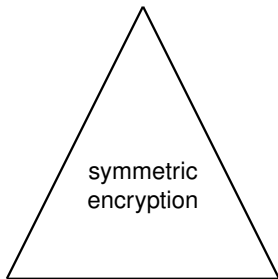
- A Cryptographic Primitive
  - Block Ciphers
  - Stream Ciphers
  - Bruteforce Inversion Algorithms
  - Subtle Bruteforce Inversion Algorithms
  - Pushing the Physical Limits
  - Formalism

# Symmetric Encryption



# Symmetric Encryption (Informal)

Alice and Bob, Generator, Encrypt, Decrypt  
**components**



symmetric  
encryption

**functionality**

$$\text{Decrypt}_K(\text{Encrypt}_K(X)) = X$$

**security**

confidentiality is preserved

# Example: Vernam Cipher

**components:** Alice and Bob, a parameter  $n$

- **Generator:** select  $K \in \{0, 1\}^n$  uniformly at random and set it up for Alice and Bob
- **Encrypt:** for  $X \in \{0, 1\}^n$ , compute  $Y = X \oplus K$ , send  $Y$  and discard  $K$
- **Decrypt:** for  $Y \in \{0, 1\}^n$ , compute  $X = Y \oplus K$  and discard  $K$

**functionality:** for any  $X$  we have  $\text{Decrypt}_K(\text{Encrypt}_K(X)) = X$

**security:** perfect secrecy ( $X$  and  $Y$  have independent distribution)

Warning: use  $K$  only once

# Two Categories of Symmetric Encryption

stream ciphers	block ciphers
<b>RC4</b>	<b>DES</b>
<b>GSM-A5/1</b>	<b>3DES</b>
Bluetooth-E0	IDEA
CSS	BLOWFISH
...	RC5
	<b>AES</b>
	KASUMI
	SAFER
	CS-Cipher
	FOX
	...

## Symmetric Encryption

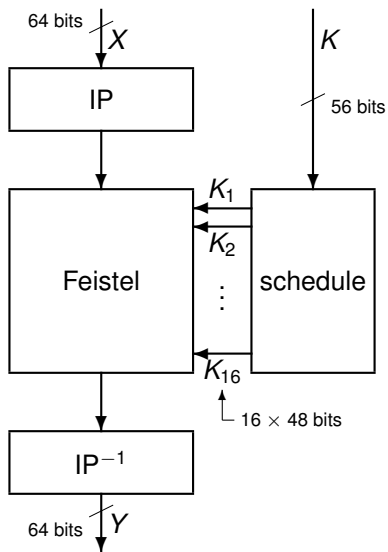
- A Cryptographic Primitive
- **Block Ciphers**
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subtle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism



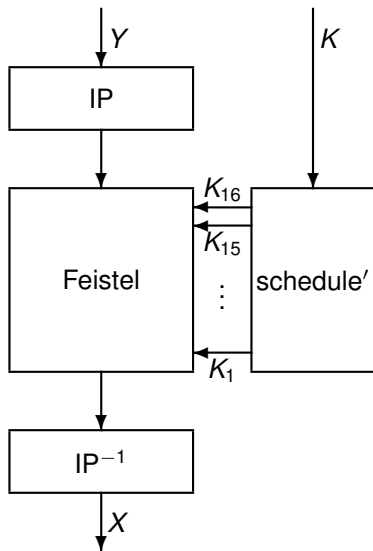
# DES: the Data Encryption Standard

- US Standard from NBS (now NIST), branch of the Department of Commerce in 1977
- secret design by IBM based on a call for proposal
- based on LUCIFER by Horst Feistel (from IBM)
- design influenced by the NSA
- rationales of the design published by Don Coppersmith in 1994
  
- dedicated to hardware implementation
- block cipher with 64-bit blocks
- key of 56 effective bits

# DES



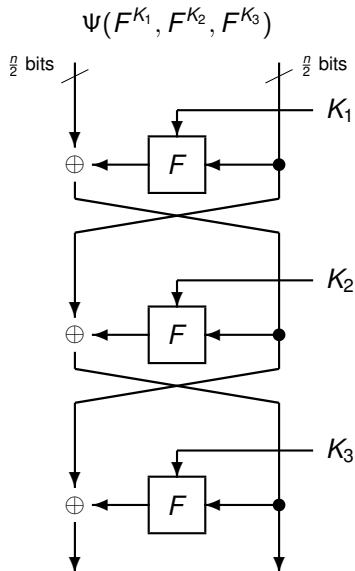
# DES<sup>-1</sup>



# Feistel Scheme

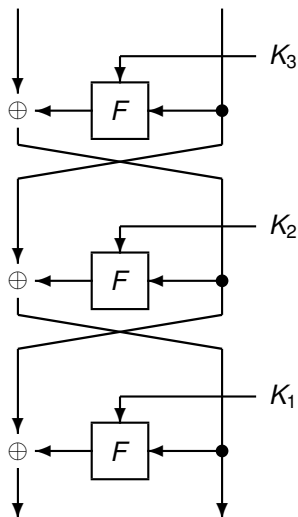
- transform function over  $\{0, 1\}^{\frac{n}{2}}$  into permutations over  $\{0, 1\}^n$
- inverse permutations have same structure
- alternate round functions and halve swaps
- final halve swap omitted

# (Direct) Feistel Scheme

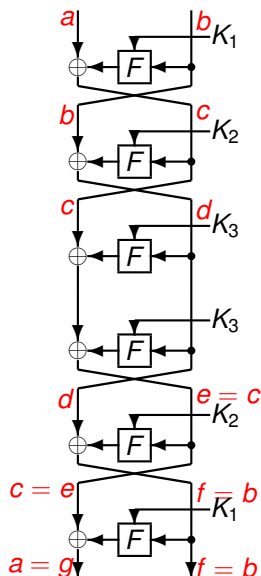


# (Inverse) Feistel Scheme

$$\Psi^{-1}(F^{K_1}, F^{K_2}, F^{K_3}) = \Psi(F^{K_3}, F^{K_2}, F^{K_1})$$

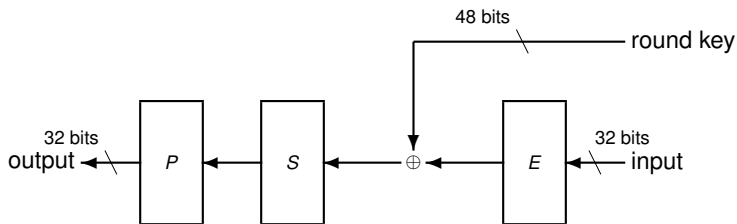


# (Direct + Inverse) Feistel Scheme



- $e = c \oplus F^{K_3}(d) \oplus F^{K_3}(d) = c$
- $f = d \oplus F^{K_2}(e) = (b \oplus F^{K_2}(c)) \oplus F^{K_2}(c) = b$
- $g = e \oplus F^{K_1}(f) = c \oplus F^{K_1}(b) = (a \oplus F^{K_1}(b)) \oplus F^{K_1}(b) = a$

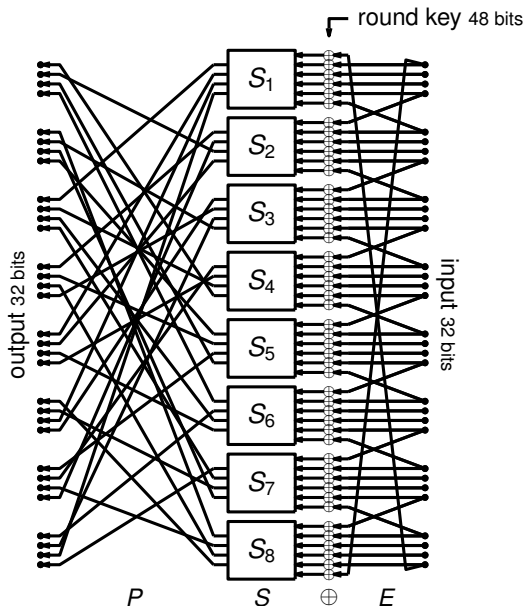
# DES Round Function Overview



- $E$ : expansion (32 to 48 bits)
- $\oplus$ : bitwise XOR to a round key
- $S$ : eight 6-bit to 4-bit S-boxes (substitution boxes)
- $P$ : permutation



# DES Round Function



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

Example:  $S_3(111000) = 0101$ :

$$1\ 1100\ 0 = 56$$

$$1100 = 12$$

$$10 = 2$$

$$0101 = 5$$

# DES Key Schedule

**schedule**( $K$ )

- 1:  $K \xrightarrow{\text{PC1}} (C, D)$
- 2: **for**  $i = 1$  to 16 **do**
- 3:    $C \leftarrow \text{ROL}_{r_i}(C)$
- 4:    $D \leftarrow \text{ROL}_{r_i}(D)$
- 5:    $K_i \leftarrow \text{PC2}(C, D)$
- 6: **end for**

$K$ : 56-bit register

$C, D$ : two 28-bit registers

$K_1, \dots, K_{16}$ : sixteen 48-bit registers

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r_i$	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

# DES Inverse Key Schedule

**schedule'**( $K$ )

- 1:  $K \xrightarrow{\text{PC1}} (C, D)$
- 2: **for**  $i = 16$  down to 1 **do**
- 3:    $K_i \leftarrow \text{PC2}(C, D)$
- 4:    $C \leftarrow \text{ROR}_{r_i}(C)$
- 5:    $D \leftarrow \text{ROR}_{r_i}(D)$
- 6: **end for**

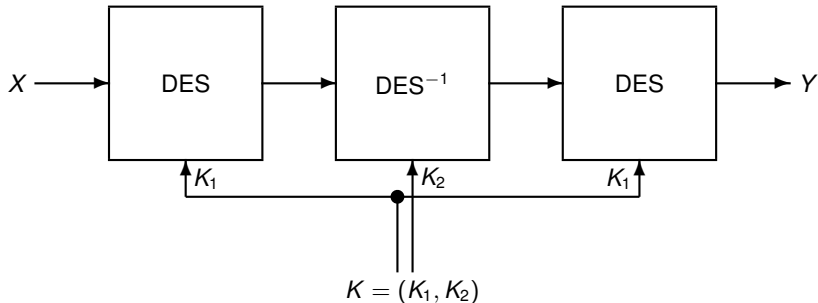
$K$ : 56-bit register

$C, D$ : two 28-bit registers

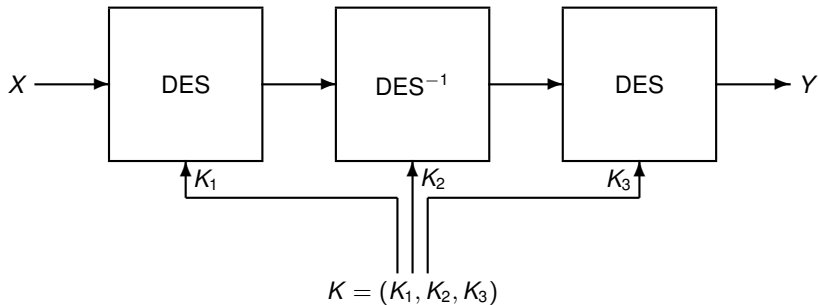
$K_1, \dots, K_{16}$ : sixteen 48-bit registers

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r_i$	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

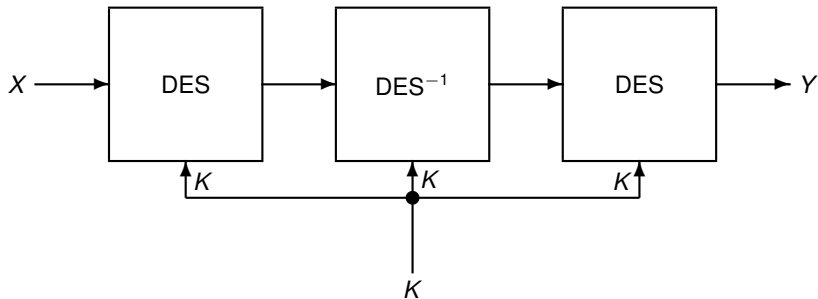
# Two-Key Triple DES



# Three-Key Triple DES



# From Triple DES to DES



a 3K 3DES chip can emulate DES

# Security Notions

- adversary objective: learn confidential information
- typically: **key recovery**
- **ciphertext only attack**: using ciphertexts in transit only
- **known plaintext attack**: same + know (or guess) the corresponding plaintexts
- **chosen plaintext attack**: force the sender to encrypt some messages selected by the adversary
- **chosen ciphertext attack**: force the receiver to decrypt some messages selected by the adversary



# Attacks on DES

- weak keys (1977)
- optimized exhaustive search (Hellman 1980)
- chosen plaintext attack against 2-key TDES using  $2^{56}$  chosen plaintexts,  $2^{56}$  time and  $2^{56}$  memory (Merkle-Hellman 1981)
- known plaintext attack against 2-key TDES using  $2^t$  known plaintexts,  $2^{120-t}$  time (van Oorschot-Wiener 1990)
- study on dedicated hardware (Diffie-Hellman 1977, Wiener 1993)
- chosen plaintext attack with  $2^{47}$  chosen plaintexts (Biham-Shamir 1992)
- known plaintext attack with  $2^{43}$  known plaintexts (Matsui 1994) or actually a little less  $2^{40}$  (Junod 2001)
- optimized exhaustive search within 4 days on a dedicated hardware (EFF 1998)
- bruteforce on 3-key TDES using  $2^{32}$  known plaintexts,  $2^{113}$  time and  $2^{88}$  memory (Lucks 1998)

# AES: the Advanced Encryption Standard

- US Standard from NIST, branch of the Department of Commerce in 2001
- public process based on a call for proposal
- standard version of **Rijndael**
- Rijndael was designed by Joan **Daemen** and Vincent **Rijmen** in Belgium
  
- dedicated to software on 8-bit microprocessors
- block cipher with 128-bit blocks
- key of length 128, 192, or 256
  
- cartoon: [www.moserware.com/2009/09/stick-figure-guide-to-advanced.html](http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html)  
animation: [www.formaestudio.com/rijndaelinspector/archivos/rijndaelanimation.html](http://www.formaestudio.com/rijndaelinspector/archivos/rijndaelanimation.html)

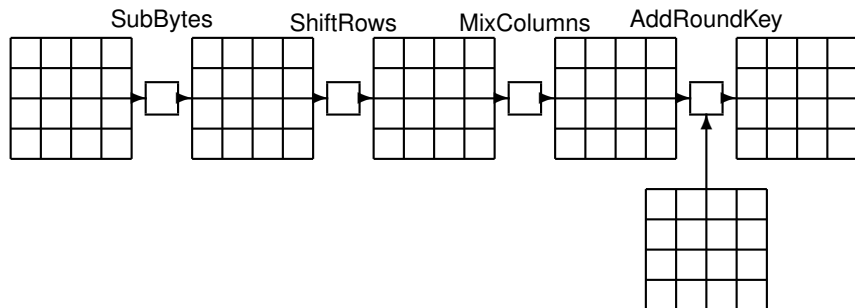
# Rijndael Skeleton

- 128-bit block  $\rightarrow$   $4 \times 4$  square matrix of bytes
- $N_r = 10, 12$  or  $14$  rounds depending on the key size of 128, 192 or 256 bits

**AES encryption**( $s, W$ )

- 1: **AddRoundKey**( $s, W_0$ )
- 2: **for**  $r = 1$  to  $N_r - 1$  **do**
- 3:     **SubBytes**( $s$ )
- 4:     **ShiftRows**( $s$ )
- 5:     **MixColumns**( $s$ )
- 6:     **AddRoundKey**( $s, W_r$ )
- 7: **end for**
- 8: **SubBytes**( $s$ )
- 9: **ShiftRows**( $s$ )
- 10: **AddRoundKey**( $s, W_{N_r}$ )

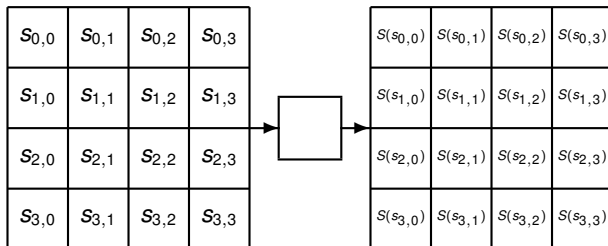
# One Non-Terminal Round of Rijndael



# SubBytes

## SubBytes( $s$ )

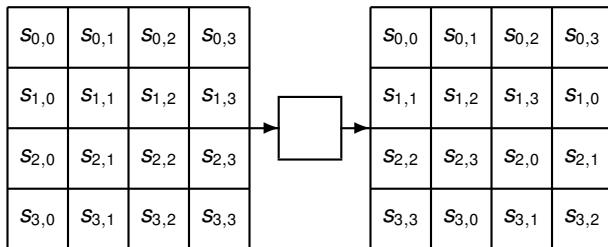
- 1: **for**  $i = 0$  to 3 **do**
- 2:     **for**  $j = 0$  to 3 **do**
- 3:          $s_{i,j} \leftarrow \text{S-box}(s_{i,j})$
- 4:     **end for**
- 5: **end for**



# ShiftRows

## ShiftRows( $s$ )

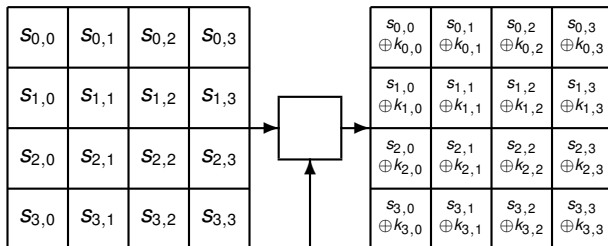
- 1: replace  $[s_{1,0}, s_{1,1}, s_{1,2}, s_{1,3}]$  by  $[s_{1,1}, s_{1,2}, s_{1,3}, s_{1,0}]$
- 2: replace  $[s_{2,0}, s_{2,1}, s_{2,2}, s_{2,3}]$  by  $[s_{2,2}, s_{2,3}, s_{2,0}, s_{2,1}]$
- 3: replace  $[s_{3,0}, s_{3,1}, s_{3,2}, s_{3,3}]$  by  $[s_{3,3}, s_{3,0}, s_{3,1}, s_{3,2}]$



# AddRoundKey

**AddRoundKey**( $s, k$ )

- 1: **for**  $i = 0$  to 3 **do**
- 2:     **for**  $j = 0$  to 3 **do**
- 3:          $s_{i,j} \leftarrow s_{i,j} \oplus k_{i,j}$
- 4:     **end for**
- 5: **end for**



# Introduction to GF Arithmetics in Rijndael

look at [slide 330](#)

- we use the following representation rule

byte	bit string	polynomial
$B$	$b_7 \cdots b_2 b_1 b_0$	$b_7 \cdot x^7 + \cdots + b_2 \cdot x^2 + b_1 \cdot x + b_0$

- we replace every 2 by 0 in polynomials  
hence  $3 = 2 + 1$  is replaced by  $0 + 1 = 1$ , 4 is replaced by 0, ...  
→ monomial coefficients are binary
- we replace every  $x^8$  by  $x^4 + x^3 + x + 1$  in polynomials  
hence  $x^9 = x^8 \times x$  is replaced by  $x^5 + x^4 + x^2 + x$ , ...  
→ polynomials have degree at most 7



# Examples

- $0x5c + 0x2a = 0x76$

	byte	bit string	polynomial
	0x5c	01011100	$x^6 + x^4 + x^3 + x^2$
+	0x2a	00101010	$x^5 + x^3 + x$
=			$x^6 + x^5 + x^4 + 2x^3 + x^2 + x$
=	0x76	01110110	$x^6 + x^5 + x^4 + x^2 + x$

- $0x9e \times 0x02 = 0x27$

	byte	bit string	polynomial
	0x9e	10011110	$x^7 + x^4 + x^3 + x^2 + x$
×	0x02	00000010	$x$
=			$x^8 + x^5 + x^4 + x^3 + x^2$
=			$x^5 + 2x^4 + 2x^3 + x^2 + x + 1$
=	0x27	00100111	$x^5 + x^2 + x + 1$

# GF Arithmetics

A byte  $a = a_7 \dots a_1 a_0$  represents an element of the finite field  $\text{GF}(2^8)$  as a polynomial  $a_0 + a_1.x + \dots + a_7.x^7$  modulo  $x^8 + x^4 + x^3 + x + 1$  and modulo 2

byte	bit string	polynomial
0x00	00000000	0
0x01	00000001	1
0x02	00000010	$x$
0x03	00000011	$x + 1$
0x1b	00011011	$x^4 + x^3 + x + 1$

**Addition:** a simple XOR

**Multiplication by 0x01:** nothing

**Multiplication by 0x02:** shift and XOR with 0x1b if carry

**Multiplication by 0x03:** XOR of multiplications by 0x01 and 0x02

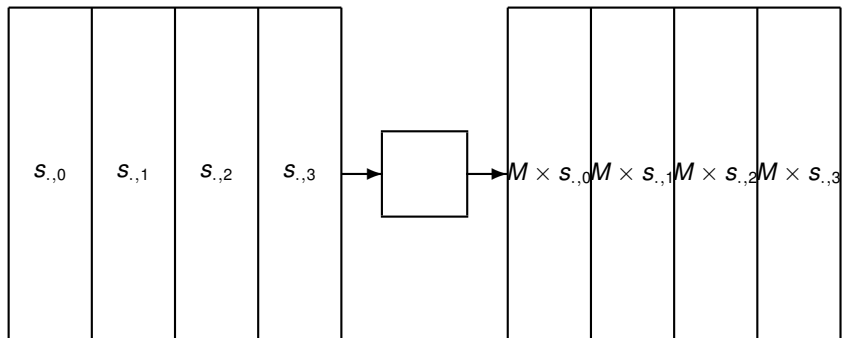
# MixColumns

## MixColumns( $s$ )

- 1: **for**  $i = 0$  to  $3$  **do**
- 2: let  $v$  be the 4-dimensional vector with coordinates  $s_{0,i} s_{1,i} s_{2,i} s_{3,i}$
- 3: replace  $s_{0,i} s_{1,i} s_{2,i} s_{3,i}$  by  $M \times v$
- 4: **end for**

$$M = \begin{pmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{pmatrix}.$$

# MixColumns



# InvMixColumns

## InvMixColumns( $s$ )

- 1: **for**  $i = 0$  to  $3$  **do**
- 2: let  $v$  be the 4-dimensional vector with coordinates  $s_{0,i} s_{1,i} s_{2,i} s_{3,i}$
- 3: replace  $s_{0,i} s_{1,i} s_{2,i} s_{3,i}$  by  $M^{-1} \times v$
- 4: **end for**

$$M^{-1} = \begin{pmatrix} 0x0e & 0x0b & 0x0d & 0x09 \\ 0x09 & 0x0e & 0x0b & 0x0d \\ 0x0d & 0x09 & 0x0e & 0x0b \\ 0x0b & 0x0d & 0x09 & 0x0e \end{pmatrix}.$$

# Key Expansion

- we consider  $W$  as a sequence of  $4(Nr + 1) = 44$  (resp. 52, 60) rows (32-bit words)  $w$
- we consider the key as a sequence of  $Nk = 4$  (resp. 6, 8) rows
- the  $w_i$  are iteratively loaded:
  - the first  $w_i$  are loaded with the key
  - $w_i$  is loaded with  $w_{i-Nk} \oplus w_{i-1}$
  - every  $Nk$  iterations, the  $w_i$  is modified before the XOR
  - for  $Nk = 8$ , we add an extra modification

# Key Expansion

**KeyExpansion**(key, Nk)

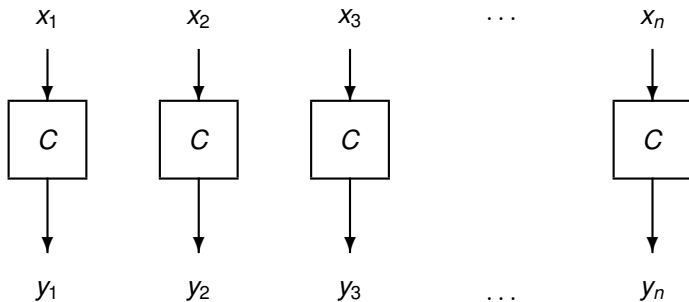
- 1: **for**  $i = 0$  to  $Nk - 1$  **do**
- 2:    $w_i \leftarrow \text{key}_i$
- 3: **end for**
- 4: **for**  $i = Nk$  to  $4(Nr + 1) - 1$  **do**
- 5:    $t \leftarrow w_{i-1}$
- 6:   **if**  $i \bmod Nk = 0$  **then**
- 7:     replace  $[t_1, t_2, t_3, t_4]$  by  $[t_2, t_3, t_4, t_1]$  in  $t$
- 8:     apply S-box to the four bytes of  $t$
- 9:     XOR  $x^{i/Nk-1}$  (in GF) onto the first byte of  $t$
- 10:   **else if**  $Nk = 8$  and  $i \bmod Nk = 4$  **then**
- 11:     apply S-box to the four bytes of  $t$
- 12:   **end if**
- 13:    $w_i \leftarrow w_{i-Nk} \oplus t$
- 14: **end for**

# Modes of Operation

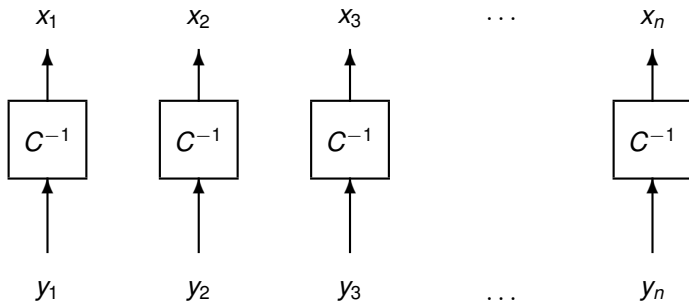
- transform a block cipher into a symmetric encryption with variable message length
- encrypt and decrypt “on the fly” (online encryption)
- in some sense: transform a block cipher into a stream cipher
- may require an **Initialization Vector** (IV)
- typically: message length must be multiple of the block length



# ECB Mode



# ECB Decryption





# ECB vs CBC

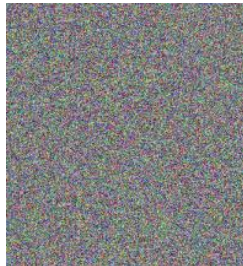
original



ECB

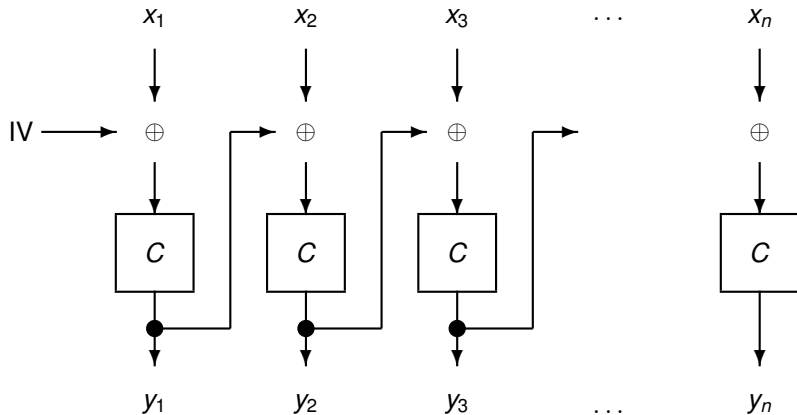


CBC

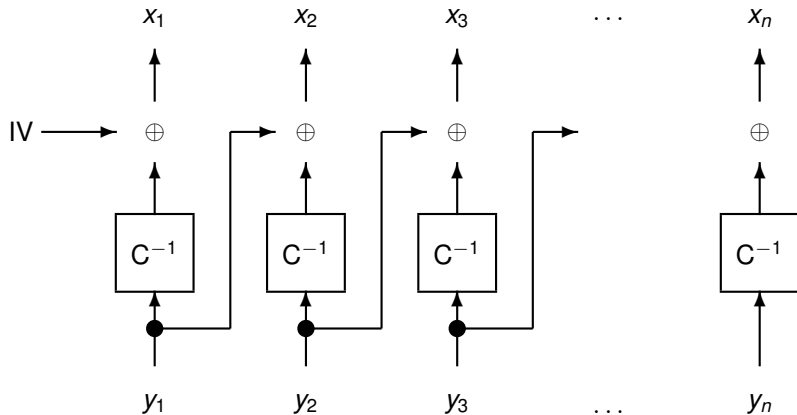


[en.wikipedia.org](http://en.wikipedia.org)

# CBC Mode



# CBC Decryption

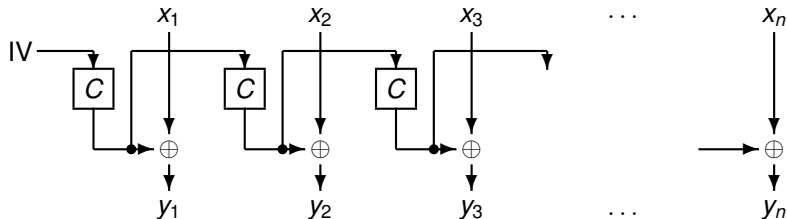


# Note on the CBC Mode

Three possibilities for dealing with IV

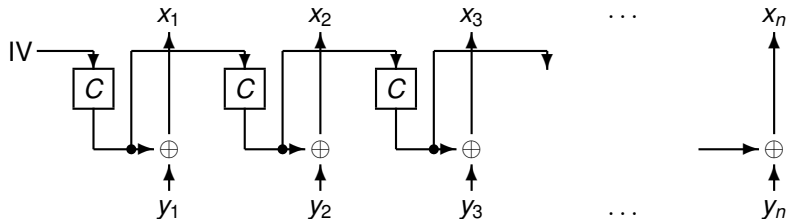
- Using a (non secret) constant IV  
example: **MRTD** (IV= 0)
- Using a secret IV which is part of the key  
example: **TLS**
- Using a random IV which is sent in clear with the ciphertext

# OFB Mode





# OFB Decryption

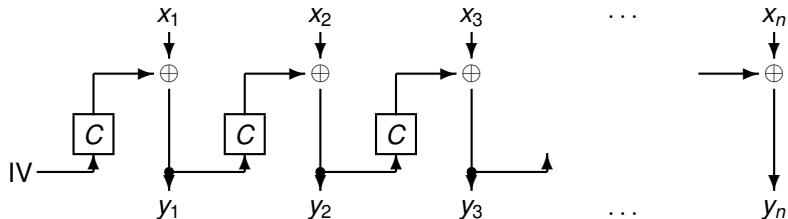


## Note on the OFB Mode

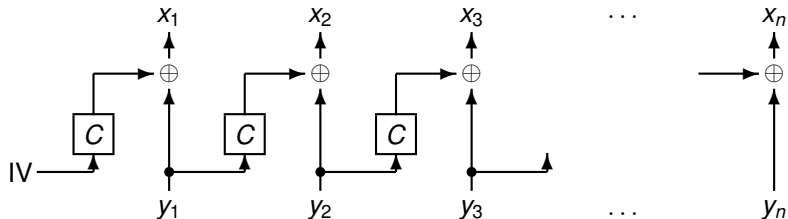
- IV must be new for every plaintext!
- Use a random one which is sent in clear...
- ... or use a counter-based IV
- This is not only a property of the OFB mode:  
property of stream ciphers
- OFB actually transforms a block cipher into a stream cipher

IV is used as a **nonce** (number used once)

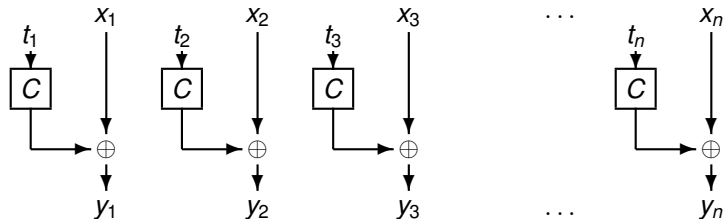
# CFB Mode



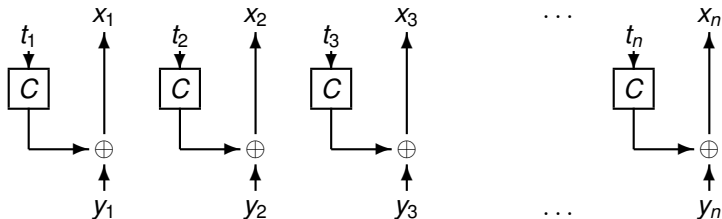
# CFB Decryption



# CTR Mode



# CTR Decryption



# Note on the CTR Mode

- $t_i$  must be new for every block! (a nonce)
  - Example 1:  $t_i = \text{msg counter} \parallel \text{blk counter}$
  - Example 2:  $t_i = t_1 + (i - 1)$  where  $t_1$  is the last  $t_n$  plus 1
  - Example 3:  $t_i = t_1 + (i - 1)$  where  $t_1$  is a (unique) nonce
- CTR also transforms a block cipher into a stream cipher

# XTS Mode

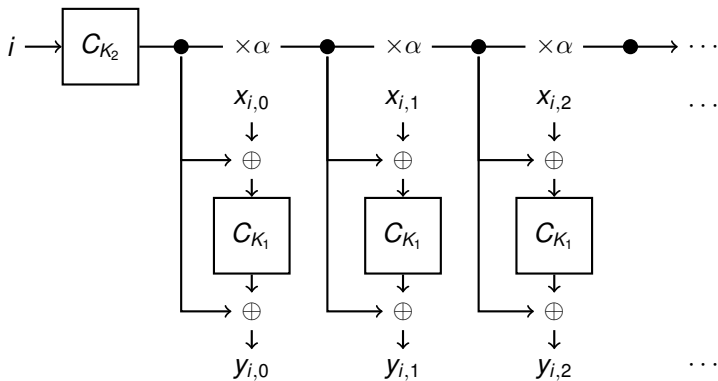
- used to encrypt a hard disk
- hard disks are made of “sectors” of various lengths  
length may not be a multiple of the block length
- requirements:  
encryption shall not increase space  
encryption shall allow random access with small overhead
- uses two keys ( $K_1, K_2$ )
- for a block of index  $j$  in sector of index  $i$ :

$$y_{i,j} = \text{Enc}_{i,j}(x_{i,j}) = C_{K_1}(x_{i,j} \oplus t_{i,j}) \oplus t_{i,j} \quad t_{i,j} = \alpha^j \times C_{K_2}(i)$$

in a GF structure, with a constant  $\alpha$

- use **ciphertext stealing** for the last two blocks





# Ciphertext Stealing

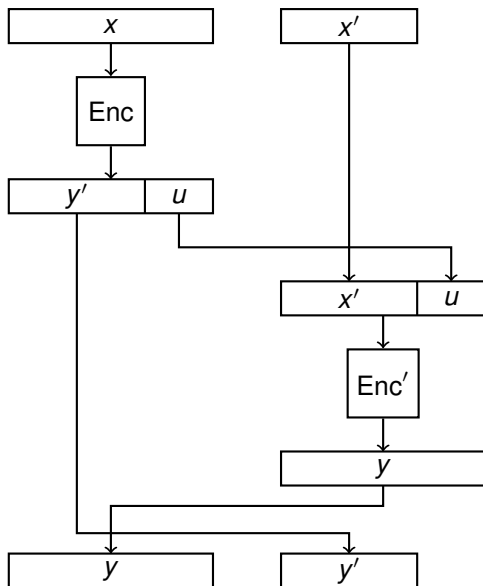
- used to encrypt two blocks  $x$  and  $x'$  (typically, the last two)
- Case 1 (easy): if  $x$  and  $x'$  have regular length, encrypt normally  
 $y = \text{Enc}(x)$ ,  $y' = \text{Enc}'(x')$
- Case 2: if  $x'$  is shorter than usual.

- 1: split  $\text{Enc}(x) = y' \| u$  with  $y'$  of same length as  $x'$
- 2:  $y = \text{Enc}'(x' \| u)$
- 3: give  $y$  and  $y'$

to decrypt  $y$  and  $y'$ :

- 1: split  $\text{Dec}'(y) = x' \| u$  with  $x'$  of same length as  $y'$
- 2:  $x = \text{Dec}(y' \| u)$
- 3: give  $x$  and  $x'$

# Ciphertext Stealing



# To Be Known About Modes of Operation

- ECB should be avoided
- CBC (very popular) requires IV
- OFB (stream cipher) requires a nonce
- CTR (stream cipher) requires a nonce

# Classical Skeletons for Block Ciphers

- Feistel schemes  
...and extensions  
DES, 3DES, BLOWFISH, KASUMI
- Lai-Massey scheme  
IDEA, FOX
- Substitution-permutation network (SPN)  
SAFER, CS-Cipher, AES

# Block Ciphers Characteristics

cipher	release	block	key	# rounds	comment
DES	1977	64	56	16	secretly developed
3DES	1985	64	112,168	48	pragmatic solution
IDEA	1990	64	128	8.5	
SAFER K-64	1993	64	64	6	
BLOWFISH	1994	64	0-448	16	
RC5	1996	2-256	0-255	0-255	64/128/12 recommended
CS-Cipher	1998	64	0-128	8	
AES	2001	128	128,192,256	10,12,14	dependent parameters
KASUMI	2002	64	128	8	dedicated
FOX	2003	64,128	0-256	12-255	

## Symmetric Encryption

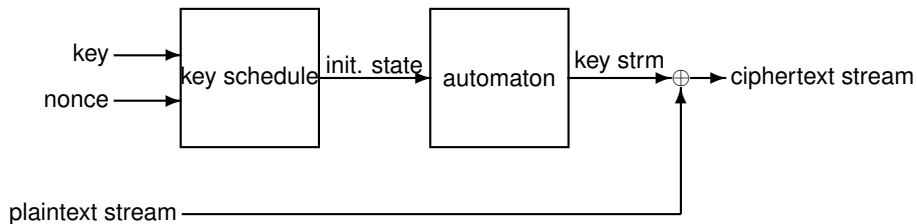
- A Cryptographic Primitive
- Block Ciphers
- **Stream Ciphers**
- Bruteforce Inversion Algorithms
- Subtle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

# Stream Ciphers

- adapt the Vernam cipher
- use a pseudorandom generator to generate a **key stream**  
the PRNG avoids having to store large secret keys
- seed the PRNG with a fixed secret key and a **nonce**: a *number*  
to be used only *once*  
the nonce avoids reuse of the same keystream
- variant 1: participants are synchronized to a nonce (e.g. a counter or the clock value)
- variant 2: the encrypting nonce is sent in clear with the ciphertext (asynchronous)



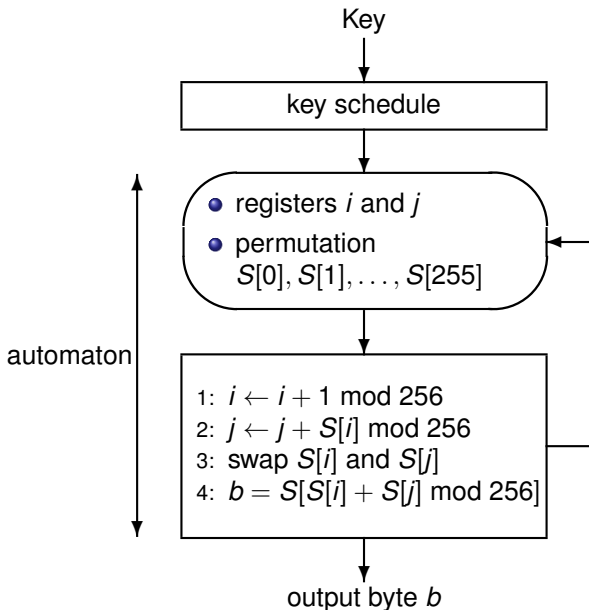
# Stream Ciphers from a High Level



# RC4

- Designed at MIT in 1987 by Ronald Rivest
- Trade secret of RSA Security Inc.
- illegally disclosed in 1994
- well known to be used in SSL/TLS
  
- dedicated to software on 8-bit microprocessors
- stream cipher with bytes streams
- key length from 40 bits to 256 ( $\ell = 5$  to 32 bytes)

# RC4 (Alleged)



# RC4 Key Schedule

```
1:  $j \leftarrow 0$ 
2: for  $i = 0$  to 255 do
3:    $S[i] \leftarrow i$ 
4: end for
5: for  $i = 0$  to 255 do
6:    $j \leftarrow j + S[i] + K[i \bmod \ell] \bmod 256$ 
7:   swap  $S[i]$  and  $S[j]$ 
8: end for
9:  $i \leftarrow 0$ 
10:  $j \leftarrow 0$ 
```

# RC4 in Security Protocols

- In SSL/TLS:
  - key is used only once
  - state is kept from one message to the other
- In WEP:
  - key is the concatenation of a 3-byte nonce (sent in clear) and a 5-byte key

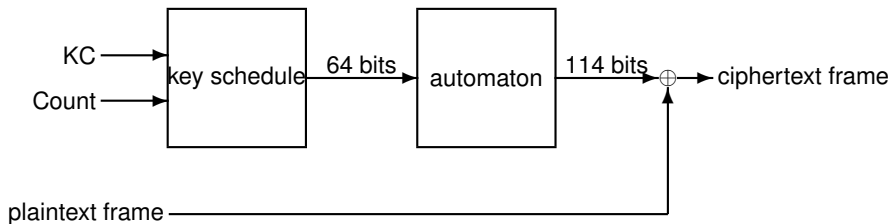
# Known Weaknesses

- some correlations between some output bytes and key bytes when the nonce is known
  - (passive) key recovery attack in WEP after seeing 22500 packets
- output bytes are not uniformly distributed
  - ciphertext-only decryption attacks in TLS if a plaintext is encrypted several times (e.g. secure http cookies)
- speculations that some state agencies can break RC4
- RC4 is now prohibited (RFC 7465 and similar recommendations)

# GSM A5/1

- Designed at ETSI by the SAGE group
- Trade secret of the GSM consortium
- reverse engineered
  
- dedicated to lightweight hardware
- stream cipher with bit streams
- 64-bit key and 22-bit counter

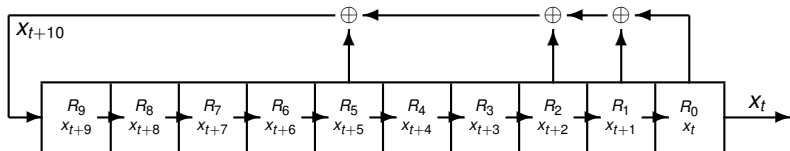
# A5/1 from a High Level





# Linear Feedback Shift Register (LFSR)

- when  $\text{CLK} = 1$ , increment  $t$ , load  $R_i$  with  $R_{i+1}$  and  $R_{d-1}$  with a XOR of some  $R_i$ 's

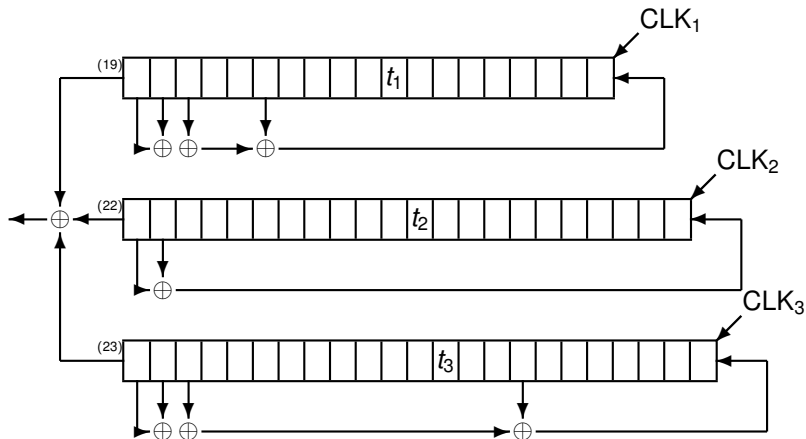


- at time  $t$ ,  $R_i = x_{t+i}$
- $x_{t+d} = a_{d-1}x_{t+d-1} \oplus \cdots \oplus a_0x_t$  for any  $t$  (linear recursion)
- $a_dx_{t+d} \oplus \cdots \oplus a_1x_{t+1} \oplus a_0x_t = 0$  for any  $t$  ( $a_d = 1$ )
- connection polynomial:  $a_dx^d + \cdots + a_1x + a_0$   
example:  $x^{10} + x^5 + x^2 + x + 1$
- maximal period  $\iff$  primitive polynomial  $\implies$  irreducible polynomial

# A5/1 Automaton

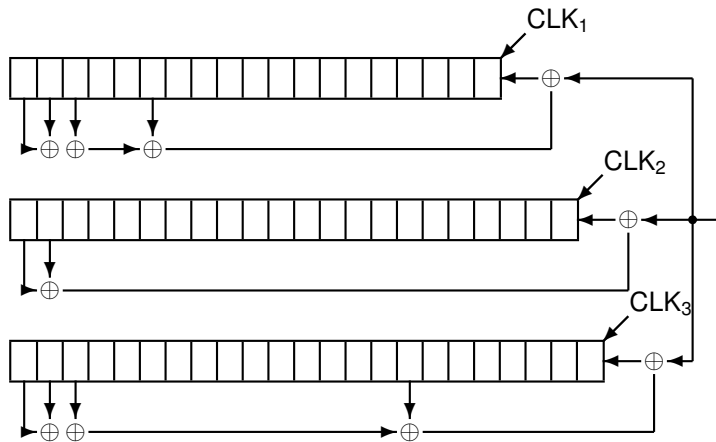
$$x^{19} + x^{18} + x^{17} + x^{14} + 1$$

$$x^{22} + x^{21} + 1$$

$$x^{23} + x^{22} + x^{21} + x^8 + 1$$


asynchronous:  $CLK_i = CLK$  if  $t_i = \text{majority}(t_1, t_2, t_3)$ , 0 otherwise

