Cryptography and Security

Serge Vaudenay

http://lasec.epfl.ch/

LASEC

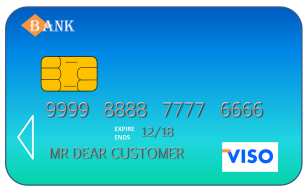
Roadmap

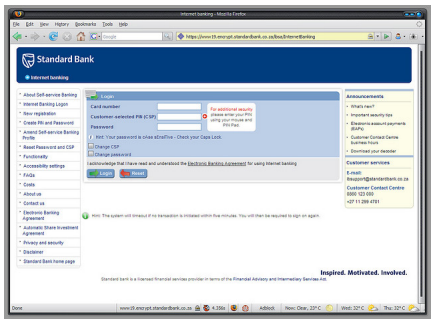
- scope of cryptography
- prehistory (before XX-th Century)
  - transposition and substitution
  - Vigenère
- pre-modern cryptography
  - Kerckhoffs principle
  - Enigma
- cryptography by information theory
  - Vernam
  - Shannon

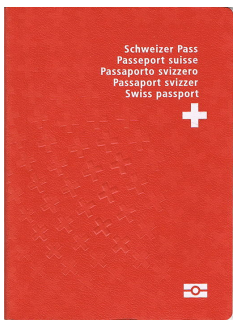
1 Ancient Cryptography

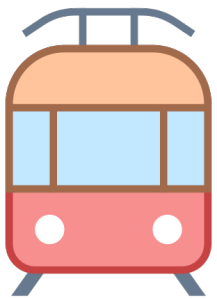
- Scope of Cryptography
- Cryptography Prehistory
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

Cryptography = Science of Information and Communication Security









Evolution

- **Prehistory**
  secret development
  cryptography before communication systems
  (confidentiality/privacy)
- **Modern cryptography**
  academic research
  for mass communication
  (confidentiality/privacy, detection of malicious modification, data authentication, non-repudiation, access control, timestamping, fair exchange, digital rights management, etc)

Applications

- bank cards
- Internet (e-commerce)
- mobile telephony (DECT, GSM, GPRS, EDGE, 3G, 4G, 5G, ...)
- mobile communication (Bluetooth, WiFi...)
- e-passport
- traceability, logistic & supply chains (RFID)
- pay-TV, DRM
- access control (car lock systems, metro...)
- payment (e-cash)
- electronic voting

Cryptography versus Security

- cryptography: a toolbox for setting up security infrastructure
- security experts often assume cryptography does a good job
- cryptographic tools are pretty good, but not for everything
  some can easily be misused
- proper usage of cryptography still requires to master it

Cryptography vs Coding Theory

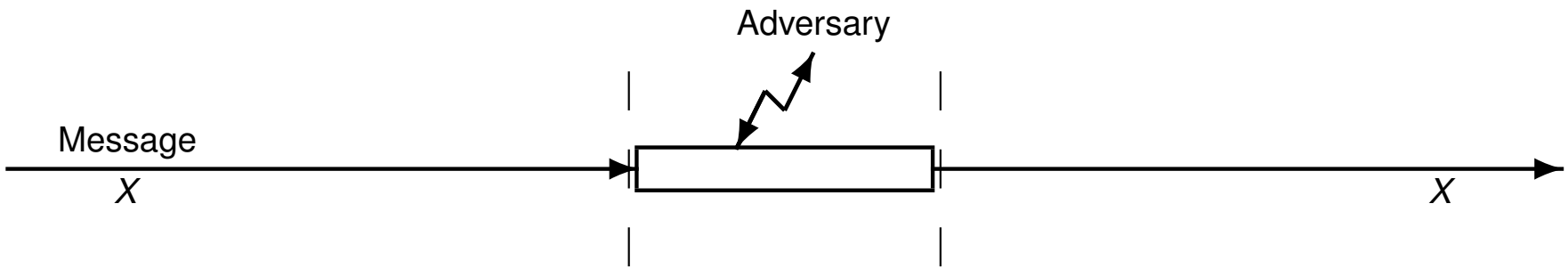
- **Code**
  a system of symbols which represent information
- **Coding theory**
  science of code transformation which enables to send information through a communication channel in a reliable and efficient way ($\rightarrow$ dummy adversary)
- **Cryptography**
  (obsolete definition) the science of secret codes, enabling the confidentiality of communication through an insecure channel ($\rightarrow$ 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)