# A5/1 in Key Schedule



synchronous:  $CLK_1 = CLK_2 = CLK_3 = CLK$

# A5/1 Key Schedule

- 1: set all registers to zero
- 2: **for**  $i = 0$  to 63 **do**
- 3:    $R_1[0] \leftarrow R_1[0] \oplus \text{KC}[i]$
- 4:    $R_2[0] \leftarrow R_2[0] \oplus \text{KC}[i]$
- 5:    $R_3[0] \leftarrow R_3[0] \oplus \text{KC}[i]$
- 6:   clock registers (synchronous)
- 7: **end for**
- 8: **for**  $i = 0$  to 21 **do**
- 9:    $R_1[0] \leftarrow R_1[0] \oplus \text{Count}[i]$
- 10:    $R_2[0] \leftarrow R_2[0] \oplus \text{Count}[i]$
- 11:    $R_3[0] \leftarrow R_3[0] \oplus \text{Count}[i]$
- 12:   clock registers (synchronous)
- 13: **end for**
- 14: **for**  $i = 0$  to 99 **do**
- 15:   clock registers (asynchronous)
- 16: **end for**

# Known Weaknesses

- key recovery known plaintext attack  
(kind of time-memory tradeoff)
- active attacks on GSM (chosen cipher attack)
- ciphertext-only key recovery attack  
(optimized bruteforce)

## Symmetric Encryption

- A Cryptographic Primitive
- Block Ciphers
- Stream Ciphers
- **Bruteforce Inversion Algorithms**
- Subtle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

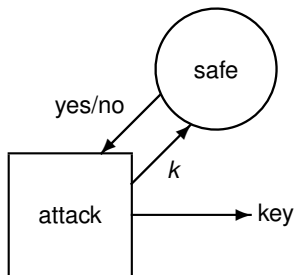
# The Random Key Guessing Game

Parameters: a set  $\mathcal{K}$ , a setup algorithm

- 1 (setup) the challenger runs the setup algorithm to select an element  $K \in \mathcal{K}$   
he may send some clue  $w$  to the adversary
- 2 (guessing) the adversary may send some chosen  $k$ 's to the challenger who would respond if  $k \neq K$
- 3 the adversary wins if  $K = k$

## Example: Opening a Safe (Online Attack)

For any  $k$ , we can ask the safe whether the key  $K$  is equal to  $k$





# Distribution Cases

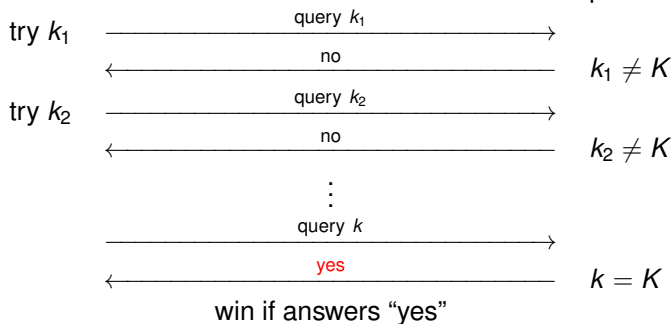
Setup selects  $K$  following a probability distribution

- $D$  is uniform
- $D$  is arbitrary
- $D$  is fixed and known to the adversary

# Key Recovery Game - Online with no Clue

**Adversary**

**Challenger**  
pick  $K \in_D \mathcal{K}$



# Exhaustive Search Algorithm (Uniform Case)

(online with no clue and  $D$  uniform)

**Input:** a set of possible keys  $\mathcal{K} = \{k_1, \dots, k_N\}$

**Challenger interface:** input is an element of  $\mathcal{K}$ ,  
output is Boolean

- 1: **for** all  $i = 1$  to  $N$  **do**
- 2:   **if** query  $k_i$  answers yes **then**
- 3:     yield  $k_i$  and stop
- 4:   **end if**
- 5: **end for**

$$\begin{aligned} E(\# \text{iterations}) &= \sum_{i=1}^N \Pr[K = k_i] i \\ &= \sum_{i=1}^N \frac{1}{N} i \\ &= \frac{N+1}{2} \end{aligned}$$

# Exhaustive Search Algorithm (Optimal Case)

(online with no clue and  $D$  known)

**Input:** a set of possible keys  $\mathcal{K} = \{k_1, \dots, k_N\}$

**Challenger interface:** input is an element of  $\mathcal{K}$ ,  
output is Boolean

- 1: take the permutation  $\sigma$  of  $\{1, \dots, N\}$  sorting  $k_{\sigma(i)}$  by decreasing order of likelihood
- 2: **for** all  $i = 1$  to  $N$  **do**
- 3:     **if** query  $k_{\sigma(i)}$  answers yes **then**
- 4:         yield  $k_{\sigma(i)}$  and stop
- 5:     **end if**
- 6: **end for**

$$E(\#iterations) = \min_{\sigma} \left( \sum_{i=1}^N \Pr[K = k_{\sigma(i)}] i \right)$$

which is sometimes called the **guesswork entropy** of  $D$

# Exhaustive Search Algorithm (Any Case)

(online with no clue)

**Input:** a set of possible keys  $\mathcal{K} = \{k_1, \dots, k_N\}$

**Challenger interface:** input is an element of  $\mathcal{K}$ ,  
output is Boolean

- 1: pick a random permutation  $\sigma$  of  $\{1, \dots, N\}$
- 2: **for** all  $i = 1$  to  $N$  **do**
- 3:   **if** query  $k_{\sigma(i)}$  answers yes **then**
- 4:     yield  $k_{\sigma(i)}$  and stop
- 5:   **end if**
- 6: **end for**

$$E(\#iterations) = \sum_{i=1}^N E(\Pr[K = k_{\sigma(i)}])i$$

since  $\sigma$  is random we have  $E(\Pr[K = k_{\sigma(i)}]) = \frac{1}{N}$  for all  $i$ :

$$E(\#iterations) = \sum_{i=1}^N \frac{1}{N}i = \frac{N+1}{2}$$

# Complexity Analysis (All Cases)

key of distribution  $D$  in a set of  $N$  elements

		number of iterations
worst case complexity		$N$
average complexity	$D$ unknown $D$ known	$\frac{N+1}{2}$ smaller

# Metrics of Algorithms

for comparing algorithms, we must look at:

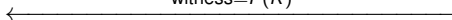
- precomputation time
- memory complexity
- time complexity
- number of online queries
- probability of success

# Key Recovery Game - Offline with a Clue

**Adversary**

**Challenger**  
pick a random  $K$

witness =  $F(K)$



⋮  
yield  $k$

win if  $k = K$



# Using Deterministic Clues

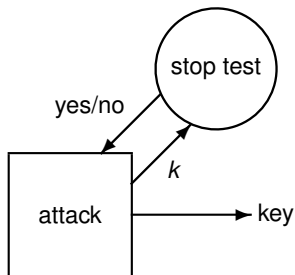
- **chosen plaintext attack:**  
get  $C_K(x)$  for some fixed  $x$  chosen by the adversary
- **password hash** (coming up in next slides)  
get  $C_K(x_0)$  for some constant  $x_0$  (e.g.  $x_0 = 0$ )

# Using Non-Deterministic Clues

- no chosen plaintext attack:  
**known plaintext attack** with random  $W = (x, C_K(x))$  pair  
**ciphertext only attack** with redundant plaintexts
- **randomized key hash**:  
instead of leaking  $C_K(x_0)$ , leak  $W = (F(K, \text{salt}), \text{salt})$  with **salt** randomly selected by the challenger

## More General Clues

We use a stop test function which tells whether the key candidate is consistent with the witness



Examples:

	witness	stop test
known plaintext attack	$W = (x, C_K(x))$	$C_k(W_1) = W_2$
ciphertext only attack	$W = \text{ciphertext}$	$C_k^{-1}(W)$ meaningful
salted key hash	$W = (F(K, \text{salt}), \text{salt})$	$F(k, W_2) = W_1$

# Password Recovery from a Password Hash

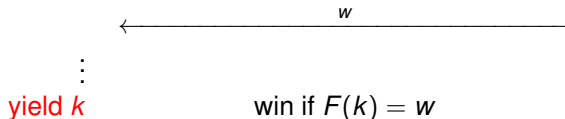
- **assumption:** at enrolment, the hash by  $F$  of the password is stored in a database  
to check a typed password, just hash it and compare with the hash in database
- remark: in this case, we do not care if the password is wrong; we just want one with the right hash to pass authentication  
→ the adversary has to find **one** password with correct hash (the problem is to **invert**  $F$ )

# Inversion (Preimage) Game

(assume a deterministic function  $F$ )

**Adversary**

**Challenger**



# Inversion by Exhaustive Search

**Input:** an image  $w$

- 1: shuffle  $\mathcal{K}$  with a random permutation
- 2: **for** all  $i = 1$  to  $N$  **do**
- 3:   **if**  $F(k_i) = w$  **then**
- 4:     yield  $k_i$  and stop
- 5:   **end if**
- 6: **end for**

If  $F$  is a uniformly distributed random function,  $\#\mathcal{X} = N$ ,  $\#\mathcal{Y} = M$ :

$$\Pr[\text{success}] = 1 - \left(1 - \frac{1}{M}\right)^M \approx 1 - e^{-\frac{N}{M}} \text{ for } N \gg M$$

$$\Pr[\text{complexity} > i] = \left(1 - \frac{1}{M}\right)^i$$

# Complexity of an Inversion Attack

$$\begin{aligned} E(\text{complexity}) &= \sum_{i=0}^{N-1} i \Pr[\text{complexity} = i] \\ &= \sum_{i=0}^{N-1} \Pr[\text{complexity} > i] \\ &= \sum_{i=0}^{N-1} x^i \quad \text{with } x = 1 - \frac{1}{M} \\ &= \frac{1 - x^N}{1 - x} \\ &\sim \frac{1 - e^{-\frac{N}{M}}}{1 - x} \quad \text{as } \frac{N}{M} \rightarrow +\infty \\ &= M \left(1 - e^{-\frac{N}{M}}\right) \\ &\approx M \text{ for } N \gg M \end{aligned}$$

# Dictionary Inversion Attack (Full Book)

(assume a deterministic function  $F$ )

## Preprocessing

**Input:** access to function  $F$

- 1: **for** all candidates  $K$  **do**
- 2:   compute  $F(K)$
- 3:   insert  $(F(K), K)$  in a dictionary
- 4: **end for**
- 5: output the dictionary

## Attack

**Input:** a witness  $w = F(K)$ , a dictionary

- 6: look at  $w$  in the dictionary
- 7: **for** all  $(w, K)$  in the dictionary **do**
- 8:   yield  $K$  and stop
- 9: **end for**



# Dictionary Inversion Attack (Smaller Dictionary)

(assume a deterministic function  $F$ )

## Preprocessing

**Input:** access to function  $F$

- 1: **for**  $D$  different candidates  $K$  **do**
- 2:   compute  $F(K)$
- 3:   insert  $(F(K), K)$  in a dictionary
- 4: **end for**
- 5: output the dictionary

## Attack

**Input:** a witness  $w = F(K)$ , a dictionary

- 6: look at  $w$  in the dictionary
- 7: **for** all  $(w, K)$  in the dictionary **do**
- 8:   yield  $K$  and stop
- 9: **end for**
- 10: search failed

# Complexity Analysis

**Precomputation time**  $D$

**Memory complexity**  $D$

**Time complexity**  $\approx 1$

**Probability of success** (with randomly selected dictionary keys)  
 $D/N$

# Summary of Single-Target Brute Force Attacks

strategy	preprocessing	memory	time	success proba.
exhaustive search	0	1	$N$	1
dictionary attack	$N$	$N$	1	1
tradeoffs	$N$	$N_{\text{opt}}^2$	$N_{\text{opt}}^2$	cte
partial ex. search	0	1	$D$	$D/N$
dictionary attack	$D$	$D$	1	$D/N$

# Application to DES

strategy	preprocessing	memory	time
exhaustive search	0	1	$2^{56}$
dictionary attack	$2^{56}$	$2^{56}$	1
tradeoffs	$2^{56}$	$2^{37}$	$2^{37}$

→ the key of DES is too short!

# Security of Passwords with less than 48 Bits of Entropy

An 8 i.u.d. random characters password in  $\{a, \dots, z, A, \dots, Z, 0, \dots, 9\}$  has less than 48 bits of entropy

- classical conventional cryptography may require about 300 cycles on a P4 2GHz to check a guess ( $= 2^{22.6}$  guesses per second)  
→ 256d to find a password with a PC
- time-memory tradeoffs cracked a (36-bit entropy) password within a few seconds (complexity  $N^{\frac{2}{3}}$  + precomputation  $N$ )  
→ 1h to find a password (+ a year of precomputation)
- special purpose hardwares cracked 56-bit keys within a day  
→ 5 min to find a password
- distributed.net cracked 64-bit keys in 2002 after 1757 days  
→ 40 min to find a password

# Extension: Multi-Target Dictionary Inversion Attack

(assume a deterministic function  $F$ )

## Preprocessing

**Input:** access to function  $F$

- 1: **for**  $D$  different candidates  $K$  **do**
- 2:   compute  $F(K)$
- 3:   insert  $(F(K), K)$  in a dictionary
- 4: **end for**
- 5: output the dictionary

## Attack

**Input:**  $T$  many witnesses  $w_i = F(K_i)$ , a dictionary

- 6: **for**  $i = 1$  to  $T$  **do**
- 7:   look at  $w_i$  in the dictionary
- 8:   **for** all  $(w_j, K)$  in the dictionary **do**
- 9:     yield  $i, K$
- 10:   **end for**
- 11: **end for**

# Complexity Analysis

**Precomputation time**  $D$

**Memory complexity**  $D$

**Time complexity**  $T$

**Probability of success**  $1 - \left(1 - \frac{D}{N}\right)^T \approx 1 - e^{-\frac{DT}{N}}$

This is quite interesting when  $D \approx T \approx \sqrt{N}...$

# The Role of Salt

mitigates dictionary attacks and tradeoffs

(makes dictionaries much larger)

best offline inversion attack with large enough salt:

**Input:** a set of possible keys  $\mathcal{K} = \{k_1, \dots, k_N\}$ , a salted witness  $W = (W_1, W_2)$  (salt is  $W_2$ )

**Challenger interface:** input is an element of  $\mathcal{K}$ , output is Boolean

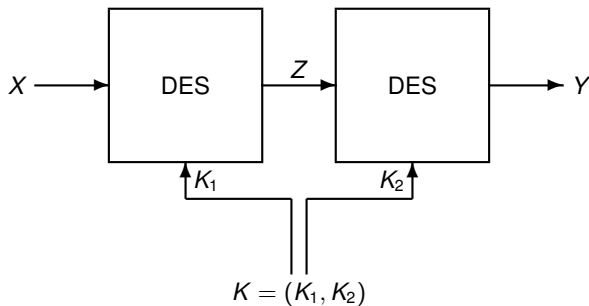
- 1: pick a random permutation  $\sigma$  of  $\{1, \dots, N\}$
- 2: **for** all  $i = 1$  to  $N$  **do**
- 3:   **if**  $F(k_{\sigma(i)}, W_2) = W_1$  **then**
- 4:     yield  $k_{\sigma(i)}$  and stop
- 5:   **end if**
- 6: **end for**
- 7: search failed



## Symmetric Encryption

- A Cryptographic Primitive
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- **Subtle Bruteforce Inversion Algorithms**
- Pushing the Physical Limits
- Formalism

# Double DES



this is not much more secure than single DES

# Meet-in-the-Middle Attack

**Input:** two encryption schemes  $C'$  and  $C''$  with two corresponding sets of possible keys  $\mathcal{K}'$  and  $\mathcal{K}''$ , an  $(x, y)$  pair with  $y = C''_{k_2}(C'_{k_1}(x))$

- 1: **for** all  $k_1 \in \mathcal{K}'$  **do**
- 2:   compute  $z = C'_{k_1}(x)$
- 3:   insert  $(z, k_1)$  in a hash table (indexed with the first entry)
- 4: **end for**
- 5: **for** all  $k_2 \in \mathcal{K}''$  **do**
- 6:   compute  $z = C''_{k_2}^{-1}(y)$
- 7:   **for** all  $(z, k_1)$  in the hash table **do**
- 8:     yield  $(k_1, k_2)$  as a possible key
- 9:   **end for**
- 10: **end for**

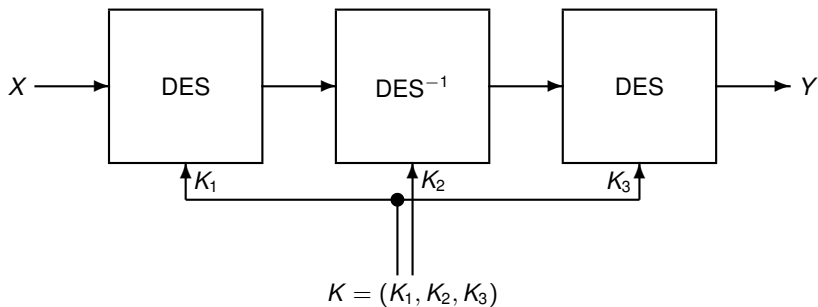
# Complexity Analysis

**Memory complexity**  $\#\mathcal{K}'$  ( $2^{56}$  for double DES)

**Time complexity**  $\#\mathcal{K}' + \#\mathcal{K}''$  ( $2^{57}$  for double DES)

**Probability of success** 1

# Triple DES



- 3-key triple DES:  $K_1, K_2, K_3$
- 2-key triple DES:  $K_1 = K_3, K_2$
- DES:  $K_1 = K_2 = K_3$

# Generic Attacks on Triple DES

## 2 keys

- key length: 112
- chosen plaintext ( $\times 2^{56}$ ):  
time complexity  $2^{57}$   
memory complexity  $2^{57}$   
[Merkle-Hellman 1981]  
[exercise 2.5 in exercise book]
- known plaintext ( $\times 2^{32}$ ):  
time complexity  $2^{88}$   
memory complexity  $2^{57}$   
[van Oorschot-Wiener 1990]

## 3 keys

- key length: 168
- known plaintext ( $\times 3$ ):  
time complexity  $2^{113}$   
memory complexity  $2^{56}$   
[meet-in-the-middle]

# Time-Memory Tradeoffs — i

**Input:** a deterministic function  $F$

**Parameter:**  $\ell, m, t$

**Preprocessing**

- 1: **for**  $s = 1$  to  $\ell$  **do**
- 2:   pick a reduction function  $R_s$  at random and  
   define  $f_s : k \mapsto R_s(F(k))$
- 3:   **for**  $i = 1$  to  $m$  **do**
- 4:     pick  $k'$  at random
- 5:      $k \leftarrow k'$
- 6:     **for**  $j = 1$  to  $t$  **do**
- 7:       compute  $k \leftarrow f_s(k)$
- 8:     **end for**
- 9:     insert  $(k, k')$  in table  $T_s$
- 10:   **end for**
- 11: **end for**

# Precomputed Tables

$$\begin{array}{l}
 T_1 : \\
 \left[ \begin{array}{c} k_{1,0}^1 \\ k_{2,0}^1 \\ k_{3,0}^1 \\ \vdots \\ k_{m,0}^1 \end{array} \right] \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \begin{array}{c} k_{1,1}^1 \\ k_{2,1}^1 \\ k_{3,1}^1 \\ \vdots \\ k_{m,1}^1 \end{array} \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \begin{array}{c} k_{1,2}^1 \\ k_{2,2}^1 \\ k_{3,2}^1 \\ \vdots \\ k_{m,2}^1 \end{array} \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \begin{array}{c} k_{1,3}^1 \\ k_{2,3}^1 \\ k_{3,3}^1 \\ \vdots \\ k_{m,3}^1 \end{array} \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \cdots \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \begin{array}{c} k_{1,t-1}^1 \\ k_{2,t-1}^1 \\ k_{3,t-1}^1 \\ \vdots \\ k_{m,t-1}^1 \end{array} \begin{array}{c} \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \xrightarrow{f_1} \\ \vdots \\ \xrightarrow{f_1} \end{array} \left[ \begin{array}{c} k_{1,t}^1 \\ k_{2,t}^1 \\ k_{3,t}^1 \\ \vdots \\ k_{m,t}^1 \end{array} \right] \Rightarrow \begin{array}{c} (k_{1,t}^1, k_{1,0}^1) \\ (k_{2,t}^1, k_{2,0}^1) \\ (k_{3,t}^1, k_{3,0}^1) \\ \vdots \\ (k_{m,t}^1, k_{m,0}^1) \end{array} \\
 \vdots \\
 T_\ell : \\
 \left[ \begin{array}{c} k_{1,0}^\ell \\ k_{2,0}^\ell \\ k_{3,0}^\ell \\ \vdots \\ k_{m,0}^\ell \end{array} \right] \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \begin{array}{c} k_{1,1}^\ell \\ k_{2,1}^\ell \\ k_{3,1}^\ell \\ \vdots \\ k_{m,1}^\ell \end{array} \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \begin{array}{c} k_{1,2}^\ell \\ k_{2,2}^\ell \\ k_{3,2}^\ell \\ \vdots \\ k_{m,2}^\ell \end{array} \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \begin{array}{c} k_{1,3}^\ell \\ k_{2,3}^\ell \\ k_{3,3}^\ell \\ \vdots \\ k_{m,3}^\ell \end{array} \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \cdots \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \begin{array}{c} k_{1,t-1}^\ell \\ k_{2,t-1}^\ell \\ k_{3,t-1}^\ell \\ \vdots \\ k_{m,t-1}^\ell \end{array} \begin{array}{c} \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \xrightarrow{f_\ell} \\ \vdots \\ \xrightarrow{f_\ell} \end{array} \left[ \begin{array}{c} k_{1,t}^\ell \\ k_{2,t}^\ell \\ k_{3,t}^\ell \\ \vdots \\ k_{m,t}^\ell \end{array} \right] \Rightarrow \begin{array}{c} (k_{1,t}^\ell, k_{1,0}^\ell) \\ (k_{2,t}^\ell, k_{2,0}^\ell) \\ (k_{3,t}^\ell, k_{3,0}^\ell) \\ \vdots \\ (k_{m,t}^\ell, k_{m,0}^\ell) \end{array}
 \end{array}$$



## Time-Memory Tradeoffs — ii

### Attack

**Attack input:**  $y = F(K)$

```
1: for  $s = 1$  to  $\ell$  do
2:   set  $i$  to 0
3:   set  $k$  to  $R_s(y)$ 
4:   while  $T_s$  contains no  $(k, \cdot)$  entry
      and  $i < t$  do
5:     increment  $i$ 
6:      $k \leftarrow f_s(k)$ 
7:   end while
8:   if  $T_s$  contains a  $(k, \cdot)$  entry then
9:     get the  $(k, k')$  entry from table
       $T_s$ 
10:    while  $F(k') \neq y$  and  $i < t$  do
11:      increment  $i$ 
12:       $k' \leftarrow f_s(k')$ 
13:    end while
14:    if  $F(k') = y$  then
15:      yield  $k'$  as a possible key
16:    end if
17:  end if
18: end for
```

# Complexity Analysis

**Precomputation time**  $\ell \times m \times t$

**Memory complexity**  $\ell \times m$

**Time complexity**  $\ell \times t$

**Probability of success** can be shown to be greater than  $\frac{1}{2}$  for

$$\ell \approx m \approx t \approx \sqrt[3]{N}$$

time and memory complexity of  $N^{\frac{2}{3}}$

## Symmetric Encryption

- A Cryptographic Primitive
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subtle Bruteforce Inversion Algorithms
- **Pushing the Physical Limits**
- Formalism

# Order of Magnitudes

for exhaustive search on a 128-bit key:

- # clock cycles needed to perform a typical cryptographic operation (encryption of one block): 300
- clock rate in 2007: 2GHz
- age of the universe: 15BY =  $15 \times 10^9 \text{Y} \approx 473 \times 10^{15} \text{s}$
- # machines to do the exhaustive search within 15BY:  $108 \times 10^{12}$



# Better Strategy (of Metaphysical Interest)

create the universe then take 15BY of vacations

humankind will create itself, invent computers, and solve the problem

# Energy Bill

- we can compute without burning energy! [Bennett 1973]  
need superconductors and invertible computation gates

but all computations must be invertible!

exhaustive search must keep lots of garbage in memory

- minimal energy spent to erase one bit:  $kT \ln 2$  [Landauer 1961]

$k = 1.38 \times 10^{-23} J/K$  (Boltzmann constant)

$T$ : absolute temperature (absolute 0 is  $-273C$ )

- example: assume we run an exhaustive search with  $2^{128}$  loops  
but we erase 128 bits per loop

assume the computer operates at  $3\mu K$  (very cold!)

energy bill:  $1.2 \times 10^9 J$

if we want to do it within 1s we need a 1 200MW nuclear powerplant

# Conclusion

- **symmetric encryption**: stream ciphers (RC4, A5/1), block ciphers (DES, AES), modes of operation (ECB, CBC, OFB, CFB, CTR, XTS)
- **bruteforce inversion** within complexity  $\mathcal{O}(\#\text{domain})$
- **tradeoffs** within complexity  $\mathcal{O}\left((\#\text{domain})^{\frac{2}{3}}\right)$  after precomputation with complexity  $\mathcal{O}(\#\text{domain})$



# Ciphers to Remember

cipher	release	block	key	design
DES	1977	64	56	Feistel scheme
3DES	1985	64	112,168	triple DES
RC4	1987	8	40–256	stream cipher
AES	2001	128	128,192,256	SPN

# Several Types of Symmetric Encryption

- **fixed message length** vs **variable message length**  
**block ciphers**: use fixed message length  
**modes of operation**: adapt to variable message length  
**stream ciphers**: encrypt messages “on-the-fly”
- **deterministic** vs **probabilistic**  
most common case for symmetric encryption: deterministic
- **synchronous** (stateful) vs **asynchronous** (stateless)
- **authenticating** or not (not in this chapter)

# Stream Ciphers vs Block Ciphers

stream cipher	block cipher
<ul style="list-style-type: none"><li>● small granularity (encrypt bits or bytes)</li><li>● based on the Vernam cipher, requires a <i>nonce</i> (number to be used only once)</li><li>● very high speed rate, very cheap on hardware</li><li>● low confidence on security</li></ul>	<ul style="list-style-type: none"><li>● large granularity (encrypt blocks of 64 or 128 bits), require padding techniques for messages with arbitrary length</li><li>● high rate, nice for software implementation, can be adapted to various platforms (8-bit, 32-bit, or 64-bit microprocessors)</li><li>● well established security</li></ul>

## Symmetric Encryption

- A Cryptographic Primitive
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subtle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

# Block Cipher

## Definition

A **block cipher** is a tuple  $(\{0, 1\}^k, \{0, 1\}^n, \text{Enc}, \text{Dec})$  with a key domain  $\{0, 1\}^k$ , a block domain  $\{0, 1\}^n$ , and two efficient deterministic algorithms Enc and Dec. It is such that

$$\forall K \in \{0, 1\}^k \quad \forall X \in \{0, 1\}^n \quad \text{Dec}(K, \text{Enc}(K, X)) = X$$

Write  $C_K(\cdot) = \text{Enc}(K, \cdot)$  and  $C_K^{-1}(\cdot) = \text{Dec}(K, \cdot)$ .

(operate on bitstrings)

# Variable-Length Symmetric Encryption

## Definition

A **(variable-length, length-preserving) symmetric encryption scheme** is a tuple  $(\{0, 1\}^k, \mathcal{D}, \text{Enc}, \text{Dec})$  with a key domain  $\{0, 1\}^k$ , a plaintext domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , and two efficient deterministic algorithms Enc and Dec.

It is such that

$$\forall K \in \{0, 1\}^k \quad \forall X \in \mathcal{D} \quad \begin{cases} \text{Dec}(K, \text{Enc}(K, X)) & = & X \\ |\text{Enc}(K, X)| & = & |X| \end{cases}$$

Write  $C_K(\cdot) = \text{Enc}(K, \cdot)$  and  $C_K^{-1}(\cdot) = \text{Dec}(K, \cdot)$ .

→ can be made from block ciphers using a mode of operation

# Nonce-Based Symmetric Encryption

## Definition

A (**nonce-based, variable-length, length-preserving**) **symmetric encryption scheme** is a tuple  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  with a key domain  $\{0, 1\}^k$ , a plaintext domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , a nonce domain  $\mathcal{N}$ , and two efficient deterministic algorithms Enc and Dec.

It is such that

$$\forall K \in \{0, 1\}^k \quad \forall X \in \mathcal{D} \quad \forall N \in \mathcal{N} \quad \left\{ \begin{array}{l} \text{Dec}(K, N, \text{Enc}(K, N, X)) = X \\ |\text{Enc}(K, N, X)| = |X| \end{array} \right.$$

$N$  is supposed to be used only once for encryption  
random nonce (beware of random repetitions), counter, sent in clear  
or synchronized

→ could be a mode of operation (IV...), a stream cipher

# Security against Key Recovery

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is  $(q, t, \varepsilon)$ -**secure against key recovery under chosen plaintext attacks** if for any nonce-respecting probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots)} \rightarrow K] \leq \varepsilon$$

where  $K \in \{0, 1\}^k$  is random.

It is  $(q, t, \varepsilon)$ -**secure against key recovery under chosen plaintext/ciphertext attacks** if for any similar  $\mathcal{A}$ ,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots), \text{Dec}(K, \dots)} \rightarrow K] \leq \varepsilon$$

(nonce-respecting:  $\mathcal{A}$  is not allowed to make two encryption queries with the same nonce; it is ok to repeat nonces for decryption queries)



# CCA Security is Stronger than CPA Security

If  $\mathcal{A}^{\text{Enc}(K, \dots)}$  is a CPA adversary, we can define it as  $\mathcal{A}^{\text{Enc}(K, \dots), \text{Dec}(K, \dots)}$  but making no use of  $\text{Dec}(K, \dots)$ .

So,

CPA-breaking  $\implies$  CCA-breaking

So,

CCA-secure  $\implies$  CPA-secure

# Security against Decryption

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is  $(q, t, \epsilon)$ -**secure against decryption under chosen plaintext attacks** if for any nonce-respecting probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots)}(N, \text{Enc}(K, N, X)) \rightarrow X] \leq \epsilon$$

where  $K \in \{0, 1\}^k$ ,  $N \in \mathcal{N}$ , and  $X \in \mathcal{D}$  are random. It is  $(q, t, \epsilon)$ -**secure against decryption under chosen plaintext/ciphertext attacks** if for any similar  $\mathcal{A}$ ,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots), \text{Dec}(K, \dots)}(N, \text{Enc}(K, N, X)) \rightarrow X] \leq \epsilon$$

( $\mathcal{A}$  is not allowed to query the decryption oracle with its input  $(N, \text{Enc}(K, N, X))$ )

# Decryption Security is Stronger than Key Recovery Security

If  $\mathcal{A}$  is a key recovery adversary, we can define

- 1: run  $\mathcal{A} \rightarrow K$
- 2: compute  $X' = \text{Dec}(K, N, Y)$
- 3: return  $X'$

So,

key recovery-breaking  $\implies$  decryption-breaking

So,

decryption-secure  $\implies$  key recovery-secure

# Not Good Enough Security

- some parts of the plaintext may be more private than others  
how about a cipher letting half of the plaintext in clear and strongly encrypting the other half?  
it would be secure against decryption
- the “ideal cipher”: given  $K$ , for each  $N$ , we pick a random permutation  $\Pi_N$  and define

$$\text{Enc}(K, N, X) = \Pi_N(X)$$

- security would mean that we cannot tell the real cipher and the ideal one apart from a black-box usage

# Security against Distinguisher

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is  $(q, t, \varepsilon)$ -**secure under chosen plaintext attacks** if for any nonce-respecting probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots)} \rightarrow 1] - \Pr[\mathcal{A}^{\Pi(\dots)} \rightarrow 1] \leq \varepsilon$$

where  $K \in \{0, 1\}^k$  is random and  $\Pi(N, \cdot)$  is a random length-preserving permutation over  $\mathcal{D}$  for every  $N$ .

It is  $(q, t, \varepsilon)$ -**secure under chosen plaintext/ciphertext attacks** if for any similar  $\mathcal{A}$ ,

$$\Pr[\mathcal{A}^{\text{Enc}(K, \dots), \text{Dec}(K, \dots)} \rightarrow 1] - \Pr[\mathcal{A}^{\Pi(\dots), \Pi^{-1}(\dots)} \rightarrow 1] \leq \varepsilon$$

# Security Notions

	<b>key recovery</b>	<b>decryption</b>	<b>distinguisher</b>
<b>CPA</b>	weaker security		
<b>CCA</b>			stronger security

- if we can recover the key, we can decrypt
- if we can decrypt, we can recognize from the ideal cipher
- if we can break without chosen ciphertext, we can also break with

# References

- **Schneier.** *Applied Cryptography*. Wiley & Sons. 1996.  
Crypto for dummies!
- **Ferguson–Schneier.** *Practical Cryptography*. Wiley & Sons.  
2003.  
Crypto for dummies!

# Must be Known

- types of symmetric encryption
- parameters of block ciphers: DES, 3DES, AES
- modes of operation: ECB, CBC, OFB, CTR
- Feistel scheme
- parameters of stream ciphers: RC4
- exhaustive search
- meet-in-the-middle



# Train Yourself

- encryption:  
final exam 2013–14 ex1  
midterm exam 2012–13 ex3
- modes of operation:  
midterm exam 2009–10 ex3  
midterm exam 2011–12 ex1
- Moore's law:  
midterm exam 2008–09 ex1
- multitarget password recovery:  
final exam 2014–15 ex3















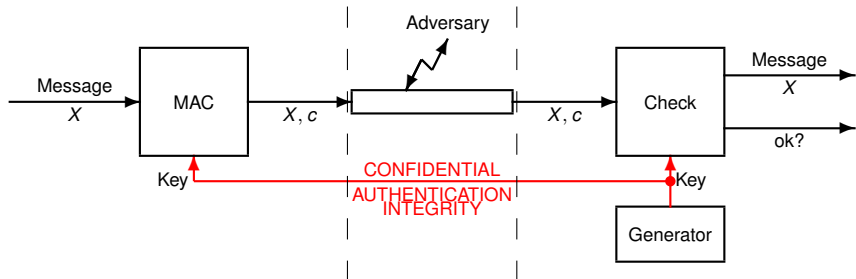


- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication**
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

# Roadmap

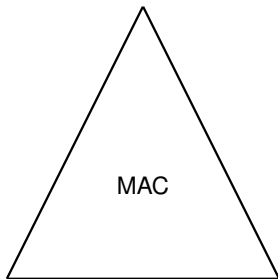
- hash functions: MD5, SHA-1
- message authentication codes: HMAC, CBCMAC, WC-MAC
- other primitives: commitment, key derivation
- birthday paradox

# Message Authentication Code



# Message Authentication Code (Informal)

Alice and Bob, Generator, MAC, Check  
**components**



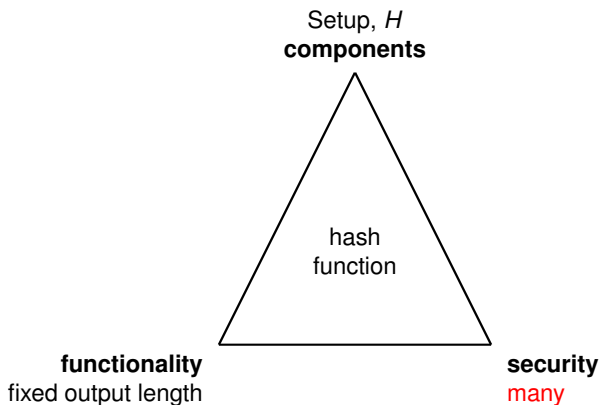
**functionality**

$\text{Check}_K(\text{MAC}_K(X)) = (X, \text{ok})$

**security**

cannot forge

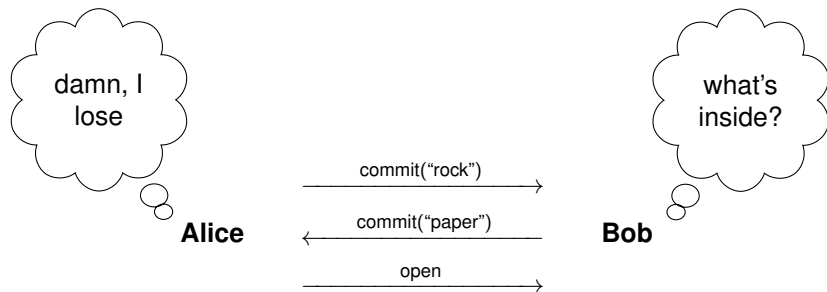
# Hash Function (Informal)



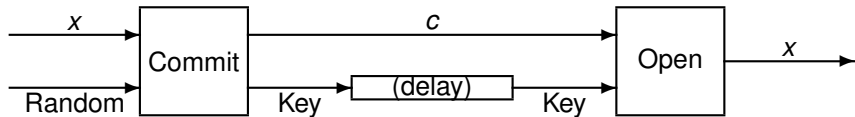
## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses

# Commitment to Play Rock-Paper-Scissors



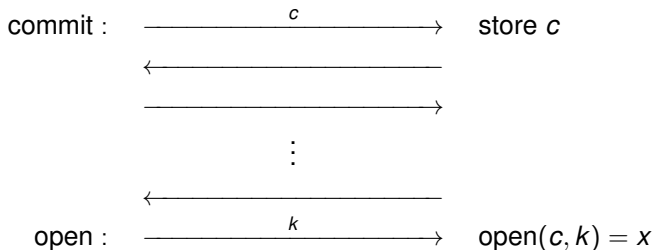
# Commitment





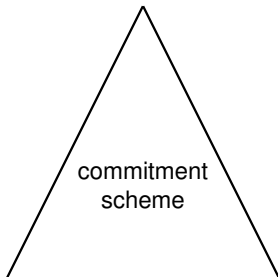
# Using a Commitment Scheme

pick  $r$  at random  
 $(c, k) \leftarrow \text{Commit}(x; r)$



# Commitment Scheme (Informal)

Alice and Bob, Setup, Commit, Open  
**components**



**functionality**

if  $\text{Commit}(X; r) = (c, k)$   
then  $\text{Open}(c, k) = X$

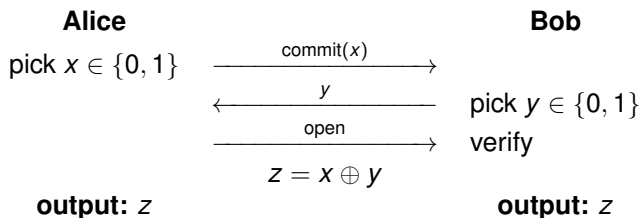
**security**

hiding, binding

- hiding: Bob does not get a clue on  $X$  from  $c$
- binding: Alice cannot produce  $c, k, k'$  such that  $\text{Open}(c, k) \neq \text{Open}(c, k')$

# Application Example: Tossing a Coin

how to toss a coin:



$z$  is the outcome of the tossed coin

# Application Example: Playing Dices

how to throw a 6-face die:

**Alice**

pick  $x \in \{1, \dots, 6\}$

commit( $x$ )

$y$

open

$$z = 1 + ((x + y) \bmod 6)$$

**output:  $z$**

**Bob**

pick  $y \in \{1, \dots, 6\}$

verify

**output:  $z$**

$z$  is the outcome of the thrown die

# Several Types of Commitment Schemes

- interactive vs non-interactive
- perfectly/statistically/computationally hiding
- perfectly/statistically/computationally binding
- using a common reference string or not

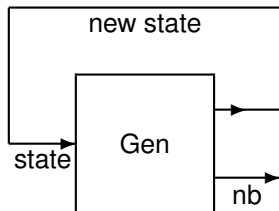
# Examples

- a **BAD one**:  $\text{Commit}(x; r) = (\text{Enc}_r(x), (x, r))$   
(not binding)
- a **BAD one**:  $\text{Commit}(x; r) = (H(x), x)$   
(not hiding)
- a not-too-bad one:  $\text{Commit}(x; r) = (H(r||x), (x, r))$   
(problem: most likely,  $H$  was not designed for that)

## Integrity and Authentication

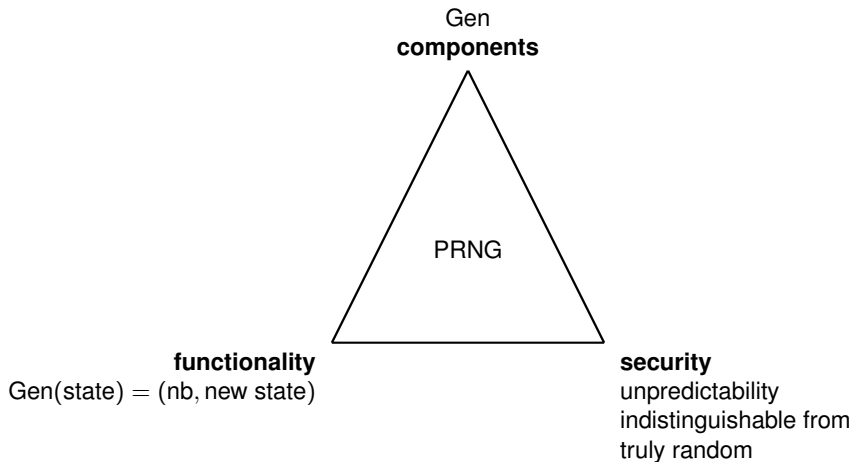
- Commitment Scheme
- **Key Derivation Function and Pseudorandom Generator**
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses

# Pseudorandom Number Generator (PRNG)





# PRNG (Informal)



# PRNG Examples

- stream ciphers: RC4, A5/1...
- block ciphers with OFB or CTR mode of operation
- finite automaton with an internal state (time, key, Seed)  
(time is updated by hardware)

$$J = \text{Enc}_{\text{key}}(\text{time})$$

$$r = \text{Enc}_{\text{key}}(J \oplus \text{Seed})$$

and the seed is replaced by

$$\text{NextSeed} = \text{Enc}_{\text{key}}(J \oplus r)$$

and the output is  $r$

# Famous Failure Cases

- early version of SSL (Goldberg-Wagner 1996):  
initial seed computed from the time in microseconds and the pid and ppid numbers (not enough entropy)

<http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html>

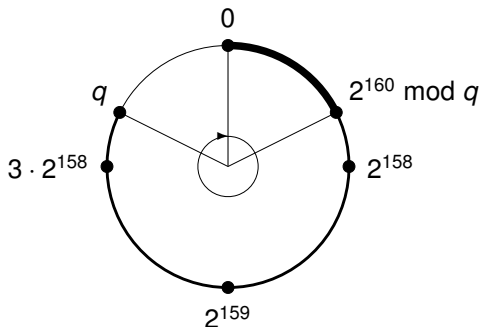
- Debian OpenSSL implementation until 2008:  
initial seed computed from the pid (15 bits) (other randomness removed due to complains by the compiler purify tool)

<http://metasploit.com/users/hdm/tools/debian-openssl/>

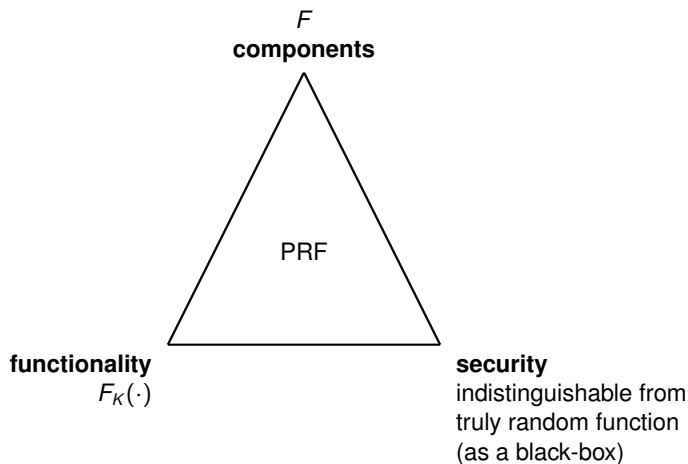
```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
             // guaranteed to be random.  
}
```

## Other Famous Failure Case

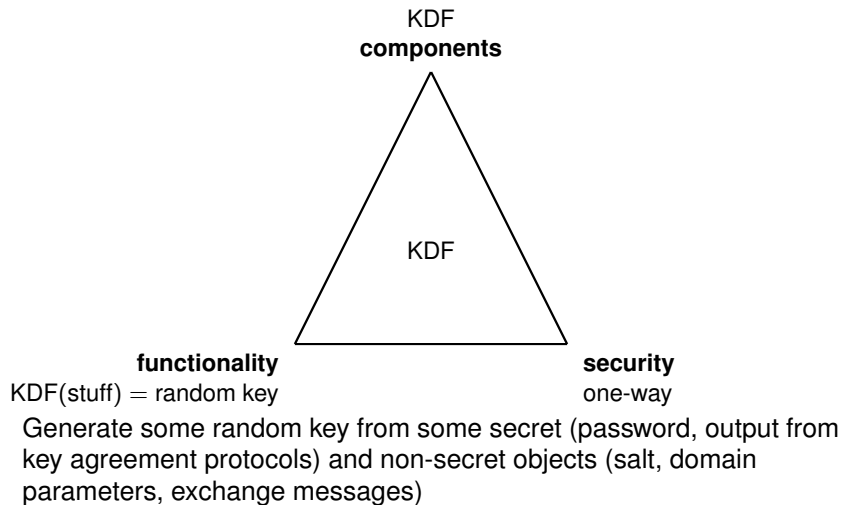
- DSA (Bleichenbacher 2001): the 160-bit random number was reduced modulo a 160-bit prime number  $q$  so that the final distribution was biased



# Pseudorandom Function (PRF)



# Key Derivation Function (KDF)



# KDF Examples

- typically: a standard hash function (MD5, SHA-1, ...)
- PKCS#5/RFC 2898  
example:

$$\text{PBKDF1}(\text{password}, \text{salt}, c, \ell) = \text{trunc}_{\ell}(H^c(\text{password} \parallel \text{salt}))$$

where  $H^c$  is  $H$  iterated  $c$  times

NB:  $\ell$  shall not be larger than the  $H$  length

- HKDF (RFC 5869)

$$\text{HKDF}(\text{salt}, \text{input}, \text{extra}, L) = K_1 \parallel K_2 \parallel \dots \parallel \text{trunc} \left( K_{\lceil \frac{L}{\text{HMAC.length}} \rceil} \right)$$

$$\text{PRK} = \text{HMAC}_{\text{salt}}(\text{input})$$

$$K_1 = \text{HMAC}_{\text{PRK}}(\text{extra} \parallel 0)$$

$$K_{i+1} = \text{HMAC}_{\text{PRK}}(K_i \parallel \text{extra} \parallel i)$$

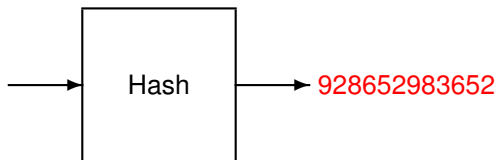
## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- **Cryptographic Hash Function**
- Message Authentication Codes
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses



# Cryptographic Hashing

La cigale ayant  
chanté tout l'été  
se trouva fort  
dépourvue quand  
la bise fut venue  
pas un seul petit  
morceau de mouche  
ou de vermisseau  
elle alla trouver  
famine chez la  
fourmie sa voisine ...