Problem of this Lecture: Secure Communication over an Insecure Channel

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

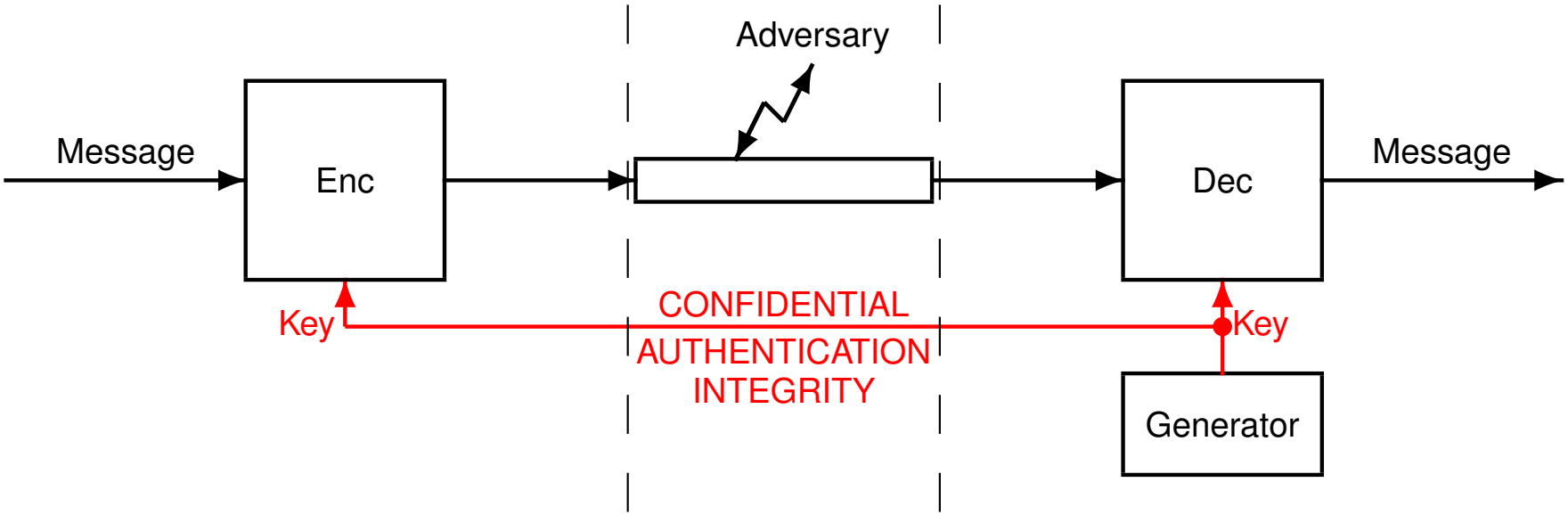
Basic Security Properties

- **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

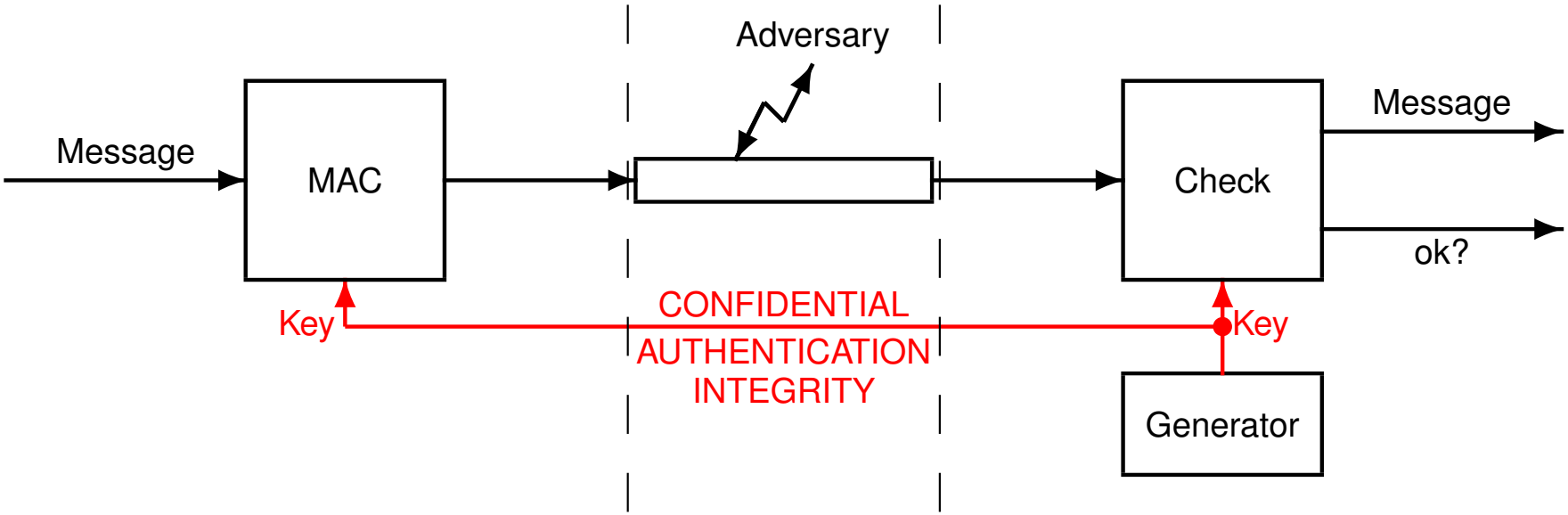
Main Cryptographic Primitives in this Lecture

- symmetric encryption
- message authentication code
- key agreement protocol
- public-key cryptosystem
- digital signature

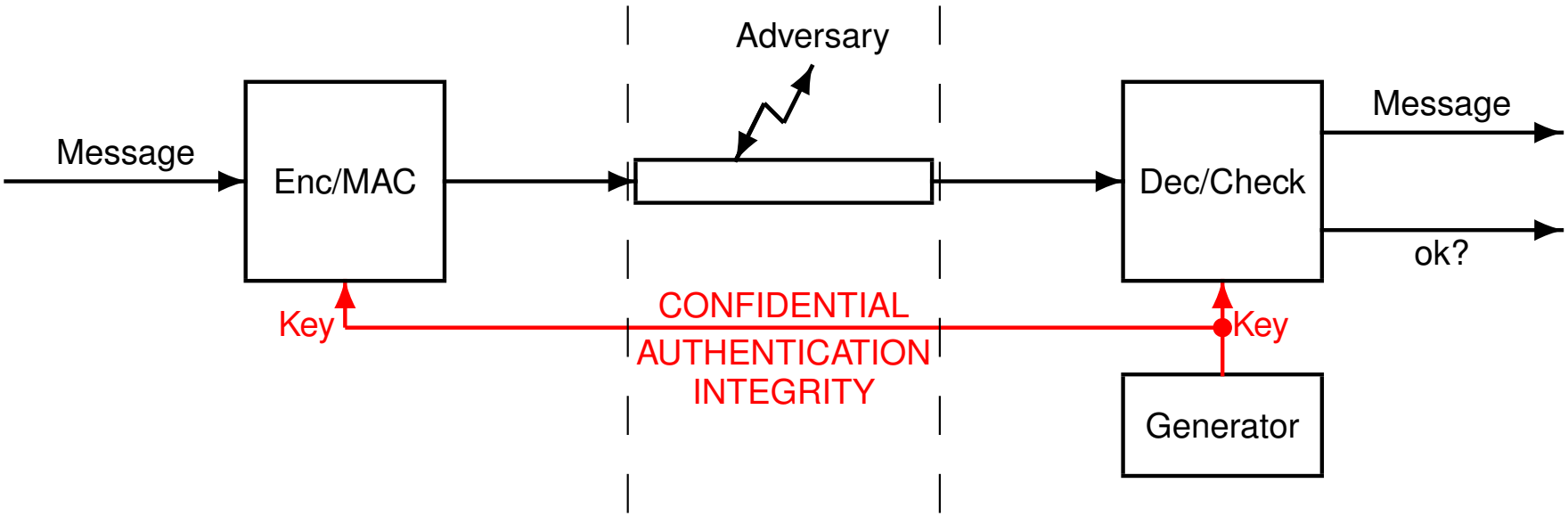
Symmetric Encryption



Message Authentication Code



Secure Comm. based on Conventional Cryptography

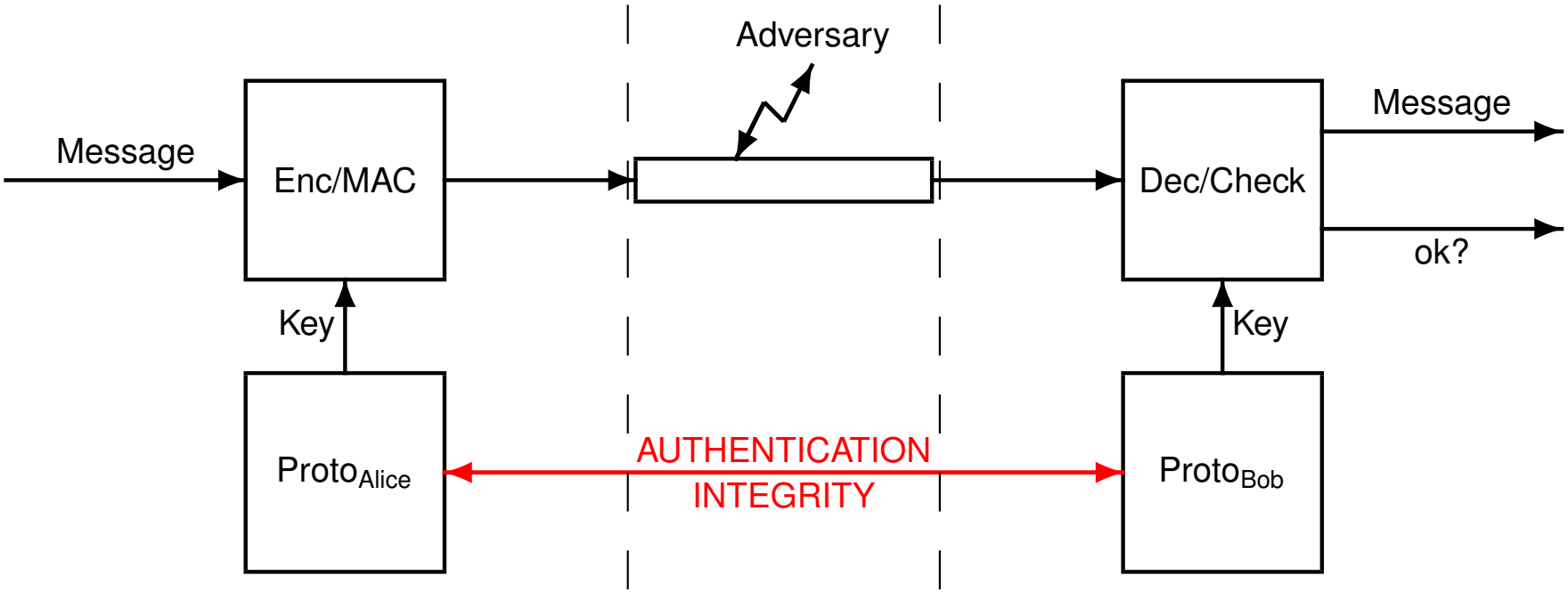


Problem of Symmetric Cryptography

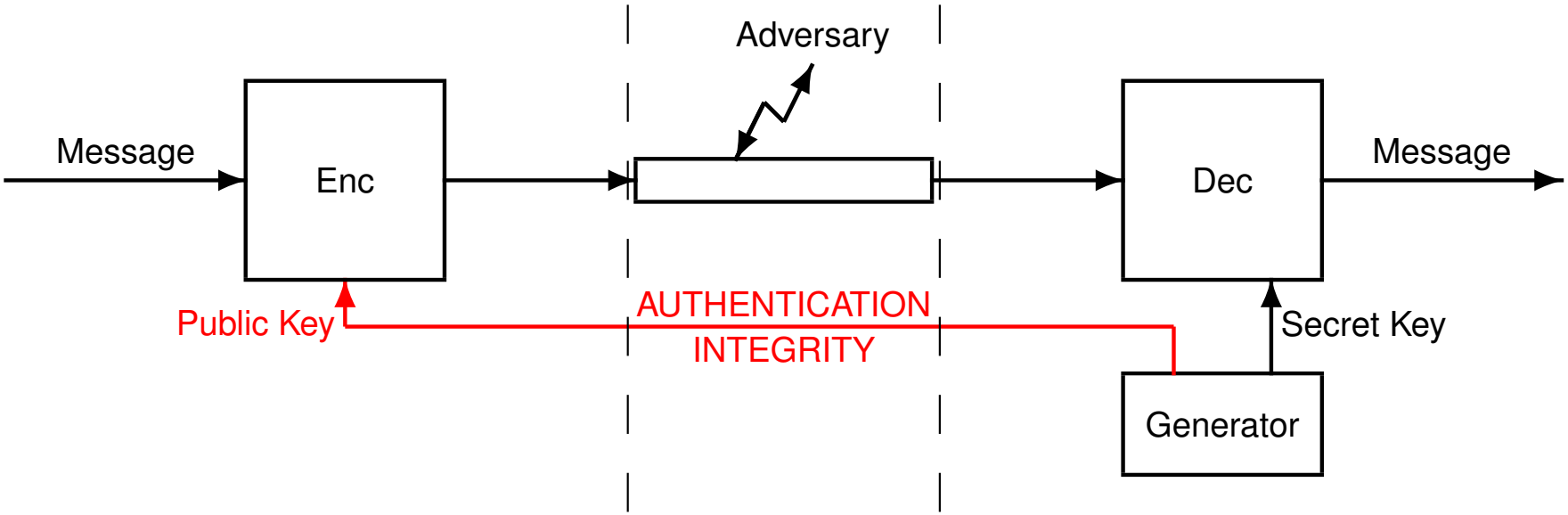
Q: `What is the main problem of symmetric-key cryptography?`
A: `Jr zhfg frg hc n flzzrgevp xrl`

▸ ROT13

Key Agreement Protocol



Public-Key Cryptosystem (Key Transfer)

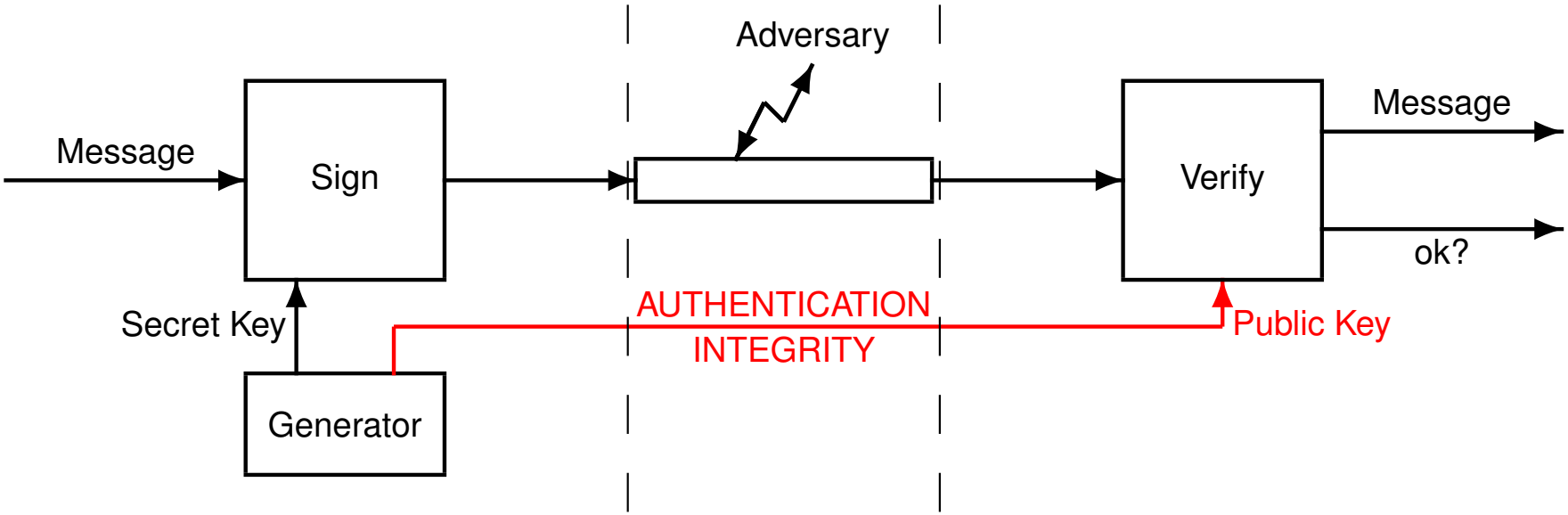


Problem of Public-Key Cryptography

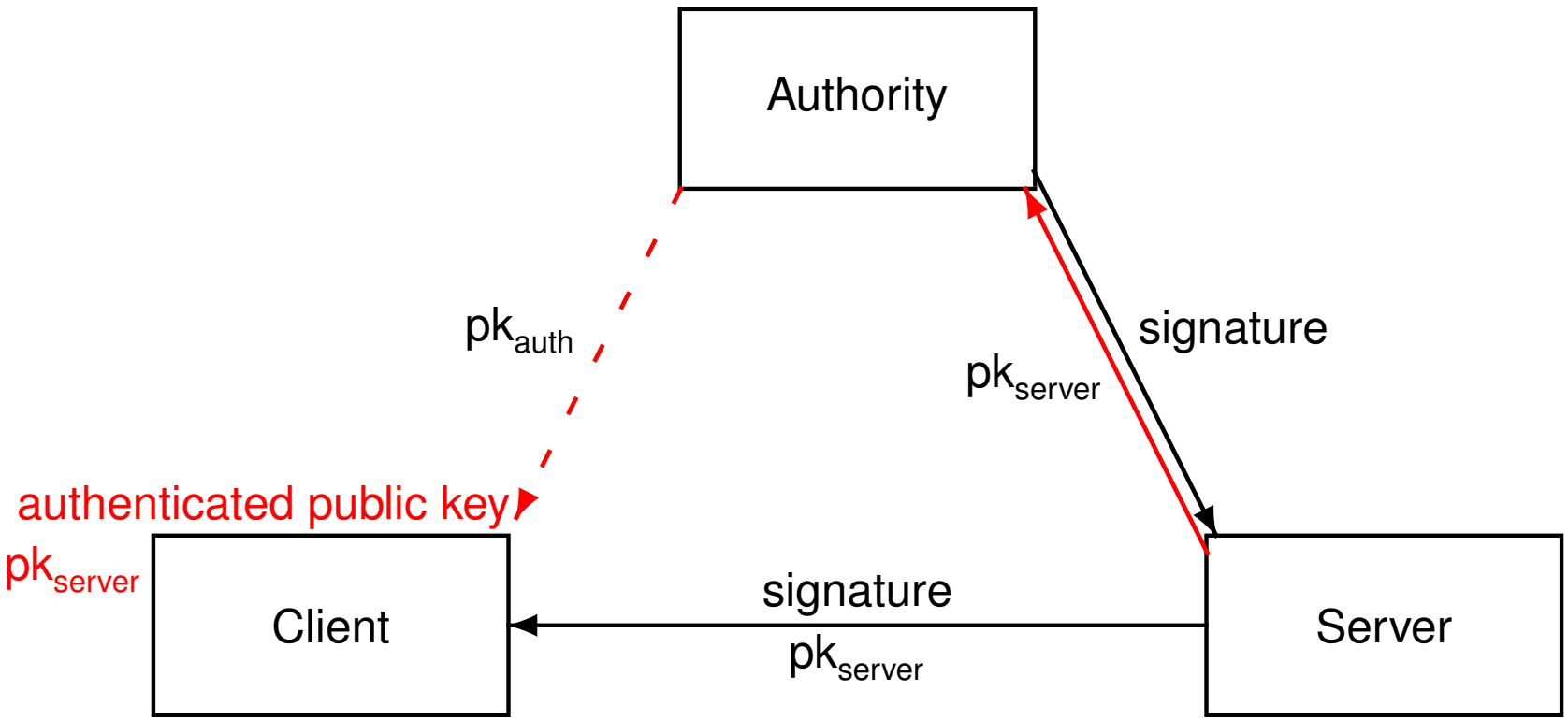
Q: `What is the main problem of public-key cryptography?`
A: `Jr zhfg nhguragvpngr n choyvp xrl`