- can hash a string of arbitrary length
- produce digests (hashes) of standard length (e.g. 160 bits)

# A Swiss Army Knife Cryptographic Primitive

**Domain expander:** hash bitstrings of arbitrary length into bitstrings of fixed length.

Application: instead of specifying digital signature algorithms on set of bitstring with arbitrary length, we specify them with bitstrings of fixed length and use the hash-and-sign paradigm.

**Commitment:** “uniquely” characterizes a bitstring without revealing information on it.

Application: commitment which is binding and hiding.

**Pseudorandom generator:** generate bitstrings from seeds which are unpredictable.

Application: generation of cryptographic keys from a seed.

# Constructing Other Primitives with Hash Functions

- commitment:

$$\text{Commit}(X; \text{random}) = (H(\text{Key}), \text{Key} = X \parallel \text{random})$$

$$\text{Open}(c, X \parallel \text{random}) = \begin{cases} X & \text{if } H(X \parallel \text{random}) = c \\ \perp & \text{otherwise} \end{cases}$$

- PRNG:

$$\text{generation} = H(\text{seed} \parallel \text{counter})$$

- KDF:

$$\text{seed} \longrightarrow \text{trunc}(H(\text{seed} \parallel 1) \parallel H(\text{seed} \parallel 2) \parallel H(\text{seed} \parallel 3) \parallel \dots)$$

- domain extender for authentication (MAC or signature):

$$\text{Authenticate}(H(X))$$

# Security Properties for Hash Functions

**Collision resistance:** hash function  $h$  for which it is hard to find  $x$  and  $x'$  such that  $h(x) = h(x')$  and  $x \neq x'$ .

→ *digital fingerprint of the bitstring*

**One-wayness:** hash function  $h$  for which given  $y$  it is hard to find even one  $x$  such that  $y = h(x)$ .

→ *witness for a password*

**Pseudo-randomness** : hash function  $h$  such that for any given  $f$  and  $g_i = h(f^i(x))$  for  $i = 0, \dots, n - 1$  with a random (unknown)  $x$  such that  $f^i(x)$  is not cycling, it is hard to predict  $h(f^n(x))$ .

→ *pseudo-random generation*

# Threat Models for Hash Functions

**Collision attack:** find  $x$  and  $x'$  such that  $x \neq x'$  and  $h(x) = h(x')$ .

**First preimage attack:** given  $y$  find  $x$  such that  $y = h(x)$ .

**Second preimage attack:** given  $x$  find  $x'$  such that  $x \neq x'$  and  $h(x) = h(x')$ .

# Bruteforce First Preimage Attack

**Input:** access to a hash function  $h$ , an image  $y$

**Output:**  $x$  such that  $h(x) = y$

1: pick a random ordering of all inputs  $x_1, x_2, \dots$

2: **for all**  $i$  **do**

3:     compute  $h(x_i)$

4:     **if**  $h(x_i) = y$  **then**

5:         yield  $x = x_i$  and stop

6:     **end if**

7: **end for**

8: search failed

# Bruteforce Second Preimage Attack

**Input:** access to a hash function  $h$  onto a domain of size  $N$ , an input  $x$

**Output:**  $x'$  such that  $x \neq x'$  and  $h(x) = h(x')$

- 1: compute  $h(x)$
- 2: pick a random ordering of all inputs  $x_1, x_2, \dots$
- 3: **for all**  $i$  such that  $x_i \neq x$  **do**
- 4:   compute  $h(x_i)$
- 5:   **if**  $h(x_i) = h(x)$  **then**
- 6:     yield  $x' = x_i$  and stop
- 7:   **end if**
- 8: **end for**
- 9: search failed

# Scenarii for Threat Models

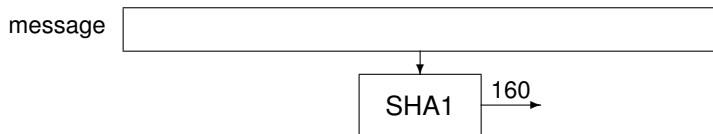
- Substitution in the integrity check process  
→ second preimage attack
- Substitution in a commitment scheme  
→ collision attack
- Information retrieval in a commitment scheme  
→ first preimage attack



# Cryptographic Hashing

- “Message Digest” (MD) devised by Ronald Rivest
- “Secure Hash Algorithm” (SHA) standardized by NIST
- MD4 in 1990 (128-bit digest)
- MD5 in 1991 (128-bit digest) published as RFC 1321 in 1992
- SHA in 1993 (160-bit digest) (obsolete, sometimes called SHA0)
- SHA-1 in 1995 (160-bit digest)
- collision found on MD4 (Dobbertin 1996)
- preimage attack on MD4 (Dobbertin 1997)
- SHA-2 in 2002: SHA256, SHA384, SHA512 (256-, 384-, and 512-bit digest)
- collision found on SHA0 (Joux+ 2004)
- collision found on MD5 (Wang+ 2004)
- theoretical attack on SHA1 (Wang+ 2005)
- SHA-3

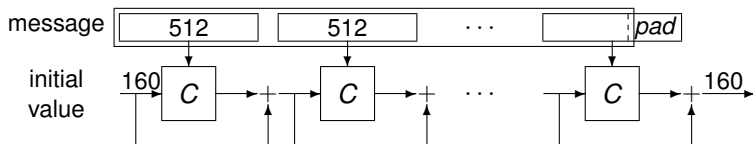
# Cryptographic Hashing



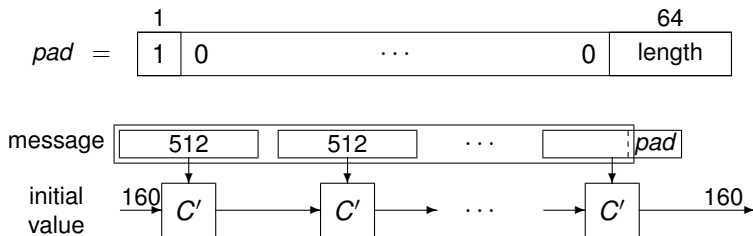
# Encryption to Hashing

On-line hashing:

- the message is padded following the Merkle-Damgård scheme;
- each block is processed using an encryption function  $C$  in a feedback mode according to the Davies–Meyer.



# Merkle-Damgård's Extension



Note: maximal length is  $2^{64} - 1$  bits

$$\text{SHA1} : \bigcup_{\ell=0}^{2^{64}-1} \{0, 1\}^{\ell} \longrightarrow \{0, 1\}^{160}$$

# Merkle-Damgård Theorem

## Theorem (Merkle-Damgård 1989)

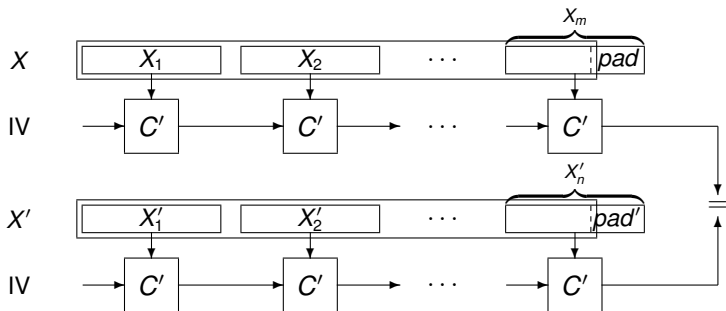
*We construct a cryptographic hash function  $h$  from a compression function  $C'$  by using the Merkle-Damgård scheme. If the compression function  $C'$  is collision-resistant, then the hash function  $h$  is collision-resistant as well.*

### **Proof.**

Case 1: messages of different length

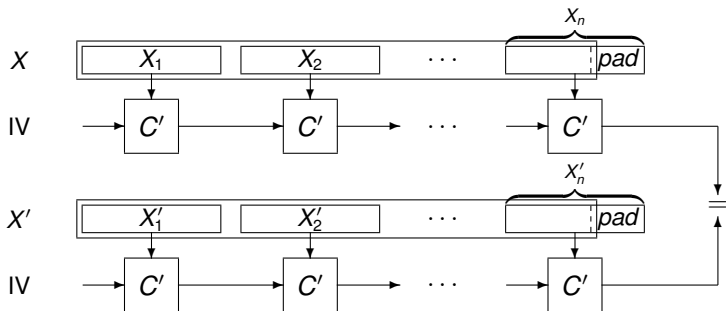
Case 2: messages of same length

# Proof of Merkle-Damgård Theorem - Case 1



$$C'(H_m, X_m) = C'(H'_n, X'_n)$$

# Proof of Merkle-Damgård Theorem - Case 2

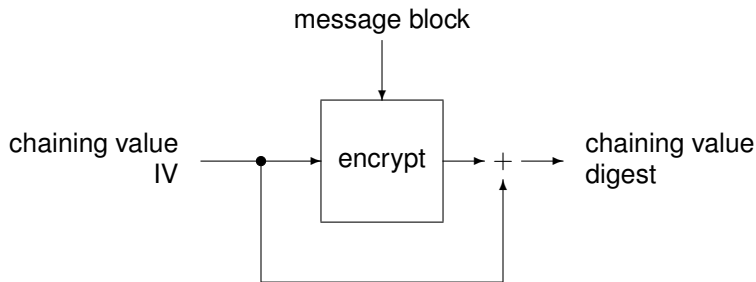


$$C'(H_i, X_i) = C'(H'_i, X'_i)$$

where  $i$  is the last index such that  $H_i \neq H'_i$  or  $X_i \neq X'_i$



# Davies–Meyer Scheme



+ is a group law



# Bitwise Boolean Functions in SHA1

$$\begin{aligned}f_1(b, c, d) &= \text{if } b \text{ then } c \text{ else } d \\ &= (b \text{ AND } c) \text{ OR } (\text{NOT}(b) \text{ AND } d) \\ f_2(b, c, d) &= b \text{ XOR } c \text{ XOR } d \\ f_3(b, c, d) &= \text{majority}(b, c, d) \\ &= (b \text{ AND } c) \text{ OR } (c \text{ AND } d) \text{ OR } (d \text{ AND } b) \\ f_4(b, c, d) &= b \text{ XOR } c \text{ XOR } d\end{aligned}$$

# Implementation of SHA-1 Compression

**Input:** an initial hash  $a, b, c, d, e$ , a message block  $x_0, \dots, x_{15}$

**Output:** a hash  $a, b, c, d, e$

1: **for**  $i = 16$  to  $79$  **do**

2:      $x_i \leftarrow \text{ROTL}^1(x_{i-3} \text{ XOR } x_{i-8} \text{ XOR } x_{i-14} \text{ XOR } x_{i-16})$

3: **end for**

4: **FOR**  $i = 1$  to  $4$  **DO**

5:     **FOR**  $j = 0$  to  $19$  **DO**

6:          $t \leftarrow \text{ROTL}^5(a) + f_i(b, c, d) + e + x_{20(i-1)+j} + k_i$

7:          $e \leftarrow d$

8:          $d \leftarrow c$

9:          $c \leftarrow \text{ROTL}^{30}(b)$

10:         $b \leftarrow a$

11:         $a \leftarrow t$

12:        **end for**

13:     **end for**

14:      $a \leftarrow a + a_{\text{initial}}$

15:      $b \leftarrow b + b_{\text{initial}}$

16:      $c \leftarrow c + c_{\text{initial}}$

17:      $d \leftarrow d + d_{\text{initial}}$

18:      $e \leftarrow e + e_{\text{initial}}$

# SHA-3 based on Keccak

- designed by Bertoni, Daemen, Peeters, and Van Assche (STMicroelectronics and NXP Semiconductors, Belgium)
- based on a **sponge** construction
- uses a permutation Keccak- $f[b]$  (or just  $f$ ) with  $b = 1\,600 = 25 \times 2^6$  (could use  $b = 25 \times 2^\ell$  with  $0 \leq \ell \leq 6$ )
- operates on states bitstrings  $s$  represented as 3-dimensional  $5 \times 5 \times 2^\ell$  arrays  $a$  of bits

$$a_{x,y,z} = s[2^\ell(5y + x) + z]$$

in what follows,  $x, y, z$  are taken modulo their dimension

- $f$  is a sequence of  $n_r = 12 + 2\ell$  rounds

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

# One Round of $f$ — i

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- $\theta$  is a linear diffusion layer using the parity of columns

$$\theta(\mathbf{a})_{x,y,z} = a_{x,y,z} \oplus \bigoplus_{j=0}^4 a_{x-1,j,z} \oplus \bigoplus_{j=0}^4 a_{x+1,j,z-1}$$

- $\rho$  permutes some lanes

$$\rho(\mathbf{a})_{x,y,z} = a_{x,y,z - \frac{(t+1)(t+2)}{2}} \quad \text{with} \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

for  $t = 0, \dots, 23$  (+ use  $\rho(\mathbf{a})_{0,0,z} = a_{0,0,z}$ )

## One Round of $f$ — ii

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- $\pi$  permutes the slices

$$\pi(\mathbf{a})_{X,Y,Z} = \mathbf{a}_{x,y,z} \quad \text{with} \quad \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- $\chi$  has degree two

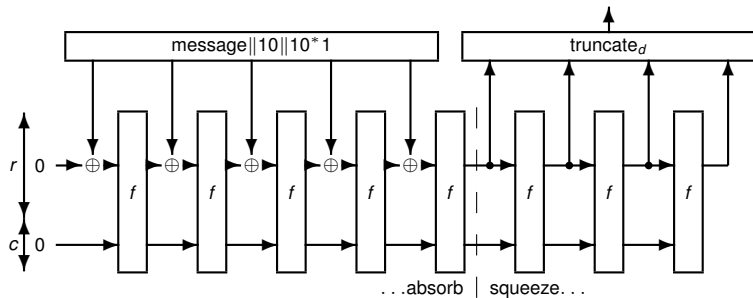
$$\chi(\mathbf{a})_{x,y,z} = \mathbf{a}_{x,y,z} \oplus (\mathbf{a}_{x+1,y,z} \oplus 1)\mathbf{a}_{x+2,y,z}$$

- $\iota$  adds a constant for  $x = y = 0$

$$\iota(\mathbf{a})_{x,y,z} = \begin{cases} \mathbf{a}_{0,0,z} \oplus \text{RC}[i_r]_z & \text{if } x = y = 0 \\ \mathbf{a}_{x,y,z} & \text{otherwise} \end{cases}$$

where  $i_r$  is the round index

# The Sponge



algo	$r$	$c$	$d$
SHA3-224	1 152	448	224
SHA3-256	1 088	512	256
SHA3-384	832	768	384
SHA3-512	576	1 024	512

# Hash Functions to Remember

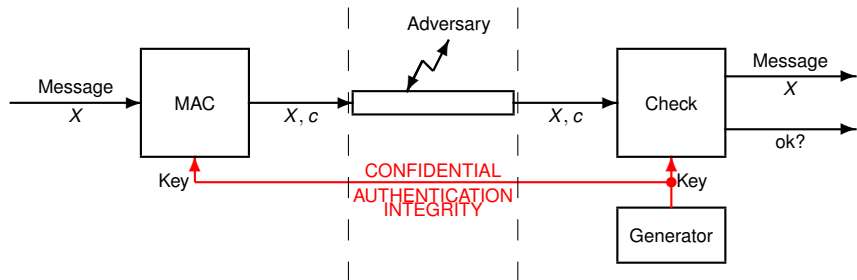
algorithm	release	digest	comment
MD5	1991	128	broken
SHA1	1995	160	still surviving
SHA3	2015	224, 256, 384, 512	

## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- **Message Authentication Codes**
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses

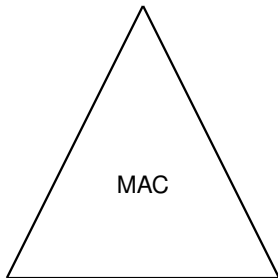


# MAC



# MAC Primitive

Alice and Bob, Gen, MAC, Check  
**components**



**functionality**

$\text{Check}_K(\text{MAC}_K(X)) = (X, \text{ok})$

typically: MAC appends a code  $c$  to  $X$ , Check recomputes  $c$  and compares

**security**

unforgeability

# Security

- adversary objective: forge new messages
- typically: **key recovery**
- **known message attack** (previous picture): using authenticated messages in transit only
- **chosen message attack**: force the sender to authenticate some messages selected by the adversary

# Hashing to Authentication: HMAC [RFC 2104]

Computing the MAC of  $t$  bytes for a message  $X$  with a key  $K$  using a Merkle-Damgård hash function with block size  $B$  bytes, digest size  $L$  bytes. ( $t = L$  by default.) E.g.  $H = \text{SHA-1}$ ,  $B = 64$ ,  $L = 20$ .

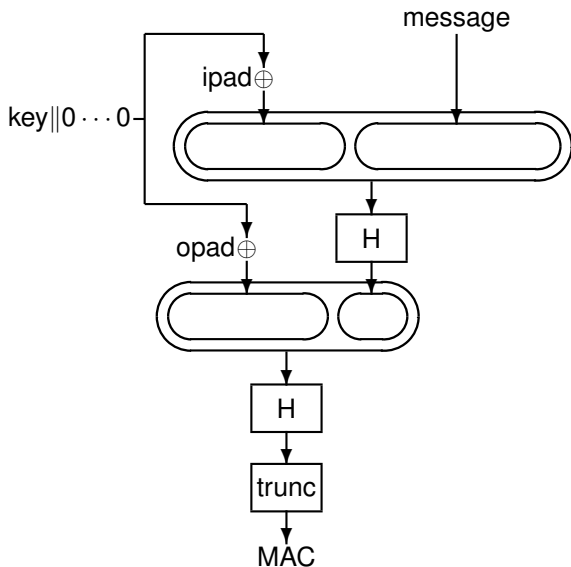
- 1 If  $K$  has more than  $B$  bytes, we first replace  $K$  by  $H(K)$ .  
(Having a key of such a long size does not increase the security.)
- 2 We append zero bytes to the right of  $K$  until it has exactly  $B$  bytes.
- 3 We compute

$$H((K \oplus \text{opad}) \| H((K \oplus \text{ipad}) \| X))$$

where  $\text{ipad}$  and  $\text{opad}$  are two fixed bitstrings of  $B$  bytes. The  $\text{ipad}$  consists of  $B$  bytes equal to  $0x36$  in hexadecimal. The  $\text{opad}$  consists of  $B$  bytes equal to  $0x5c$  in hexadecimal.

- 4 We truncate the result to its  $t$  leftmost bytes. We obtain  $\text{HMAC}_K(X)$ .

# HMAC [RFC 2104]



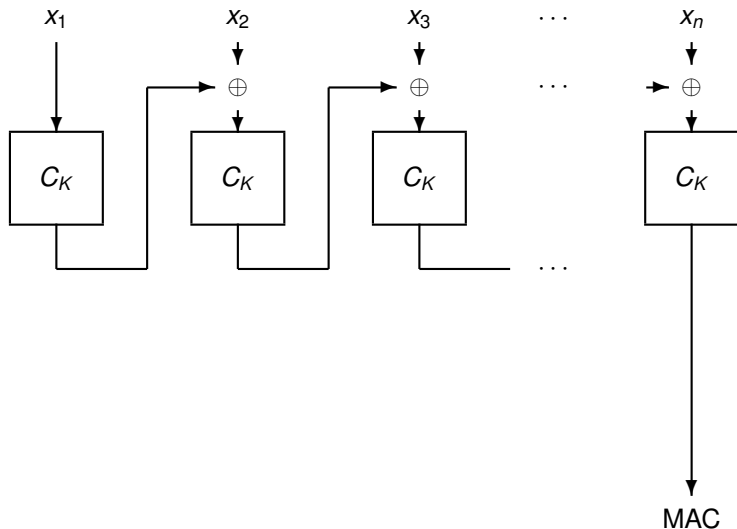
# Examples

algo	hash	$B$	$L$	$t$
------	------	-----	-----	-----

TLS				
MD5	MD5	64	16	16
SHA	SHA1	64	20	20

SSH				
hmac_md5	MD5	64	16	16
hmac_md5_96	MD5	64	16	12
hmac_sha1	SHA1	64	20	20
hmac_sha_96	SHA1	64	20	12

# CBCMAC - (A Bad MAC)



= last ciphertext block of CBC encryption ( $IV = 0$ )

# A MAC Forgery

$X_1$  = random

$X_2$  = random

$X_3$  =  $X_1 || B$

$X_4$  =  $X_2 || B'$

$B'$  =  $B \oplus c \oplus c'$

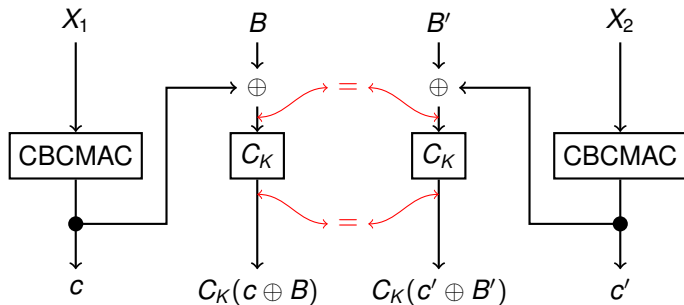
$\text{MAC}(X_1)$  =  $c$

$\text{MAC}(X_2)$  =  $c'$

$\text{MAC}(X_3)$  =  $C_K(c \oplus B)$

$\text{MAC}(X_4)$  =  $C_K(c' \oplus B')$

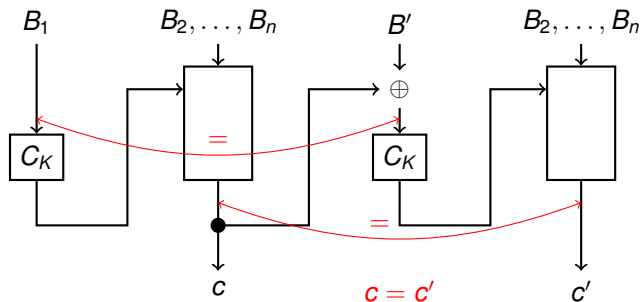
$\text{MAC}(X_4)$  =  $\text{MAC}(X_3)$





# Other Attack with 1 Known Message

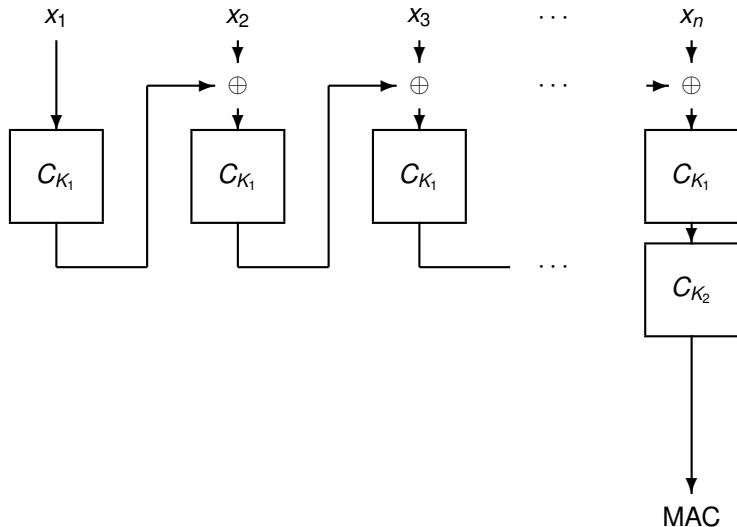
- $X_1 = B_1 \parallel \dots \parallel B_n$  arbitrary
- $c = \text{MAC}(X_1)$
- $X_2 = X_1 \parallel B' \parallel B_2 \parallel \dots \parallel B_n$  with  $B' = c \oplus B_1$
- forgery:  $c = \text{MAC}(X_2)$



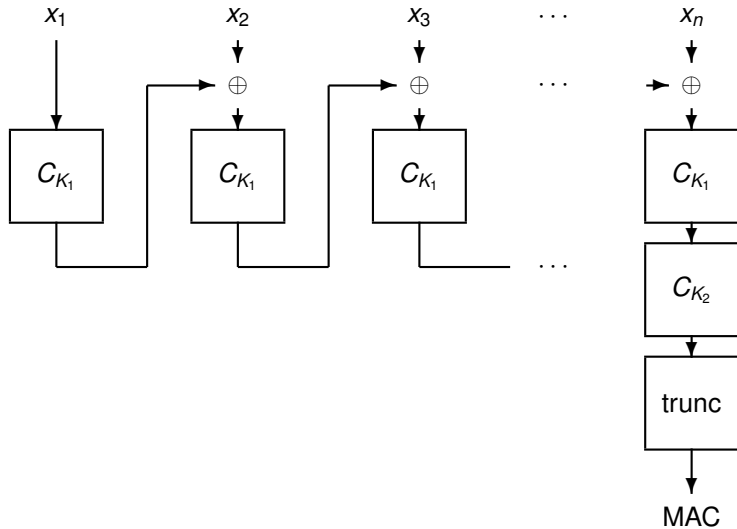
# Result on CBCMAC

- insecure when used alone as a MAC
- secure when restricted to messages of same fixed length
- might be secure if encrypted (next constructions)

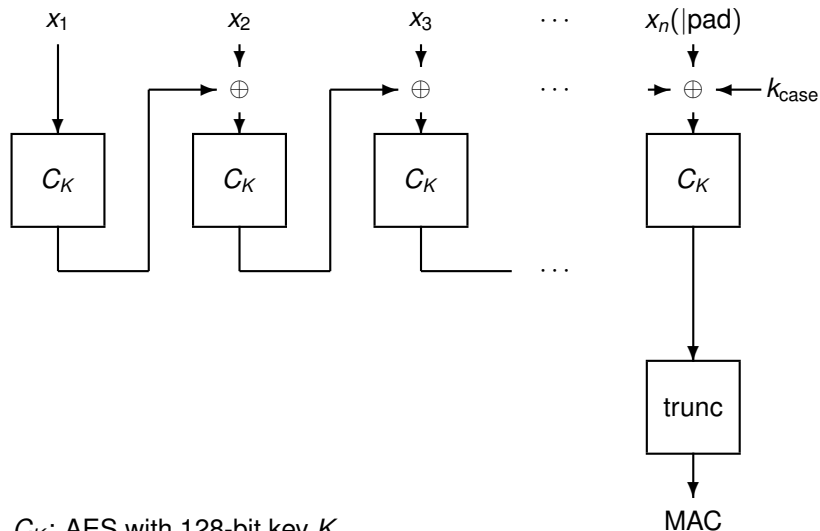
# EMAC (Encrypted MAC) - (A Better CBCMAC Variant)



# ISO/IEC 9797 - (An Even Better CBCMAC Variant)



# CMAC [RFC4493] - (Best CBCMAC Variant)



# CMAC

(previously called OMAC1)

- case 1:  $x_n$  was not padded
- case 2: the message length is not multiple of the block length pad it with a bit 1 and as many bits 0 as required to reach this length
- $L = C_K(0)$  (encryption of the zero block)
- $k_1$  is  $L$  shifted to the left by one bit XOR the carry constant if any
- $k_2$  is  $k_1$  shifted to the left by one bit XOR the carry constant if any
- actually, this is the GF multiplication by the variable  $x$  carry constant:  
0x0000000000000001b for 64-bit blocks and  
0x0000000000000000000000000000000087 for 128-bit blocks

Let  $(h_K)_{K \in \mathcal{U}\mathcal{K}}$  be a  $\varepsilon$ -**XOR-universal** family of hash functions, over the output domain  $\{0, 1\}^m$ , defined by a random key  $K$  which is chosen uniformly at random in  $\mathcal{K}$ .

Given  $K$  and a sequence of keys  $K_1, K_2, \dots$  which are independent and uniformly distributed over  $\{0, 1\}^m$ , we define a MAC algorithm which changes the key for every new message: the MAC of the message  $x_i$  of sequence number  $i$  is a pair  $(i, c_i)$  with  $c_i$  defined by

$$c_i = h_K(x_i) \oplus K_i$$

## Theorem (Wegman-Carter 1981)

*No chosen message attack can forge a new authenticated message with a probability of success greater than  $\varepsilon$ .*

# Universal Hash Function

## Definition (Krawczyk 1994)

Let  $(h_K)_{K \in \mathcal{U}\mathcal{K}}$  be a family of hash functions over the output domain  $\{0, 1\}^m$  defined by a random key  $K$  which is chosen uniformly at random in a key space  $\mathcal{K}$ .

This family is  $\epsilon$ -**XOR-universal** if for any  $a$  and  $x \neq y$  we have

$$\Pr[h_K(x) \oplus h_K(y) = a] \leq \epsilon.$$

Note:  $1 = \sum_a \Pr[h_K(x) \oplus h_K(y) = a] \leq 2^m \epsilon$  so  $\epsilon \geq 2^{-m}$



# WC-MAC - Proof — i

## Proof.

At the end, the attacker collects  $d$  triplets  $(x_i, i, c_i)$  for  $i = 1, \dots, d$  and forges  $(x, j, c)$  with  $x \neq x_i$  for any  $i$ .

If  $j$  is not in the  $[1, d]$  interval, then  $K_j$  is uniformly distributed and independent from this information, so the probability that  $c$  is a valid MAC of  $(x, j)$  is  $2^{-m}$ . (Note that  $2^{-m} \leq \epsilon$ .)

## WC-MAC - Proof — ii

If  $j$  is in the interval  $[1, d]$ , let  $I = \{h_K(x_i) \oplus K_i = c_i; i \in [1, d], i \neq j\}$ .  
The success probability is

$$P = \Pr[h_K(x) \oplus K_j = c | h_K(x_j) \oplus K_j = c_j, I]$$

Due to the distribution of  $K_1, \dots, K_{j-1}, K_{j+1}, \dots, K_d$ , we can see that  $I$  is useless in the probability.

$$\begin{aligned} P &= \Pr[h_K(x) \oplus K_j = c | h_K(x_j) \oplus K_j = c_j] \\ &= \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j | h_K(x_j) \oplus K_j = c_j] \\ &= \frac{\Pr[K_j = h_K(x_j) \oplus c_j | h_K(x) \oplus h_K(x_j) = c \oplus c_j]}{\Pr[K_j = h_K(x_j) \oplus c_j]} \\ &\quad \times \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j] \quad (\text{Bayes}) \\ &= \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j] \leq \epsilon \end{aligned}$$

since  $K_j$  is independent from  $K$ .

## WC-MAC - Proof — iii

$$\begin{aligned}\Pr[\text{success}] &= \Pr[\text{success}|j > d] \Pr[j > d] + \Pr[\text{success}|j \leq d] \Pr[j \leq d] \\ &\leq \varepsilon \Pr[j > d] + \varepsilon \Pr[j \leq d] \\ &= \varepsilon\end{aligned}$$



# Example of Universal Hashing (Krawczyk 1994)

(LFSR-based Toeplitz hash function)

- Given  $m$  and  $n$ , we define a family of hash functions  $h_K$  from  $\{0, 1\}^*$  to  $\{0, 1\}^m$
- $\mathcal{K}$  is the set of all  $K = (p, s)$  where  $p(x) = \sum_{j=0}^m p_j x^j$  is an irreducible polynomial of degree  $m$  over  $\text{GF}(2)$  and an  $s = (s_0, \dots, s_{m-1})$  is an  $m$ -bit string.
- $K$  defines an LFSR with connexion polynomial  $p(x)$  and initial state  $s$

$$s_{t+m} = \bigoplus_{j=0}^{m-1} p_j s_{t+j} \quad h_K(x_0, \dots, x_{n-1}) = \bigoplus_{\substack{0 \leq t < n \\ x_t = 1}} (s_t, \dots, s_{t+m-1})$$

- For any  $m$  and  $n$ , the family of all  $h_K$  defined from  $\{0, 1\}^{\leq n}$  to  $\{0, 1\}^m$  is  $n2^{1-m}$ -XOR-universal

# Example

$$p(x) = 1 + x + x^4, s = (1, 0, 0, 0)$$

compute

$$h_K(1, 1, 0, 1, 0)$$

	1	0	0	0	1
$\oplus$	0	0	0	1	1
$\oplus$	0	0	0	0	0
$\oplus$	0	1	0	0	1
$\oplus$	0	0	0	0	0
<hr/>					
=	1	1	0	1	

$$h_K(1, 1, 0, 1, 0) = (1, 1, 0, 1)$$

# WC-MAC using a Stream Cipher

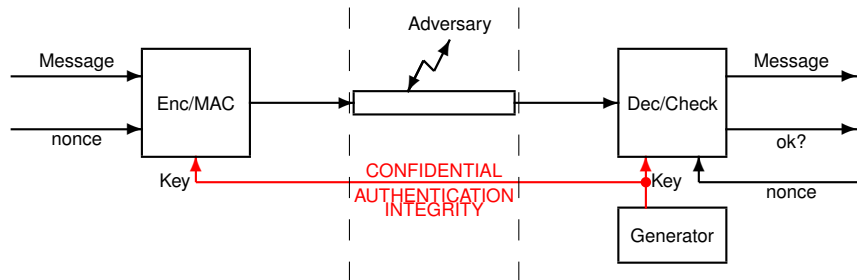
$$N \leftarrow \text{nonce}$$
$$\text{MAC}_{K,K'}(x) = (N, h_K(x) \oplus \text{Keystream}_{K',N})$$

idea: “encrypt  $h_K(x)$  using a stream cipher”

## Example (Taken From GCM Mode)

- (mac)  $\text{GMAC}_K(\text{IV}, A)$ 
  - 1: set  $H = C_K(0^{128})$
  - 2: set  $S = \text{GHASH}_H(A\|0^v\|\text{length}(A)\|0^{128})$
  - 3: set  $T = \text{trunc}(\text{GCTR}_K((\text{IV}\|0^{31}1), S))$
  - 4: return  $T$
- (hash)  $\text{GHASH}_H(X_1, \dots, X_m) = X_1H^m + \dots + X_mH$  in  $\text{GF}(2^{128})$
- (CTR encryption)  $\text{GCTR}_K(\text{ct}, X) = \text{trunc}_{\text{length}(X)}(C_K(\text{ct})\|C_K(\text{ct} + 1)\|C_K(\text{ct} + 2)\dots) \oplus X$

# Authenticated Modes of Operation





# CCM (Counter with CBC-MAC)

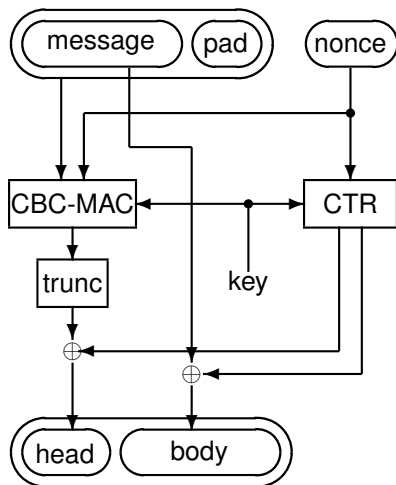
Roughly speaking:

- 1: select a nonce  $N$  (way to select and synchronize are free)
- 2: let  $T = \text{CBCMAC}(\text{message})$  using  $N$
- 3: encrypt  $T\|\text{message}$  in CTR mode using  $N$

More precisely, the CCM mode is defined by

- a block cipher which accepts 16-Byte blocks
- an even parameter  $M$  between 4 and 16 (size of the CBCMAC in bytes)
- a parameter  $L$  between 2 and 8 (size of the length field in bytes)

# CCM



# CCM Processing

- pad  $X$  with enough zero bytes to reach the block boundary
- split  $X\|\text{pad}$  as  $B_1\|\dots\|B_n$
- make  $B_0 = \text{byte}_1\|N\|\text{length}(X)$  where  $\text{byte}_1$  encodes  $M$  and  $L$
- compute the CBCMAC of  $B_0\|B_1\|\dots\|B_n$ , truncate it to  $M$  bytes, and get  $T$
- make  $A_i = \text{byte}_2\|N\|i$  where  $\text{byte}_2$  encodes  $L$
- encrypt  $T\|X$  by

$$Y = (T\|X) \oplus (\text{trunc}_M(C_K(A_0))\|\text{trunc}(C_K(A_1))\|\dots\|C_K(A_n))$$

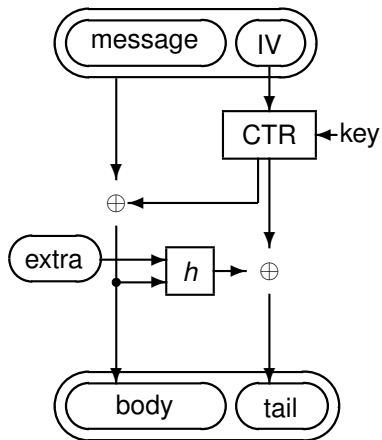
# Processing with an Extra Data

If we wish to send  $X$  together with a protocol data  $a$  which also needs to be authenticated (e.g. a sequence number, and IP address...)

- add a special bit in  $\text{byte}_1$  which tells that  $a$  is used
- if  $a$  has a length between 1 and 65279 bytes, encode this length on two bytes, make  $\text{length}(a) \| a \| \text{pad}'$  where  $\text{pad}'$  consists of enough zero bytes to reach the block boundary
- insert it between  $B_0$  and  $B_1$  before the CBCMAC computation

# GCM Mode

- (authenticated encryption)  $\text{GCM}_{AE_K}(IV, P, A)$  with plaintext  $P$  and extra data  $A$ 
  - 1: set  $H = C_K(0^{128})$
  - 2: set  $J_0 = IV || 0^{31} 1$  (IV concatenated with a 32-bit counter)
  - 3: set  $C = \text{GCTR}_K(J_0 + 1, P)$
  - 4: concatenate  $A$  and  $C$  with 0 bits to reach a length multiple of 128 and get  $A || 0^v$  and  $C || 0^u$
  - 5: set  $S = \text{GHASH}_H(A || 0^v || C || 0^u || \text{length}(A) || \text{length}(C))$
  - 6: set  $T = \text{trunc}(\text{GCTR}_K(J_0, S))$
  - 7: return  $(C, T)$
- (MAC)  $\text{GMAC}_K(IV, A) = \text{GCM}_{AE_K}(IV, \emptyset, A)$
- (hash)  $\text{GHASH}_H(X_1, \dots, X_m) = X_1 H^m + \dots + X_m H$  in  $\text{GF}(2^{128})$
- (CTR encryption)  $\text{GCTR}_K(\text{ct}, X) = \text{trunc}_{\text{length}(X)}(C_K(\text{ct}) || C_K(\text{ct} + 1) || C_K(\text{ct} + 2) \dots) \oplus X$



# Authenticated Modes to Remember

mode	comment
CCM	CTR + CBCMAC
GCM	CTR + WC-MAC

## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- **Formalism**
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses



# Hash Function

## Definition

A **hash function** is a tuple  $(\mathcal{D}, \{0, 1\}^\tau, h)$  with a message domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , an output domain  $\{0, 1\}^\tau$ , and one efficient deterministic algorithm  $h$  implementing a function

$$\begin{aligned} h: \mathcal{D} &\longrightarrow \{0, 1\}^\tau \\ X &\longmapsto h(X) \end{aligned}$$

# One-Wayness

## Definition

A hash function  $(\mathcal{D}, \{0, 1\}^\tau, h)$  is  $(t, \varepsilon)$ -**one-way** if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$ ,

$$\Pr[h(\mathcal{A}(y)) = y] \leq \varepsilon$$

where  $y \in \{0, 1\}^\tau$  is random.

(= first preimage attack)

# Security Against Collision Attack (Bad Definition)

## Definition

A hash function  $(\mathcal{D}, \{0, 1\}^\tau, h)$  is  $(t, \epsilon)$ -**secure against collision attacks** if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$ ,

$$\Pr[h(x) = h(x'), x \neq x'] \leq \epsilon$$

where  $(x, x')$  is the output of  $\mathcal{A}$

Following this definition, no hash function with  $\#\mathcal{D} > 2^\tau$  is secure: collision exist, so  $\mathcal{A}$  can just print one!

Making a correct definition is beyond the scope of this course

# Message Authentication Code

(most common construction)

## Definition

A **message authentication code** is a tuple  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  with a key domain  $\{0, 1\}^k$ , a message domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , an output domain  $\{0, 1\}^\tau$ , and one efficient deterministic algorithm  $\text{MAC}$  implementing a function

$$\begin{array}{lcl} \text{MAC} : & \{0, 1\}^k \times \mathcal{D} & \longrightarrow \{0, 1\}^\tau \\ & (K, X) & \longmapsto \text{MAC}_K(X) \end{array}$$

# Security against Key Recovery

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  is  $(q, t, \varepsilon)$ -**secure against key recovery under chosen message attacks** if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{MAC}(K, \cdot)} \rightarrow K] \leq \varepsilon$$

where  $K \in \{0, 1\}^k$  is random.

(+ similar notion with **known message attacks**)

# Security against Forgery

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  is  $(q, t, \varepsilon)$ -**secure against forgery under chosen message attacks** if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{MAC}(K, \cdot)} \text{ forges}] \leq \varepsilon$$

where  $K \in \{0, 1\}^k$  is random,  $(X, c)$  a pair of random variables defined as the output of  $\mathcal{A}^{\text{MAC}(K, \cdot)}$ , and “ $\mathcal{A}^{\text{MAC}(K, \cdot)}$  forges” is the event that  $\text{MAC}_K(X) = c$  and that  $\mathcal{A}$  did not query  $X$  to the authentication oracle.

(+ similar notion with **known message attacks**)

# Forgery Security is Stronger than Key Recovery Security

If  $\mathcal{A}$  is a key recovery adversary, we can define

- 1: run  $\mathcal{A} \rightarrow K$
- 2: pick a fresh  $X$  arbitrarily
- 3: compute  $c = \text{MAC}(K, X)$
- 4: return  $(X, c)$

So,

key recovery-breaking  $\implies$  forge

So,

forgery-secure  $\implies$  key recovery-secure

# Security against Distinguisher

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  is a  $(q, t, \varepsilon)$ -**pseudorandom function (PRF)** if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{MAC}(K, \cdot)} \rightarrow 1] - \Pr[\mathcal{A}^{F(\cdot)} \rightarrow 1] \leq \varepsilon$$

where  $K \in \{0, 1\}^k$  is random and  $F(\cdot)$  is a random function from  $\mathcal{D}$  to  $\{0, 1\}^\tau$ .



# Security Notions

	<b>key recovery</b>	<b>forgery</b>	<b>PRF</b>
<b>KMA</b>	weaker security		
<b>CMA</b>			stronger security

## 6

## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- **Bruteforce Collision Search Algorithms**
- How to Select Security Parameters?
- Other Reasons why Security Collapses

# Birthday Paradox

## Theorem

If we pick independent random numbers in  $\{1, 2, \dots, N\}$  with uniform distribution,  $\theta\sqrt{N}$  times, we get at least one number twice with probability

$$1 - \frac{N!}{N^{\theta\sqrt{N}}(N - \theta\sqrt{N})!} \xrightarrow{N \rightarrow +\infty} 1 - e^{-\frac{\theta^2}{2}}.$$

For  $N = 365$ , we obtain the following figures.

$\theta\sqrt{N}$	10	15	20	25	30	35	40
$\theta$	0.52	0.79	1.05	1.31	1.57	1.83	2.09
probability	12%	25%	41%	57%	71%	81%	89%

# Birthday Paradox - Informal Proof

$$n = \theta\sqrt{N}$$

$$\begin{aligned} p &\approx 1 - \left(1 - \frac{1}{N}\right)^{\binom{n}{2}} \\ &\approx 1 - \left(1 - \frac{1}{N}\right)^{\frac{n^2}{2}} \\ &= 1 - e^{\frac{n^2}{2} \ln\left(1 - \frac{1}{N}\right)} \\ &\approx 1 - e^{-\frac{n^2}{2N}} \\ &= 1 - e^{-\frac{\theta^2}{2}} \end{aligned}$$

# Birthday Paradox - Proof — i

**Proof.** We use the Stirling Approximation

$$n! \underset{n \rightarrow +\infty}{\sim} \sqrt{2\pi n} e^{-n} n^n$$

We have

$$\begin{aligned} 1 - p &= \frac{N!}{N^{\theta\sqrt{N}}(N - \theta\sqrt{N})!} \\ &\sim \left(1 - \frac{\theta}{\sqrt{N}}\right)^{-N + \theta\sqrt{N}} e^{-\theta\sqrt{N}} \\ &= \exp\left[-\theta\sqrt{N} + (-N + \theta\sqrt{N}) \log\left(1 - \frac{\theta}{\sqrt{N}}\right)\right] \end{aligned}$$

## Birthday Paradox - Proof — ii

We now use  $\log(1 - \varepsilon) = -\varepsilon - \frac{\varepsilon^2}{2} + o(\varepsilon^2)$

$$\begin{aligned}1 - p &\sim \exp\left[-\theta\sqrt{N} + (-N + \theta\sqrt{N})\log\left(1 - \frac{\theta}{\sqrt{N}}\right)\right] \\ &\sim \exp\left[-\frac{\theta^2}{2} + o(1)\right] \\ &\rightarrow e^{-\frac{\theta^2}{2}}\end{aligned}$$



# Collision Search I

**Input:** a cryptographic hash function  $h$  onto a domain of size  $N$

**Output:** a pair  $(x, x')$  such that  $x \neq x'$  and  $h(x) = h(x')$

- 1: **for**  $\theta\sqrt{N}$  many different  $x$  **do**
- 2:   compute  $y = h(x)$
- 3:   **if** there is a  $(y, x')$  pair in the hash table **then**
- 4:     yield  $(x, x')$  and stop
- 5:   **end if**
- 6:   insert  $(y, x)$  in the hash table
- 7: **end for**
- 8: search failed

# Collision Search II

**Input:** a cryptographic hash function  $h$  onto a domain of size  $N$

**Output:** a pair  $(x, x')$  such that  $x \neq x'$  and  $h(x) = h(x')$

- 1: **repeat**
- 2: pick a (new) random  $x$
- 3: compute  $y = h(x)$
- 4: insert  $(y, x)$  in the hash table
- 5: **until** there is already another  $(y, x')$  pair in the hash table
- 6: yield  $(x, x')$

we can show that the expected number of iterations is  $\sqrt{\frac{\pi}{2}} \times \sqrt{N}$   
(Buffon's needles...)