▸ ROT13

Digital Signature (Public-Key Certificate)



Example of TLS



Secure Communication Standards

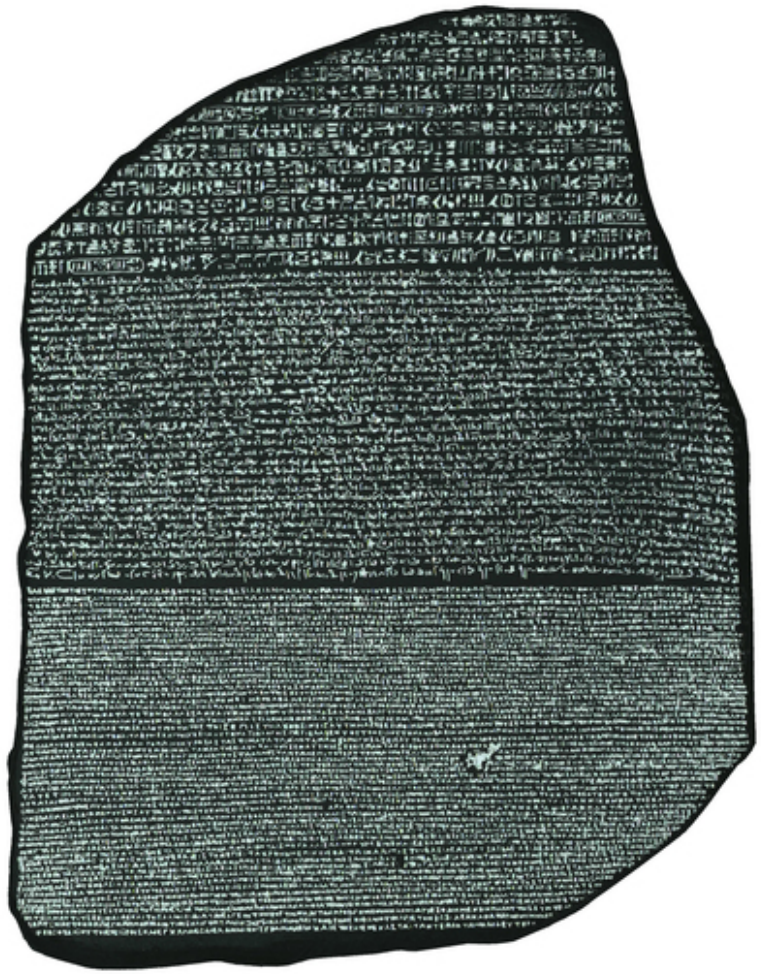
- TLS (e-commerce, business-to-customer)
- IPSEC (VPN, corporate networks)
- SSH (secure remote connections)
- PGP (secure peer-to-peer, secure email)
- GSM/GPRS/3G/4G/5G (mobile telephones)
- Bluetooth (wireless local networks)
- WPA (WiFi)
- MRTD (e-passports)
- ...

1 **Ancient Cryptography**

- Scope of Cryptography
- Cryptography Prehistory
- Pre-Modern Industrial Cryptography
- Cryptography and Information Theory

Secret Writing

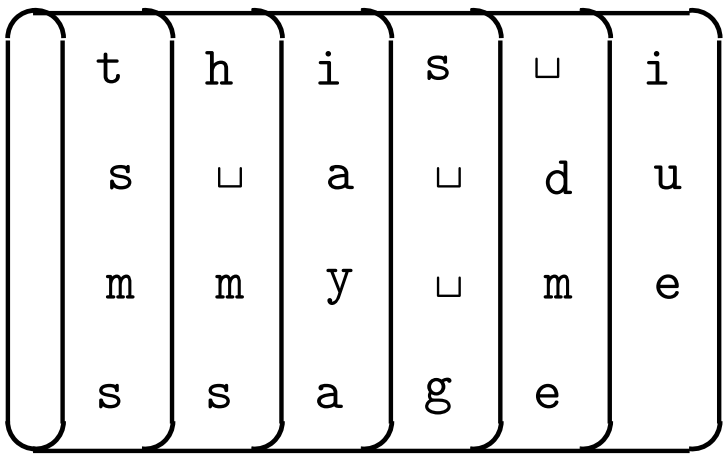
Hieroglyphs!



Transpositions

Spartan scytales

this␣is␣a␣dummy␣message



TSMSH␣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 $\longrightarrow$ FDHXDV

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 $\longrightarrow$ EBG

Application: quiz

Q: Where can we find good quiz?

A: va pnenzone pnaqvrf

▶ 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 $\longrightarrow$ LCNMRT

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

Quiz:

Q: How to break this?

A: ol fgngvfgvpny nanylfvf

▸ ROT13

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.

Vigenère Cipher

Plaintext: `this is a dummy message`

Key: `ABC`

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

Ciphertext: `TIKSJUAEWMNAMFUSBIE`

e.g. $\mathtt{y} + \mathtt{C} = \mathtt{A}$.

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) $a \leftrightarrow 0$, $b \leftrightarrow 1$, $c \leftrightarrow 2$, ... this is the addition modulo 26
(group $\mathbf{Z}_{26}$)

Column-Dependent Substitution

A	B	C
t	h	i
s	i	s
a	d	u
m	m	y
m	e	s
s	a	g
e		

$\longrightarrow$

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

Kasiski Test Example

→ look at unexpectedly frequent patterns

CHR EEVOAHMAERATB IAXXWTNXBEEOPHBSBQMQEQERBW
RVXUOAKXAOSXXWE AHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTU I ADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJ I KNBHWRJGNMGJSGLXFEYPHAGNRB I EQJT
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBB I
PEEWEVKAKOEWADREMXMTBHHCHRTKDNVRZCHRCLQOHP
WQA I I WXNRMGWO 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 occurences 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 $n = 4$, $t = 2$: $\{0,1\}$, $\{0,2\}$, $\{0,3\}$, $\{1,2\}$, $\{1,3\}$, $\{2,3\}$

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

  Application: if we draw n balls with replacement in a jar of $1/p$ numbered balls, the probability to pick ball number u exactly t times is $\binom{n}{t} p^t (1-p)^{n-t}$
  (binomial distribution)

Are 5 Occurrences Significant?

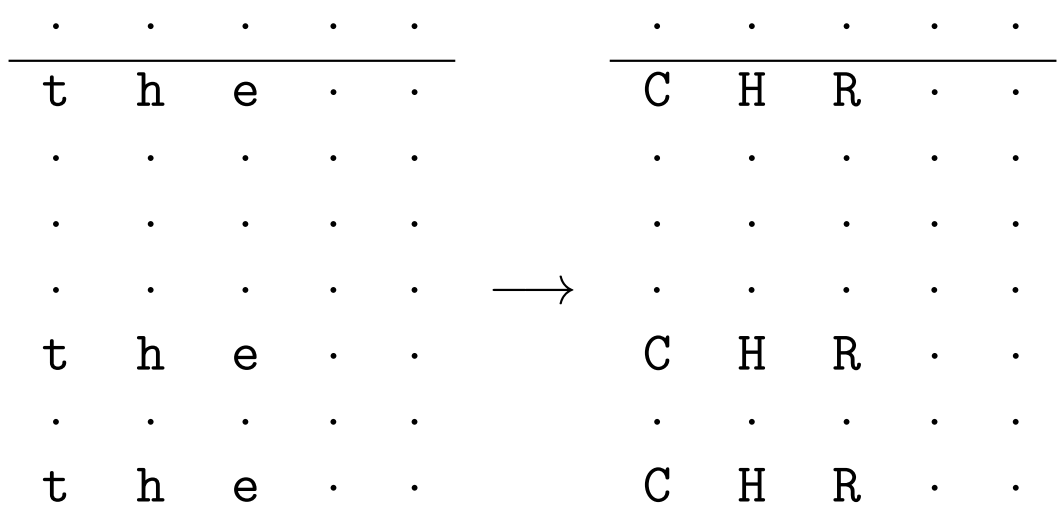
In a truly random sequence of 313 characters $x_1 x_2 \ldots x_{313}$ with alphabet of 26 letters

- 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
- $\Pr[n_{abc} = t] = \binom{n}{t} p^t (1-p)^{n-t}$
  ($\lambda = n \times p$ is small so $\Pr[n_{abc} = t] \approx \frac{\lambda^t}{t!} e^{-\lambda}$: Poisson distribution)
- Application: $\Pr[\exists a, b, c \quad n_{abc} \geq 5] \approx 2.42 \times 10^{-7}$

  observing 5 occurrences of `CHR` is significantly odd

Where does CHR Come From?

key of length multiple of 5 + frequent trigram



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

Index of Coincidence

$$\begin{aligned}\text{Index}(x_1,\ldots,x_n) &= \Pr_{I,J}[x_I = x_J | I \neq J] \\ &= \frac{1}{n(n-1)} \sum_{\substack{1\leq i,j\leq n \\ i\neq j}} 1_{x_i=x_j} \\ &= \sum_{c\in Z} \frac{n_c(n_c-1)}{n(n-1)}\end{aligned}$$

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

Proposition

For any permutation σ over Z, we have

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

For any permutation σ of $\{1,\ldots,n\}$, we have

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

the index of coincidence is invariant by substitution and transposition

Expected Index of Coincidence

$$
\begin{aligned}
E(\text{Index}(x_1, \ldots, x_n)) &= \frac{1}{n(n-1)} \sum_{\substack{1 \le i,j \le n \\ i \neq j}} \Pr[x_i = x_j] \\
&= \sum_{c \in Z} f_c^2
\end{aligned}
$$

if all x_i have i.i.d. distribution with frequency table f_c

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

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:

CEHRIWBPBEBXKSEWMKVTWLSTDTXIGSXLVUNWGXLSXLZLSNRMGEABJRRNXMNHAVEPEKAMBHDZCHAXGIE

(string of 79 characters)

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

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)

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

this is high enough!

Example — iii

Next:

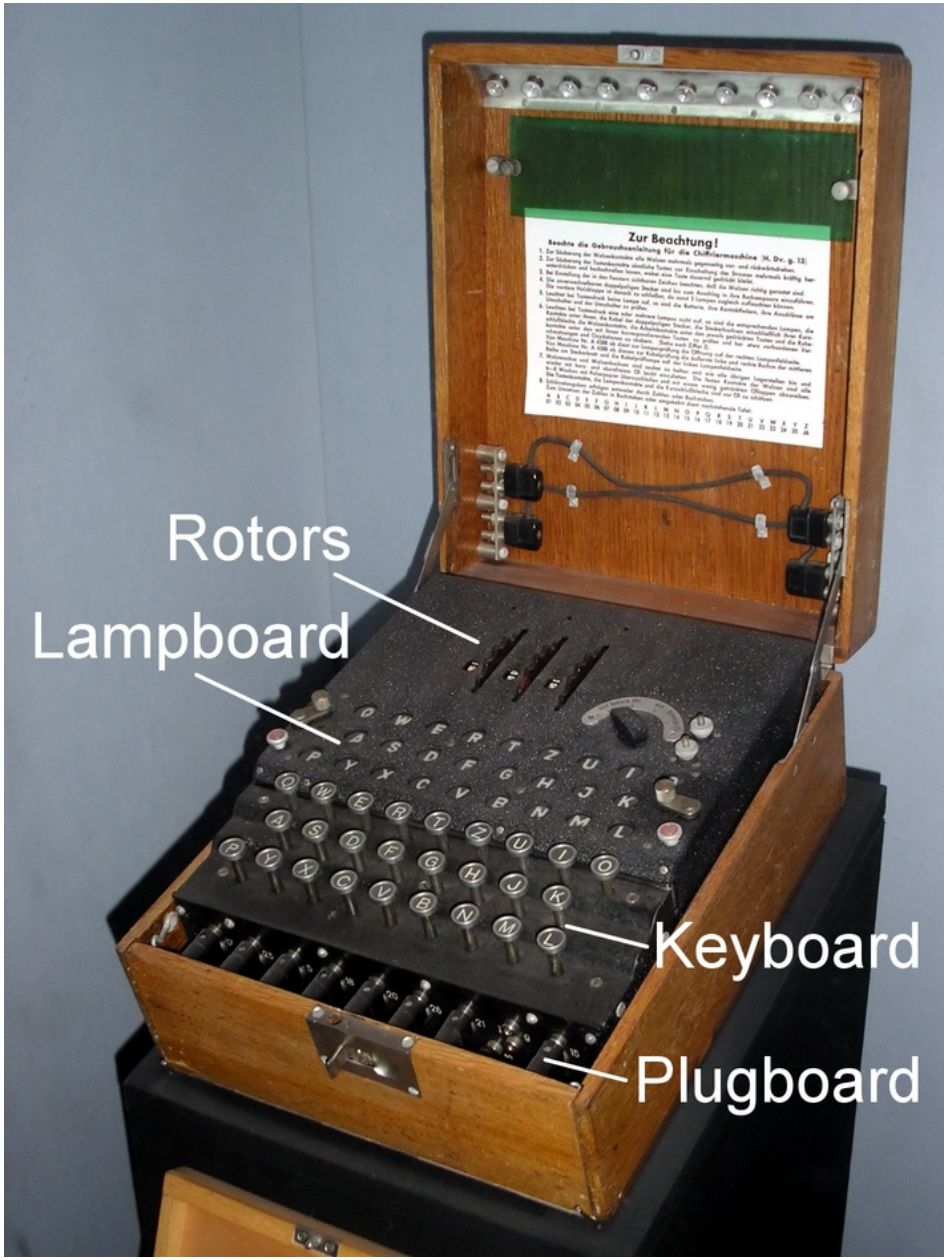
- do a statistical analysis in each column
- look at cross-column indices
  (find the difference between two letters of the key)