# Collision Search Complexity

strategy	memory	time	success proba.
collision search I	$\theta\sqrt{N}$	$\theta\sqrt{N}$	$1 - e^{-\frac{\theta^2}{2}}$
collision search II	$\sqrt{\frac{\pi}{2}} \times \sqrt{N}$	$\sqrt{\frac{\pi}{2}} \times \sqrt{N}$	1

example for SHA1:  $N = 2^{160}$ , complexity  $\sim 2^{80}$

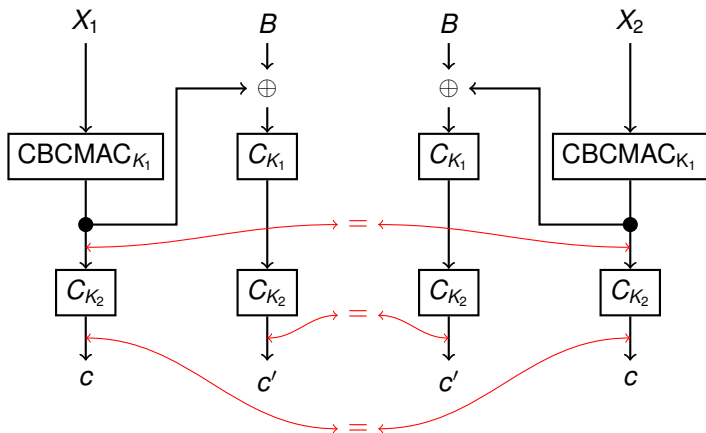
## Example: Birthday Attack on EMAC

First get  $\sqrt{2^{\text{MAC length}}}$  many messages until we get two messages  $X_1$  and  $X_2$  such that  $\text{MAC}(X_1) = \text{MAC}(X_2)$  by using the birthday paradox.

Deduce  $\text{CBCMAC}(X_1) = C_{K_2}^{-1}(c) = \text{CBCMAC}(X_2)$

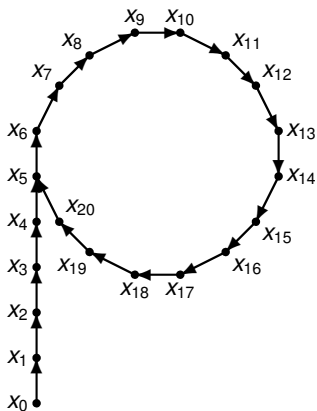
Pick  $B$  arbitrarily. Query  $\text{MAC}(X_1 \| B) = c'$

Deduce  $\text{MAC}(X_2 \| B) = c'$



# (Almost) Memoryless Collision Search

## The Rho ( $\rho$ ) Effect



- $x_{i+1} = F(x_i)$
- $\rho$  shape (due to finite set)
- tail  $\lambda = 5$
- loop  $\tau = 16$
- collision  $F(x_{\lambda-1}) = F(x_{\lambda+\tau-1})$

### Lemma

If  $F$  is a random function over a set of cardinality  $N$ , we have  $E(\lambda) = E(\tau) = \sqrt{\pi N/8}$ .

# Floyd Cycle Finding Algorithm (1967)

## Tortoise and the Hare

**Output:** a collision for  $F$

**Complexity:**  $\mathcal{O}(\sqrt{N})$   $F$  mappings

```
1: set  $x_0$  at random
2:  $a \leftarrow x_0$  (tortoise)
3:  $b \leftarrow x_0$  (hare)
4: repeat
5:    $a \leftarrow F(a)$ 
6:    $b \leftarrow F(F(b))$ 
7: until  $a = b$ 
8:  $a \leftarrow x_0$ 
9: while  $a \neq b$  do
10:   $a_{\text{old}} \leftarrow a$ 
11:   $b_{\text{old}} \leftarrow b$ 
12:   $a \leftarrow F(a)$ 
13:   $b \leftarrow F(b)$ 
14: end while
15: output  $a_{\text{old}}, b_{\text{old}}$ 
```

- whenever  $x_{2i} = x_i$  we must have  $\tau|i$
- we find  $i = \tau \times \lceil \frac{\max(\lambda, 1)}{\tau} \rceil$
- exact complexity is  $5i$  computations  $F$
- which is on average

$$5 \times \left( E(\lambda) + \frac{1}{2} E(\tau) \right) \\ = 7.5 \sqrt{\pi/8} \times \sqrt{N}$$

# Why it Works

let  $x_i = F(x_{i-1})$

- after iteration  $i$  of the **repeat-until** loop, we have  $a = x_i$  and  $b = x_{2i}$   
 $a = b$  is equivalent to  $(i \geq \lambda$  and  $\tau | i)$   
there exists a minimum  $i = i_0 = \tau \times \lceil \frac{\lambda}{\tau} \rceil$  satisfying this condition
- after iteration  $i$  of the **while-endwhile** loop, we have  $a = x_i$  and  $b = x_{i_0+i}$   
 $a = b$  is equivalent to  $i \geq \lambda$   
so, the loop ends with the correct value of  $\lambda$
- the correct value of  $\tau$  is found with the additional **repeat-until** loop

# Cycle Detection Algorithms

- Floyd (1967)
- Gosper (1972)
- Brent (1980)
- Sedgewick-Szymanski-Yao (1982)
- Quisquater-Delescaille (1989)
- van Oorschot-Wiener (1999)
- Nivasch (2004)

## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- Brute-force Collision Search Algorithms
- **How to Select Security Parameters?**
- Other Reasons why Security Collapses

# Summary of Generic Attacks against Symmetric Encryption

if we have a  $n$ -bit key, ( $N = 2^n$ )

strategy	preprocessing	memory	time	success proba.
exhaustive search	0	1	$2^n$	1
dictionary attack	$2^n$	$2^n$	1	1
tradeoffs	$2^n$	$2^{\frac{2}{3}n}$	$2^{\frac{2}{3}n}$	cte

Want a security of  $2^s$ ?

- select  $n \geq s$



# Summary of Generic Attacks against Hash Functions

if we hash onto  $n$  bits, ( $N = 2^n$ )

attack	complexity
preimage attack	$2^n$
collision attack	$2^{\frac{n}{2}}$

Want a security of  $2^s$ ?

- want security against inversion only: **select  $n \geq s$**
- want security against collisions: **select  $n \geq 2s$**

# Breaking Symmetric Cryptography

- we do not know how to prove security
- we know generic attacks are unavoidable
- empirical security: assume (hope) there is no better attack than known ones
- security  $\implies$  generic attacks are untractable
- security parameter for encryption/authentication: key length
- Caveat: hash length must be twice the security parameter due to the birthday paradox

## 6

## Integrity and Authentication

- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses

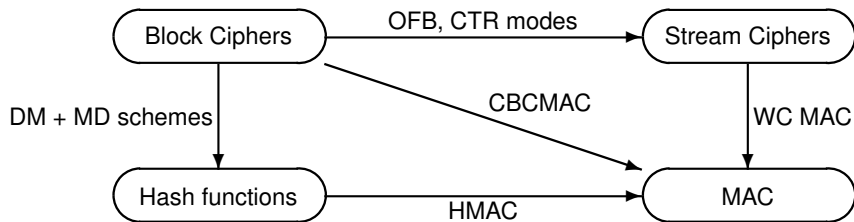
# Cryptanalytic Advances

- security is often empirical
  - dedicated attacks
- heuristic security against attack methods
  - arguments may be wrong, other attack methods can be discovered
- all eggs in the same basket (lack of crypto-diversity)
  - more exposure, attacks more devastating
- the quantum threat
  - quantum computers to factor, compute discrete logarithms, or even half security parameters [Grover 1996]
- side channel attacks
- wrong proofs, wrong models
- security interference: secure + secure may be insecure

# Conclusion

- **MAC:** HMAC, CBCMAC, WC-MAC, CCM mode, GCM mode
- **hash functions:** MD5, SHA-1
- **commitment**
- **bruteforce collision** within complexity  $\mathcal{O}(\sqrt{\#\text{range}})$

# Dedicated Primitives and Reductions



# References

- **Schneier**. *Applied Cryptography*. Wiley & Sons. 1996.  
Crypto for dummies!
- **Ferguson–Schneier**. *Practical Cryptography*. Wiley & Sons. 2003.  
Crypto for dummies!
- **Oechslin**. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *CRYPTO 2003*, LNCS 2729.

other references:

- **Barkan-Biham-Shamir**. Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs. In *CRYPTO 2006*, LNCS 4117.
- **Nivasch**. Cycle Detection Using a Stack. *Information Processing Letters* vol. 90 pp. 135–140, 2004.

# Must be Known

- Merkle-Damgård and Davies-Meyer schemes
- parameters of hash functions: MD5, SHA1
- MAC: (principles of) HMAC, CBCMAC
- existence of authenticated encryption modes: CCM, GCM
- collision search based on the birthday paradox
- security from key length



# Train Yourself

- hash functions:
  - final exam 2008–09 ex3
  - midterm exam 2011–12 ex3
- collisions:
  - final exam 2013–14 ex2
  - final exam 2012–13 ex2
  - final exam 2010–11 ex1

















- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I**
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II

# Roadmap

- mobile telephony: GSM, 3G
- WiFi: WEP, WPA
- Bluetooth
- access control: password, challenge-response, strong authentication

## 7

## Case Studies I

- Mobile Telephony
  - WEP/WPA/WPA2
  - Bluetooth
  - Cryptography Based on Short Authenticated Strings
  - Access Control
  - Forward Secrecy: the Case of Signal
  - Block Chains

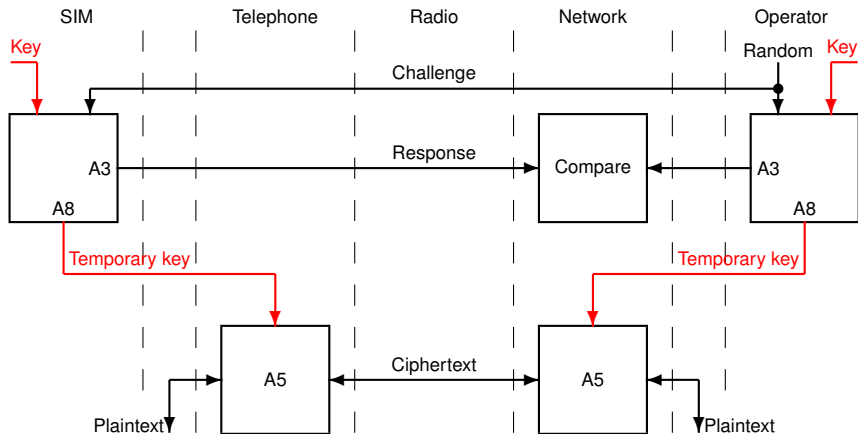
# GSM Architecture

- principle 1: authentication of mobile system
- principle 2: privacy protection in the wireless link
  
- challenge-response protocol based on Ki
- encryption key for a limited period of time (derived from Ki)
- identity IMSI replaced by a pseudonym TMSI as soon as possible
- Ki never leaves the security module (SIM card) or home security database (HLR)

# GSM Slang

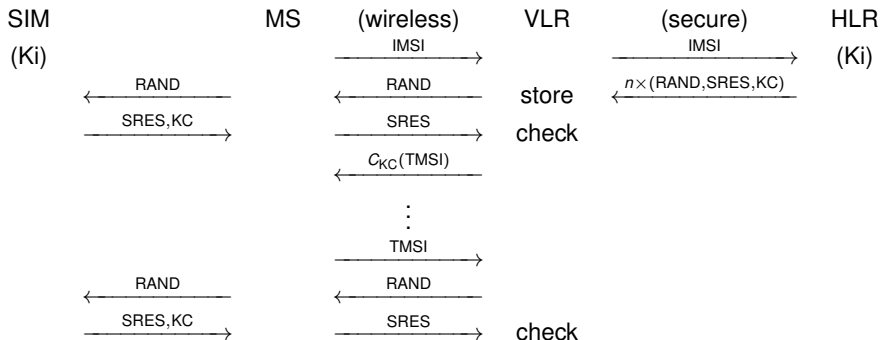
- GSM: Global System for Mobile telecommunications
- MS: Mobile Station
- SIM: Subscriber Identity Module (part of MS)
- HLR: Home Location Register
- VLR: Visitor Location Register
- IMSI: International Mobile Subscriber Identity (stored in SIM)
- Ki: subscriber Integrity Key (securely stored in SIM)

# GSM Protocol



# GSM Authentication

$$A3/8(K_i, \text{RAND}) = (\text{SRES}, \text{KC})$$



# Security of Authentication

- Ki never leaves the SIM card or the secure database of the operator (assuming SIM card is tamper proof and HLR is secure)
- assuming that A3/8 are secure PRF then authentication to network is secure
- A3/8 not standard: chosen by operator
- problem with weak A3/8 (e.g. COMP128)

security: 😊



# GSM Encryption

- several standard algorithms: A5/0, **A5/1**, A5/2, A5/3
- cipher imposed by network
- new KC for each session
- synchronized frame counter (see [A5/1 on slide 455](#))

# Security of Privacy protections

- blinding the identity: telephone identifies itself in clear at the first time then using a pseudonym given by the local network  
not effective at all:
  - challenges can be replayed to trace mobile telephones
  - fake network can force identification in clear (re-synchronization protocol)
- security of A5/0 (no encryption) void
- security of A5/2 weak
- security of A5/1 not high
- security of A5/3 high
- fake network can force to weak encryption (they all use the same key)
- replaying a challenge will force reusing a key in one-time pad
- message integrity protection is ineffective

security: 😞

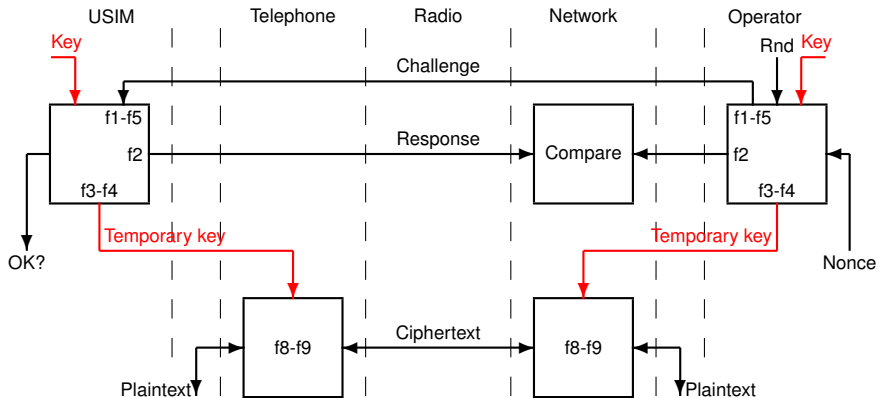
# Improvements in 3G Mobile Telephony

- challenges are authenticated (fake network cannot forge them)
- integrity protection (MAC)
- protection against challenge-replay attacks
- uses a block cipher KASUMI instead of the stream cipher A5/1

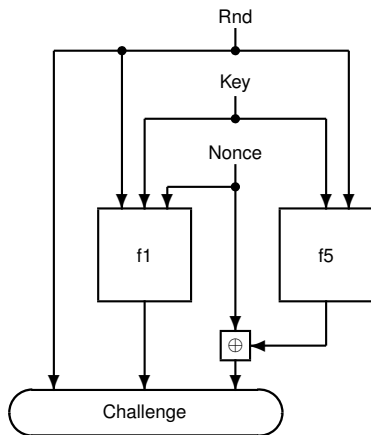
# The UMTS Crypto Menagery

- communication: f8 (encryption) and f9 (MAC) based on KASUMI
- signaling communication: f6 (encryption) and f7 (MAC) based on AES
- challenge pseudorandom generator: f0
- MILENAGE (key establishment): f1, f1\*, f2, f3, f4, f5, f5\*
  - f1 and f5: challenge computation for synchronized entities
  - f1\* and f5\*: challenge computation for re-synchronization
  - f2: response to challenge (replaces A3)
  - f3: key derivation for encryption (replaces A8)
  - f4: key derivation for MAC

# MILENAGE Protocol

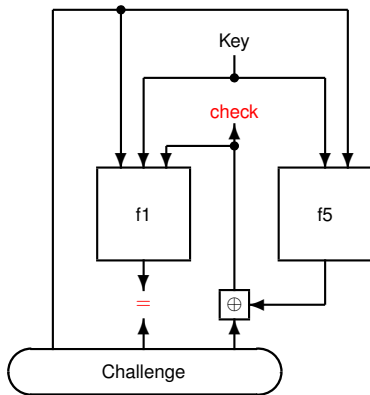


# MILENAGE Challenges



- challenge authenticated based on **f1**
- freshness protection based on a nonce  
nonce may be counter-based (USIM and operator synchronized)
- privacy protection: the nonce is encrypted by **f5**

# MILENAGE Challenge Verification





















- 1 extract Rnd
- 2 decrypt Nonce by computing  $f5(\text{Key}, \text{Rnd})$
- 3 check authentication (f1)
- 4 check Nonce is correct

# Security Misses

- network is not authenticated (network only proves he received authorization from operator)
  - attack by fake network rerouting through expensive networks of unencrypted network
- no encryption awareness



# Mobile Telephony (In)security

	2G	3G
• confidentiality		
• message authentication		
• message integrity		
• challenge freshness		
• mobile authentication		
• network authentication		
• key establishment		
• frame sequentiality		
• privacy		

# Other Standards

- DECT: wireless telephone (connected to fixed base line)  
DSAA: DECT standard Authentication Algorithm  
DSC: DECT standard Cipher  
standard is not public (but published and broken!)
- EDGE (used to be GPRS)  
GEA: GPRS Encryption Algorithm  
standard is not public
- cdmaOne (also called IS-95 or CDMA)  
no SIM card  
CAVE: Cellular Authentication and Voice Encryption  
ORYX: encryption algorithm (stream cipher)  
CMEA: Cellular Message Encryption Algorithm

## 7 Case Studies I

- Mobile Telephony
- **WEP/WPA/WPA2**
- Bluetooth
- Cryptography Based on Short Authenticated Strings
- Access Control
- Forward Secrecy: the Case of Signal
- Block Chains

# IEEE 802.11 in a Nutshell

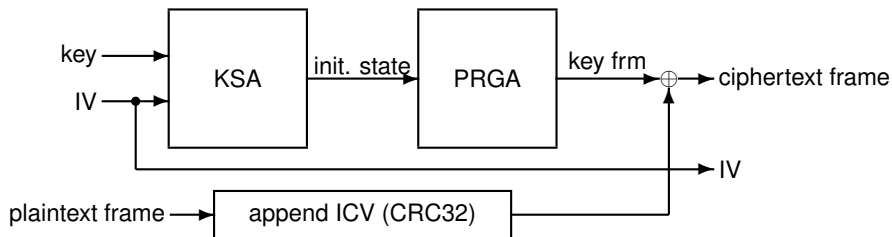
- protocol for wireless local area network (WLAN) at the link level
- since 1997
- corporate or ad-hoc mode
- secure communication by **wired equivalent privacy** (WEP)
- ~~station authentication by **Shared Key Authentication** (SKA)~~
- since 2003: interim **Wi-Fi Protected Access** (WPA)  
due to security issues
- since 2004: added WPA2 (complete change)

# WEP Security Goals

- privacy as if communication was through a wired connection
  - protect against unauthorized access
- 

- use up to 4 (common) pre-shared key to be manually set
  - key not frequently changed
  - key not too long (40 or 104 bits)
  - key stored at many places
- entirely based on RC4 stream cipher

# WEP Encryption



- self-synchronizing stream cipher (24-bit IV sent in clear)
- integrity protection using CRC32

→ packets are easily malleable (Borisov-Goldberg-Wagner 2001)





$$\text{Enc}_{\text{key}}(\text{IV}, \text{plaintext}) \oplus [\Delta \parallel \text{CRC32}(\Delta)] = \text{Enc}_{\text{key}}(\text{IV}, \text{plaintext} \oplus \Delta)$$

# WEP and SKA Issues

- **collision on IV's**  
a 24-bit IV repeats itself, sooner or later
- **use linearity of CRC32**  
if modification injected, make it coherent with CRC32 encoding
- **dedicated attack on WEP/RC4 encryption**  
Fluhrer-Mantin-Shamir 2001 and follow up's
- **passive ciphertext only attack**  
(with some bytes of each frame known)  
after sniffing 20 000 packets, probability to recover the key is  $\frac{1}{2}$   
Sepehrdad-Vaudenay-Vuagnoux 2012

# WEP (In)security

security is snake oil:

- confidentiality 
- message authentication 
- message integrity 
- message freshness no protection
- key establishment (pre-shared)
- message sequentiality no protection
- privacy 



# WPA: a Dirty Quick Fix

- WPA-TKIP (Temporal Key Integrity Protocol):  
make the RC4 key change for every packet (based on a master key)
- message integrity (with a bad MAC...)
- check IV increases to protect against replay attacks
- set up master key using EAP (Extensible Authentication Protocol)
  - PSK (Pre-Shared Key)
  - one of the possible authentication protocols form 802.1x using an authentication server (e.g. RADIUS)

# EAP

- EAP-PSK (Pre Shared Key) derive master key from passphrase
- EAP-TLS need a certificate for server and station
- EAP-TTLS (Tunneled TLS) need a certificate for server then a login and password for station
- EAP-PEAP similar as TTLS with a different protocol
- EAP-SIM using a SIM card in GSM network
- EAP-AKA same as SIM but for UMTS
- EAP-LEAP (Cisco protocol) no longer recommended
- EAP-FAST (Cisco protocol to replace LEAP)

# WPA (In)security

- confidentiality / (academic attacks)
- message authentication  (MICHAEL is broken)
- message integrity 
- message freshness 
- key establishment depends
- message sequentiality no protection (packet drops)
- privacy 

RC4 replaced by AES CCMP (CCM Protocol = AES in CCM mode)  
128 or 256 bit key

# WPA2 (In)security

- confidentiality 
- message authentication 
- message integrity 
- message freshness 
- key establishment depends
- message sequentiality no protection (packet drops)
- privacy 

## 7

## Case Studies I

- Mobile Telephony
- WEP/WPA/WPA2
- **Bluetooth**
- Cryptography Based on Short Authenticated Strings
- Access Control
- Forward Secrecy: the Case of Signal
- Block Chains

# The Bluetooth Project

- short-range wireless technology
- designed to transmit voice and data
- for a variety of mobile devices (computing, communicating, ...)
- bring together various markets



- 1Mbit/sec up to 10 meters over the 2.4-GHz radio frequency
- robustness, low complexity, low power, low cost

# Bluetooth History



- 10th Century: Viking King Harald Blåtand (Harold Bluetooth) tried to unify Denmark, Norway, and Sweden
- 1994: Ericsson initiated a study to investigate the feasibility
- May 20, 1998: Bluetooth announced, controlled by the Special Interest Group (SIG) formed by  
Ericsson, IBM, Intel, Nokia, and Toshiba
- 1999: Bluetooth 1.0 Specification Release
- 2004: Bluetooth 2.0 Specification Release
- 2007: Bluetooth 2.1 Specification Release (add SSP)
- 2009: Bluetooth 3.0 Specification Release (add 802.11)
- 2010: Bluetooth 4.0 Specification Release (add LE)



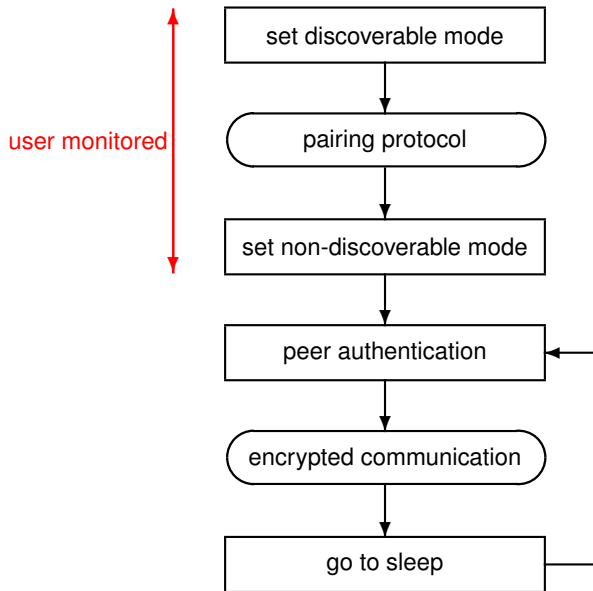
# Bluetooth Security

- mode 1: non-secure
- mode 2: service level enforced security
- mode 3: link level enforced security
- mode 4 (since v4.0): service+link level enforced security

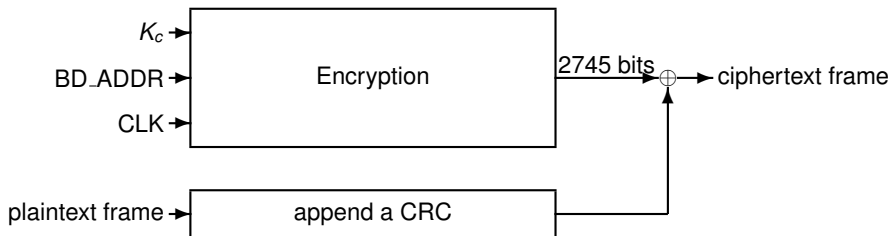
# Bluetooth Security Basics (Link Level)

- can switch device to
  - non connectable (Bluetooth is off)
  - connectable but not discoverable (invisible without knowing the MAC address)
  - discoverable (introduce itself upon any broadcast request)
- pairing to set up link keys between devices
  - typically based on a random PIN
  - (dummy device) using a built-in PIN
- can manage a database of paired devices

# Cycles in Bluetooth

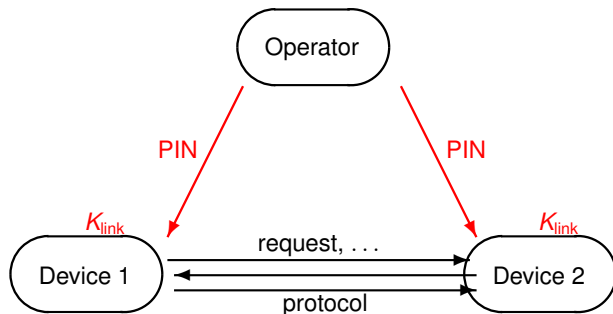


# A Strange Integrity Protection

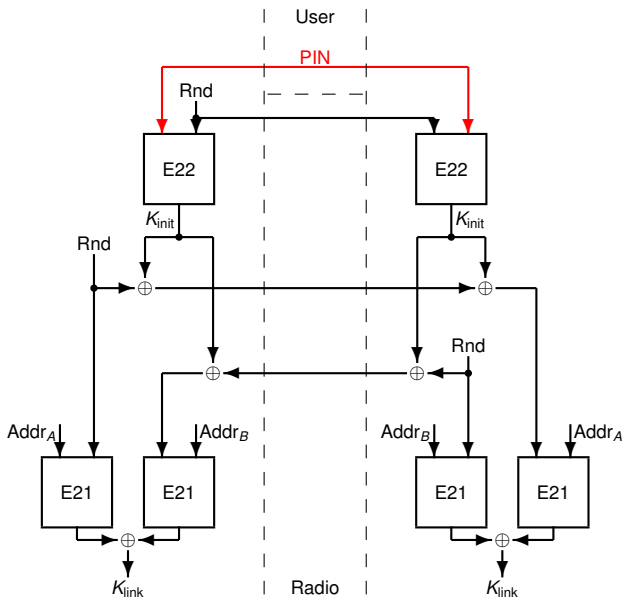


→ packets are easily malleable (Borisov-Goldberg-Wagner 2001)

# Device Pairing



# Legacy Pairing Protocol



# Legacy Pairing Protocol

## Master A

user inputs PIN code

pick  $IN\_RAND$

$$K_{init} = E_{22}(PIN, IN\_RAND)$$

pick  $LK\_RAND_A$

$$C_A = LK\_RAND_A \oplus K_{init}$$

$$LK\_RAND_B = C_B \oplus K_{init}$$

compute  $K$

## Slave B

user inputs PIN code

$$K_{init} = E_{22}(PIN, IN\_RAND)$$

pick  $LK\_RAND_B$

$$C_B = LK\_RAND_B \oplus K_{init}$$

$$LK\_RAND_A = C_A \oplus K_{init}$$

compute  $K$

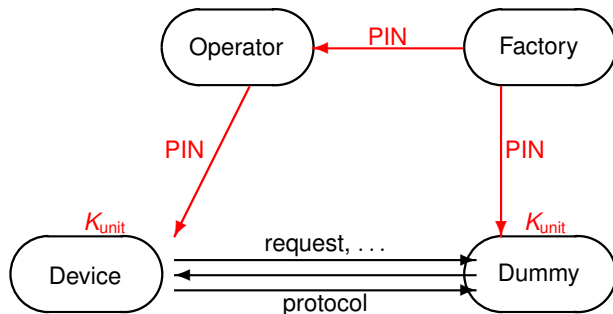
$IN\_RAND$  →

$C_A$  →

$C_B$  ←

$$K = E_{21}(LK\_RAND_A, BD\_ADDR_A) \oplus E_{21}(LK\_RAND_B, BD\_ADDR_B)$$

# Pairing with a Dummy Device





# Legacy Pairing with a Dummy Device

## Master A

user inputs PIN code

pick IN\_RAND

$$K_{\text{init}} = E_{22}(\text{PIN}, \text{IN\_RAND})$$

$$K = C_B \oplus K_{\text{init}}$$

## Slave B

user inputs PIN code (or not)

$$K_{\text{init}} = E_{22}(\text{PIN}, \text{IN\_RAND})$$

$$C_B = K_{\text{unit}} \oplus K_{\text{init}}$$

$$K = K_{\text{unit}}$$

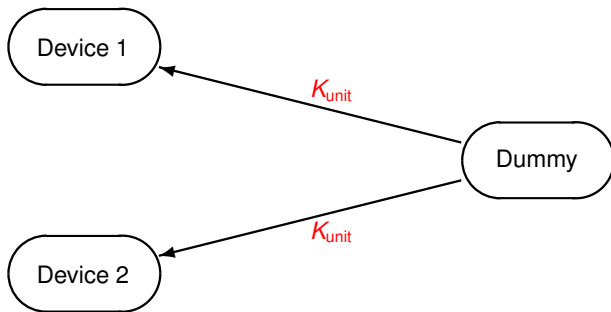
IN\_RAND →

← C<sub>B</sub>

link key is forced to be the unit key

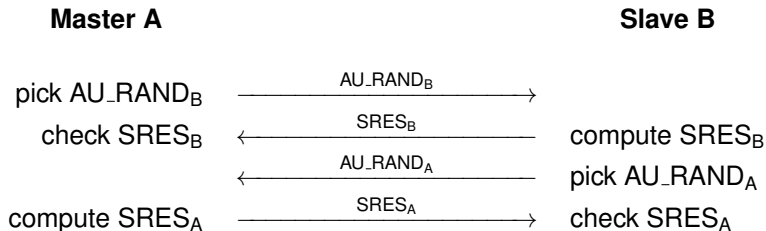
→ problem if dummy device is (or has been) paired with multiple devices

# Dummy Devices: Unit Key is Shared with Many Devices



scenario: user *A* paired his headset (Dummy) with his telephone (Device 1) then user *B* took the headset for a few seconds to pair it with his computer (Device 2)...

# Peer Authentication



$$\text{SRES}_d = E1(K, \text{AU\_RAND}_d, \text{BD\_ADDR}_d)$$

# Missing Security Protection

- Cryptographic pseudorandom generator  
→ some device may have poor generators
- Liveliness + session sequentiality  
→ some packets may be removed (Kügler 2003)
- Strong anonymity  
→ traceability (Jakobsson-Wetzel 2001)

# Insecurity Summary

- dummy devices use the same key with many devices
- suspicious security of cryptographic primitives
- academic attacks on E0 encryption
- integrity protection is void
- messages can be maliciously erased in the radio channel
- privacy protection is weak (low entropy BD\_ADDR)
- pairing protocol weak against passive attacks (next slides)

# Key Establishment (In)security

## Theorem

*The pairing protocol is secure if either PIN has large entropy or the protocol is run through a private channel (under some “reasonable assumptions” about the cryptographic algorithms).*

- 😊 a cheap pragmatic security
- 😞 pretty weak security

devastating sniffing attacks in other cases! (Jakobsson-Wetzel 2001)

# Sniffing + Offline Attack

Assumption: pairing not made in a private environment (channel not confidential) and guessable PIN (lazy operator)

- 1 sniff the pairing protocol, get  $IN\_RAND$ ,  $C_A$ ,  $C_B$
- 2  $\rightarrow$  can compute  $K_{link}$  from PIN
- 3 sniff a peer-authentication protocol, get  $rand$ ,  $F(rand, K_{link})$
- 4  $\rightarrow$  can check a guess on  $K_{link}$
- 5 run an offline exhaustive search on PIN

# Online Impersonation Attack

**Adversary**

**Slave**

receive PIN

IN\_RAND →

$C_A$  →

$C_B$  ←

AU\_Rand<sub>B</sub> →

RES<sub>B</sub> ←

compute  $K_{\text{link}}$

$RES_B = E1(K_{\text{link}}, AU\_Rand_B)$

exhaustive search on PIN s.t.

$RES_B = E1(f(\text{PIN}, \text{IN\_RAND}, C_A, C_B), \text{AU\_Rand}_B)$

compute  $K_{\text{link}} = f(\text{PIN}, \text{IN\_RAND}, C_A, C_B)$

AU\_Rand<sub>A</sub> ←

$RES_A = E1(K_{\text{link}}, \text{AU\_Rand}_B)$

RES<sub>A</sub> →



# Possible Countermeasures

- do not use short PIN  
→ not realistic
- only make pairing in a bunker  
→ not realistic
- **live with it** and make it resilient  
→ feasible by refreshing  $K_{\text{link}}$

# Pairing in Two Phases: Preparing and Repairing

## Master A

user inputs PIN code

pick IN\_RAND

$$K_{\text{init}} = E_{22}(\text{PIN}, \text{IN\_RAND})$$

pick LK\_RAND<sub>A</sub>

$$C_A = \text{LK\_RAND}_A \oplus K_{\text{init}}$$

$$\text{LK\_RAND}_B = C_B \oplus K_{\text{init}}$$

compute  $K_{\text{link}}$

## Slave B

user inputs PIN code

$$K_{\text{init}} = E_{22}(\text{PIN}, \text{IN\_RAND})$$

pick LK\_RAND<sub>B</sub>

$$C_B = \text{LK\_RAND}_B \oplus K_{\text{init}}$$

$$\text{LK\_RAND}_A = C_A \oplus K_{\text{init}}$$

compute  $K_{\text{link}}$

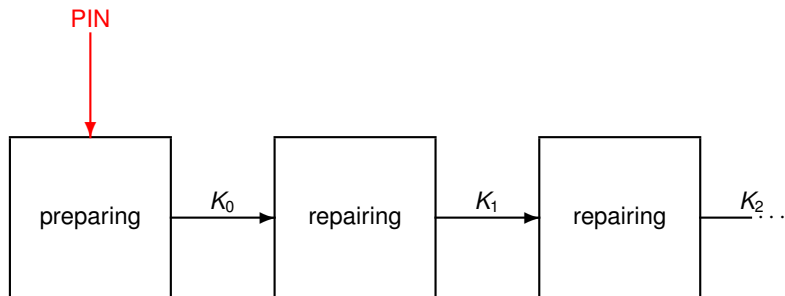
→ IN\_RAND

→ C<sub>A</sub>

← C<sub>B</sub>

preparing and repairing

# A Possible Better Usage









- if  $K_{t-1}$  is compromised and repairing is private, then  $K_t$  is safe
- if  $K_i$  is securely set up and if  $K_{i+t}$  is the first compromised key, all communications using  $K_i, \dots, K_{i+t-1}$  are safe (forward secrecy)

# Bluetooth v2.0 Summary

- light weight cryptography
- initial authenticated channel by human interaction with devices
- key exchanged based on a PIN and E21, E22 (pairing)
- derivation of a single 128-bit long term link key
- secure channel based on E0, E1, E3
  
- several missing security properties: packet authentication, detection of packet loss, privacy, ...

# Bluetooth v2.0 (In)security

Current (mode 3) security is rather poor:

- confidentiality  (attacks still academic so far)
- message integrity 
- message authentication  (auth. by encryption without integrity)
- frame freshness  (based on clock value)
- key establishment v2.0  (pragmatic repairing possible)
- frame sequentiality  (message loss)
- privacy 

# Moral

PIN has low entropy

(humans cannot generate ephemeral PINs with high entropy)

- offline passive key recovery:  
key agreement is based on conventional cryptography (so cannot resist to passive adversaries)
  - online impersonation attack:  
assuming the adversary is second to authenticate itself, the password-based key agreement does not even resist impersonation
  - next generation needs
    - be user friendly
    - be device friendly (no expensive crypto)
    - resist passive and active adversaries
- use SAS-based cryptography (to be seen in Chapter 9)

# Bluetooth v2.1: Secure Simple Pairing

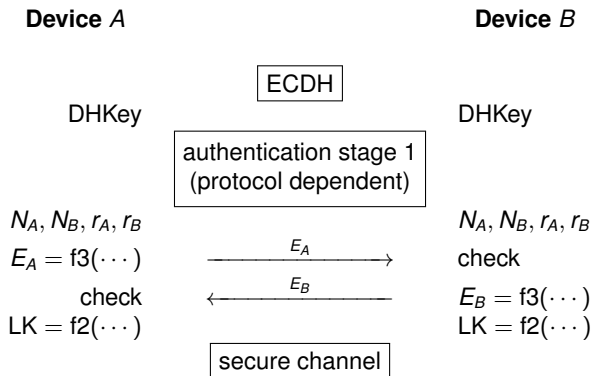
4 variants

- numeric comparison
- passkey entry
- just works (same as numeric comparison with no human work)
- out-of-band (use an ad-hoc secure channel e.g. cable or near field communication)

resist active adversary

resist passive adversary only (out-of-band may resist to active adversaries depending on the secure channel)

# Common Protocol





# Common Protocol

- **step 1:** public key exchange  
exchange ECDH public keys using standard parameters (may be ephemeral or static) leading to a key DHKey
- **steps 2–8:** authentication stage 1 (**protocol dependent**)  
this stage authenticates the ECDH public keys and exchange some values  $N_A, N_B, r_A, r_B$
- **steps 9–11:** authentication stage 2  
mutual authentication after ECDH protocol using  $N_A, N_B, r_A, r_B$ :  
 $A$  resp.  $B$  produces  $E_A$  resp.  $E_B$  and checks  $E_B$  resp.  $E_A$

$$E_A = f_3(\text{DHKey}, N_A, N_B, r_B, \text{IOcap}_A, \text{BD\_ADDR}_A, \text{BD\_ADDR}_B)$$

$$E_B = f_3(\text{DHKey}, N_B, N_A, r_A, \text{IOcap}_B, \text{BD\_ADDR}_B, \text{BD\_ADDR}_A)$$

- **step 12:** link key calculation  
key derivation from DHKey,  $N_a$ ,  $N_b$ , and the addresses

$$\text{LK} = f_2(\text{DHKey}, N_{\text{master}}, N_{\text{slave}}, \text{btlk}, \text{BD\_ADDR}_{\text{master}}, \text{BD\_ADDR}_{\text{slave}})$$

- **step 13:** encryption (business as usual)

# ECDH Common Protocol

- domain parameters:  
use secp192r1 = P192, the elliptic curve of order  $r$  over the  $\mathbf{Z}_p$  field defined by  $y^2 = x^3 + ax + b$  which is generated by  $G$ :

$$\begin{aligned} p &= 2^{192} - 2^{64} - 1 \\ a &= -3 \bmod p \\ b &= 2455155546008943817740293915197451784769108058161191238065 \\ r &= 6277101735386680763835789423176059013767194773182842284081 \\ G_x &= 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012 \\ &= 602046282375688656758213480587526111916698976636884684818 \\ G_y &= 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811 \\ &= 174050332293622031404857552280219410364023488927386650641 \end{aligned}$$

note that  $2^{192} - 2^{95} < r < 2^{192}$  and  $r$  is prime

- key agreement function: given an integer  $u$  and a point  $V$ ,  $P192(u, V)$  is the  $x$ -coordinate of the point  $uV$

$$\text{DHKey} = P192(\text{SK}_A, \text{PK}_B) = P192(\text{SK}_B, \text{PK}_A)$$

# The New Bluetooth Menagery

$$f1(U, V, X, Z) = \text{trunc}_{128}(\text{HMAC}_X(U\|V\|Z))$$

$$g(U, V, X, Y) = \text{SHA256}_X(U\|V\|X\|Y) \bmod 2^{32}$$

$$f2(W, N_1, N_2, \text{keyID}, A_1, A_2) = \text{trunc}_{128}(\text{HMAC}_W(N_1\|N_2\|\text{keyID}\|A_1\|A_2))$$

$$f3(W, N_1, N_2, R, \text{IOcap}, A_1, A_2) = \text{trunc}_{128}(\text{HMAC}_W(N_1\|N_2\|R\|\text{IOcap}\|A_1\|A_2))$$

variable	$A_i$	$N_i$	$U$	$V$	$W$	$X$	$Y$	$Z$	keyID	IOcap
# bits	48	128	192	192	192	128	128	8	32	48

- HMAC is HMAC-SHA256
- the value of keyID for “btlk” is 0x62746c6b

# Bluetooth Simple Secure Pairing Variants — i

## Numeric Comparison

### Device A

input:  $PK_A, \widehat{PK}_B$

pick  $N_A \in_U \{0, 1\}^{128}$

set  $r_A = r_B = 0$

$\hat{c}_B \stackrel{?}{=} f_1(\widehat{PK}_B, PK_A, \hat{N}_B, 0)$

$V_A \leftarrow g(PK_A, \widehat{PK}_B, N_A, \hat{N}_B)$   
display  $V_A$

output:  $N_A, \hat{N}_B, r_A, r_B$

### Device B

input:  $\widehat{PK}_A, PK_B$

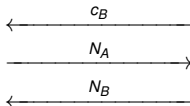
pick  $N_B \in_U \{0, 1\}^{128}$

set  $r_A = r_B = 0$

$c_B \leftarrow f_1(PK_B, \widehat{PK}_A, N_B, 0)$

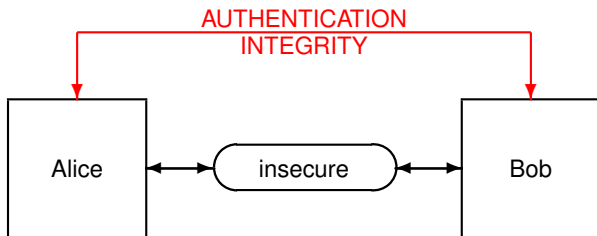
$V_B \leftarrow g(\widehat{PK}_A, PK_B, \hat{N}_A, N_B)$   
display  $V_B$

output:  $\hat{N}_A, N_B, r_A, r_B$



check  $V_A = V_B$

# Security from Human-Monitored Short String Authentication



- communication over a cheap/efficient but insecure channel
- security set up with the help of a short authenticated string (SAS)
- authentication based on human monitoring

# Numeric Comparison Analysis

**Device A**  
input:  $PK_A, \widehat{PK}_B$

pick  $N_A \in_U \{0, 1\}^{128}$   
set  $r_A = r_B = 0$

$\hat{c}_B \stackrel{?}{=} f_1(\widehat{PK}_B, PK_A, \hat{N}_B, 0)$   
 $V_A \leftarrow g(PK_A, \widehat{PK}_B, N_A, \hat{N}_B)$   
display  $V_A$

**output:**  $N_A, \hat{N}_B, r_A, r_B$

**Adversary**

$\xleftarrow{\hat{c}_B} ?? \xleftarrow{c_B}$   
 $\xrightarrow{N_A} ?? \xrightarrow{\hat{N}_A}$   
 $\xleftarrow{\hat{N}_B} ?? \xleftarrow{N_B}$

check  $V_A = V_B$

**Device B**  
input:  $\widehat{PK}_A, PK_B$

pick  $N_B \in_U \{0, 1\}^{128}$   
set  $r_A = r_B = 0$

$c_B \leftarrow f_1(PK_B, \widehat{PK}_A, N_B, 0)$

$V_B \leftarrow g(\widehat{PK}_A, PK_B, \hat{N}_A, N_B)$   
display  $V_B$

**output:**  $\hat{N}_A, N_B, r_A, r_B$

if  $(PK_A, \widehat{PK}_B) \neq (\widehat{PK}_A, PK_B)$ :

Adversary does not know  $V_B$  before he selects  $\hat{N}_A$

Adversary does not know  $V_A$  (cannot influence it) before  $N_A$  is given

# Note on Numerical Comparison

- idea:  $c_B$  is a commitment to  $N_B$  which is revealed after  $N_A$  is received,  $V$ 's are hash of the public keys and  $N$ 's  
if the public keys are corrupted, the adversary must adapt the  $N$ 's so that the  $v$ 's match but no  $N$  is free as soon as one is revealed
- **“just works”** is a variant where no check is made (vulnerable to active attacks)
- presumably, not many human users will carefully compare the 32-bit strings  $V_A$  and  $V_B$