1. **Ancient Cryptography**
   - Scope of Cryptography
   - Cryptography Prehistory
   - Pre-Modern Industrial Cryptography
   - Cryptography and Information Theory

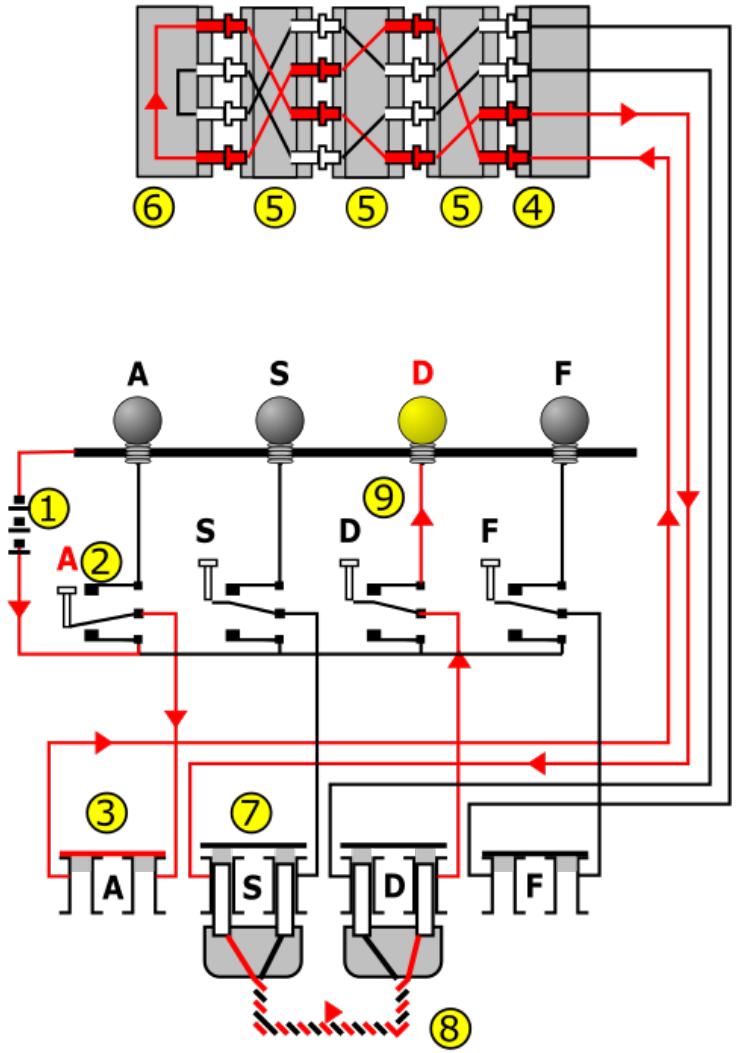
Enigma

- electro-mechanical encryption device (typewriter)
- could be plugged to a radio transmiter
- patented (1918)
- developped 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

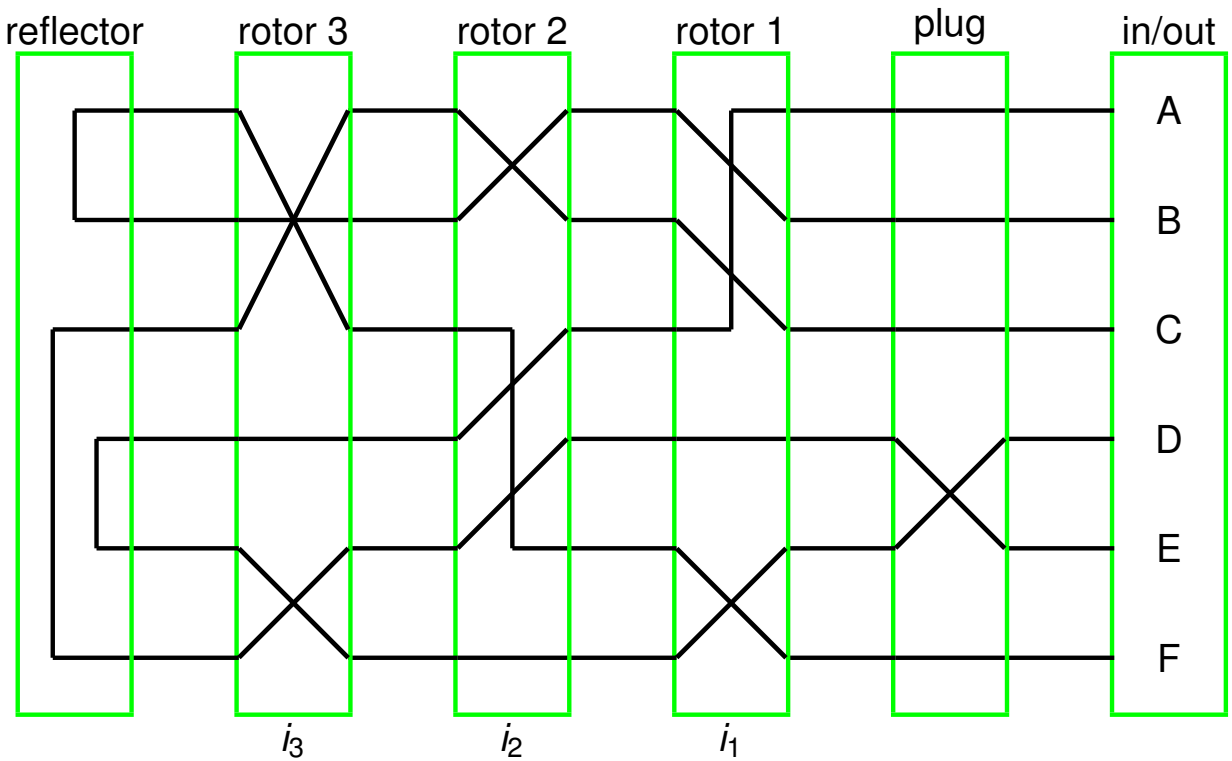


Enigma Circuit



https://en.wikipedia.org/wiki/Enigma_machine

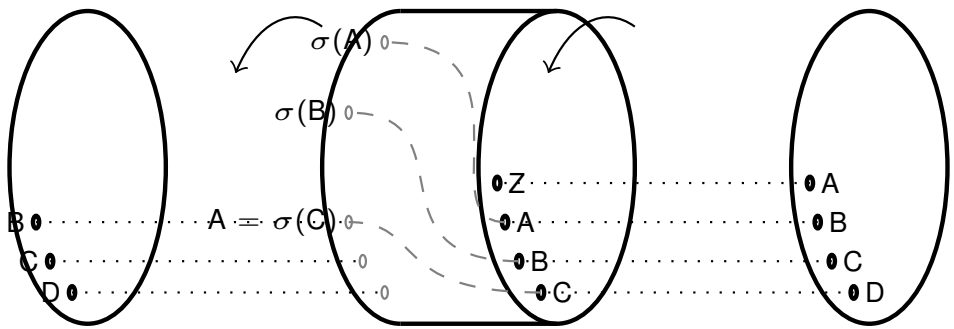
Example: DEAD BEEF



`deadbeef` $\longrightarrow$ `AADCCBBB`

Enigma Building Blocks

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



The Enigma Cipher (Mathematically)

Secret key: 3 components:

- σ (involution made of 6 pairs)
- an ordered choice $\alpha, \beta, \gamma \in S$ of pairwise different permutations (from a box of 5 rotors)
- a number a (initial position of rotors)