# Bluetooth Simple Secure Pairing Variants — ii

## Passkey Entry

**Device A**

input:  $PK_A, \widehat{PK}_B$

pick  $N_A \in_U \{0, 1\}^{128}$

$C_A \leftarrow f_1(PK_A, \widehat{PK}_B, N_A, r_i)$

$\hat{C}_B \stackrel{?}{=} f_1(\widehat{PK}_B, PK_A, \hat{N}_B, r_i)$

output:  $N_A, \hat{N}_B, r, r$

**Device B**

input:  $\widehat{PK}_A, PK_B$

pick  $N_B \in_U \{0, 1\}^{128}$

$C_B \leftarrow f_1(PK_B, \widehat{PK}_A, N_B, r_i)$

$\hat{C}_A \stackrel{?}{=} f_1(\widehat{PK}_A, PK_B, \hat{N}_A, r_i)$

output:  $\hat{N}_A, N_B, r, r$

type  $r_1 \dots r_k$

FOR  $i = 1$  to  $k$

$\xrightarrow{C_A}$

$\xleftarrow{C_B}$

$\xrightarrow{N_A}$

$\xleftarrow{N_B}$

ENDFOR

keep the last  $N_A$  and  $N_B$

note: not really SAS-based since  $r$  must be secret until the end of the protocol



# Pass Entry Analysis

If  $(PK_A, \widehat{PK}_B) \neq (\widehat{PK}_A, PK_B)$ :

- Adversary cannot forge  $\hat{c}_A$  and  $\hat{c}_B$  with a probability higher than  $\frac{1}{2}$  in each iteration (by trying to guess  $r_i$ )
- So, he cannot pass with probability higher than  $2^{-k}$

# Bluetooth Simple Secure Pairing Variants — iii

## Out-of-Band

### Device A

input:  $PK_A, \widehat{PK}_B$

pick  $r_A \in_U \{0, 1\}^{128}$

$c_A \leftarrow f1(PK_A, PK_A, r_A, 0)$

$c_B \stackrel{?}{=} f1(\widehat{PK}_B, \widehat{PK}_B, r_B, 0)$

pick  $N_A \in_U \{0, 1\}^{128}$

output:  $N_A, \widehat{N}_B, r_A, r_B$

### Device B

input:  $\widehat{PK}_A, PK_B$

pick  $r_B \in_U \{0, 1\}^{128}$

$c_B \leftarrow f1(PK_B, PK_B, r_B, 0)$

$c_A \stackrel{?}{=} f1(\widehat{PK}_A, \widehat{PK}_A, r_A, 0)$

pick  $N_B \in_U \{0, 1\}^{128}$

output:  $\widehat{N}_A, N_B, r_A, r_B$

$\xrightarrow{\text{authenticate}_A(r_A, c_A)}$

$\xleftarrow{\text{authenticate}_B(r_B, c_B)}$

$\xrightarrow{N_A}$

$\xleftarrow{N_B}$

# Bluetooth Low Energy (LE) in v4.0

previously known as WiBree (developped by Nokia)

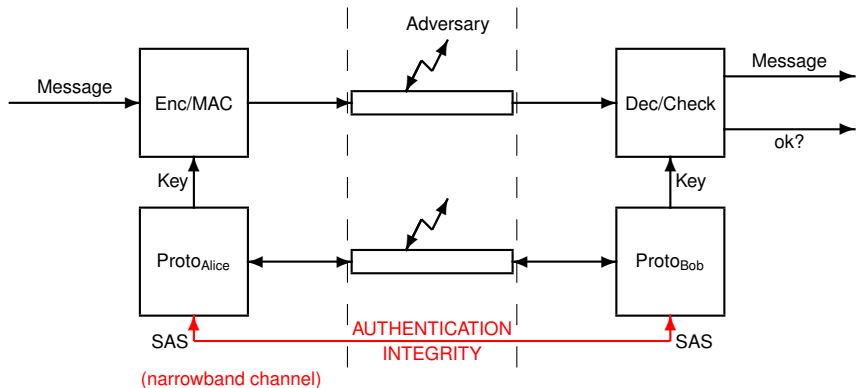
- similar association models, but no public-key crypto anymore
- some ill-designed association model
- a strange key hierarchy with not so much entropy in session key derivation

## 7

## Case Studies I

- Mobile Telephony
- WEP/WPA/WPA2
- Bluetooth
- **Cryptography Based on Short Authenticated Strings**
- Access Control
- Forward Secrecy: the Case of Signal
- Block Chains

# SAS-Based Secure Communication

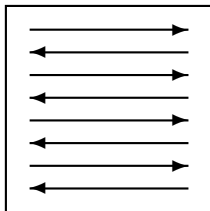


# Message Authentication Protocols

**Alice** ( $ID_A$ )

**input:**  $m$

**Bob**



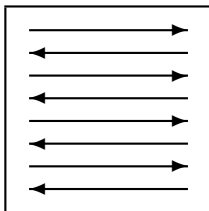
**output:**  $ID, \hat{m}$

- functionality:  $ID = ID_A$  and  $\hat{m} = m$
- security: if  $ID = ID_A$  then  $\hat{m} = m$
- application: semi-A key agreement  
( $m$  is a symmetric key for secure channel so that Bob knows he is talking to Alice)

# Message Cross-Authentication Protocols

**Alice** ( $ID_A$ )  
**input:**  $m_A$

**Bob** ( $ID_B$ )  
**input:**  $m_B$

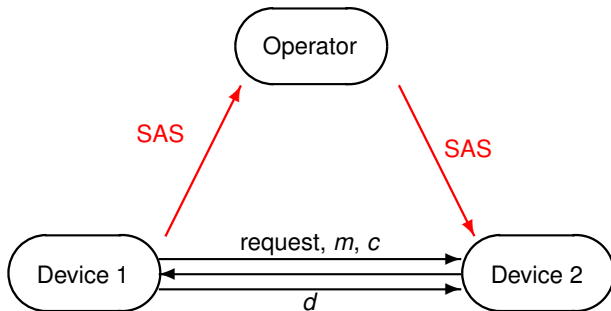


**output:**  $ID_B, m_B$

**output:**  $ID_A, m_A$

- two message authentication protocols at the same time
- application: authenticated key agreement ( $m_A$  and  $m_B$  are Diffie-Hellman public keys)

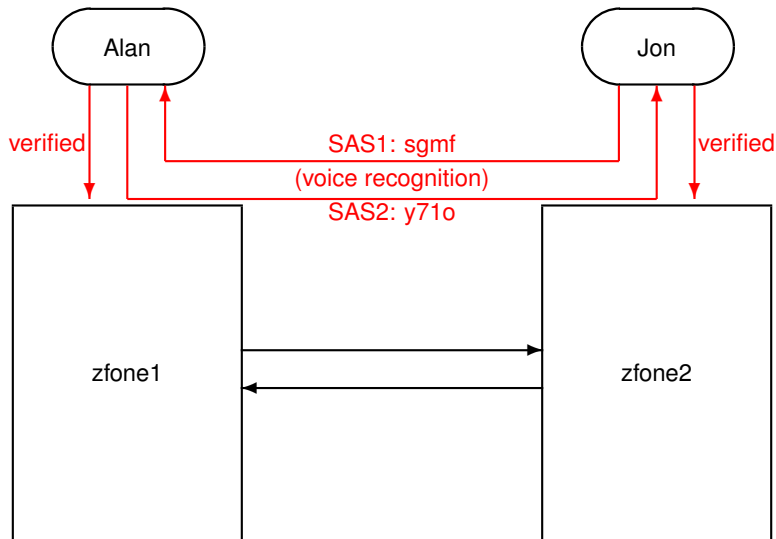
# Application I: Personal Area Network Setup





# Application II: Voice over IP

Existing Standard: ZRTP



# Application III: Peer-to-Peer PGP Channel Setup



# Application IV: Disaster Recovery

- on the road, after a key loss (computer crash, stolen laptop)  
→ set up of a security association
- PKI collapse (company bankrupt, main key sold, act of God)  
→ set up of a security association

# Semi-Authenticated Non-Interactive: Application

Faculté informatique et communications  
Institut de systèmes de communication  
Laboratoire de sécurité et de cryptographie

---



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

**Serge VAUDENAY**

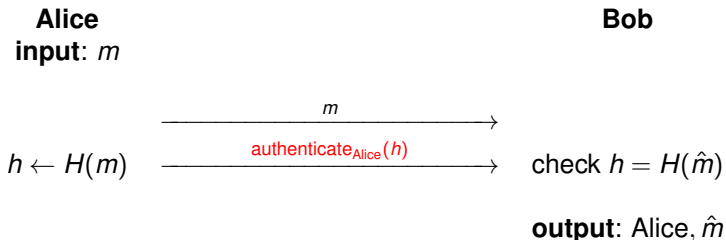
Professeur

EPFL IC ISC LASEC  
INF 241  
Station 14  
CH - 1015 Lausanne

Tél.: +41 21 6937696  
Fax: +41 21 6936870  
serge.vaudenay@epfl.ch  
<http://lasecwww.epfl.ch>

12E7 CAE2 2119 086C DC3D 8EAB 2EA5 9621 5E8C 7956

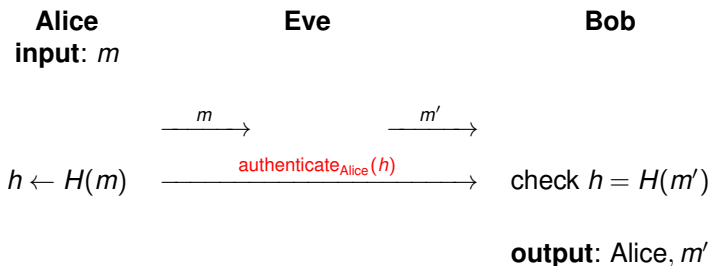
# Folklore (Balfanz-Smetters-Stewart-Chi Wong 2002)



- 😊 efficient, provably security assuming collision resistance
- 😞 this requires SAS of at least 160 bits

# A Collision Attack

if SAS is so short that we can find collisions  $h(m) = h(m')$ ,  $m \neq m'$ ,  
make Alice run the protocol with  $m$  but change the message to Bob to  $m'$



# Pasini-Vaudenay 2006: SAS-Based NIMAP

**Alice**  
input:  $m$

$c \leftarrow \text{commit}(m; r)$

$\text{SAS} \leftarrow H(c)$

$m||r$

$\text{authenticate}_{\text{Alice}}(\text{SAS})$

**Bob**

$\hat{c} \leftarrow \text{commit}(\hat{m}, \hat{r})$

check  $\text{SAS} = H(\hat{c})$

**output:** Alice,  $\hat{m}$

- 😊 provable security, efficient
- 😊 can work with SAS of 80 bits (the least possible for NIMAP)

# Semi-Authenticated Interactive

Vaudenay 2005

**Alice**  
input:  $m$

pick  $R_A \in_U \{0, 1\}^k$

$c \leftarrow \text{commit}(m, R_A; r)$

$\xrightarrow{m||c}$

$\xleftarrow{R_B}$

$\xrightarrow{R_A||r}$

$\text{SAS} \leftarrow R_A \oplus \hat{R}_B$

$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$

**Bob**

pick  $R_B \in_U \{0, 1\}^k$

$\hat{c} \stackrel{?}{=} \text{commit}(\hat{m}, \hat{R}_A; \hat{r})$

check  $\text{SAS} = \hat{R}_A \oplus R_B$

**output:** Alice,  $\hat{m}$

- 😊 provable security, efficient
- 😊 can work with SAS of 20 bits



# Authenticated Interactive

Zimmermann 1995: PGPfone

**Alice**

pick  $x_A, y_A \leftarrow g^{x_A}$

commit to  $(y_A)$  →

←  $y_B$

$z_A \leftarrow \hat{y}_B^{x_A}$

open commitment →

$SAS \leftarrow \text{truncH}(y_A || \hat{y}_B)$

authenticate<sub>Alice</sub>(SAS) →

check SAS is the same

← authenticate<sub>Bob</sub>(SAS)

**output:** Bob,  $z_A$

**Bob**

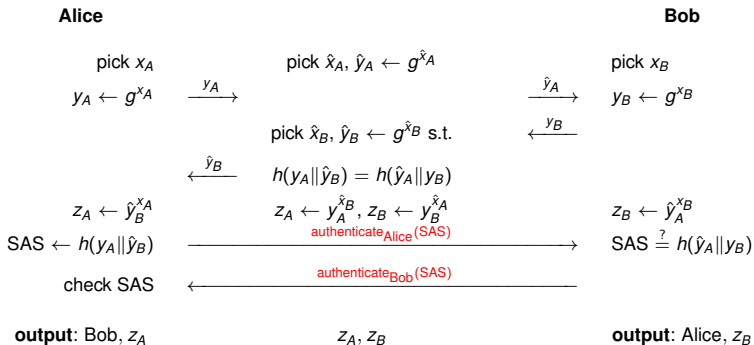
pick  $x_B, y_B \leftarrow g^{x_B}$

$z_B \leftarrow \hat{y}_A^{x_B}$

$SAS \stackrel{?}{=} \text{truncH}(\hat{y}_A || y_B)$

**output:** Alice,  $z_B$

# Attack on a Variant Without Commitment



# Conclusion on Manual Key Establishment

- secure communications over insecure channels *can* be manually set up by a human operator
- public-key -less solutions: although pretty weak, Bluetooth standards *can* offer a pragmatic costless security when properly used
- applications: personal area network, VoIP, peer-to-peer, disaster rescue

# References on SAS-Based Cryptography

- 1 D. Balfanz, D. K. Smetters, P. Stewart, H. Chi Wong.**  
Talking to Strangers: Authentication in Ad-Hoc Wireless Networks.  
In *Network and Distributed System Security Symposium Conference (NDSS 02)*, 2002.
- 2 C. Gehrman, C. Mitchell, K. Nyberg.**  
Manual Authentication for Wireless Devices.  
In *RSA Cryptobytes*, vol. 7, pp. 29–37, 2004.
- 3 S. Vaudenay.**  
Secure Communications over Insecure Channels Based on Short Authenticated Strings.  
In *Advances in Cryptology (CRYPTO'05)*, LNCS vol. 3621, pp. 309–326, 2005.
- 4 S. Pasini, S. Vaudenay.**  
Secure Communications over Insecure Channels Using an Authenticated Channel.  
In *Topics in Cryptology (CT-RSA'06)*, LNCS vol. 3860, pp. 280–294, 2006.
- 5 S. Pasini, S. Vaudenay.**  
SAS-Based Authenticated Key Agreement.  
In *Public Key Cryptography (PKC'06)*, LNCS vol. 3958, pp. 385–409, 2006.

## 7

## Case Studies I

- Mobile Telephony
- WEP/WPA/WPA2
- Bluetooth
- Cryptography Based on Short Authenticated Strings
- **Access Control**
- Forward Secrecy: the Case of Signal
- Block Chains

# Application: Access Control

many scenarios:

- access of human user to a computer
- access of a person to a door: “Sésame”
- access of human user to a mailbox
- access of human user to a service through the Internet

access control = peer authentication

# Password Authentication Protocol (Step 1)

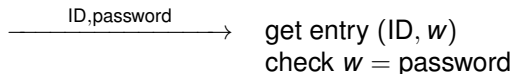
- PROBLEM: authenticate a client to a server
- HYPOTHESIS 1: channel to server keeps confidentiality
- example:
  - physical access
  - secure channel from semi-authenticated setup (client authenticates the server e.g. using a PKI)

# Password Authentication Protocol — i

- server keeps a database of (ID, password) entries
- channel to server keeps confidentiality

**Client**

**Server**



Problem: if adversary has access to database he can get the password

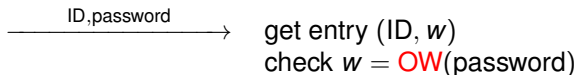


# Password Authentication Protocol — ii

- server keeps a database of  $(ID, OW(\text{password}))$  entries
- channel to server keeps confidentiality

**Client**

**Server**



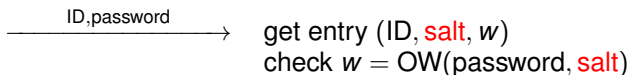
Problem: multi-target inversion attacks  
(specially when password have low entropy)

# Password Authentication Protocol — iii

- server keeps a database of (ID, salt, OW(password, salt)) entries
- channel to server keeps confidentiality

**Client**

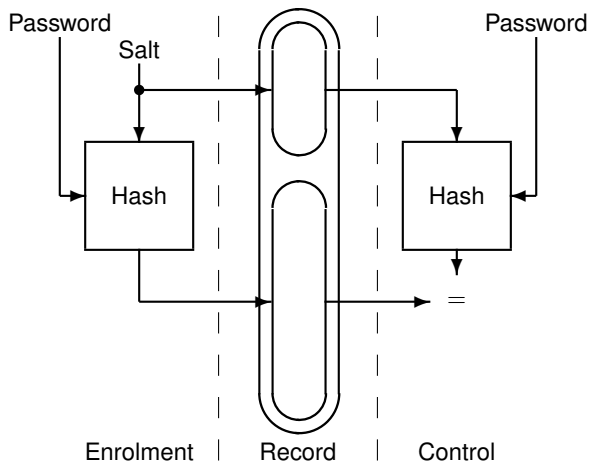
**Server**



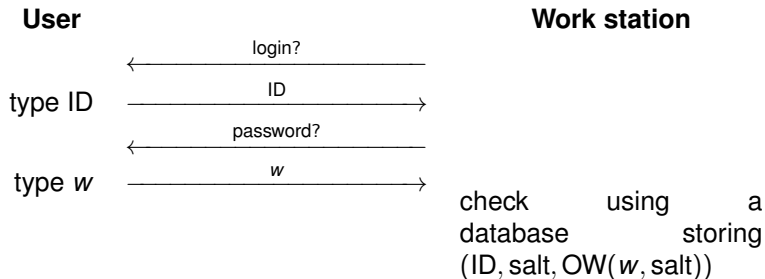
advantages:

- avoid multi-target bruteforce attacks from database  
(does not avoid single-target exhaustive search from database)

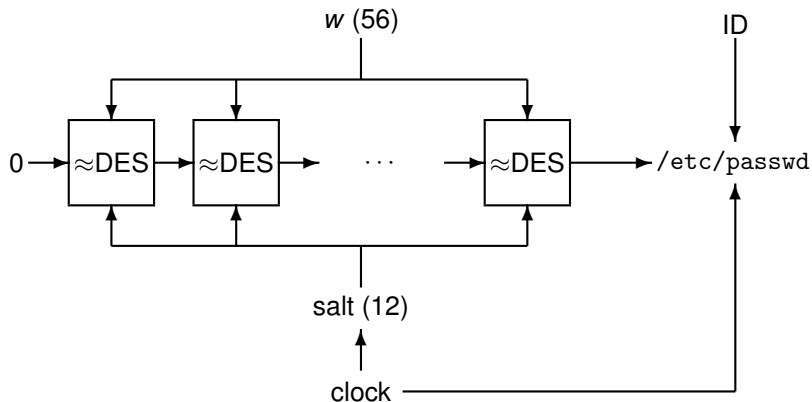
# Password Access Control Using Salt



# Example: UNIX Password Access Protocol



# UNIX Passwords



# Online and Offline Passwords Recovery

	<b>online</b>	<b>offline</b>
method	try to connect using a guess for the password until it works	get a witness look for a guess which is consistent with the witness
countermeasure	<ul style="list-style-type: none"><li>• increasing delay before new attempt</li><li>• blocked after xx trials</li></ul>	<ul style="list-style-type: none"><li>• password with large entropy</li><li>• use salt</li></ul>

# Examples

- Basic Access Control in HTTP [RFC2617]
- IMAP4rev1 [RFC2060]
- tequila authentication at EPFL

# Pros and Cons

## Pros

- the server does not keep the password (only a digest)
- the client need not run any calculation (nice for human clients!)

## Cons

- does not work through a channel without confidentiality protection: the password can be compromised



# Password Authentication Protocol (Step 2)

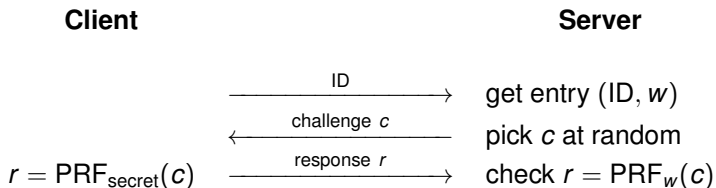
- PROBLEM: authenticate a client to a server
- HYPOTHESIS 2: adversary is passive
- example: unencrypted semi-authenticated channel (client authenticates the server e.g. using a PKI but they are not allowed to use encryption)

# Passive vs Active Adversary

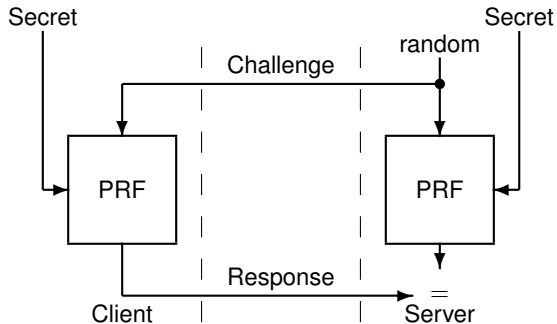
- **passive adversary**: only listen to communications and tries to get credential to later pass access control
- **active adversary**: can interfere with client or server communications e.g. man-in-the-middle

# Challenge/Response Protocol

- server keeps a database of (ID, secret) entries
- adversary is passive



# Challenge/Response Protocol



# Pros and Cons

## Pros

- resistance to passive adversary (if secret has large entropy)

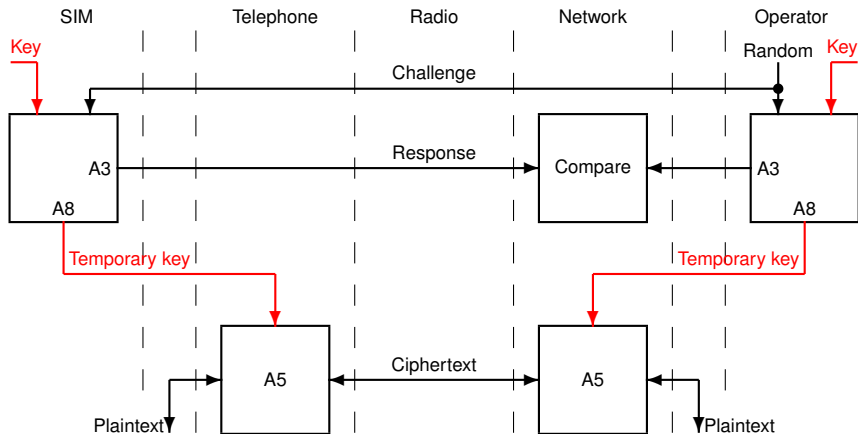
## Cons

- the server must keep the secret and strongly protect the database
- vulnerable to relay attacks
- vulnerable to passive offline attacks (if secret has low entropy)
- vulnerable to active adversary

# Examples

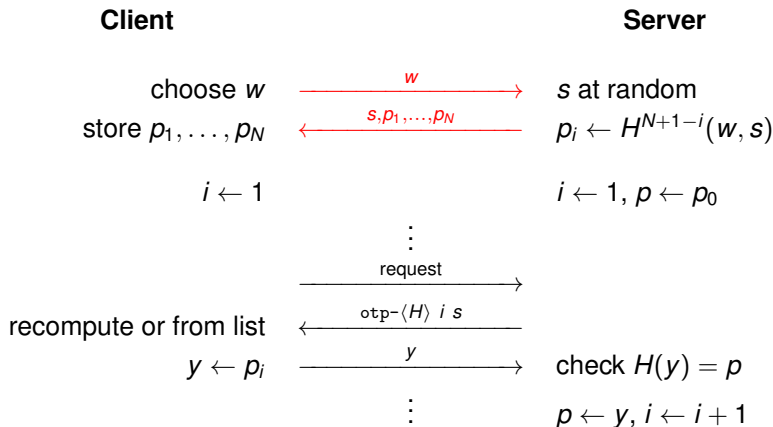
- GSM
- CHAP Access Control in PPP [RFC1334]
- Digest Access Control in HTTP [RFC2617]
- Bluetooth peer authentication
- access control to UBS account (later in this chapter)

# The GSM Case



# S/Key - OTP [RFC2289]

possible hash function  $H$ : md4, md5, sha1



challenges must be authenticated

responses shall be protected against delays in delivery



# Pros and Cons

## Pros

- the server does not keep the secret
- resistance to passive adversary

## Cons

- used with a single server (or securely synchronized ones)
- not user-friendly: user has to work (e.g. wear a long list and check passwords in it)
- still vulnerable to relay attacks

# Human Factor against Password Access Control

- weak passwords: short, trivial (in dictionaries, first name)
- long passwords are hard to remember
- people are lazy (or don't want to be bothered)  
write passwords on post'it, bypass security protocols, ...

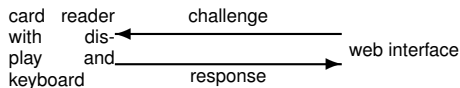
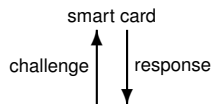
# Alternate Authentication Means

- from **what you know**: password
  - 😊 always available (unless forgotten)
  - 😞 must address the human factor
- from **what you possess**: secure token (smart card, dongle, secureID, key lock)
  - 😊 tamper proof, can perform cryptographic operations
  - 😞 can be stolen, lost, forgotten
- from **what you are**: biometrics
  - 😊 always available
  - 😞 fuzzy, not very secure, threat to humankind

strong authentication = authentication using at least two methods  
(2-way authentication)

example: smart card + PIN code

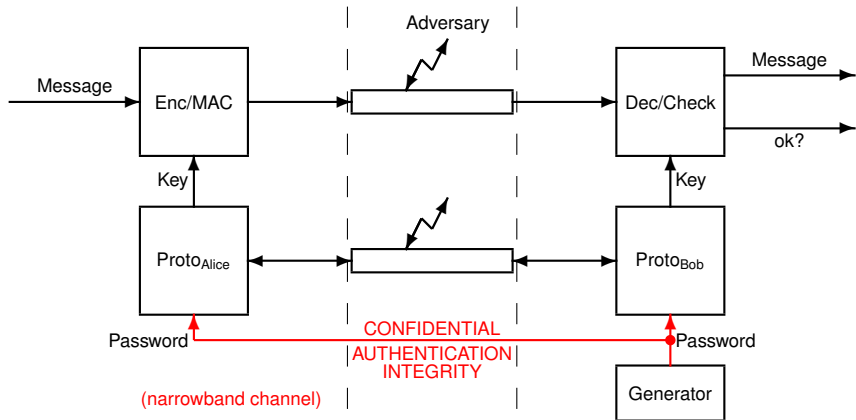
# Example: UBS E-Banking



- 1 type contract number
- 2 insert smart card
- 3 switch calculator on
- 4 type PIN code
- 5 read challenge, type it on calculator keyboard
- 6 read response, type it on browser interface

- smart card + external reader (calculator)
- challenge-response protocol

# Password-based Authenticated Key Exchange (PAKE)



# Password vs Secret Keys

- secret keys are stored by computers (can be pretty long)
- passwords are also kept in human memories
- typically: password have less than 48 bits of entropy

# How to Solve the Problem?

- with no other setup assumptions (no secure token)
- cannot assume a password with large entropy
- find a pragmatic and technical solution
  - leak no information which could be used to run **offline** attacks
  - live with online dictionary attacks (slow down tests, audit, ...)

# Online Dictionary Attack: a Generic Attack

## generic

- 1: **repeat**
- 2: make a new guess  $\hat{w}$  following a dictionary
- 3: simulate Alice with password  $\hat{w}$
- 4: launch an instance of the Bob protocol
- 5: make the simulator and Bob talk to each other
- 6: **until** Bob accepts
- 7: print  $\hat{w}$

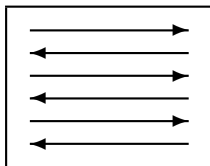
a protocol is secure if this attack is the best one



# Password-Based Authenticated Key Agreement

**Alice**  
password:  $w$   
random tape:  $r_A$

**Bob**  
password:  $w$   
random tape:  $r_B$

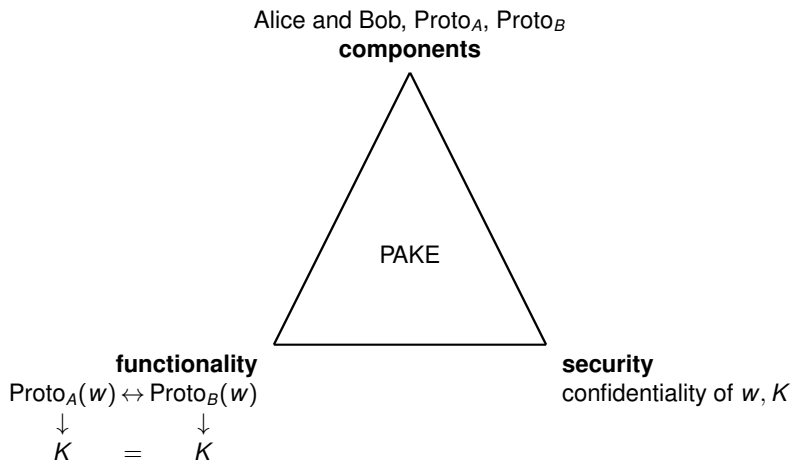


**output:**  $K_A$

**output:**  $K_B$

- functionality:  $K_A = K_B = K$
- security
  - active adversary learns (almost) nothing about  $w$
  - if party ends on  $K$  the active adversary has no clue about  $K$

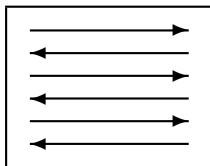
# A New Primitive



# Subproblem: Password-based Access Control

**Alice**  
password:  $w$   
random tape:  $r_A$

**Bob**  
password:  $w$   
random tape:  $r_B$

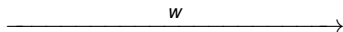


**output:** ok

- functionality: the protocol completes
- security
  - active adversary learns (almost) nothing about  $w$
  - if Bob completes then Alice has the same view on the protocol

# 1st (Bad) Example: Password Access Control

**Alice**  
password:  $w$



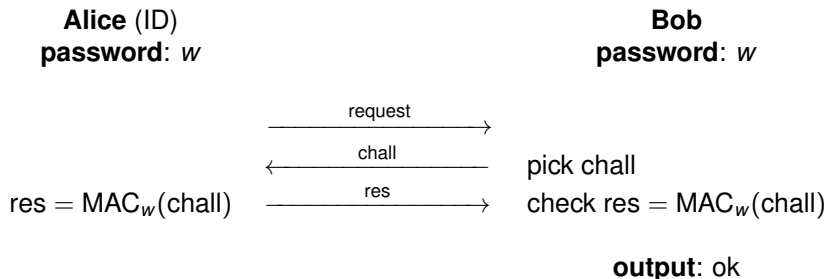
**Bob**  
password:  $w$

check  $w = \hat{w}$

**output:** ok

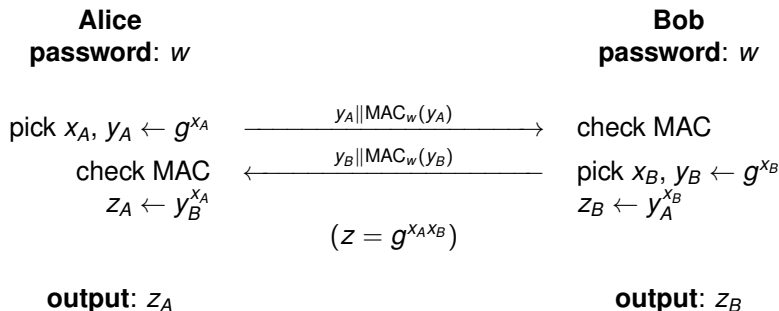
$w$  is disclosed

## 2nd (Bad) Example: Challenge/Response Protocol



subject to offline exhaustive search

# Key Agreement: a (Bad) Idea



subject to offline exhaustive search

# Key Agreement: Another (Bad) Idea

**Alice**  
password:  $w$

**Bob**  
password:  $w$

RSA.Gen  $\rightarrow (N, e, d)$   $\xrightarrow{N, e}$  pick  $K$   
 $\hat{K} \leftarrow \text{Dec}_w(\hat{c})^d \bmod N$   $\xleftarrow{c}$   $c \leftarrow \text{Enc}_w(K^e \bmod N)$

**output:  $\hat{K}$**

**output:  $K$**

if  $K$  can later be tested, offline exhaustive search possible

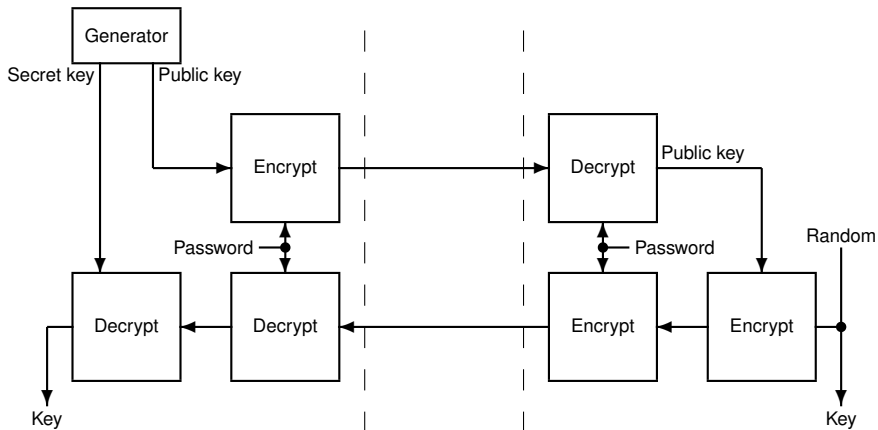
partition attack: eliminate all  $\hat{w}$  such that  $\text{Dec}_{\hat{w}}(c) \geq N$

# Existing Protocols

- Bellovin-Merritt 1992: EKE
  - general construction paradigm
  - can be based on ElGamal, Diffie-Hellman, RSA or other
  - informal (no security proof)
- Lucks 1997: OKE (later broken)
- Wu 1997: SRP (Secure Remote Password), quite popular
- EKE variants based on Diffie-Hellman
  - Bellare-Pointcheval-Rogaway 2000: EKE2
  - Boyko-MacKenzie-Patel 2000: PAK
  - Bellare-Rogaway 2000: AuthA (several variants)
  - Katz-Ostrovski-Yung 2001 (security proof without random oracles)
  - MacKenzie 2002: the PAK suite (PPK, PAK-X, PAK-Y, PAK-Z, ...)
  - Abdalla-Chevassut-Pointcheval 2005: another EKE+AuthA variant
  - others: SPEKE, augmented EKE, B-SPEKE, AMP, Jiang-Gong, ...
- protocols based on RSA
  - MacKenzie-Patel-Swaminathan 2000: SNAP1
  - Zhang 2004: PEKEP



# EKE (Bellare-Meritt 1992)



# EKE (Bellare-Meritt 1992) based on ElGamal

**Alice**  
password:  $w$

pick  $x_A, y_A \leftarrow g^{x_A}$   
pick  $R_A \in_U \{0, 1\}^k$ ,

$c_A \leftarrow \text{enc}_w(y_A)$

$\xrightarrow{c_A}$

$\hat{y}_B \parallel \hat{z} \leftarrow \text{dec}_w(\hat{c}_B)$

$k_A \leftarrow \hat{z} / \hat{y}_B^{x_A}$

$\hat{R}_B \leftarrow \text{dec}_{k_A}(\hat{d}_B)$

$d_A \leftarrow \text{enc}_{k_A}(R_A \parallel \hat{R}_B)$

$\xleftarrow{c_B \parallel d_B}$

$\xrightarrow{d_A}$

check  $R_A = \text{dec}_{k_A}(\hat{e}_B)$

$\xleftarrow{e_B}$

**output:**  $k_A$

**Bob**  
password:  $w$

pick  $x_B, y_B \leftarrow g^{x_B}$   
pick  $R_B \in_U \{0, 1\}^k$

$\hat{y}_A \leftarrow \text{dec}_w(\hat{c}_A)$

pick  $k_B, z \leftarrow k_B \hat{y}_A^{x_B}$

$d_B \leftarrow \text{enc}_{k_B}(R_B)$

$c_B \leftarrow \text{enc}_w(y_B \parallel z)$

$\hat{R}_A \parallel \hat{R} \leftarrow \text{dec}_{k_B}(\hat{d}_A)$

check  $\hat{R} = R_B$

$e_B \leftarrow \text{enc}_{k_B}(\hat{R}_A)$

**output:**  $k_B$

# Wu 2002: Secure Remote Password Protocol (SRP-6)

**Alice**  
password:  $w$

**Bob**  
secret:  $V = g^{H(s||ID_A||w)}$   
db entry:  $ID_A||s||group||V$

	$\xrightarrow{ID_A}$	
check $\widehat{group}$ , $W \leftarrow H(\hat{s}  ID_A  w)$	$\xleftarrow{s  group}$	retrieve $s  group  V$
pick $x_A, y_A \leftarrow g^{x_A}$	$\xrightarrow{y_A}$	pick $x_B, t \leftarrow 3V + g^{x_B}$
$\hat{r} \leftarrow H(y_A  \hat{t})$	$\xleftarrow{t}$	$r \leftarrow H(\hat{y}_A  t)$
$k_A \leftarrow H((\hat{t} - 3g^W)^{x_A + \hat{r}W})$		$k_B \leftarrow H((\hat{y}_A V^r)^{x_B})$
$d_A \leftarrow H(y_A  \hat{t}  k_A)$	$\xrightarrow{d_A}$	check $\hat{d}_A = H(\hat{y}_A  t  k_B)$
check $\hat{d}_B = H(y_A  d_A  k_A)$	$\xleftarrow{d_B}$	$d_B \leftarrow H(\hat{y}_A  \hat{d}_A  k_B)$

**output:**  $H(k_A)$

**output:**  $H(k_B)$

(group =  $(g, p)$ ,  $g$  generator of  $\mathbf{Z}_p^*$ ,  $p$  and  $\frac{p-1}{2}$  prime)

# References on Password-Based Cryptography

- **C. Boyd, A. Mathuria.** Protocols for Authentication and Key Establishment. Information Security and Cryptography, Springer Verlag, 2003.
- **S. M. Bellovin, M. Merritt.** Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *IEEE symposium on Research in Security and Privacy*, IEEE Computer Society Press, pp. 72–84, 1992.

## 7

## Case Studies I

- Mobile Telephony
- WEP/WPA/WPA2
- Bluetooth
- Cryptography Based on Short Authenticated Strings
- Access Control
- **Forward Secrecy: the Case of Signal**
- Block Chains

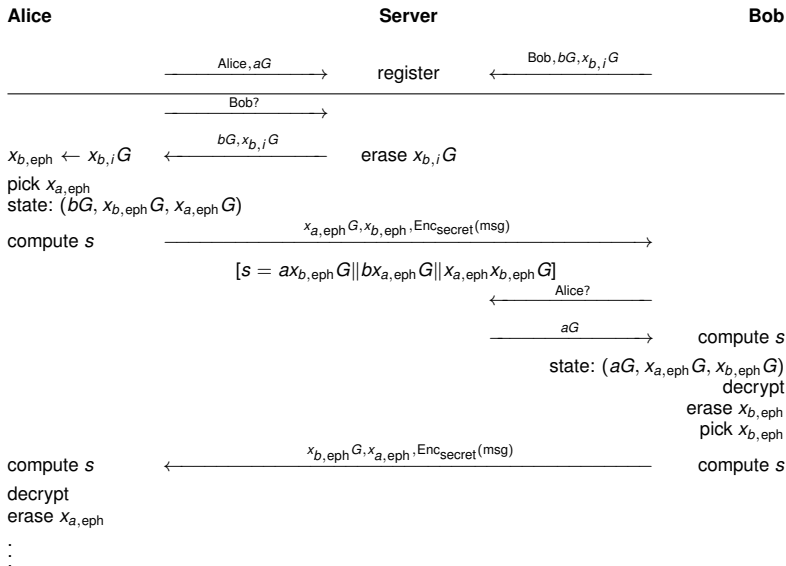
# Signal

used in WhatsApp

- **secure messaging** (confidentiality, authenticity, integrity of messages)
- **forward and future secrecy** (confidentiality preserved even though secrets leak)
- **deniability** (no transferable proof of message authorship leaks)
- **asynchronous** (can be done offline)
- detect replay/reorder/deletion attacks
- allow decryption of out-of-order messages
- don't leak metadata

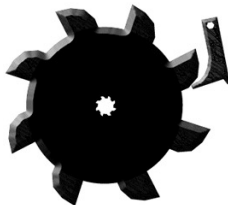
# Initial Key Agreement

(keys are in Curve25519)



# Ratchet

A ratchet is a mechanical device which can only move forward.



- **forward secrecy:** protects past sessions against future compromises of *long-term* secret keys
- **future secrecy:** protects future sessions against compromises of *ephemeral* secret keys



# Double Ratchet in Signal

- 3DH: a ratchet for every time the direction of exchange changes  
needs synchronization between the two participants  
good forward and future secrecy
- a ratchet for every message in the same direction  
no real future secrecy  
plausible deniability

# Ratchet for Messages

given  $x_{a,eph} x_{b,eph} G$ , Alice and Bob derive a sequence  $CK_0, MK_0, CK_1, MK_1, \dots$

$$MK_{i+1} = \text{HMAC-SHA256}_{CK_i}(0)$$

$$CK_{i+1} = \text{HMAC-SHA256}_{CK_i}(1)$$

the message  $i$  is encrypted using  $MK_i$  with encrypt-then-MAC using AES256 and HMAC-SHA256

other techniques to send sequence numbers, total number of messages, etc

## 7 Case Studies I

- Mobile Telephony
- WEP/WPA/WPA2
- Bluetooth
- Cryptography Based on Short Authenticated Strings
- Access Control
- Forward Secrecy: the Case of Signal
- **Block Chains**

# Bitcoins

- virtual currency
- launched in 2009 by an anonymous guy (pseudo Satoshi Nakamoto)
- **completely decentralized**, there is no authority
- anyone creates its own account
- broadcast transactions on a public ledger

# A Bitcoin Transaction

*“I, pk, holder of UTXO  $link_1, \dots, link_n$  pay  $x_1$  to  $pk_1, \dots, x_m$  to  $pk_m$ ”  
[signature]*

- UTXO = unspent transaction output
- requirement:  $x_1 + \dots + x_m$  equals sum of given UTXO
- then, amounts from  $[link_1], \dots, [link_n]$  to  $pk$  become spent and amounts from transaction become new UTXO with a link
- problem: how to make sure that UTXO is really unspent
- equivalent problem: how to make everybody “see” the same list of transactions

# Block Chain

a block from the block chain:

- hash of the previous block (except for the genesis block)
- list of transactions from the last period
- proof-of-work (PoW) based on the above

scheme for miners (every 10 minutes):

- take the longest valid block chain
- collect all broadcast valid transactions with respect to this chain
- make a new block and PoW
- broadcast it
- (the block can be used as an UTXO reward to the miner)

# Proof-ok-Work

block shall contains for PoW value such that

SHA256(block) starts with 69 zero bits

69 is the difficulty of June 2016  
it is constantly calibrated

# Conclusion

- Lightweight networks based on conventional cryptography only (GSM, Bluetooth, ...)
- Although limited, we can make many protocols with only conventional cryptography
- Assembling cryptographic primitives in a protocol is not trivial
- access control based on
  - what you know (password)
  - what you have (a key in a secure token for challenge-response)
  - what you are (biometrics)
- New notions: forward secrecy, plausible deniability, block chain, proof-of-work



# References

- **Borisov-Goldberg-Wagner.** Intercepting Mobile Communications: the Insecurity of 802.11. In *MOBICOM 2001*, ACM.
- **Jakobsson-Wetzel.** Security Weaknesses in Bluetooth. In *CT-RSA 2001*, LNCS 2020.
- **Vaudenay.** On Bluetooth Repairing: Key Agreement based on Symmetric-Key Cryptography. In *CISC 2005*, LNCS 3822.
- **Beck-Tews.** Practical Attacks against WEP and WPA. In *WiSec 2009*, ACM 2009.

# Must Be Known

- GSM security infrastructure
- mobile telephony security
- Bluetooth pairing
- challenge-response protocol
- password-based cryptography
- techniques for access control
- how PAKE works
- forward secrecy

# Train Yourself

- bad EKE variant:  
final exam 2014–15 ex4













- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography**
- 9 Trust Establishment
- 10 Case Studies II

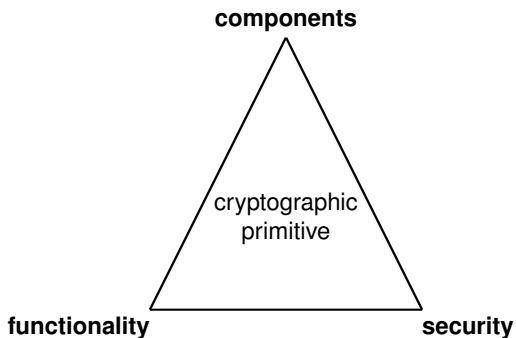
# Roadmap

- Diffie-Hellman: new directions in cryptography
- RSA standards for encryption and signature
- the ElGamal signature dynasty

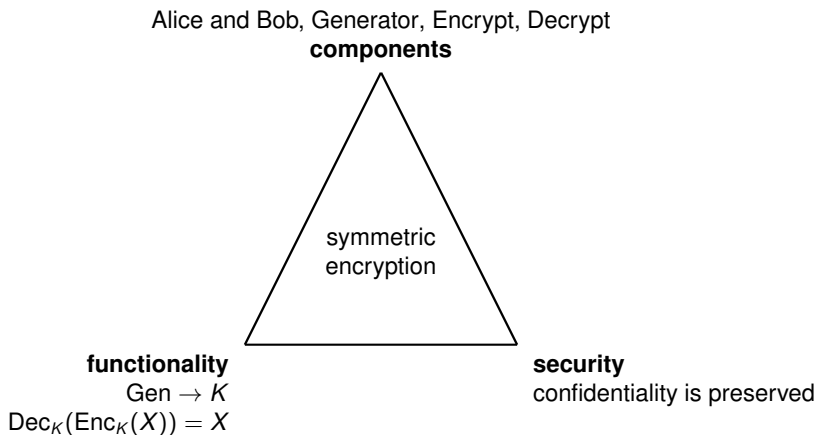
## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

# Cryptographic Primitive (Reminder)



# Symmetric Encryption (Reminder)



# Diffie-Hellman

“New Directions in Cryptography” (1976)

[Merkle, Hellman, Diffie]

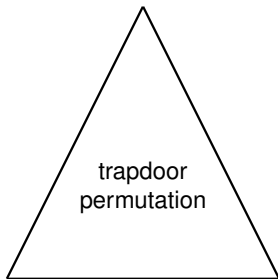
- notion of “trapdoor permutation” (no instance)
- building a public-key cryptosystem from it
- building a digital signature scheme from it
- key agreement protocol

# Trapdoor Permutation

- we use an encryption Perm that is easy to compute in one way
- ...but hard in the other (to compute InvPerm)
- ...except using a trapdoor  $K$

# Trapdoor Permutation

Alice and Bob, Generator, Perm, InvPerm  
**components**



**functionality**

$\text{Gen} \rightarrow (\text{param}, K)$

$\text{InvPerm}_K(\text{Perm}_{\text{param}}(X)) = X$

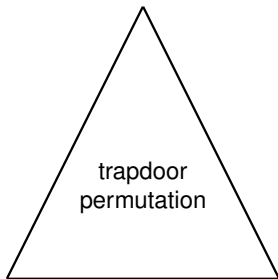
**security**

confidentiality is preserved



# Public-Key Cryptosystem

Alice and Bob, Generator, Encrypt, Decrypt  
**components**



**functionality**

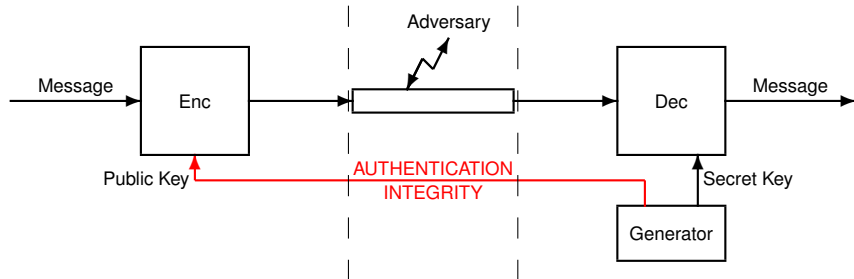
$\text{Gen} \rightarrow (K_p, K_s)$   
 $\text{Dec}_{K_s}(\text{Enc}_{K_p}(X)) = X$

**security**

confidentiality is preserved

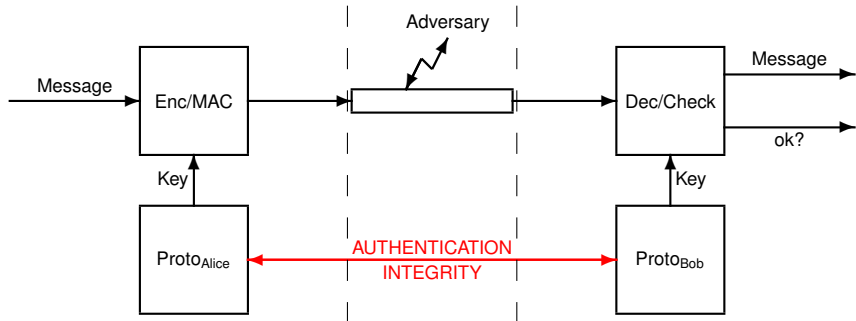
# Confidentiality using an Authenticated Channel

## Public Key Cryptosystem

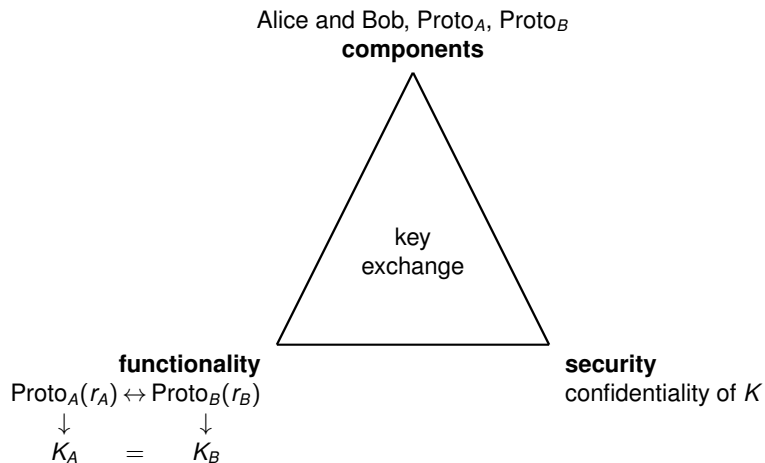


# Confidentiality using an Authenticated Channel

## Key Exchange Protocol



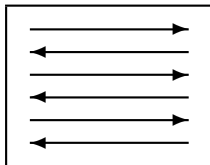
# Key Exchange Protocol



# Key Exchange Protocol

**Alice**  
random tape:  $r_A$

**Bob**  
random tape:  $r_B$



**output:**  $K_A$

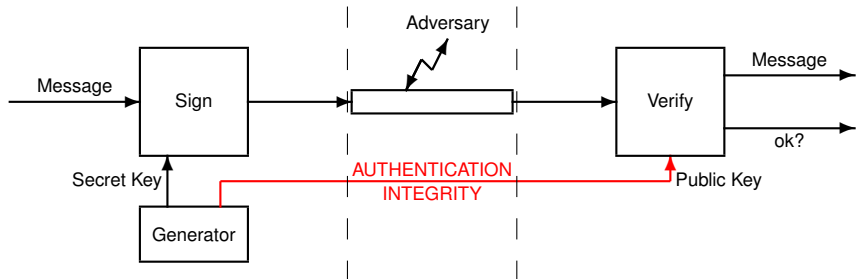
**output:**  $K_B$

- functionality:  $K_A = K_B = K$
- security: passive adversary cannot infer  $K$  from the exchanges

# Terminology

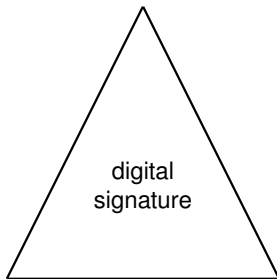
- **key exchange**: there is no exchange of keys, just exchange of data to derive a common secret key  
often assumes no prior common secret
- = **key agreement**: same
- = **key establishment**: same (may be more general)
- **key transfer**: one participant chooses a key and sends it to the second participant

# Digital Signature Scheme



# Digital Signature Primitive

Alice and Bob, Gen, Sig, Ver  
**components**



digital  
signature

**functionality**

$\text{Gen} \rightarrow (K_p, K_s)$

$\text{Ver}_{K_p}(\text{Sig}_{K_s}(X; r)) = X$

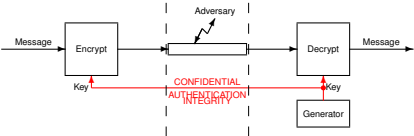
**security**

signature is non-repudiable

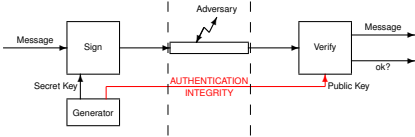
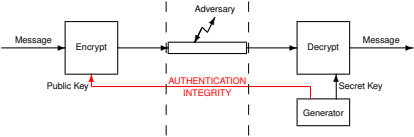
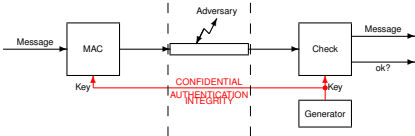


# Big Picture

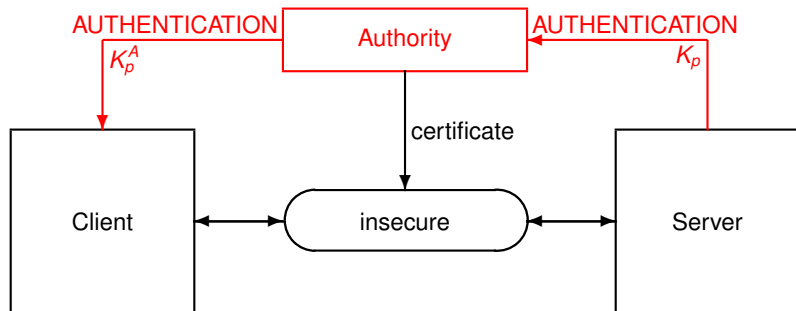
## confidential transmission



## authenticated transmission



# Application: Certificates



certificate = signature("I certify that public key  $K_p$  belongs to  $S$ ")

## Public-Key Cryptography

- Public-Key Cryptography
- **Diffie-Hellman Key Exchange**
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

# Static versus Ephemeral Diffie-Hellman

- Ephemeral DH:  $X$  and  $Y$  are fresh (and destroyed after protocol completes)
- Static DH:  $X$  and  $Y$  are used like public keys
- Semi-static DH: one key is fixed (public key), the other is fresh

# Ephemeral Diffie-Hellman Key Agreement Protocol

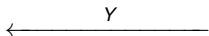
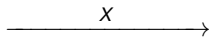
Assume a group generated by some  $g$

**Alice**

**Bob**

pick  $x$  at random

$$X \leftarrow g^x$$



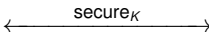
$$K \leftarrow Y^x$$

$$(K = g^{xy})$$

pick  $y$  at random

$$Y \leftarrow g^y$$

$$K \leftarrow X^y$$



# Semi-Static Diffie-Hellman Key Agreement Protocol

Assume a group generated by some  $g$

**Alice**

secret key:  $x$

public key:  $X = g^x$

**Bob**

pick  $y$  at random

$Y \leftarrow g^y$

$K \leftarrow X^y$

$K \leftarrow Y^x$

$(K = g^{xy})$

secure<sub>K</sub>

# Static Diffie-Hellman Key Agreement Protocol

Assume a group generated by some  $g$

**Alice**

secret key:  $x$   
public key:  $X = g^x$

$$K \leftarrow Y^x$$

$$(K = g^{xy})$$

**Bob**

secret key:  $y$   
public key:  $Y = g^y$

$$K \leftarrow X^y$$

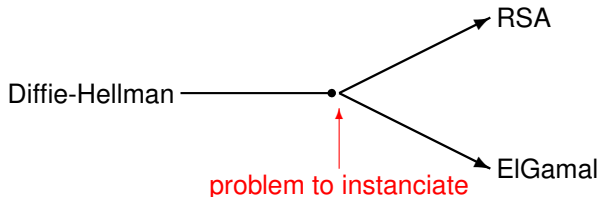
← secure<sub>K</sub> →

# Forward Secrecy

- **forward secrecy:** communication is still private if long term secret keys are disclosed  
example: ephemeral Diffie-Hellman (no long term secret)
- **no forward secrecy:** communication might be decrypted if long term secret keys leak in the future  
example: static or semi-static Diffie-Hellman



# Diffie-Hellman Cryptography



- trapdoor permutation: operation in  $\mathbf{Z}_n^*$  which can be inverted with the factorization of  $n$
- probabilistic encryption: encryption returns  $g^x$  along with  $\text{symEnc}_{\text{KDF}(Y^x)}(\text{message})$  for  $Y^x = \text{DH}(g, g^x, Y)$

## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- **RSA Cryptography**
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

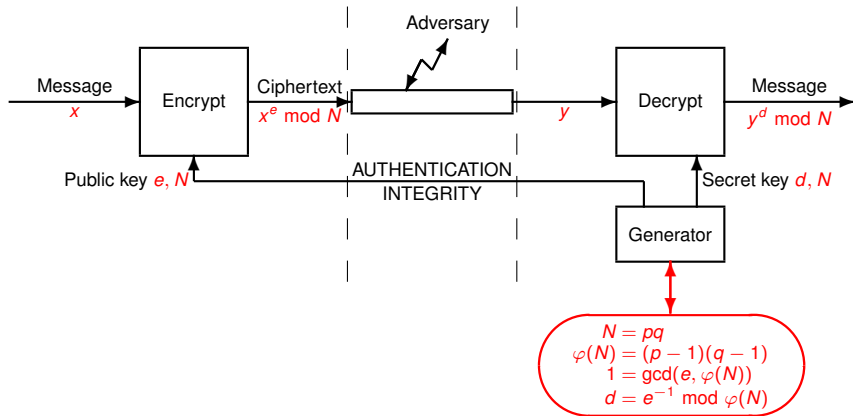
# Rivest-Shamir-Adleman (RSA)

(1978)

[Shamir, Rivest, Adleman]

- concrete trapdoor permutation
- → public-key cryptosystem
- → signature scheme

# Plain RSA



# Why “Plain” RSA

plain RSA

- = textbook RSA
- = vanilla RSA
- = raw RSA
- = RSA for mathematicians

in practice, things are a little more complicated because

- messages are not elements of  $\mathbf{Z}_N$
- RSA has homomorphic properties ( $\text{Enc}(ab) = \text{Enc}(a)\text{Enc}(b)$ ) which are quite dangerous
- RSA engineering leads us to security concerns

# PKCS#1v1.5

(Modulus of  $k$  bytes, message  $M$  of at most  $k - 11$  bytes.)

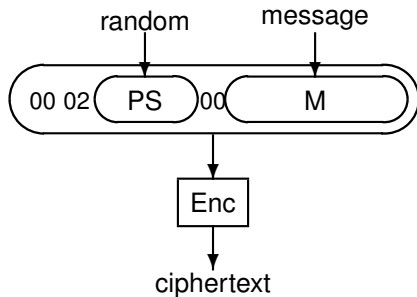
Encryption:

- 1 generate a pseudorandom string PS of non-zero bytes so that  $M\|PS$  is of  $k - 3$  bytes
- 2 construct string  $00\|02\|PS\|00\|M$  of  $k$  bytes
- 3 convert it into an integer
- 4 perform the plain RSA encryption
- 5 convert the result into a string of  $k$  bytes

Decryption:

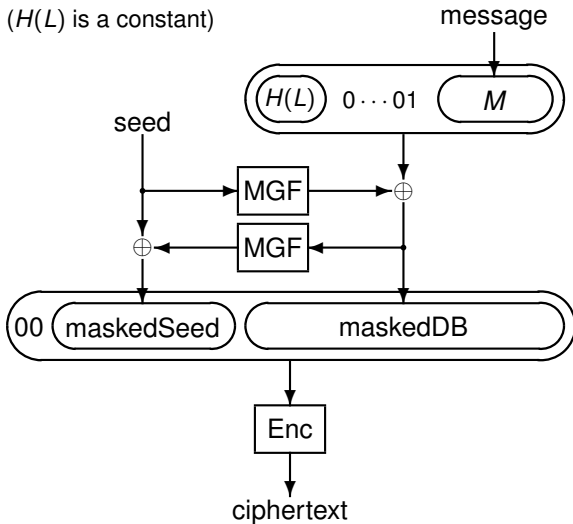
- 1 convert the ciphertext into an integer, reject it if it is greater than the modulus
- 2 perform the plain RSA decryption and obtain another integer
- 3 convert back the integer into a byte string
- 4 check that the string has the  $00\|02\|PS\|00\|M$  format for some byte strings PS and  $M$  where PS has no zero bytes
- 5 output  $M$

# PKCS#1v1.5 Encryption



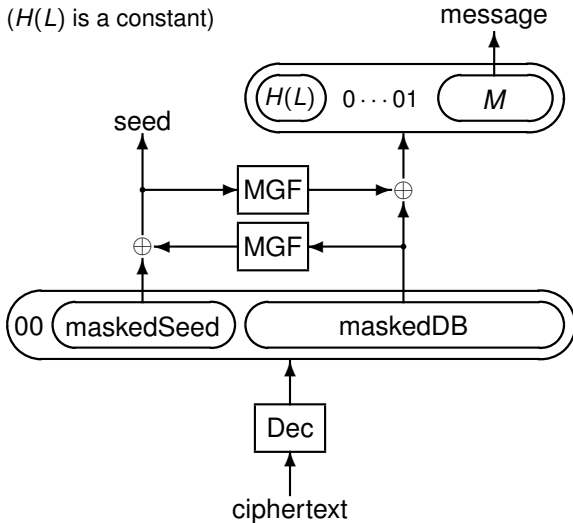
# RSA-OAEP Encryption

( $H(L)$  is a constant)





# RSA-OAEP Decryption



# Mask Generation Function in RSA-OAEP

The PKCS specifications further suggests an mask generation function MGF1 which is based on a hash function. The MGF1 <sub>$\ell$</sub> ( $x$ ) string simply consists of the  $\ell$  leading bytes of

$$H(x\|00000000)\|H(x\|00000001)\|H(x\|00000002)\|\dots$$

in which  $x$  is concatenated to a four-byte counter.

# Rabin Cryptosystem

**Set up:** find two prime numbers  $p$  and  $q$ , set  $N = pq$  and pick a random  $B \in \mathbf{Z}_N$  (e.g.  $B = 0$ )

**Messages:**  $x \in \mathbf{Z}_N$

**Public key:**  $B, N$

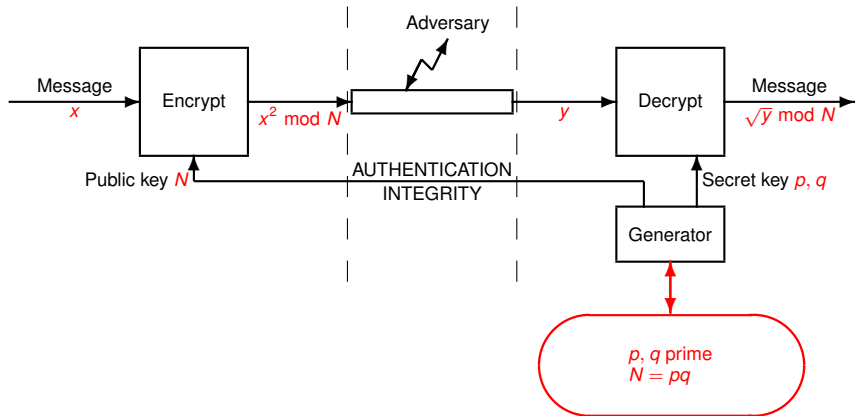
**Secret key:**  $B, p, q$

**Encryption:**  $E(x) = x(x + B) \bmod N$

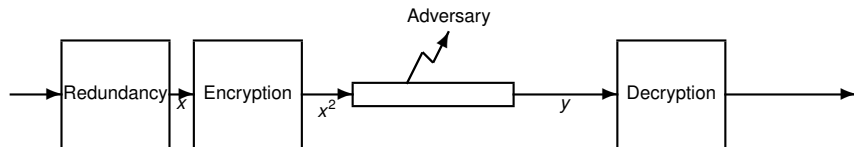
**Decryption:**  $D(y)$  is one of the four square roots of  $\frac{B^2}{4} + y$  minus  $\frac{B}{2}$

$$y = x(x + B) \iff \left(x + \frac{B}{2}\right)^2 = \frac{B^2}{4} + y$$

# Plain Rabin Encryption ( $B = 0$ )



# Ensuring Non-Ambiguity in the Decryption

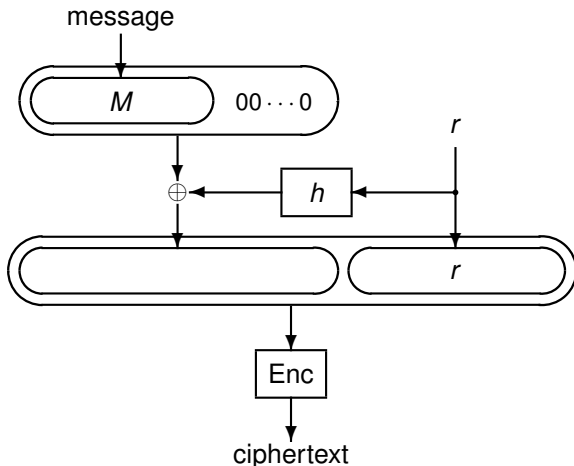


- we add redundancy in the plaintext so that valid plaintexts are sparse
- we make sure that no other square root has valid redundancy (hard without  $K_S$ )
- we take the only expected square root with valid redundancy
- we reject ciphertexts which fail to decrypt

# Rabin Complexity

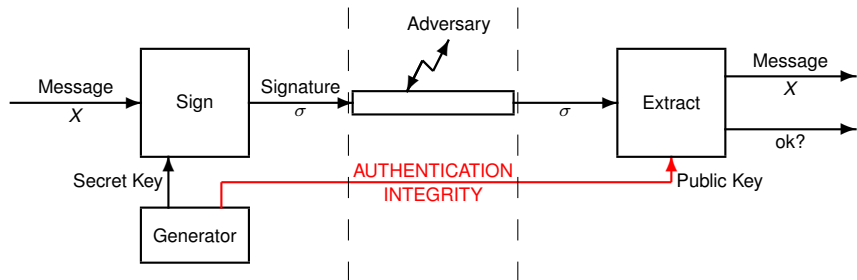
- Generator:  $\mathcal{O}(\ell^4)$  (prime numbers generation)
- Encryption:  $\mathcal{O}(\ell^2)$
- Decryption:  $\mathcal{O}(\ell^3)$

# SAEP: Simple OAEP Padding for Rabin



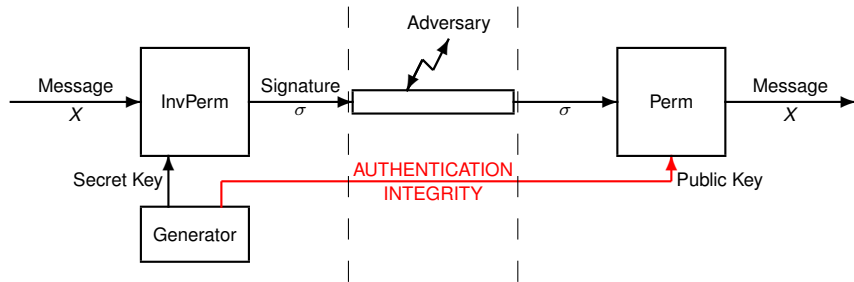
pad with enough 0's to ensure non-ambiguity

# Signature with Message Recovery





# Trapdoor Permutation to Signature with Message Recovery



# Plain RSA Signature

**Set up:** find two random different prime numbers  $p$  and  $q$  of size  $\frac{\ell}{2}$  bits. Set  $N = pq$ . Pick a random  $e$  until  $\gcd(e, (p-1)(q-1)) = 1$ . (Sometimes we pick special constant  $e$  like  $e = 17$  or  $e = 2^{16} + 1$ .) Set  $d = e^{-1} \bmod ((p-1)(q-1))$ .

**Secret key:**  $K_s = (d, N)$ .

**Public key:**  $K_p = (e, N)$ .

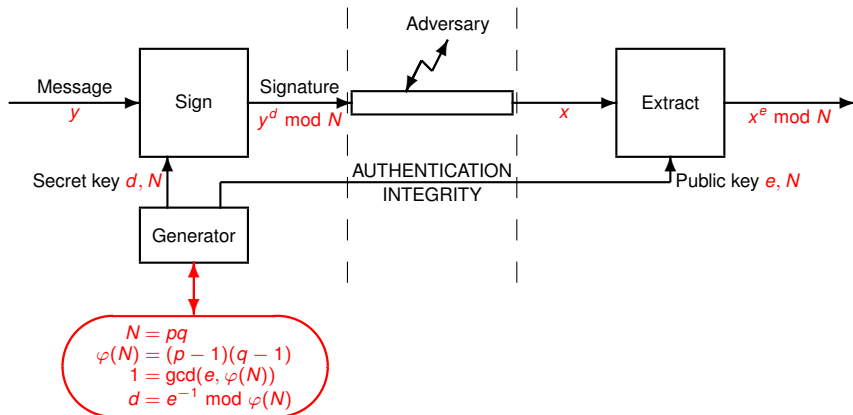
**Message:** an element  $y \in \mathbf{Z}_N$ .

**Signature generation:**  $x = y^d \bmod N$ .

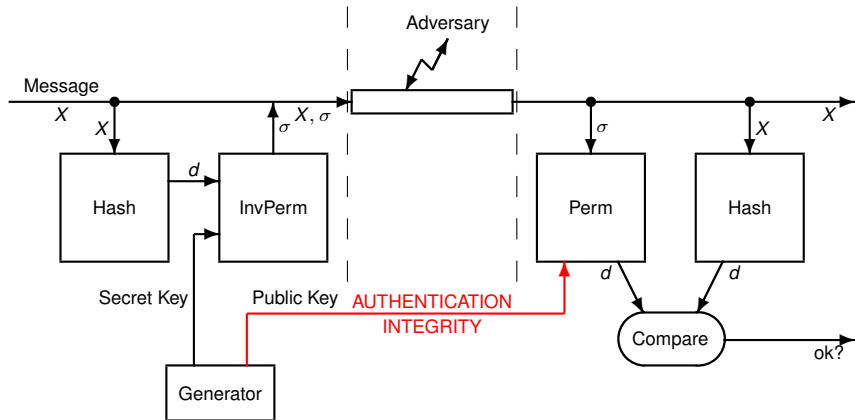
**Extraction:**  $y = x^e \bmod N$ .

(Signature with message recovery)

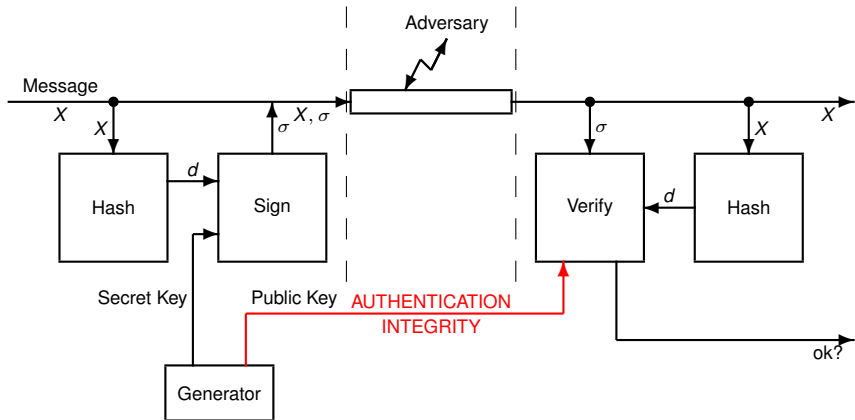
# Plain RSA Signature



# Trapdoor Permutation to Signature



# More Generally: Hash-and-Sign Paradigm



# PKCS#1v1.5

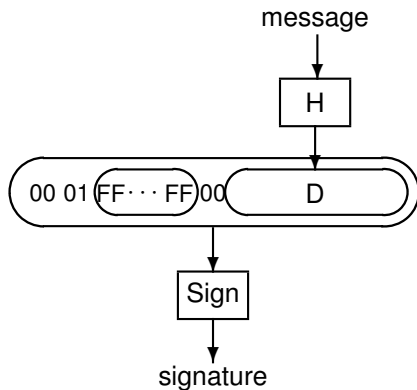
We are given a modulus  $N$  of  $k$  bytes.

- 1 hash the message (for instance with SHA-1) and get a message digest.
- 2 encode the message digest and the identifier of the hash algorithm into a string  $D$ .
- 3 pad it with a zero byte to the left, then with many FF bytes in order to reach a length of  $k - 2$  bytes, then with a 01 byte. We obtain  $k - 1$  bytes.
- 4 This byte string  $00\|01\|FF \dots FF\|00\|D$  is converted into an integer.
- 5 compute the plain RSA signature.
- 6 convert the result into a string of  $k$  bytes.

# Signature Verification

- 1 convert the signature into an integer. Reject it if it is greater than the modulus.
- 2 perform the plain RSA verification and obtain another integer.
- 3 convert back the integer into a byte string.
- 4 check that the string has the  $00\|01\|FF \dots FF\|00\|D$  format for a byte string  $D$ .
- 5 decode the data  $D$  and obtain the message digest and the hash algorithm. Check that the hash algorithm is acceptable.
- 6 hash the message and check the message digest.

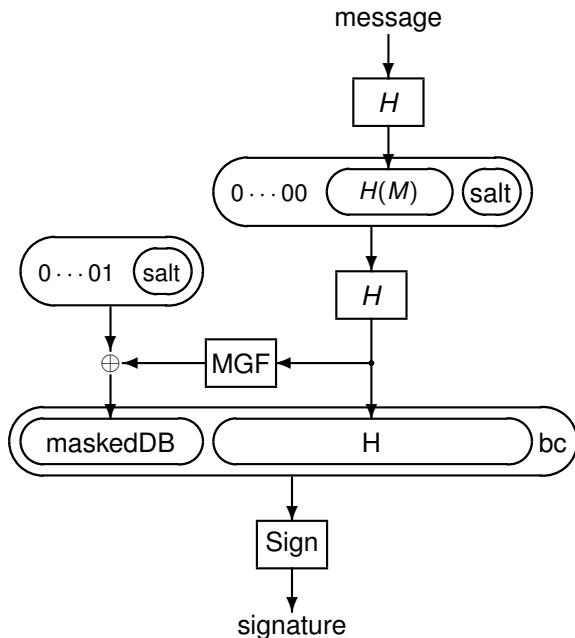
# PKCS#1v1.5 Signature



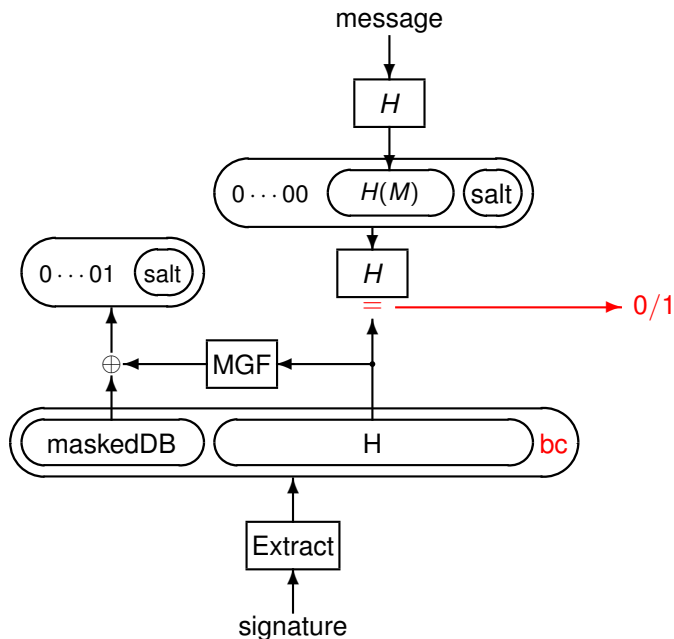
RSA signature without message recovery



# RSA-PSS



# RSA-PSS Verification



## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- **ElGamal Cryptography**
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

# ElGamal Signature

**Public parameters:** a large prime number  $p$ , a generator  $g$  of  $\mathbf{Z}_p^*$ .

**Set up:** generate a random  $x \in \mathbf{Z}_{p-1}$  and compute  
 $y = g^x \bmod p$ .

**Secret key:**  $K_s = x$ .

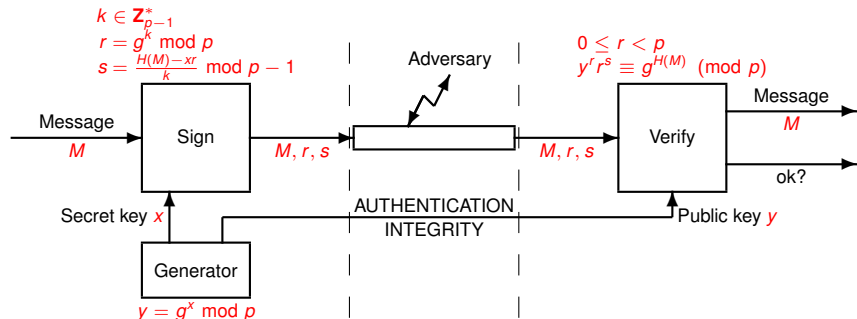
**Public key:**  $K_p = y$ .

**Message digest:**  $h = H(M) \in \mathbf{Z}_{p-1}$ .

**Signature generation:** pick a random  $k \in \mathbf{Z}_{p-1}^*$ , compute  
 $r = g^k \bmod p$  and  $s = \frac{h-xr}{k} \bmod p-1$ , the signature is  
 $\sigma = (r, s)$ .

**Verification:** check that  $y^r r^s \equiv g^h \pmod{p}$  and  $0 \leq r < p$ .

# ElGamal Signature



$p$  prime  
 $g$  generator of  $\mathbf{Z}_p^*$

# Drawbacks of ElGamal Signatures

- signatures are pretty long
- security issues related to subgroups
- lack of security proof for *arbitrary* public parameter

# The ElGamal Dynasty

- 1984 ElGamal signatures
- 1989 Schnorr signatures: introduced  $p$  and  $q$
- 1995 DSA: US signatures
- 1995 Nyberg-Rueppel signatures
- 1997 Pointcheval-Vaudenay signatures
- 1998 KCDSA: Korean signatures
- 1998 ECDSA
- ...

# Generating the Public Parameters

- pick a prime number  $q$
- take a random  $p = aq + 1$  until it is prime
- take a random number in  $\mathbf{Z}_p^*$ , raise it to the power  $a$  modulo  $p$ , and get  $g$
- if  $g = 1$ , try again (otherwise, it must be of order  $q$  in  $\mathbf{Z}_p^*$ )



# Benefits

- signatures are shorter
- no proper subgroup (only  $\{1\}$  and the group itself)
- some form of provable security (related to interactive proofs)

# DSA Signature (DSS)

**Public parameters**  $(p, q, g)$ : pick a 160-bit prime number  $q$ , a large prime number  $p = aq + 1$ ,  $h$  of  $\mathbf{Z}_p^*$  raised to the power  $a$ ,  $g = h^a \bmod p$  such that  $g \neq 1$  (an element of order  $q$ ).

**Set up:** pick  $x \in \mathbf{Z}_q$  and compute  $y = g^x \bmod p$ .

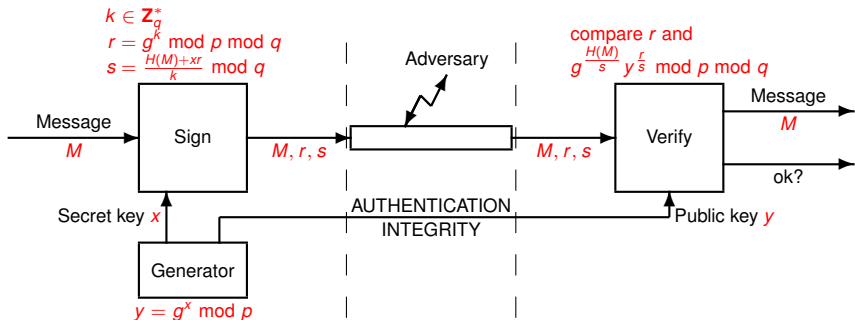
**Secret key:**  $K_s = x$ .

**Public key:**  $K_p = y$ .

**Signature generation:** pick a random  $k \in \mathbf{Z}_q^*$ , compute  $r = (g^k \bmod p) \bmod q$ , and  $s = \frac{H(M) + xr}{k} \bmod q$ , the signature is  $\sigma = (r, s)$ .

**Verification:** check that  $r = \left( g^{\frac{H(M)}{s}} \bmod q y^{\frac{r}{s}} \bmod q \bmod p \right) \bmod q$ .

# DSA Signature



$q$  prime  
 $p = aq + 1$  prime  
 $g = \text{random}^a \bmod p > 1$

# ECDSA

**Public parameters:** we use a field of cardinality  $q$  (either a power of 2, or a large prime), an elliptic curve  $C$  defined by two field elements  $a$  and  $b$ , a prime number  $n$  larger than  $2^{160}$ , and an element  $G$  of  $C$  of order  $n$ . (The elliptic curve equation over  $\text{GF}(q)$  is  $y^2 + xy = x^3 + ax^2 + b$  in the characteristic two case and  $y^2 = x^3 + ax + b$  in the prime field case.) Public parameters are subject to many security criteria.

**Set up:** pick an integer  $d$  in  $[1, n - 1]$ , compute  $Q = dG$ . Output  $(K_p, K_s) = (Q, d)$ .

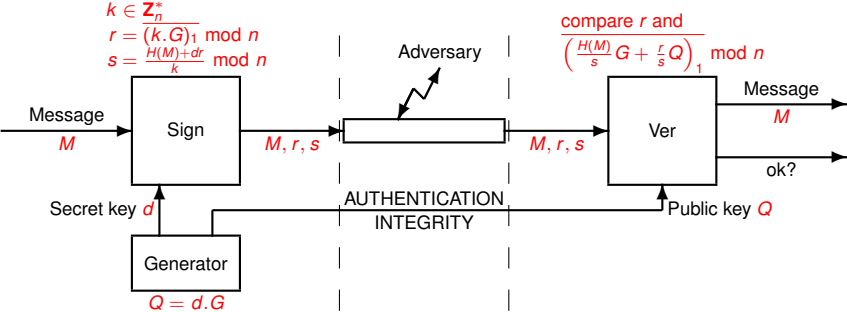
**Signature generation:** pick  $k$  in  $[1, n - 1]$  at random and compute

$$\begin{aligned}(x_1, y_1) &= kG \\ r &= \overline{x_1} \bmod n \\ s &= \frac{H(M) + dr}{k} \bmod n\end{aligned}$$

( $\overline{x_1}$  is a standard way to convert a field element  $x_1$  into an integer.) If  $r = 0$  or  $s = 0$ , try again. Output the signature  $\sigma = (r, s)$

**Verification:** check that  $Q \neq \mathcal{O}$ ,  $Q \in \mathcal{C}$ , and  $nQ = \mathcal{O}$ . Check that  $r$  and  $s$  are in  $[1, n - 1]$  and that  $r = \overline{x_1} \bmod n$  for  $(x_1, y_1) = u_1G + u_2Q$ ,  $u_1 = \frac{H(M)}{s} \bmod n$ , and  $u_2 = \frac{r}{s} \bmod n$ .

# ECDSA Signature



select field, elliptic curve  
 G point of order  $n$   
 $n$  prime

# Example of Public Parameters

secp192r1:

```
q = 6277101735386680763835789423207666416083908700390324961279
a = ffffffff ffffffff ffffffff ffffffff ffffffff fffffffc
b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
n = 6277101735386680763835789423176059013767194773182842284081
G = 03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
seed = 3045ae6f c8422f64 ed579528 d38120ea e12196d5
```

(the leading “03” is for point compression)

# Example of Keys

$d$  = 651056770906015076056810763456358567190100156695615665659  
 $Q$  = 02 62b12d60 690cdcf3 30babab6 e69763b4 71f994dd 702d16a5

(the leading “02” is for point compression)



# ECDSA Parameters Generation

- 1 Choose the finite field  $\mathbf{F}_q$ .
- 2 Pseudo-randomly generate a  $c$  from seed. Take an elliptic curve defined by some  $a$  and  $b$  such that the  $j$ -invariant is  $j = 6912 \frac{4c}{4c+27}$  for  $q$  prime (i.e.  $c = a^3/b^2$ ) and  $j = \frac{1}{c}$  (i.e.  $c = b$ ) otherwise.
- 3 For  $q$  prime, check that  $4a^3 + 27b^2 \bmod q \neq 0$ . For  $q$  a power of two, check that  $b \neq 0$ . If this is not the case, go back to Step 2.
- 4 Count the number of points on the elliptic curve and isolate a prime factor  $n$  greater than  $2^{160}$ . If this does not work or if  $n \leq 4\sqrt{q}$ , go back to Step 2.
- 5 Check the MOV and anomalous condition for  $C$ . If this does not hold, go back to Step 2.
- 6 Pick a random point on the elliptic curve and raise it to the cofactor of  $n$  power in order to get  $G$ . If  $G$  is the point at infinity, try again.

Set parameters to  $(q, \text{representation}, a, b, n, G, \text{seed})$ .

# ECDSA Parameters Validation

Parameters: ( $q$ , representation,  $a$ ,  $b$ ,  $n$ ,  $G$ , seed).

- 1 Check that  $q$  is an odd prime or a power of 2 of appropriate size. In the latter case, check that the field representation choice is valid.
- 2 Check that  $a$ ,  $b$ ,  $x_G$ ,  $y_G$  (where  $G = (x_G, y_G)$ ) lie in  $\mathbf{F}_q$ .
- 3 Check that seed certifies  $a$  and  $b$  by generating  $c$  again and checking that  $\frac{a^3}{b^2} = c$  or  $b = c$  depending on the field type.
- 4 For  $q$  prime, check that  $4a^3 + 27b^2 \bmod q \neq 0$ . For  $q$  a power of two, check that  $b \neq 0$ . Check that  $G$  lies in the elliptic curve. Check that  $n$  is a prime greater than both  $2^{160}$  and  $4\sqrt{q}$ . Check that  $nG = \mathcal{O}$ , the neutral element. Check the MOV and anomalous condition.

# ECDSA Parameters Selection: Conclusion

- making new parameters is not easy
- rather use parameters from standards

## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- ElGamal Cryptography
- **Selecting Key Lengths**
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

# Breaking RSA Cryptography by Factoring

Best attack (ideally): factoring

## Fact

If we can factor  $N = pq$  then from an RSA public key, we can compute the secret key.

- To have RSA cryptography secure, the factoring problem must be hard
- Parameter for the factoring problem: modulus length

# Breaking DH Cryptography by Discrete Logarithm

Best attack (ideally): discrete logarithm computation

## Fact

If we can compute the discrete logarithm  $x$  of  $g^x$  then from  $g, g^x, g^y$  we can compute  $g^{xy}$ .

To have DH cryptography secure then the discrete logarithm problem must be hard for the proposed parameters:

- prime order of the generated subgroup
- overall structure type:
  - **multiplicative group of a finite field**
  - **elliptic curve**
    - random over **prime field**
    - random over **binary field**
    - special

# Reading the Tables

tables give equivalent security levels over time depending on applications

- **symmetric**: bitlength of the key for symmetric encryption or MAC  
also: half of the hash length for hashing
- **asymmetric**: bitlength of the RSA modulus or of  $p$  for  $\mathbf{Z}_p^*$   
(sub)groups
- **subgroup DL**: bitlength of the order of the generator  $g$   
(in multiplicative groups and elliptic curves as well)
- **EC**: bitlength of the field cardinality on which the random elliptic curve is considered

# Meta-comparison of Cryptographic Strengths

Following <http://www.keylength.com> by Quisquater

method	year	sym.	asym.	DL		EC	hash
Lenstra-Verheul	2015	82	1613	145	1613	154	163
Lenstra updated	2015	78	1245	156	1245	156	156
ECRYPT II	2011–15	80	1248	160	1248	160	160
NIST	2011–30	112	2048	224	2048	224	224
FNISA	2010–20	100	2048	200	2048	200	200
BSI	2011–15	–	1976	224	2048	224	224



## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- **Formalism**
- Towards Post-Quantum Cryptography?
- Other Primitives

**Definition**

A **public-key cryptosystem** is a tuple  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  with a plaintext domain  $\mathcal{M}$  and three efficient algorithms  $\text{Gen}$ ,  $\text{Enc}$ , and  $\text{Dec}$ . The algorithm  $\text{Dec}$  is deterministic and outputs either something in  $\mathcal{M}$  or an error  $\perp$ . It is such that

$$\forall X \in \mathcal{M} \quad \Pr[\text{Dec}(K_s, \text{Enc}(K_p, X)) = X] = 1$$

where  $(K_p, K_s)$  is generated from running  $\text{Gen}$ . The probability is over the randomness used in  $\text{Gen}$  and  $\text{Enc}$ .

# How to Define Security?

- the adversary holds the public key so he can encrypt whatever he wants without using any external oracle
- so, for predictable plaintext, if encryption is deterministic, it is easy to recognize form the ciphertext  
example: the encryption of a salary, the encryption of “yes” or “no”
- we should add randomness in the encryption and make the encryption of arbitrary messages hard to distinguish

# Security against Distinguisher

## Definition

A PKC  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  is  $(t, \varepsilon)$ -**secure under chosen plaintext attacks** (IND-CPA-secure) if for any interactive process  $\mathcal{A}$  limited to a time complexity  $t$ , given a bit  $b$ , when we first run the following steps

- 1:  $\text{Gen} \rightarrow (K_p, K_s)$
- 2:  $\mathcal{A}(K_p) \rightarrow (m_0, m_1)$  such that  $|m_0| = |m_1|$
- 3:  $\text{Enc}(m_b) \rightarrow c$
- 4:  $\mathcal{A}(K_p, c) \rightarrow x$

we have

$$\Pr[x = 1 | b = 0] - \Pr[x = 1 | b = 1] \leq \varepsilon$$

It is  $(q, t, \varepsilon)$ -**secure under chosen plaintext/ciphertext attacks** (IND-CCA-secure) if the same holds for any similar interactive process  $\mathcal{A}^{\text{Dec}(K_s, \cdot)}$  who is limited to  $q$  queries to a decryption oracle  $\text{Dec}(K_s, \cdot)$  but not allowed to send it  $c$ .

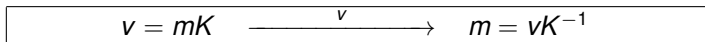
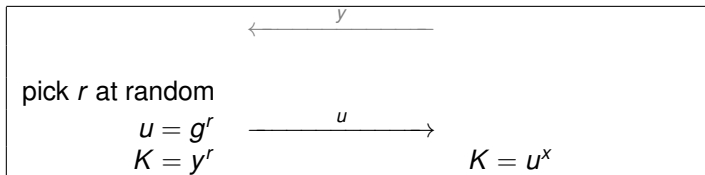
# Problem with Deterministic Cryptosystems

- this is a modern notion of security
- problem: if Enc is deterministic, then PKC is insecure!
- modern PKC are probabilistic
- example: ElGamal cryptosystem (and variants)

# ElGamal Cryptosystem Generalized (Reminder)

**Alice**  
input:  $m$

**Bob**  
secret key:  $x$   
public key:  $y = g^x$



**output:  $m$**

# Signature Scheme

## Definition

A **digital signature scheme** is a tuple  $(\text{Gen}, \mathcal{D}, \text{Sig}, \text{Ver})$  with a message domain  $\mathcal{D} \subseteq \{0, 1\}^*$  and three efficient algorithms  $\text{Gen}$ ,  $\text{Sig}$ , and  $\text{Ver}$ . The algorithm  $\text{Ver}$  is deterministic and outputs 0 (reject) or 1 (accept). It is such that

$$\forall X \in \mathcal{D} \quad \Pr[\text{Ver}(K_p, \text{Sig}(K_s, X)) = 1] = 1$$

where  $(K_p, K_s)$  is generated from running  $\text{Gen}$ . The probability is over the randomness used in  $\text{Gen}$  and  $\text{Sig}$ .