Plaintext: $x = x_1, \ldots, x_m$

Ciphertext: $y = y_1, \ldots, y_m$

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

- σ: number of involutions with 14 fixed points

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

- α, β, γ: number of choices for the rotors

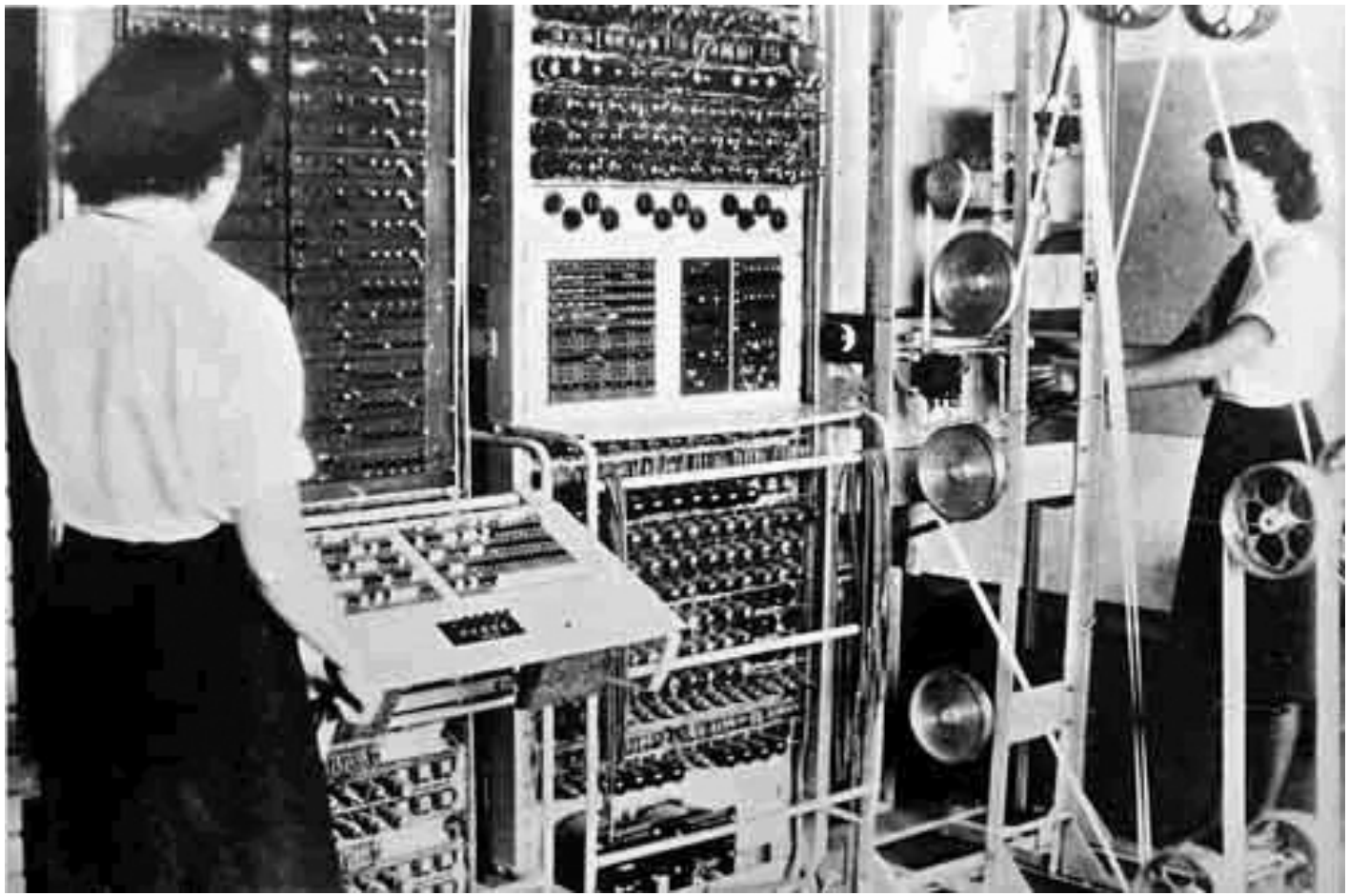
$$5 \times 4 \times 3 = 60 \approx 2^6$$

- *a*: number of initial positions

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

total: 57 bits

A Turing Machine



Q

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 assumes 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.

Kerckhoffs Principles - Translation

Google Translate:

1. The system must be materially, if not mathematically, indecipherable;
2. It must not require secrecy, and it must be able to fall between the hands of the enemy;
3. The key must be able to be communicated and retained without the aid of written notes, and be changed or modified at the discretion of the correspondents;
4. It must be applicable to telegraphic correspondence;
5. It must be portable and its handling or operation must not require assistance of several people;
6. Finally, it is necessary, given the circumstances which require its application, that the system is easy to use, requiring neither mental tension nor knowledge of a long series of rules to observe.

My translation:

1. The system must be secure.
2. Security must not depend on the secrecy of the algorithm.
3. The cryptographic key must be easy to communicate or change.
4. It must be compatible with telegraphic systems.
5. It must be portable and usable by a single person.
6. It should remain easy to use in stressful circumstances.