# EF-CMA Security

## Definition

A digital signature scheme  $(\text{Gen}, \mathcal{D}, \text{Sig}, \text{Ver})$  is  $(q, t, \varepsilon)$ -**secure against existential forgery under chosen message attacks** (EF-CMA) if for any probabilistic algorithm  $\mathcal{A}$  limited to a time complexity  $t$  and to  $q$  queries,

$$\Pr[\mathcal{A}^{\text{Sig}(K_s, \cdot)} \text{ forges}] \leq \varepsilon$$

where  $(K_p, K_s)$  is the output of  $\text{Gen}$ ,  $(X, c)$  a pair of random variables defined as the output of  $\mathcal{A}^{\text{Sig}(K_s, \cdot)}$ , and “ $\mathcal{A}^{\text{Sig}(K_s, \cdot)}$  forges” is the event that  $\text{Ver}(K_p, X) = 1$  and that  $\mathcal{A}$  did not query  $X$  to the signing oracle.



# Other Public-Key Cryptosystems

- RSA
- Rabin
- ECC
- HECC
- Paillier cryptosystem
- NTRU
- lattice-based cryptosystem
- McEliece cryptosystem
- TCHo
- ...

## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- **Towards Post-Quantum Cryptography?**
- Other Primitives

# Some Alternate Constructions Based on Lattices or Codes

- NTRU
- lattice-based crypto
- McEliece cryptosystem
- TCHo

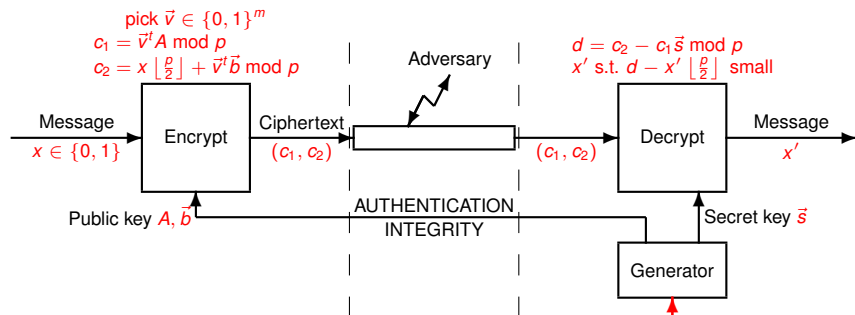
why this?

- resilience to quantum computing
- it finds other applications:
  - fully homomorphic encryption (clouds..., privacy-by-design...)
  - multilinear mapping (multiparty cryptography...)

# An Example: NTRUEncryption

- $N$  prime,  $q > p$ ,  $p$  and  $q$  coprime
- $f, g$  polynomials with degree at most  $N - 1$  and coefficients in  $\{-1, 0, 1\}$   
 $f$  must be such that there exists  $f_p$  and  $f_q$  such that  $f \times f_p = 1$  in  $\mathbf{Z}_p[x]/(x^N - 1)$  and  $f \times f_q = 1$  in  $\mathbf{Z}_q[x]/(x^N - 1)$
- secret key:  $(f, f_p)$
- public key:  $h$  such that  $h = pf_qg$  in  $\mathbf{Z}_q[x]/(x^N - 1)$
- message: a polynomial  $m$  with degree at most  $N - 1$  and coefficients in  $\{-1, 0, 1\}$
- encryption: pick a random polynomial  $r$  with degree at most  $N - 1$  and small coefficients, then  $e = rh + m$  in  $\mathbf{Z}_q[x]/(x^N - 1)$
- decryption:  $a = fe$  in  $\mathbf{Z}_q[x]/(x^N - 1)$   
note that  $a = prg + fm$   
by having  $rg$  small, we have  $a \bmod q \bmod p = fm \bmod p$ , so we compute  $b = fm$  in  $\mathbf{Z}_p[x]/(x^N - 1)$   
then  $c = f_p b = m$  in  $\mathbf{Z}_p[x]/(x^N - 1)$

# The Regev Public-Key Cryptosystem



$p$  prime,  $\varepsilon > 0$   
 $n^2 \leq p \leq 2n^2$ ,  $m = (1 + \varepsilon)(n + 1) \log_2 p$   
 $\alpha = \frac{1}{\sqrt{n \log_2^2 n}}$   
 $\chi: E_i \sim \mathcal{N}(0, \alpha p)$ ,  $e_i = \lfloor E_i \rfloor$

$\vec{s} \in \mathbb{Z}_p^n$   
 $A \in \mathbb{Z}_p^{m \times n}$   
 $e_i \leftarrow \chi \quad i = 1, \dots, n$   
 $b = A\vec{s} + \vec{e} \bmod p$

# Lattice-Based Cryptography

- **lattice**: discrete subgroup of  $\mathbf{R}^m$
- specified by a basis:

$$\mathcal{L}(\vec{a}_1, \dots, \vec{a}_n) = \left\{ \sum_{i=1}^n s_i \vec{a}_i; s_1, \dots, s_n \in \mathbf{Z} \right\}$$

- it is hard to find short vectors  $\vec{x} \in \mathcal{L}(\vec{a}_1, \dots, \vec{a}_n)$
- given  $\vec{b}$ , it is hard to find  $\vec{x} \in \mathcal{L}(\vec{a}_1, \dots, \vec{a}_n)$  making  $\|\vec{b} - \vec{x}\|$  small
- many cryptographic algorithms
- fully homomorphic encryption
- problem: public keys are a bit large
- likely to be used in practice in near future

## Public-Key Cryptography

- Public-Key Cryptography
- Diffie-Hellman Key Exchange
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography?
- Other Primitives

# Key and Data Encapsulation Mechanisms

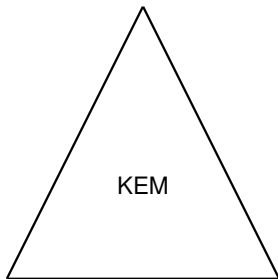
## Hybrid Encryption

- DEM: same as symmetric encryption
- KEM: public-key algorithm producing an encrypted (encapsulated) key
  - ≈ generate a random symmetric key and encrypt it using public-key encryption



# KEM Primitive

Generator, KemEnc, KemDec  
**components**



KEM

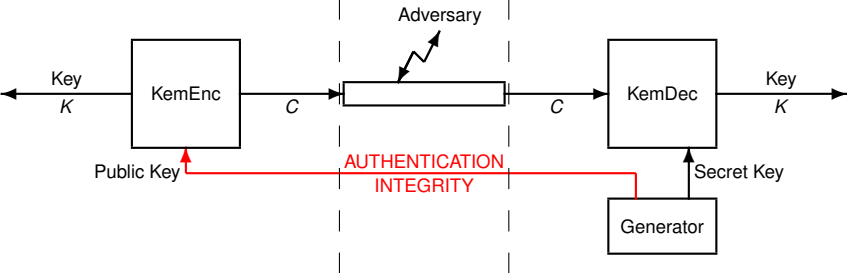
**functionality**

if  $\text{KemEnc}_{K_p} \rightarrow (K, C)$   
then  $\text{KemDec}_{K_s}(C) = K$

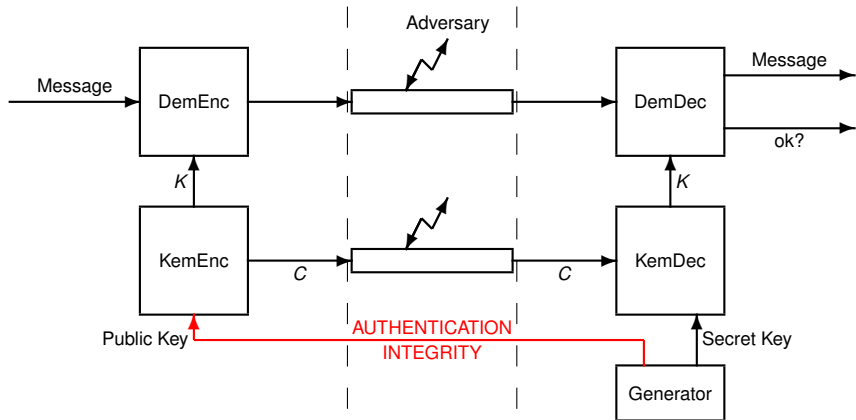
**security**

key is confidential

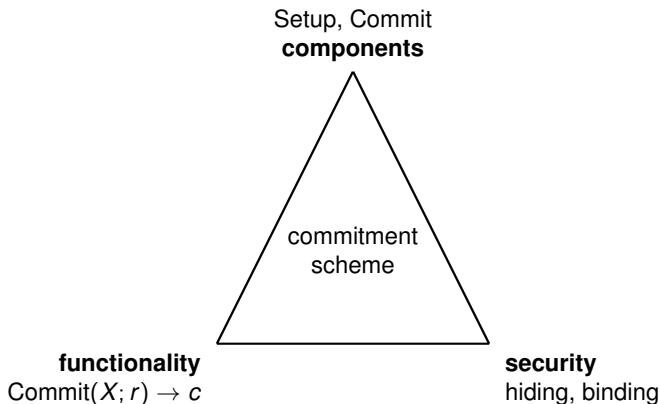
# KEM



# KEM + DEM



# Commitment Scheme



- to commit to  $X$ : pick  $r$  at random and release  $\text{Commit}(X; r)$
- to open  $c$ : release  $r$  to check  $\text{Commit}(X; r) = c$

# Pedersen Commitment

**setup** generates two large primes  $p$  and  $q$  s.t.  $q|(p-1)$  (e.g. 1024 resp. 160 bit-long), an element  $g \in \mathbf{Z}_p^*$  of order  $q$ ,  $a \in \mathbf{Z}_q^*$ , and  $h = g^a \bmod p$   
Domain parameters:  $\langle p, q, g, h \rangle$

**commit**  $\text{Commit}(X; r) = g^X h^r \bmod p$

**unconditionally hiding** given  $c$  in the subgroup spanned by  $g$ , any  $X$  has a related  $r$  such that  $\text{Commit}(X; r) = c$

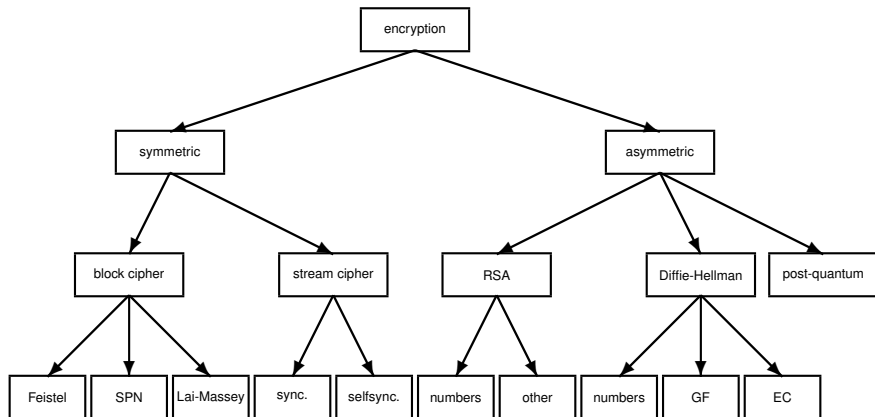
**computationally binding** committing to  $X$  and opening to  $X' \neq X$  leads to solving  $g^X h^r \equiv g^{X'} h^{r'} \pmod{p}$  hence  $a = \frac{X' - X}{r - r'} \bmod q$

This is equivalent to solving the discrete logarithm problem with the domain parameters

# Conclusion

- two families: RSA (factoring-based) and DH (discrete log-based)
- does not replace symmetric cryptography: used for key exchange only
- more compact data using elliptic curves

# Systematic Classification of Cryptography



# References

- **Lenstra-Verheul.** Selecting Cryptographic Key Sizes. *Journal of Cryptology* vol. 14, 2001.
- **Regev.** On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM* vol. 56(6), 2009.



# Must be Known

- the big picture with the 4 primitives
- Diffie-Hellman key agreement protocol
- ElGamal cryptosystem
- RSA
- PKCS#1
- Rabin cryptosystem

# Train Yourself

- RSA encryption: midterm exam 2008–09 ex2
- RSA signature: final exam 2010–11 ex2
- PKC construction: final exam 2009–10 ex3
- signature construction: final exam 2008–09 ex2
- trapdoor in DSA: final exam 2014–15 ex1
- DSA with related randomness: final exam 2014–15 ex2
- bad DL-based signature: final exam 2015–16 ex1
- Pedersen commitment: final exam 2012–13 ex5















- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment**
- 10 Case Studies II

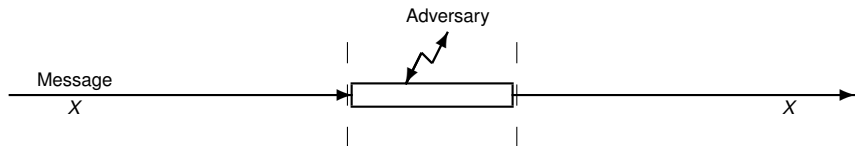
# Roadmap

- secure communication channels
- setup by password
- setup by short authenticated strings
- setup by a trusted third party: Kerberos, PKI

## Trust Establishment

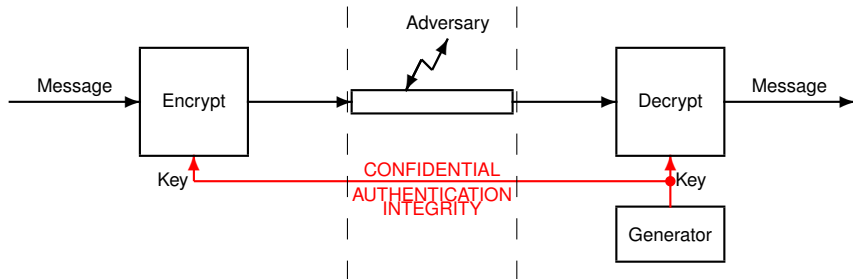
- From Secure Channel to Secure Communications
- Setup of Secure Channels
- Setup by Narrowband Secure Channel
- Setup by a Trusted Third Party
- Trust Management and Cryptography

# The Cryptographic Trilogy

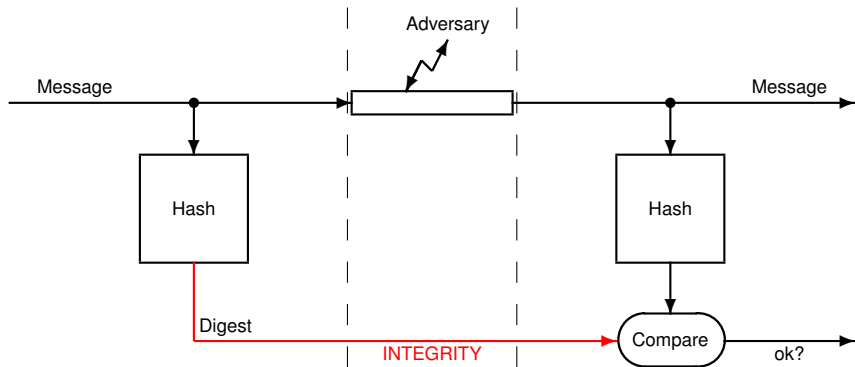


- **Confidentiality (C)**: only the legitimate receiver can get  $X$
- **Authentication + Integrity (A+I)**: only the legitimate sender can insert  $X$  and the received message must be equal to  $X$

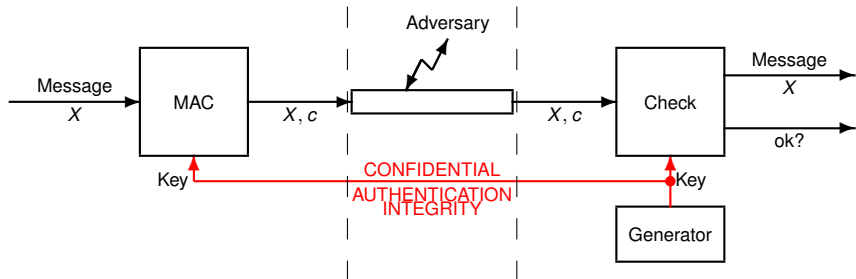
# Enforcing Confidentiality by Encryption



# Enforcing Integrity by Hash Function



# Enforcing Authenticity + Integrity by MAC

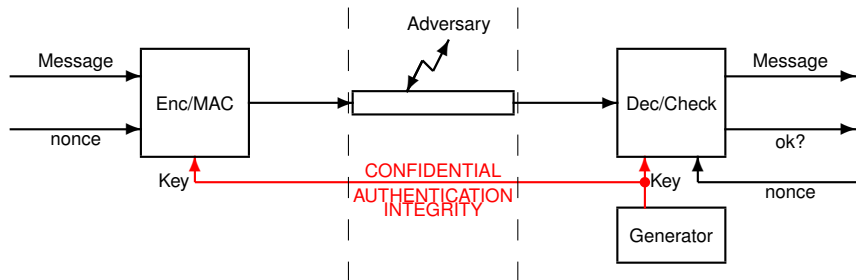


# Authentication and Integrity

- **Message integrity:** we make sure that the received message is equal to the sent one
- **Message authentication:** we make sure about who sent the message
- *good* authentication means often enforce integrity at the same time  
symmetric encryption is sometimes used for message authentication but this is a BAD practice
- but there are weird authentication means not protecting it  
example: problem in GSM/WEP/Bluetooth/... (see [slide 670](#))



# A+I+C by Symmetric Cryptography

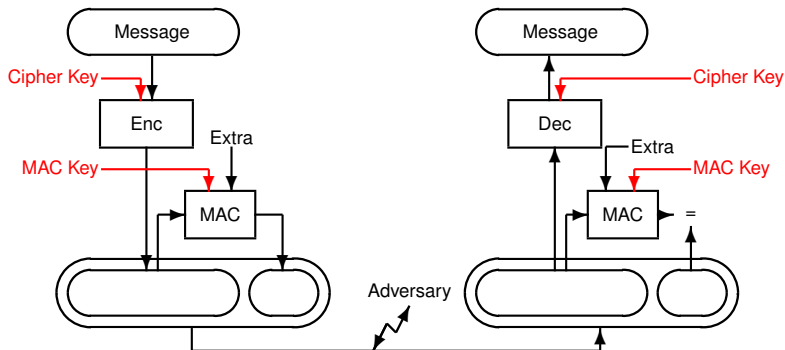


# A+I+C Symmetric Constructions

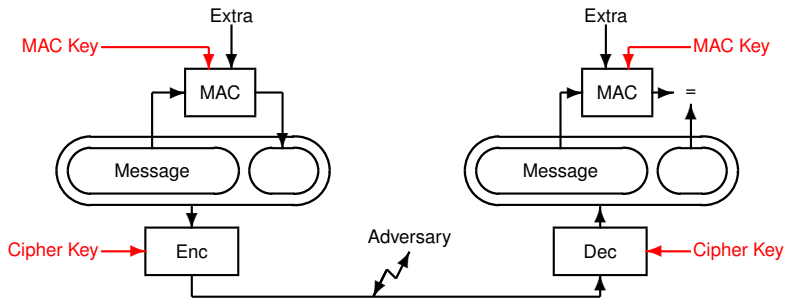
- encrypt-then-MAC
- MAC-then-encrypt
- authenticated modes of operation

**CCM** CS CWC EAX **GCM** IACBC IAPM OCB PCFB XCBC ...

# Encrypt-then-MAC



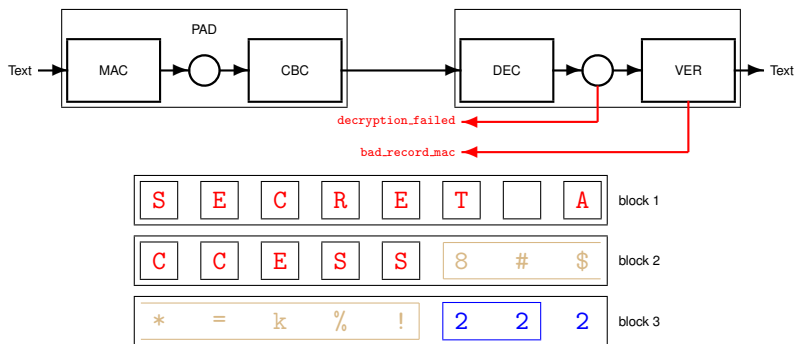
# MAC-then-Encrypt



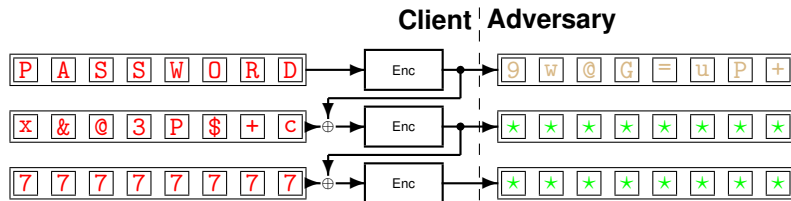
# Some Tricky Additional Things

- as soon as padding occurs, some combination may be weak
- some problems when adversary can get advantage of a return channel
- many standards weak, fixed by implementations
- example (2003): MAC-then-Pad-then-Encrypt in TLS using block ciphers is weak

# TLS using Block Ciphers

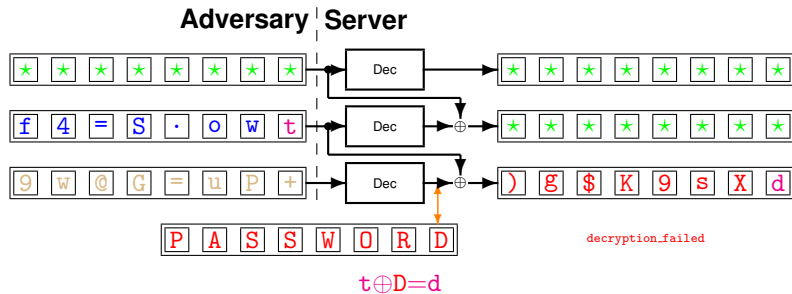


# Padding Oracle Attack: Encryption



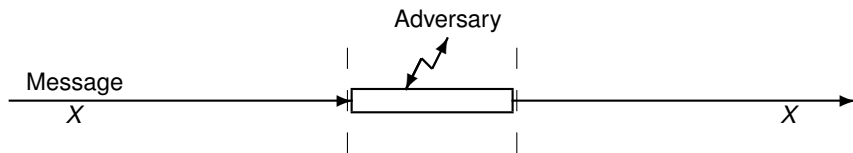
We would like to decrypt 9w@G=uP+

# Padding Oracle Attack: Decryption



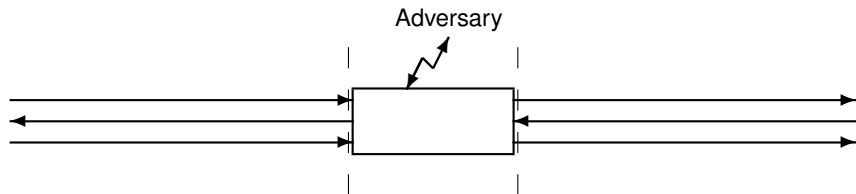


# Security Property of Communication Channels



- **Confidentiality, Authentication, Integrity**
- **Freshness:** the received  $X$  was not received before (a message in transit cannot be replayed)
- **Liveliness:** a sent message  $X$  is eventually delivered (a message in transit cannot be discarded)
- **Timeliness:** ( $>$  liveliness) time of delivery is upper bounded (a message cannot be overly delayed)

# From Packet Security to Session Security



- **Key establishment:** set up *A/I/C* key material for message security
- **Session integrity:** the sequence of protocol messages is eventually the same at both ends  
(messages in transit cannot be swapped)
- **Privacy:** many different notions at this time!  
(cannot identify sender or receiver)  
(cannot link that two messages by same sender)

# Enforcing Session Integrity

Assuming that channels enforce A+I+C and that key establishment is secure, session integrity splits in two problems

- **Sequentiality**: whenever a participant has seen a message sequence starting with  $X_1, \dots, X_t$ ,  $X_t$  coming in, then the other participant has seen a message sequence whose first  $t$  messages are  $X_1, \dots, X_t$   
😊: easy to protect: just number the messages and apply A+I protection on message numbers
- **Termination fairness**: making sure that the last message on both ends is the same one  
😞: no cheap way to enforce it if liveness is not guaranteed

# Sequentiality using A + I Message Security

common methods:

- acknowledge receipt of every message
- authenticate a sequence number in packets and check that received packets have consecutive sequence numbers
- authenticate an increasing nonce value (e.g. a clock value) + check for no packet loss by other means

TLS or SSH:  $Y = \text{Enc}(X \parallel \text{MAC}(\text{seq} \parallel X))$  where seq is implicit

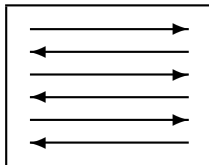
# Fair Termination Problems

- example: contract signing  
Alice and Bob have signed a contract and want to be sure that they both consider the contract as valid
- there must be one critical message in the protocol such that one participant thinks his counterpart has a valid contract the other does not think the transaction is valid
- this reduces to synchronizing on an exit status bit

# Fair Termination by Synchronization Protocol

Alice

Bob



$a \leftarrow$  exit status

$b \leftarrow$  exit status

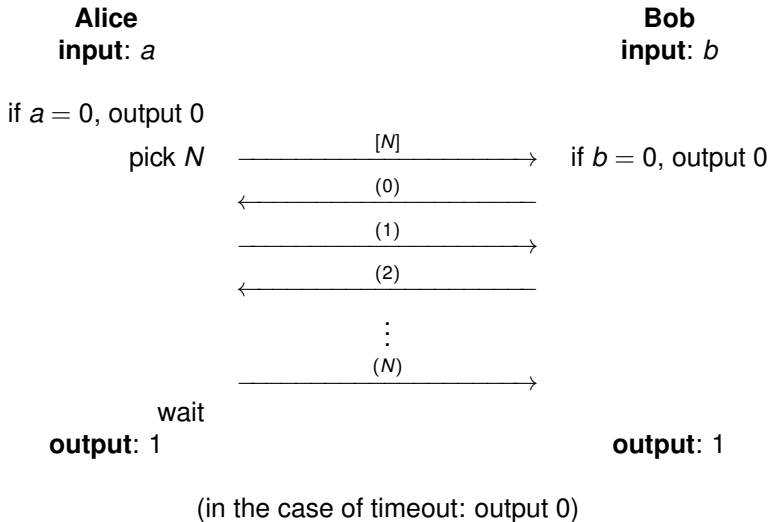
synchronization  
protocol

**output:**  $a'$

**output:**  $b'$

exit status: 1 (normal termination case) or 0 (failure case)  
functionality:  $a' = b' = a \times b$

# Keep-in-Touch (KiT) Synchronization Protocol



# KiT Protocol Security

## Theorem (Avoine-Vaudenay 2006)

**Communication complexity:** at most  $E(C) = 2 + \sum_i i \Pr[N = i]$

**Probability of asymmetric termination:** at most

$$p_a = \max_i \Pr[N = i]$$

*For any synchronization protocol with parameters  $C$  and  $p_a$ , there exists a KiT protocol with parameters  $C'$  and  $p'_a$ , such that  $\Pr[C' \leq C] = 1$  and  $p'_a \leq p_a$ .*

Example:  $\Pr[N = i] = \frac{1}{N}$  so  $p_a = \frac{1}{N}$  and  $E(C) = \frac{N+3}{2}$



# Bad News

## Theorem (Avoine-Vaudenay 2006)

For any synchronization protocol with parameters  $C$  and  $p_a$  we have

$$E(C) - 2 \geq \frac{1}{2} \left( \frac{1}{p_a} - 1 \right)$$

- example: if we want  $p_a \leq 2^{-20}$  we need  $E(C) \geq 2^{19}$
- morality: synchronization must be expensive
- morality: it is hard to beat the KiT protocol set up with  $N$  uniform

## Summary for Secure Channel (so far)

level	property	toolkit
packet	A+I confidentiality A+I+C freshness liveliness	MAC symmetric encryption integrated modes (comes with sequentiality) (must live without)
session	key establishment sequentiality termination	setup protocols (next) various protocol options synchronization protocol
all	privacy	?

## Trust Establishment

- From Secure Channel to Secure Communications
- Setup of Secure Channels
- Setup by Narrowband Secure Channel
- Setup by a Trusted Third Party
- Trust Management and Cryptography

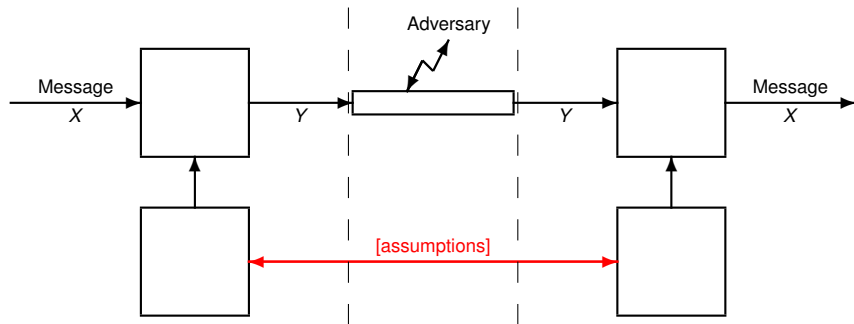
# Problem

Q: How to setup a secure channel over an insecure channel?

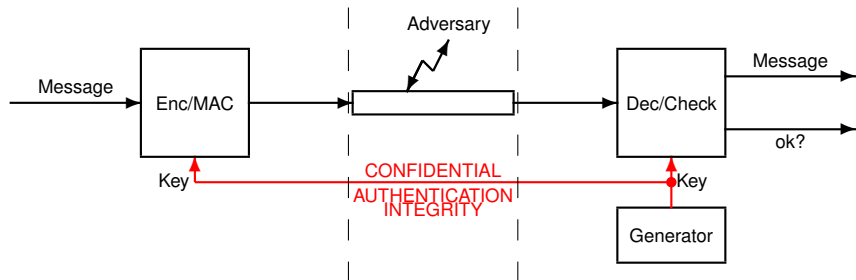
A: hfr n frpher punaary

▶ ROT13

# Virtual Channels by Combination of Channels



# Secure Channel from A+I+C Channel



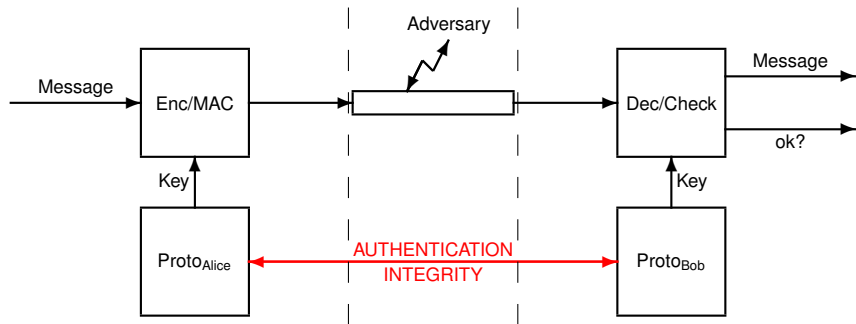
# Next Step: Strongly Secure Channel From Weakly Secure Channel

Q: How to relax security properties at setup?

A: hfr choyvp-xrl pelcgbtencul

▶ ROT13

## ... with A+I Channel: Key Agreement Protocol





# Security of Key Exchange Protocols

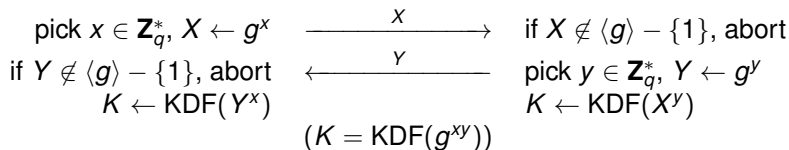
**Secrecy:** by looking at the communication protocol, it is impossible to guess the exchanged key

# The Diffie-Hellman Key Agreement Protocol

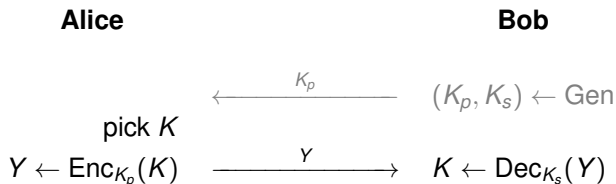
Assume a group  $\langle g \rangle$  generated by some  $g$  of prime order  $q$

**Alice**

**Bob**



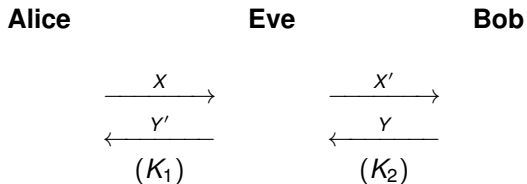
# Key Transfer by Public-Key Encryption



# Passive vs Active Adversaries

- **active adversary:** can interfere with communication (modify messages, insert messages, replay messages)  
The Diffie-Hellman protocol requires A+I channel to protect against it  
example: static keys authenticated by ad hoc means
- **passive adversary:** just listen to communications and tries to decrypt communications (e.g. by recovering the key)  
The Diffie-Hellman protocol resists to passive adversaries with no extra assumptions

# An Active Attack: Man-in-the-Middle Attack



# Approaches to Build an Initial Authenticated Channel

- **using really secure initial channel**  
setup cable, Near Field Comm. (see Bluetooth simple pairing)
- **by user monitoring**  
caution: humans are not so reliable for security (e.g. Bluetooth)  
relies on strong assumptions (genuine software, correct public keys...)  
→ password-based, SAS-based
- **using a trusted third party**  
examples: secure token, key server, certificate authority

# Summary

- we need specific means to A+I-securely transmit a public key
- we agree on a master key using public key cryptography
- we use conventional cryptography to set up secure channels
  - we derive several symmetric keys using key derivation functions
  - we use symmetric encryption and MAC
- we must live with the fear that termination may be unfair

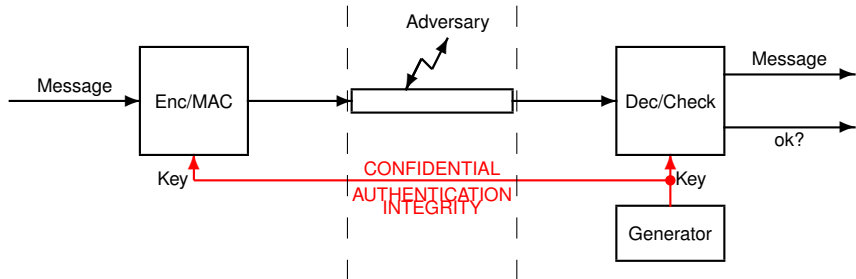
## Trust Establishment

- From Secure Channel to Secure Communications
- Setup of Secure Channels
- **Setup by Narrowband Secure Channel**
- Setup by a Trusted Third Party
- Trust Management and Cryptography



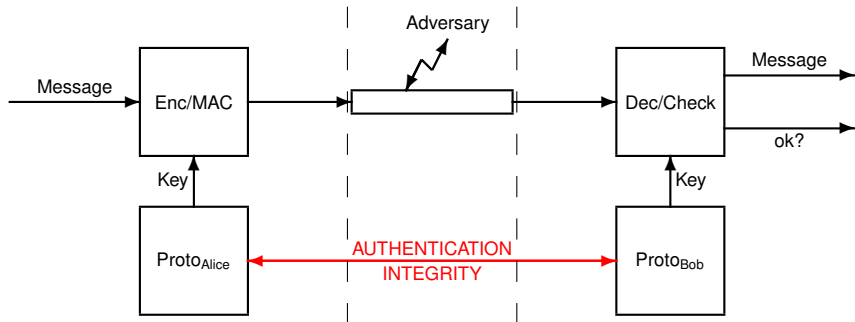
# Secure Communication Step 1

## Conventional Cryptography



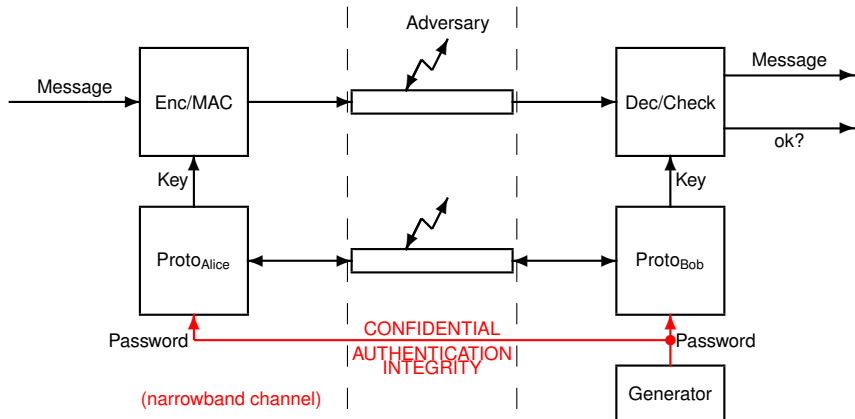
# Secure Communication Step 2

## Public-Key Cryptography



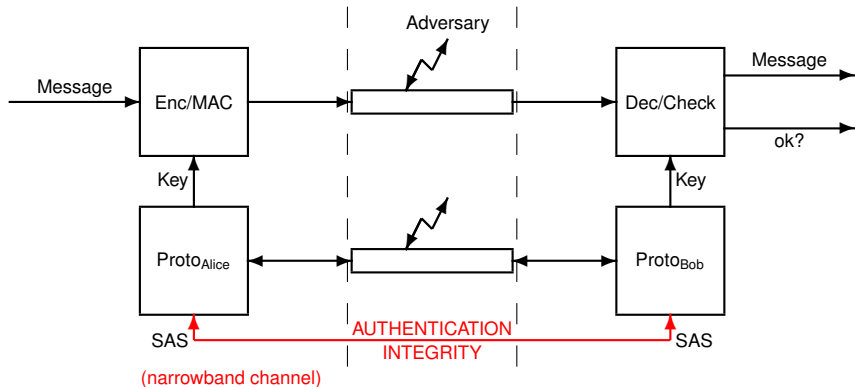
# Secure Communication Step 3

## Password-Based Cryptography



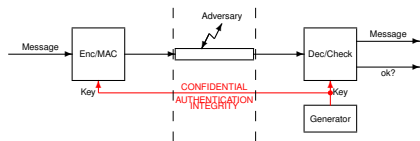
# Secure Communication Step 4

## Cryptography Based on Short Authenticated Strings

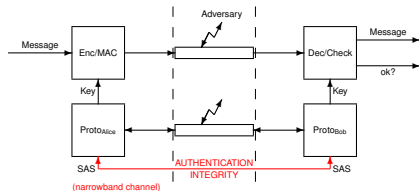
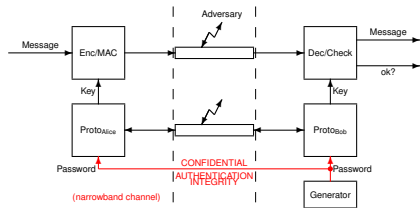
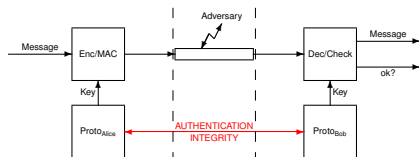


# Secure Communication Steps 1-4

with confidential channel



without confidential channel



# Adversary Capabilities on the Secure Channel

**Regular channels:** the adversary can do whatever he/she wants with the messages: modify, create, swap, remove, stall, ...

**(Weak) authenticated channels:** the adversary cannot modify nor create messages. He/she can swap, remove, stall, ...

**(Strong) authenticated channels:** same plus some additional assumptions!

E.g. messages must be either delivered at once or removed (stall-free channels).

## Trust Establishment

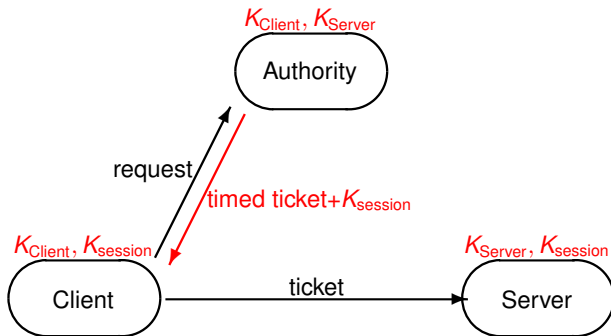
- From Secure Channel to Secure Communications
- Setup of Secure Channels
- Setup by Narrowband Secure Channel
- **Setup by a Trusted Third Party**
- Trust Management and Cryptography

# Several Trusted 3rd Party Approach

- **soft 3rd party:** user monitoring  
password-based, SAS-based
- **pervasive 3rd party:** secure token  
smart cards, secureID, trusted computing platform
- **key server:** Kerberos  
symmetric cryptography only, for corporate network
- **certificate authority:** PKI  
for global network



# Example: Kerberos



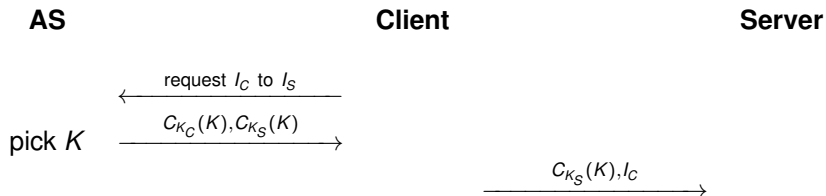
- timed ticket +  $K_{\text{session}}$  encrypted with  $K_{\text{Client}}$
- ticket encrypted with  $K_{\text{Server}}$

# Kerberos

Hypotheses:

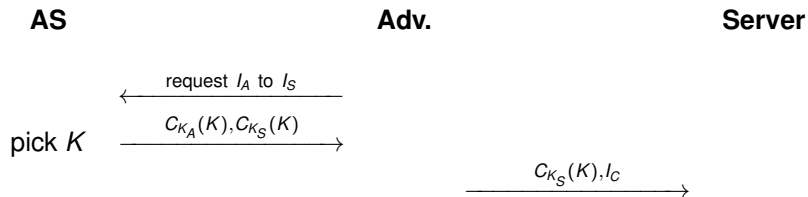
- there is an online (trusted) authentication server (AS)
- AS shares  $K_C$  with client  $I_C$
- AS shared  $K_S$  with server  $I_S$
  
- Goal: to help  $I_C$  and  $I_S$  to share a session key  $K$  (and to help careless users to get privacy)

# Server-Aided Authentication (Bad Protocol)



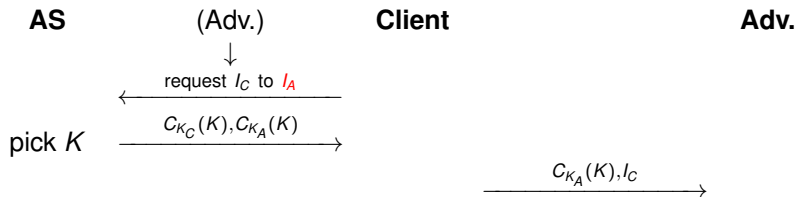
Problem: there is no authentication: an attacker can replace  $I_C$  or  $I_S$

# Attack



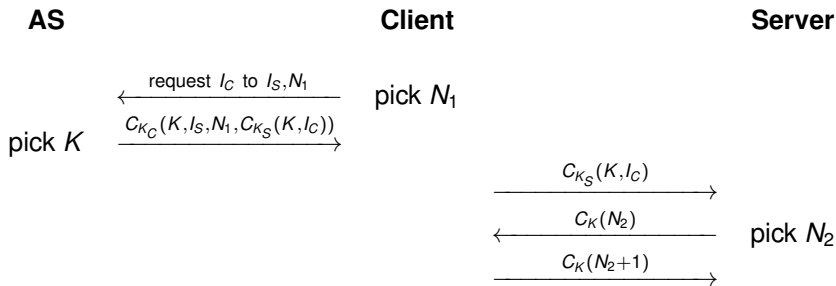
Server thinks he is talking to  $I_C$ !

# Attack



Client thinks he is talking to  $I_S$ !

# Needham-Schroeder Authentication (Still Bad)



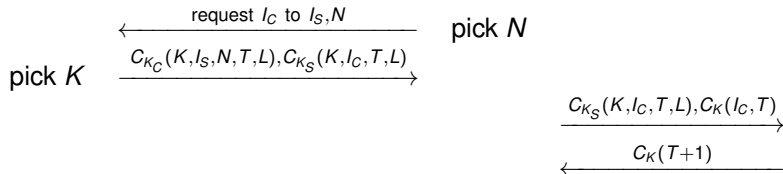
Problem: replay attack by impersonating C after  $K$  gets compromised

# Basic Kerberos Protocol

AS

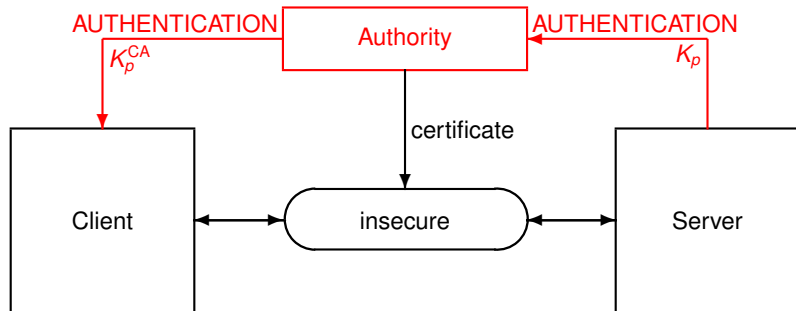
Client

Server



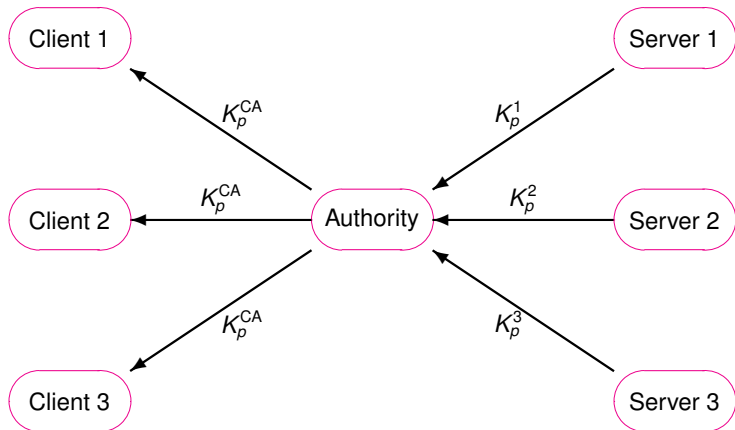
$T$ : clock value;  $L$ : validity period

# The Certificate Authority Model

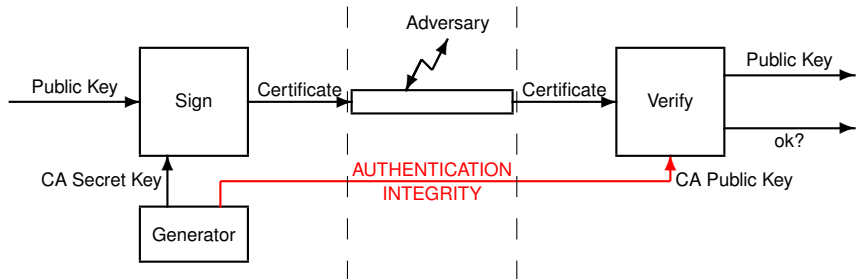




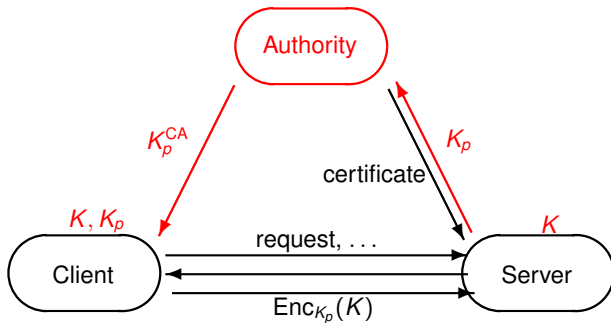
# Critical Secure Channels



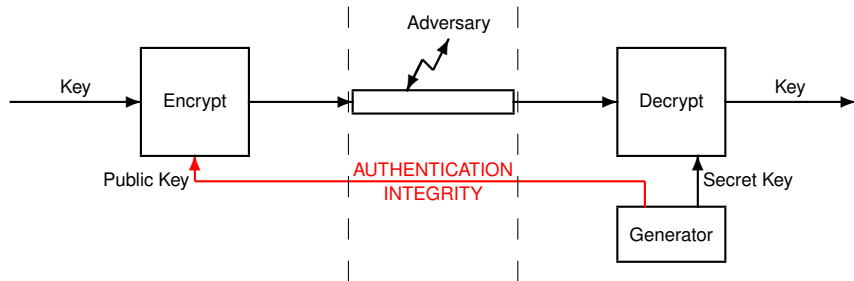
# Public-Key Certificate



# Semi-A Key Exchange Using Certificates



# Semi-Authentication: Key Transmission using PKC

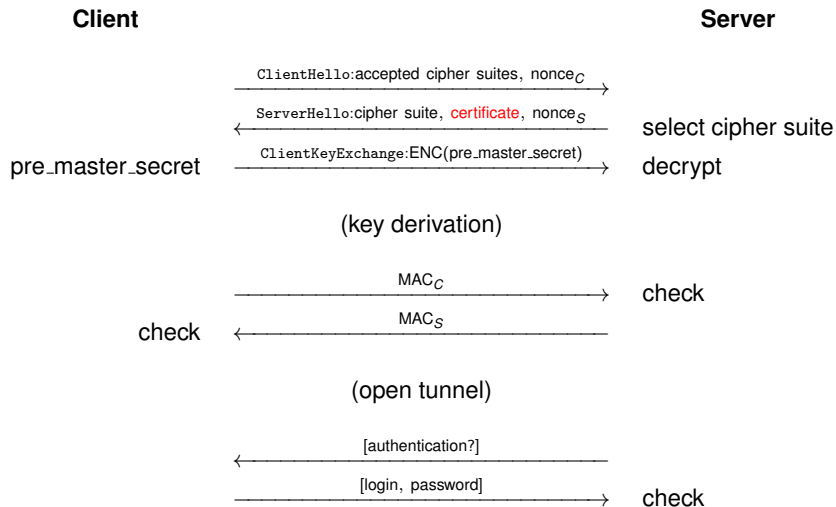


# Semi-Authenticated Channel

one participant authenticates the other  
(typical for client-server communication)

- client receives the authenticated (static) key of the server
- client and server run a key establishment protocol
- secure A+I+C channel is set up
  - client knows he is talking to the correct server
  - server has no clue to which client he is talking to

# A Typical TLS Session



# An X.509 Certificate Example: Overall Structure

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 674866 (0xa4c32)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town,
           O=Thawte Consulting cc, OU=Certification Services Division,
           CN=Thawte Server CA/Email=server-certs@thawte.com
    Validity
      Not Before: Jun  2 13:10:11 2003 GMT
      Not After : Jun 11 10:21:15 2005 GMT
    ...
    X509v3 extensions:
      X509v3 Extended Key Usage: TLS Web Server Authentication
      X509v3 Basic Constraints: critical CA:FALSE
    Signature Algorithm: md5WithRSAEncryption
    8d:7b:78:60:88:c4:13:4e:94:0d:bc:3b:1b:1c:b6:c9:bc:b1:
    0b:ed:7d:eb:6f:08:3a:ba:6d:21:36:93:38:36:66:7b:a7:bc:
    c0:3f:c4:e0:cf:b4:02:58:be:a6:b9:1d:45:a2:c4:58:38:07:
    e4:63:1a:d9:b9:8d:27:7c:93:67:31:82:6f:a3:3c:86:0c:e0:
    10:71:de:f2:e9:74:af:ac:76:b4:5b:8e:48:57:9d:8f:12:f6:
    72:63:8a:79:b4:74:e0:ba:ca:ac:1a:36:b4:16:38:c1:c5:d2:
    73:ed:e8:64:b0:ae:9e:e2:36:d7:0c:77:92:cc:c7:c0:e0:8a:
    54:24
```

# An X.509 Certificate Example: Subject

```
Subject: C=CH, ST=Bern, L=Bern,  
         O=Switch - Teleinformatikdienste fuer Lehre und Forschung,  
         CN=nic.switch.ch  
Subject Public Key Info:  
  Public Key Algorithm: rsaEncryption  
  RSA Public Key: (1024 bit)  
    Modulus (1024 bit):  
      00:d0:0e:b7:16:bf:86:59:c3:97:e6:02:33:59:90:  
      65:29:b0:69:73:64:83:03:1b:df:62:a8:4d:c0:4f:  
      3c:d9:12:6b:8c:57:95:e1:57:e8:48:a6:7f:dd:15:  
      8b:9d:ad:93:dc:78:af:06:1a:ce:0f:7b:cc:c4:6f:  
      a0:06:26:40:73:04:d3:da:7b:20:c1:15:37:8c:2f:  
      58:c4:d4:c1:4b:18:84:5c:54:f1:b1:a0:44:3c:e2:  
      0e:8a:a2:63:48:6b:34:c7:10:9d:a1:23:56:77:f5:  
      4e:3d:38:9a:70:5e:03:02:30:45:ee:81:e4:94:96:  
      47:18:9e:47:37:bb:18:f6:87  
    Exponent: 65537 (0x10001)
```



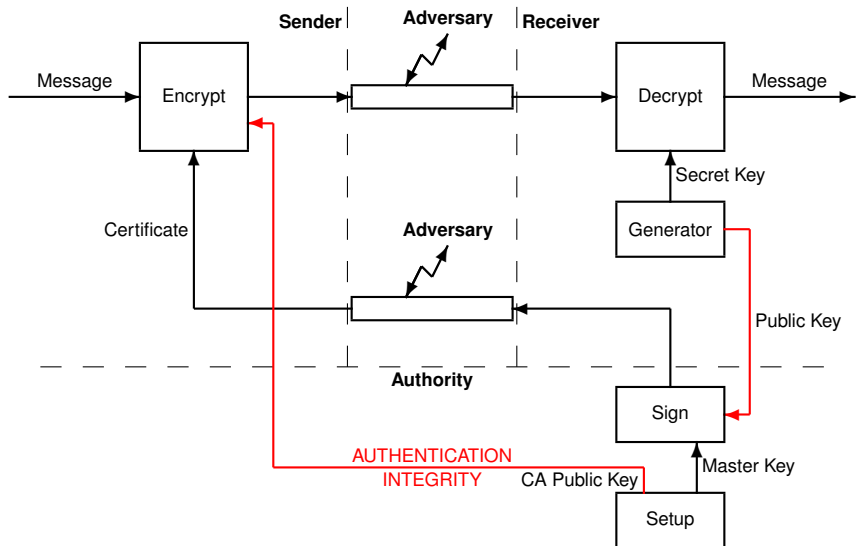
# Two Approaches to Revocations

- **certificate revocation lists (CRL):**
  - regularly, or under emergency cases, revocation lists are released by CA
  - clients should always check for new CRLs (at the nearest repository) and go through the list before treating any certificate
  - drawback: high bandwidth
- **online certificate status protocol (OCSP):**
  - clients should send certificates to the CA for approval
  - drawback: subject to DoS attacks

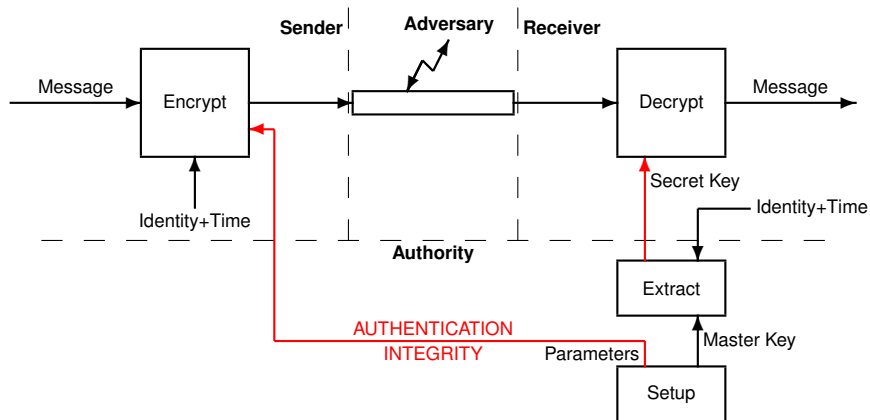
# Several 3rd-Party Based Trust Infrastructure

- **Kerberos**  
symmetric-crypto with key escrow
- **PKI**  
advantage: widely available
- **identity-based cryptography**: have public keys implicit from identities and time  
advantage: time-based revocation with small period
- **certificateless encryption**: combine the two models  
advantage: requires no key escrow
- **certificate-based encryption**: certificate is private, required for decryption  
≈ equivalent to certificateless encryption (name is confusing)

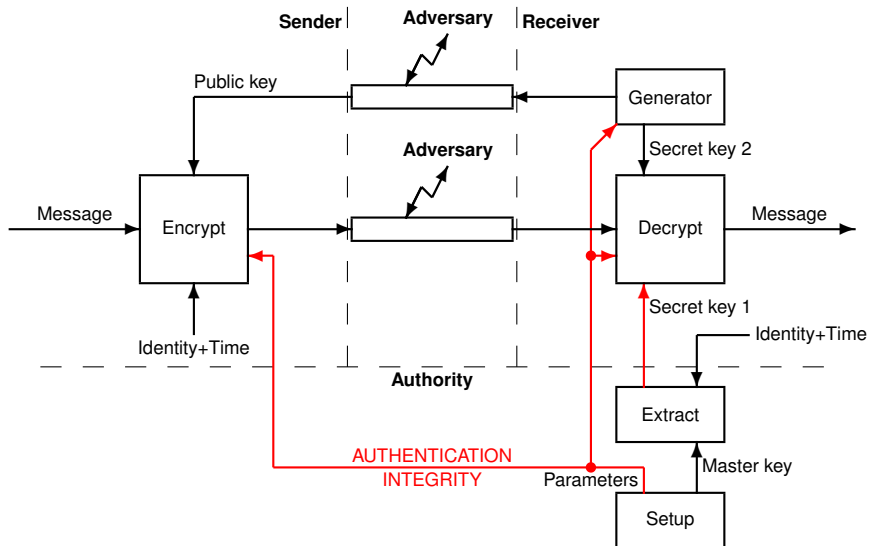
# Public-Key Infrastructure



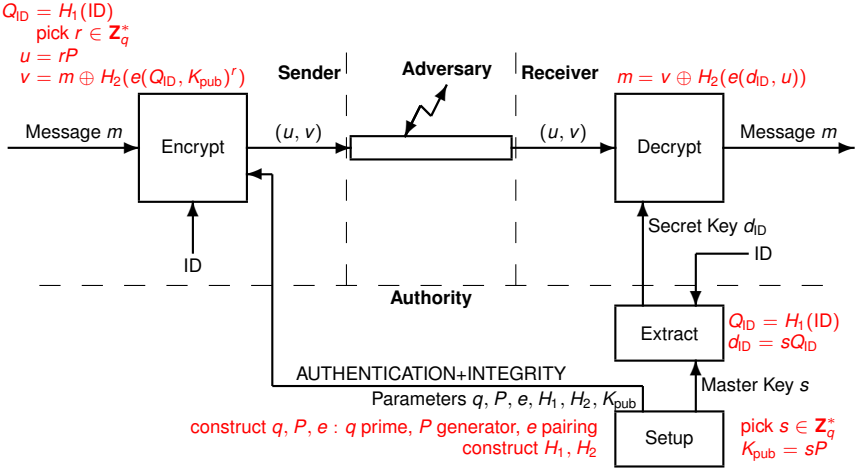
# Identity-Based Encryption



# Certificateless Encryption



# Boneh-Franklin Identity-Based Encryption



# Some Popular Trust Model

- **TLS**: trust model based on a PKI  
clients hold a list of CA public keys and retrieve server certificates
- **SSH**: trust model based on cache  
clients keep in cache the public key of servers  
(first connection may be insecure)
- **PGP**: trust model monitored by users  
users set up their confidence level in obtained public keys  
a “web of trust” can be used to check a public key  
(to check who has put a higher confidence level to this key)

## Trust Establishment

- From Secure Channel to Secure Communications
- Setup of Secure Channels
- Setup by Narrowband Secure Channel
- Setup by a Trusted Third Party
- **Trust Management and Cryptography**

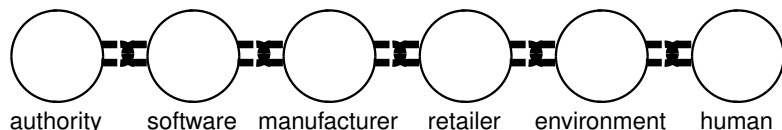


# Metacryptography

## Can we Trust Crypto?

- 2nd law of thermodynamics:  
no matter the real strength of crypto designs, security decreases with time (Moore's law or cryptanalysis)
- wrong hypotheses:  
e.g. we might figure out that factoring is easy  
→ **need for crypto-diversity**
- academic system failure:  
crypto results are done under pressure: too many conferences, too many papers, too many beans to get  
→ many results are wrong  
→ **need for automatic proof verification**
- threat model definition issues:  
some models are complicated and later happen to be irrelevant
- security does not add: secure + secure may be insecure  
→ **need for good composability models**

# Chain of Trust in the PKI Model



- CA must issue correct certificate
- software must include correct CA public keys
- hardware must execute what it is supposed to
- retailer must not add malicious software
- environment must not bypass secure software
- human user must care invalid certificates

# Chain of Trust in Real Life

- software companies add CA's on commercial basis
- some CA's are corruptable
- worms may corrupt CA lists
- users pay no attention to browser warnings

consequence: **phishing attacks**

further thoughts: this is no longer a cryptographic issue  
—→ **education, psychology, ergonomy, technology**

# Several Approaches to Certificate Verification

- **TLS**: verify a certificate every time the public key is used
- **SSH**: verify that a public key has not changed since the last time
- **PGP**: use a public key ring set up by the user (manual verification based on reputation)

# Conclusion

- **secure communication** is essentially solved as long as birth and death are secure
  - birth: need for means to authenticate public keys
  - death: no solution, just behave as if we would never die
- crypto offers many different models
  - **PKI, password-based, ID-based, certificateless, SAS-based**
- correct solution must be determined on a case-by-case basis
- trust establishment is **not a pure-crypto issue**
  - need to address the human factor
  - need to deal with trust management:
    - logistic, software engineering network security

# References

- **Merkle**. Secure Communications over Insecure Channels. *Communications of the ACM* vol. 21, 1978.
- **Gentry**. Certificate-Based Encryption and the Certificate Revocation Problem. *EUROCRYPT 2003*, LNCS 2656.
- **Bellare-Merritt**. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. *IEEE symposium on Research in Security and Privacy*, 1992.
- **C. Gehrman, C. Mitchell, K. Nyberg**. Manual Authentication for Wireless Devices. *RSA Cryptobytes* vol. 7, 2004.

# Must be Known

- secure channels
- Kerberos
- public-key cryptography and man-in-the-middle attacks
- PKI, certificate validation model
- password-based cryptography
- SAS-based cryptography

# Train Yourself

- secure channel:  
final exam 2012–13 ex3  
final exam 2009–10 ex2



- 1 Ancient Cryptography
- 2 Diffie-Hellman Cryptography
- 3 RSA Cryptography
- 4 Elliptic Curve Cryptography
- 5 Symmetric Encryption
- 6 Integrity and Authentication
- 7 Case Studies I
- 8 Public-Key Cryptography
- 9 Trust Establishment
- 10 Case Studies II**

## Case Studies II

- TLS: Transport Layer Security
- The Biometric Passport
- NFC Creditcard Payment

# Example of Critical Application

[E-banking from a browser]

# Requirements

- strong bidirectional **authentication**
- **confidentiality** of communication
- **integrity** of communication
- **non-repudiation** of transaction
- **resilience** to clients in hostile environment

# History

- SSLv1 by Netscape in 1994
- Microsoft version PCT in 1995
- SSLv3 by Netscape in 1995
- TLS/1.0 in 1999 [RFC2246]
- TLS/1.1 in 2006 [RFC4346]
- TLS/1.2 in 2008 [RFC5246]
- TLS/1.3: still draft

Goal: secure any communication (e.g. HTTP) based on TCP/IP

# TLS Record Protocols

Record Protocol is based on TCP

Here are the four protocols based on the Record Protocol:

- Handshake Protocol (for initiating a session)
- Change Cipher Spec Protocol (for setting up cryptographic algorithms)
- Alert Protocol (for managing warnings and fatal errors)
- Application Data Protocol

# Session State

- Session identifier
- Peer certificate (if any)
- Cipher suite choice
  - Algorithm for authentication and key exchange during handshake
  - Cipher Spec: symmetric algorithms (encryption and MAC)
- Master secret (a 48-byte symmetric key)
- nonces (from the client and the server)
- sequence numbers (one for each communication direction)
- compression algorithm (if any)

# Original TLS 1.0 Cipher Suites — i

CipherSuite	Key Exchange	Cipher	Hash
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4_40	MD5
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA-1
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	RSA	RC2_40	MD5
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE	SHA-1
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	DH_DSS	DES40	SHA-1
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE	SHA-1
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	DH_RSA	DES40	SHA-1
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE	SHA-1



# Original TLS 1.0 Cipher Suites — ii

CipherSuite	Key Exchange	Cipher	Hash
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	DHE_DSS	DES40	SHA-1
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE	SHA-1
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	DHE_RSA	DES40	SHA-1
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE	SHA-1
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	DH_anon	RC4_40	MD5
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH_anon	DES40	SHA-1
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE	SHA-1

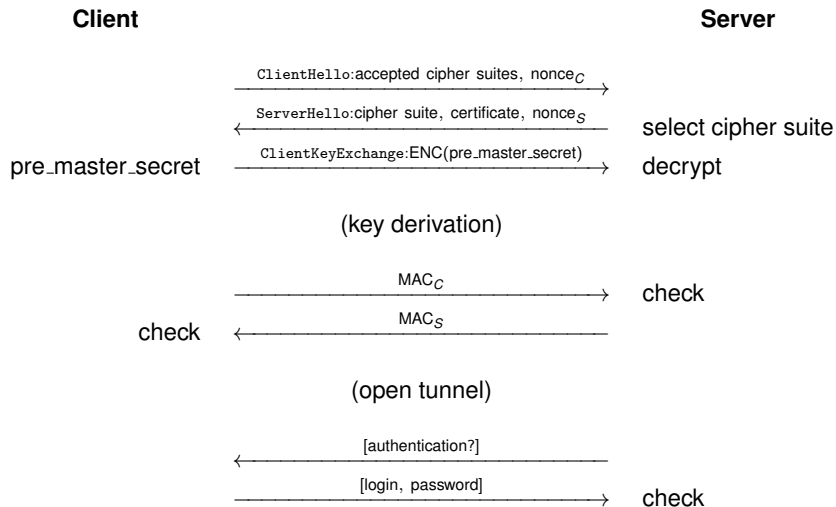
many more in 1.2:

cipher: AES\_GCM, AES\_CCM, CAMELLIA, ARIA

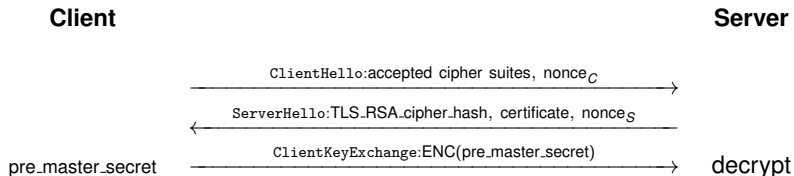
hash: SHA2

“key exchange”: ECDSA, PSK

# A Typical TLS 1.0 Session

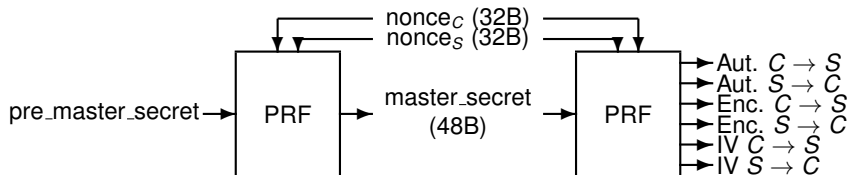


# RSA Key Exchange



- RSA encryption is PKCS#1v1.5
- the RSA public key must be authenticated

# Key Derivation

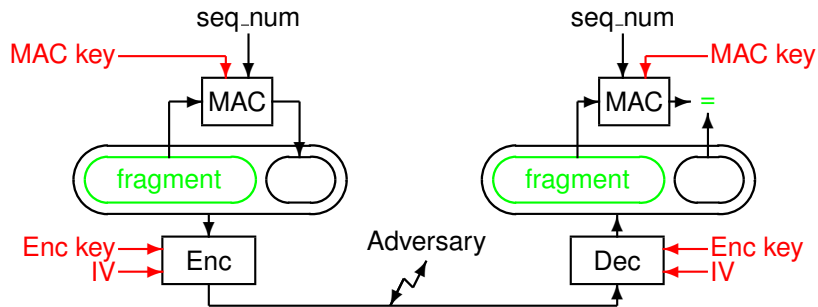


`pre_master_secret` is 48B for RSA key exchange or the obtained Diffie-Hellman key for DH\_RSA, DH\_DSS, DHE\_RSA, DHE\_DSS, and DH\_anon

# Application Data Record Protocol

- split the application data into fragments of at most  $2^{14}$  Bytes and send the fragments separately.
- (optional) compress the fragment
- append a MAC to the fragment  
The MAC is computed on a sequence number, the compression and TLS version materials, the compressed fragment.
- encrypt all this
- send this after a record header (type, version, length)

# Secure Channel in TLS (Using CBC Encryption)



# TLS 1.3

cipher suite in the form

TLS\_KEY\_AUTH\_WITH\_CIPHER\_HASH

- key exchange (KEA) and authentication (AUTH) are separated things
- KEA is (EC)DHE
- AUTH is the way to authenticate peers, it can be with a certificate (RSA or ECDSA) or PSK
- PSK:  
just makes `pre_master_secret` be the result of (EC)DH concatenated with a pre-shared key
- CIPHER: AES-GCM, AES-CCM, CHACHA20-POLY1305
- hash: SHA2

# TLS 1.3 Cipher Suites

TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256  
TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  
TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 (mandatory)  
TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 (recommended)  
TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (mandatory)  
TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (recommended)  
TLS\_DHE\_RSA\_WITH\_AES\_128\_CCM  
TLS\_DHE\_RSA\_WITH\_AES\_256\_CCM  
TLS\_DHE\_RSA\_WITH\_AES\_128\_CCM\_8  
TLS\_DHE\_RSA\_WITH\_AES\_256\_CCM\_8  
TLS\_ECDHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256 (recommended)  
TLS\_ECDHE\_ECDSA\_WITH\_CHACHA20\_POLY1305\_SHA256 (recommended)  
TLS\_DHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256  
TLS\_DHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256  
TLS\_DHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384  
TLS\_DHE\_PSK\_WITH\_AES\_128\_CCM  
TLS\_DHE\_PSK\_WITH\_AES\_256\_CCM  
TLS\_PSK\_DHE\_WITH\_AES\_128\_CCM  
TLS\_PSK\_DHE\_WITH\_AES\_256\_CCM  
TLS\_ECDHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256  
TLS\_ECDHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384  
TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CCM\_8\_SHA256  
TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CCM\_SHA256  
TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CCM\_SHA384  
TLS\_ECDHE\_PSK\_WITH\_CHACHA20\_POLY1305\_SHA256  
TLS\_DHE\_PSK\_WITH\_CHACHA20\_POLY1305\_SHA256

mandatory curve: secp256r1 (NIST P-256)

recommended curve: X25519 [RFC7748]



10

## Case Studies II

- TLS: Transport Layer Security
- The Biometric Passport
- NFC Creditcard Payment

Schweizer Pass  
Passeport suisse  
Passaporto svizzero  
Passaport svizzer  
Swiss passport



# ICAO-MRTD Objectives

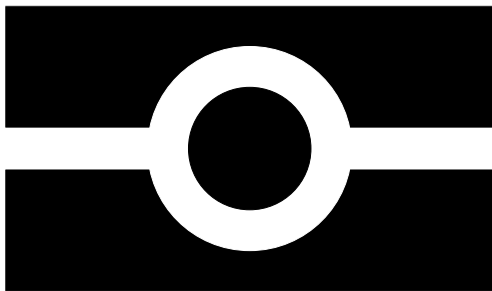
(MRTD=Machine Readable Travel Document)

more secure identification of visitors at border control

- biometrics
- contactless IC chip
- digital signature + PKI

maintained by UN/ICAO (International Civil Aviation Organization)

# How to Distinguish a Compliant MRTD

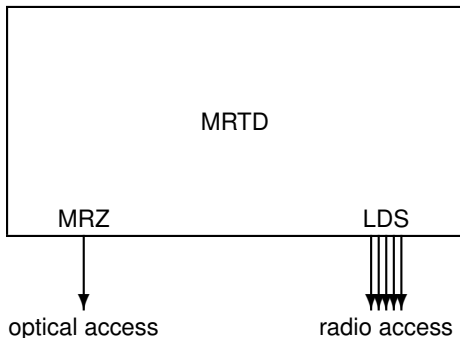


# MRTD History

- 1968: ICAO starts working on MRTD
- 1980: first standard (OCR-B **Machine Readable Zone (MRZ)**)
- 1997: ICAO-NTWG (New Tech. WG) starts working on biometrics
- 2001 9/11: US want to speed up the process
- 2002 resolution: ICAO adopts **facial recognition** (+ optional fingerprint and iris recognition)
- 2003 resolution: ICAO adopts MRTD with **contactless IC media** (instead of e.g. 2D barcode)
- **2004: version 1.1** of standard with ICC
- 2005: deployment of epassports in several countries
- 2006: **extended access control** under development in the EU
- 2007: deployment of extended access control (+ more biometrics)
- now part of Doc9303



# MRTD in a Nutshell



- data authentication by digital signature + PKI  
aka **passive authentication**
- access control + key agreement based on MRZ\_info  
aka **basic access control (BAC)**
- chip authentication by public-key cryptography  
aka **active authentication (AA)**

# Access Control Options

- **none**: anyone can query the ICC, communication in clear
- **basic**: uses secure channel with authenticated key establishment from MRZ
- **extended**: up to bilateral agreements (no ICAO standard)  
EU common criteria: now being implemented





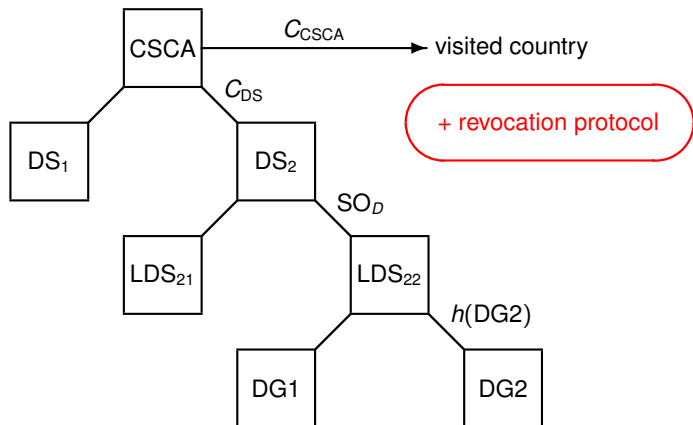
# LDS Structure

- $K_{ENC}$ ,  $K_{MAC}$ ,  $KPr_{AA}$
- COM: present data groups
- DG1: same as MRZ
- DG2: encoded face
- DG3: encoded finger(s)
- DG4: encoded eye(s)
- DG5: displayed portrait
- DG6: (reserved)
- DG7: displayed signature
- DG8: data feature(s)
- DG9: structure feature(s)
- DG10: substance feature(s)
- DG11: add. personal detail(s)
- DG12: add. document detail(s)
- DG13: optional detail(s)
- DG14: security options
- DG15:  $KPr_{uAA}$
- DG16: *person(s) to notify*
- $SO_D$

# SO<sub>D</sub> Structure

- list of hash for data groups DG1–DG15
- formatted signature by DS (include: information about DS)
- (optional)  $C_{DS}$

# (Country-wise) PKI



- one CSCA (*Country Signing Certificate Authority*)
- several DS (*Document Signer*) per country
- $SO_D$ : signature of LDS
- fingerprint of a DG

# Passive Authentication

**goal** authenticate LDS

- after getting  $SO_D$ , check the included certificate  $C_{DS}$  and the signature
  - when loading a data group from LDS, check its hash with what is in  $SO_D$
- stamp by DS on LDS

# Passport: From Paper to Bits

## paper passport

- invisible if not shown
- hard to copy
- photocopies are non-binding
- needs human check
- access control by the holder

## MRTD

- detectable, recognizable
- easy to copy with no AA
- SOD is a digital evidence
- readable automatically
- needs specific access control

# Basic Access Control

**goal** prevent from unauthorized access by the holder (privacy)

- read MRZ (OCR-B)
  - extract MRZ\_info
  - run an authenticated key exchange based on MRZ\_info
  - open secure messaging based on the exchanged symmetric key
- proves that reader knows MRZ\_info





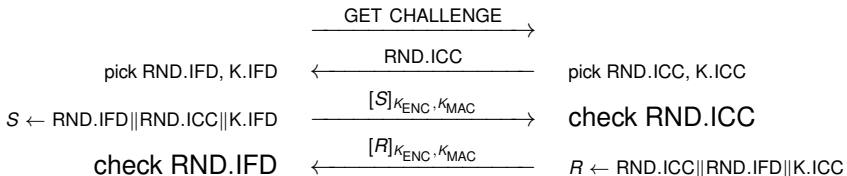
# Basic Access Control

## Authenticated Key Exchange Based on MRZ\_info

IFD

ICC

(derive  $K_{ENC}$  and  $K_{MAC}$  from MRZ\_info)



(derive  $KS_{ENC}$  and  $KS_{MAC}$  from  $K_{seed} = \text{K.ICC} \oplus \text{K.IFD}$ )

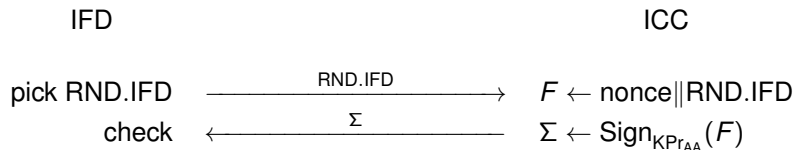
# Active Authentication

**goal** authenticate the chip

- proves that ICC knows some secret key  $KPr_{AA}$  linked to a public key  $KPu_{AA}$  by a challenge-response protocol ( $KPu_{AA}$  in LDS authenticated by passive authentication)

→ harder to clone a chip

# Active Authentication Protocol



# With vs Without Active Authentication

## No Active Authentication

- ICC can be cloned
- simple computations to perform

## Active Authentication

- protection against clones
- requires public-key cryptography in ICC

# RFID Private Collision Avoidance Protocol (ISO 14443)

- for each new singulation protocol  
ICC introduces himself with a pseudo (32-bit number)
- singulation to establish a communication link between reader and ICC of given pseudo
- pseudo is either a constant or a random number starting with 08

# Implementation Discrepancies (2007 Survey)

	shield	singulation	BAC	AA
Switzerland	none	random 08xxxxxx	used	not implemented
United Kingdom	none	random 08xxxxxx	used	not implemented
France	none	random 08xxxxxx	?	?
Australia	none	random xxxxxxxx	used	?
New Zealand	none	constant	used	?
USA	yes	random xxxxxxxx	used	?
Italy	?	constant	?	?
Belgium	none	cste then 08...	used	implemented
Czech Republic	none	random 08xxxxxx	used	implemented
Japan	none	?	not used	not implemented

# With vs Without Faraday Cages

## Regular Document

- can access to ICC without the holder approval

## Metallic Cover

- document must be opened to access to ICC
- more expensive
- not fully effective

# Algorithms (2007 Survey)

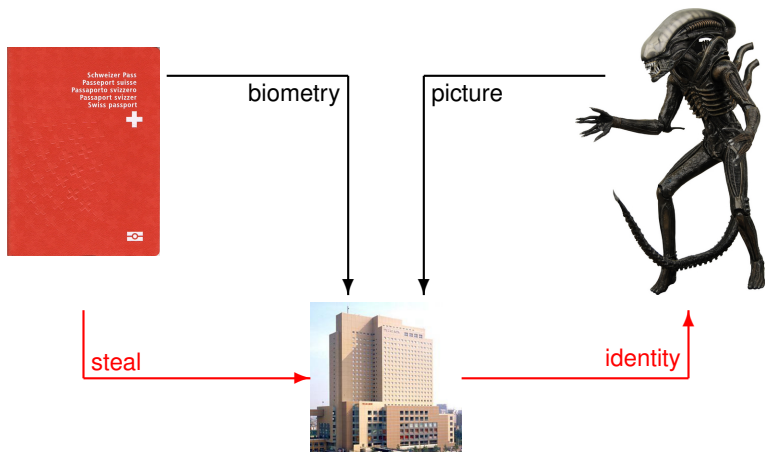
	certificate	SO <sub>D</sub>	AA
Switzerland	ecdsa_with_sha1 824b	ecdsa 512b	n/a
United Kingdom	sha256withRSA 4096b	RSA 2048b	n/a
Czech Republic	rsaPSS (sha256) 3072b	RSA 2048b	RSA 1024b
Belgium	sha1withRSA 4096b	RSA 2048b	RSA 1024b
Germany	ecdsa_with_sha1 560b	ecdsa 464b	n/a
Italy	sha1withRSA 4096b	RSA 2048b	?
New-Zealand	sha256withRSA 4096b	RSA 2048b	?
USA	sha256withRSA 4096b	RSA 2048b	?
Japan	sha256withRSA 4096b	RSA 2048b	n/a
Ireland	sha256withRSA 4096b	RSA 2048b	?
Netherland	sha256withRSA 3072b	RSA 2048b	?
South Korea	rsaPSS (sha256) 3072b	RSA 2048b	?
Sweden	rsaPSS (sha256) 2048b	RSA 2048b	?



# Security and Privacy Issues

- collision avoidance discrepancies  
→ deviating from standard induce leakages
- MRZ\_info entropy  
→ online attack or offline decryption from skimming
- underestimated wireless range limits  
→ claimed to be possible at a distance of 25m
- identity theft (by stealing/cloning MRTD)  
→ facial recognition is weak
- remote passport detection  
→ nice to find passports to steal
- relay attacks
- denial of services
- ...

# Identity Theft



a few 100 customers are enough

# Extended Access Control (EAC)

- **PACE** > BAC
- **Chip Authentication** > AA
- **Terminal Authentication** to access non-mandatory data
- more biometrics (finger) for more secure identification
  
- using state-of-the-art cryptography  
(public-key crypto, PAKE, elliptic curves)
- secure access control but requires a heavy PKI for readers
  
- in-process standard: protocols with different versions, variants,  
described in different documents, with different notations...

# Sequence of Steps for Basic Inspection

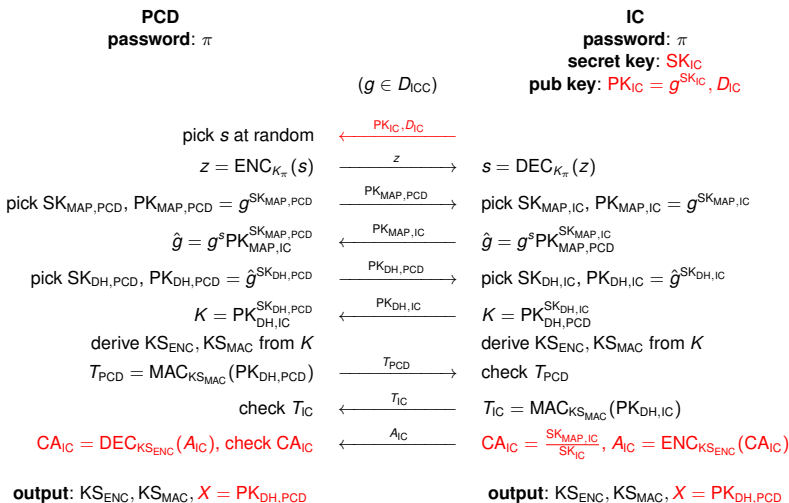
- (optional) run PACE (or BAC), start secure messaging, provide access to less-sensitive data
- passive authentication of  $SO_D$
- (optional) run AA
- read and verify less-sensitive data

# Sequence of Steps for Advanced Inspection

- (optional) run PACE (or BAC), start secure messaging, provide access to less-sensitive data
- (if not done in PACE) run Chip Authentication, restart secure messaging
- passive authentication of  $SO_D$
- (optional) run AA
- run Terminal Authentication v1, provide access to more data
- read and verify data

# PACE (GM v2)

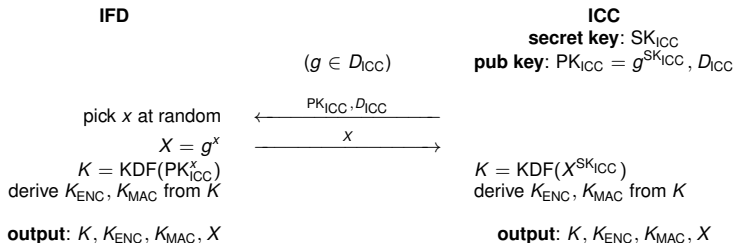
- better protocol (than BAC) based on  $\pi = \text{MRZinfo}$
- can include Chip Authentication



# Chip Authentication

- chip has a static Diffie-Hellman key in DG14 (SOD-authenticated)
- semi-static ECDH with domain parameters  $D_{ICC}$
- replace the secure messaging keys

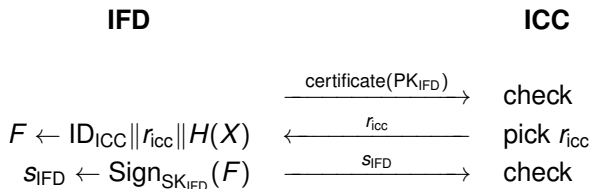
→ resists passive attacks



# Terminal Authentication

- terminal sends a certificate to chip (ECDSA)
- terminal signs a challenge + ephemeral key  $X$  from Chip Authentication
- $ID_{ICC}$  set to serial number (for BAC) or to ephemeral key of ICC (for PACE)

→ strong access control





# Terminal Authentication Issues

Terminal revocation issue:

- MRTDs are not online!
- MRTDs have no reliable clock

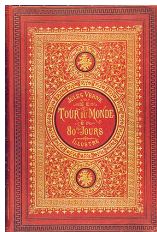
→ MRTD must trust readers to revoke themselves

# Information Leakage

- $SO_D$  leaks the digest of protected DGs before passing EAC
- could be used to recover missing parts from exhaustively search
- could be used to get a proof if DG is known

# Conclusion on MRTD

- **LDS**: contains too much private information
- **passive authentication**: leaks evidence for LDS
- **BAC**: does a poor job
- **secure messaging**: OK
- **AA**: leaks digital evidences, subject to MITM
- **EAC**: much better, but still leaks + revocation issue
- **RFID**: leaks
- **biometrics**: leaks template



“Les passeports ne servent jamais qu’à gêner les honnêtes gens et à favoriser la fuite des coquins.”

Jules Verne, 1872

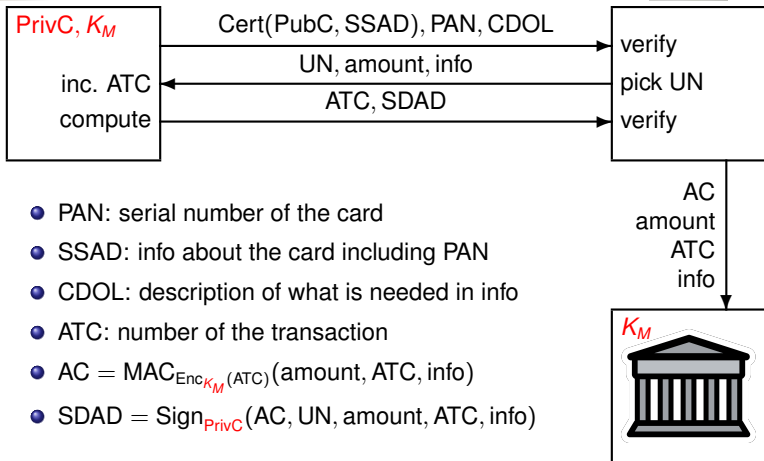
*Le tour du monde en 80 jours*

10

## Case Studies II

- TLS: Transport Layer Security
- The Biometric Passport
- NFC Creditcard Payment

# (Simplified) EMV PayPass Protocol

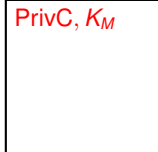


- PAN: serial number of the card
- SSAD: info about the card including PAN
- CDOL: description of what is needed in info
- ATC: number of the transaction
- $AC = MAC_{Enc_{K_M}}(ATC)(amount, ATC, info)$
- $SDAD = Sign_{PrivC}(AC, UN, amount, ATC, info)$

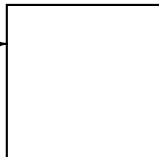
# From Paper to Bits...

- holder is not aware a payment is happening
- holder is not aware of the payment amount
- no access control of the payment terminal (no PIN)
- payee is not authenticated (info could be anyone)
- privacy issue (SSAD leaks)

# Skimming

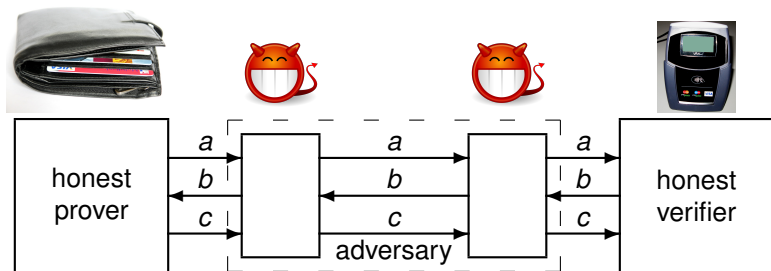


Cert(PubC, SSAD), PAN, CDOL



get name on card, credit card number, expiration date, etc

# Relay Attacks

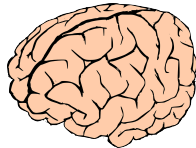
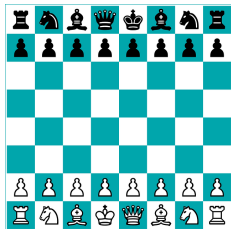
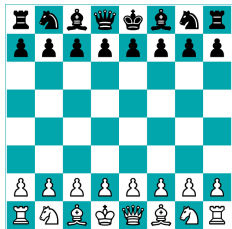




# Relay Attacks in Real

- opening cars and ignition (key with no button)
- RFID access to buildings or hotel room
- toll payment system
- NFC credit card (for payment with no PIN)
- access to public transport
- ...

# Playing against two Chess Grandmasters



# Conclusion

- TLS: standard for e-commerce, suffer from PKI weaknesses
- MRTD: secure data authentication, poor privacy
- EMV PayPass: secure for payee, not payer, poor privacy
  
- they all put together all cryptographic ingredients quite nicely
- they are permanently improved to fix mistakes and use the state-of-the-art cryptography

# References

- **Juels-Molnar-Wagner**. Security and Privacy Issues in E-Passports. In *SecureComm 2005*, IEEE.
- **Chaabouni-Vaudenay**. The Extended Access Control for Machine Readable Travel Documents. In *Biosig 2009*, LNI 155.

# Train Yourself

- biometric passport: final exam 2015–16 ex3