Evolution

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

The Laws of Modern Cryptography

Law II: scalability — 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

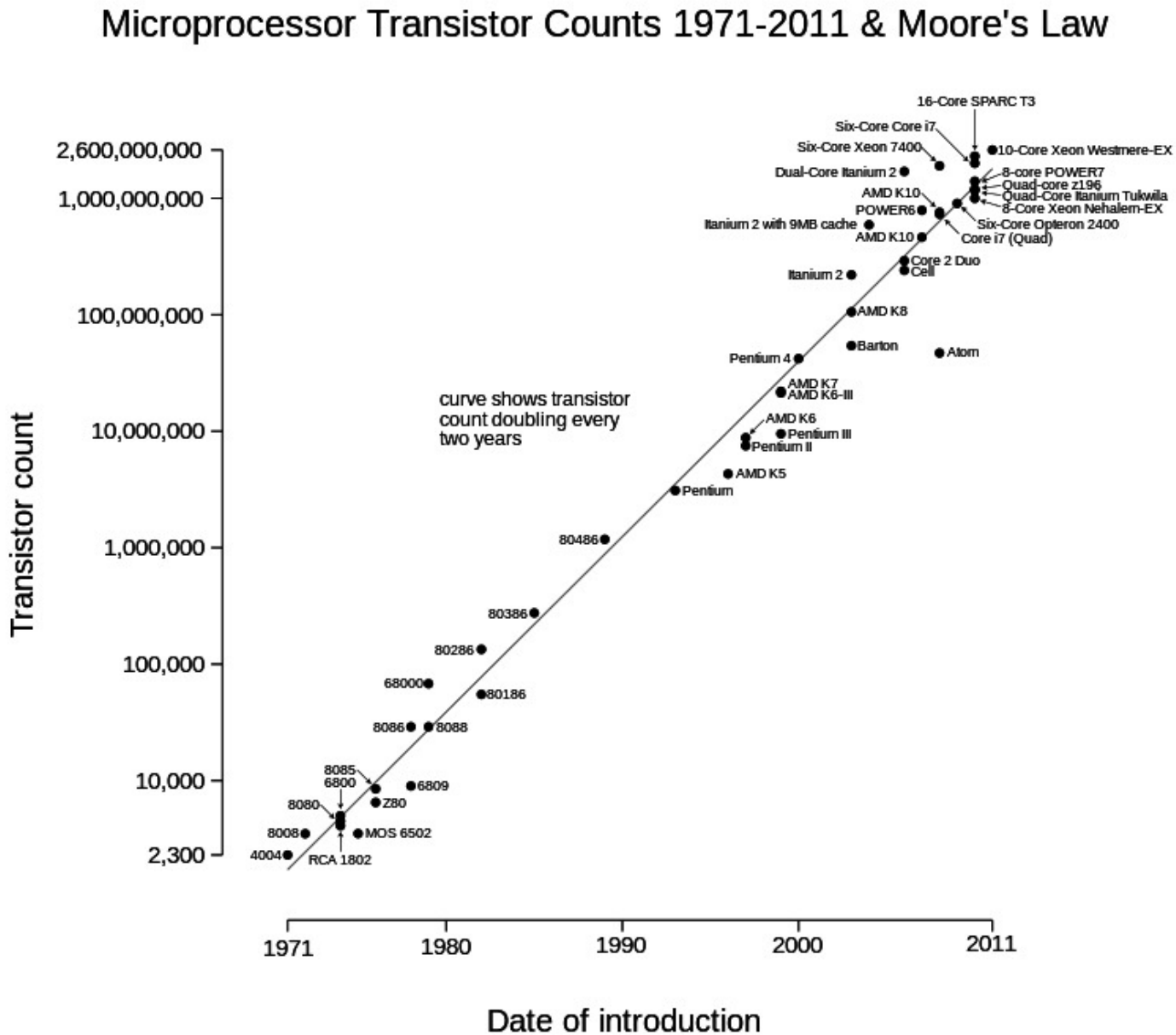
The Laws of Modern Cryptography

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
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
AMD K10
Quad-Core Itanium Tukwila
POWER6
8-Core Xeon Nehalem-EX
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802
1971
1980
1990
2000
2011
Date of introduction

Security by Key Length

- a 2007-PC could test 1 000 000 keys per second
  a 32Y times led to a 10^5 factor speed up
  $\rightarrow f_t = 10^6 \times 10^{5\frac{t-2007}{32}}$ keys per second can be tested at time t
- the number of keys we can try with one processor between time t_0 and time $t_0 + \Delta$ is

$$\begin{aligned}
\int_{t_0}^{t_0+\Delta} f_t \, dt &= \int_{t_0}^{t_0+\Delta} 10^6 \times 10^{\frac{5}{32}(t-2007)} \, dt \\
&= \frac{10^6}{\frac{5}{32}\ln 10} \times 10^{\frac{5}{32}(t_0-2007)} \left(10^{\frac{5}{32}\Delta} - 1\right) \\
&\approx \text{cte} \times 2^{0.52\times(t_0+\Delta-2007)}
\end{aligned}$$

- **assuming** that
  - f_t is correct,
  - the key length is of 128 bits,
  - and we have 2^{80} processors which we maintain up to date,

  we need to run until $t_0 + \Delta = 2100$ to break it

A 128-Bit Key

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

number of possible combinations:

$$\overbrace{2 \times 2 \times 2 \times \cdots \times 2}^{128 \text{ times}}$$

$$= \quad 2^{128}$$

$$= \quad \underbrace{340\,282\,366\,920\,938\,463\,463\,374\,607\,431\,768\,211\,456}_{39 \text{ digits}}$$

Exhaustive Search on 128 Bits

- in 2007, a standard PC could test 1 000 000 keys per second
- to run exhaustive search within 14 billion years, we need 770 000 billons of 2007-PCs!
- **if the Moore law goes on**, a single 2215-PC will do it in a second
- better create the Big Bang and take **14 billion years of vacations** to solve the problem within a second!

Two Revolutions

- **communicating**
  information theory
  mass communication (radio)
  $\rightarrow$ we need standard crypto
- **computing**
  computer science
  automata (electromechanic devices)
  $\rightarrow$ adversaries have more power

1. **Ancient Cryptography**
   - Scope of Cryptography
   - 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

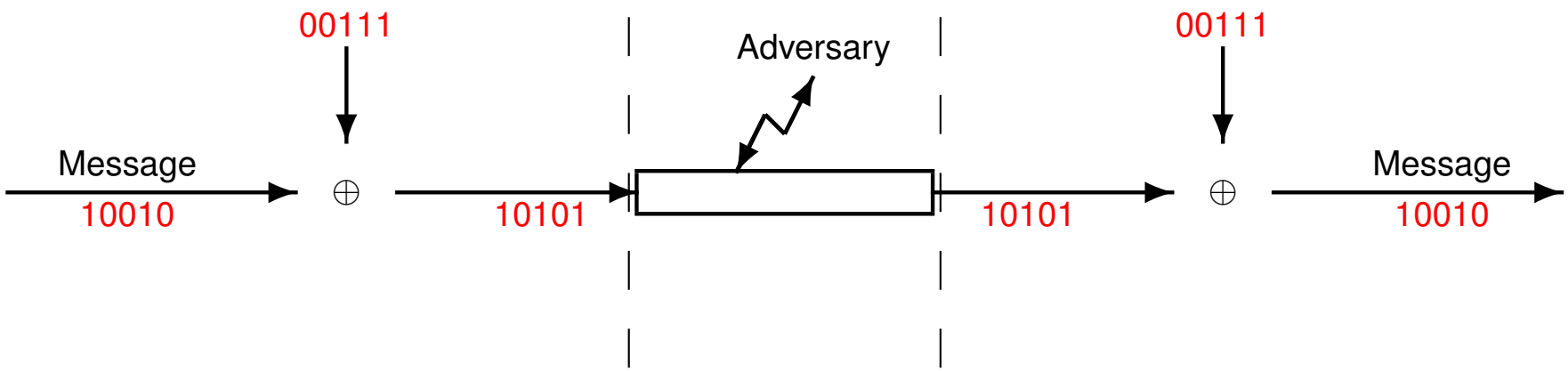
- XOR: binary addition where carry bits are ignored
- XOR: addition modulo 2
- bitwise XOR of two bitstrings:

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

- 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

⊕	0	1
0	0	1
1	1	0



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)

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

- Encrypting X with K: compute $X \oplus K$
- Decrypting Y with K: compute $Y \oplus K$

	(X)	10010
$\oplus$	(K)	00111
$=$	(Y)	10101

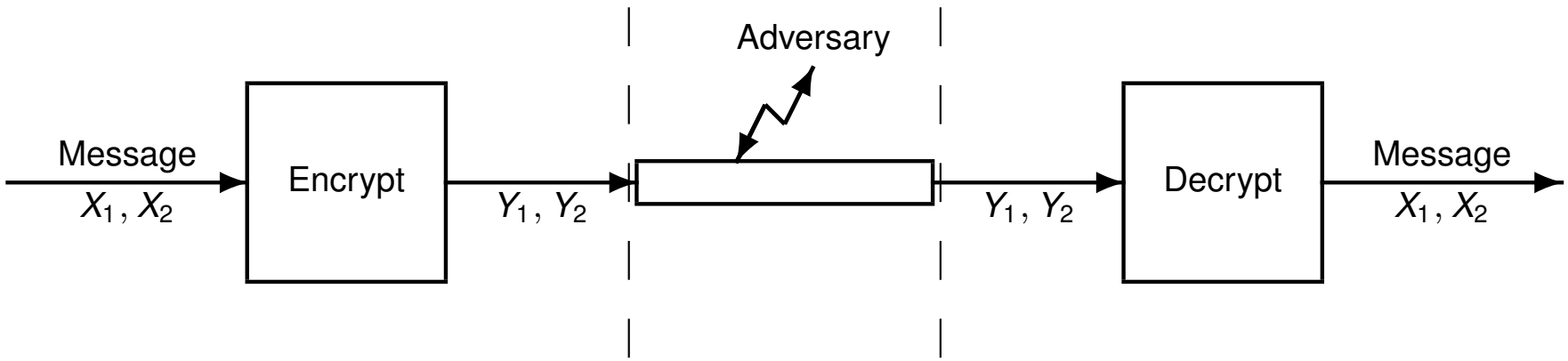
$\oplus$	(K)	00111
$=$	(X)	10010

Q

When is this insecure?

Using the Same Key Twice

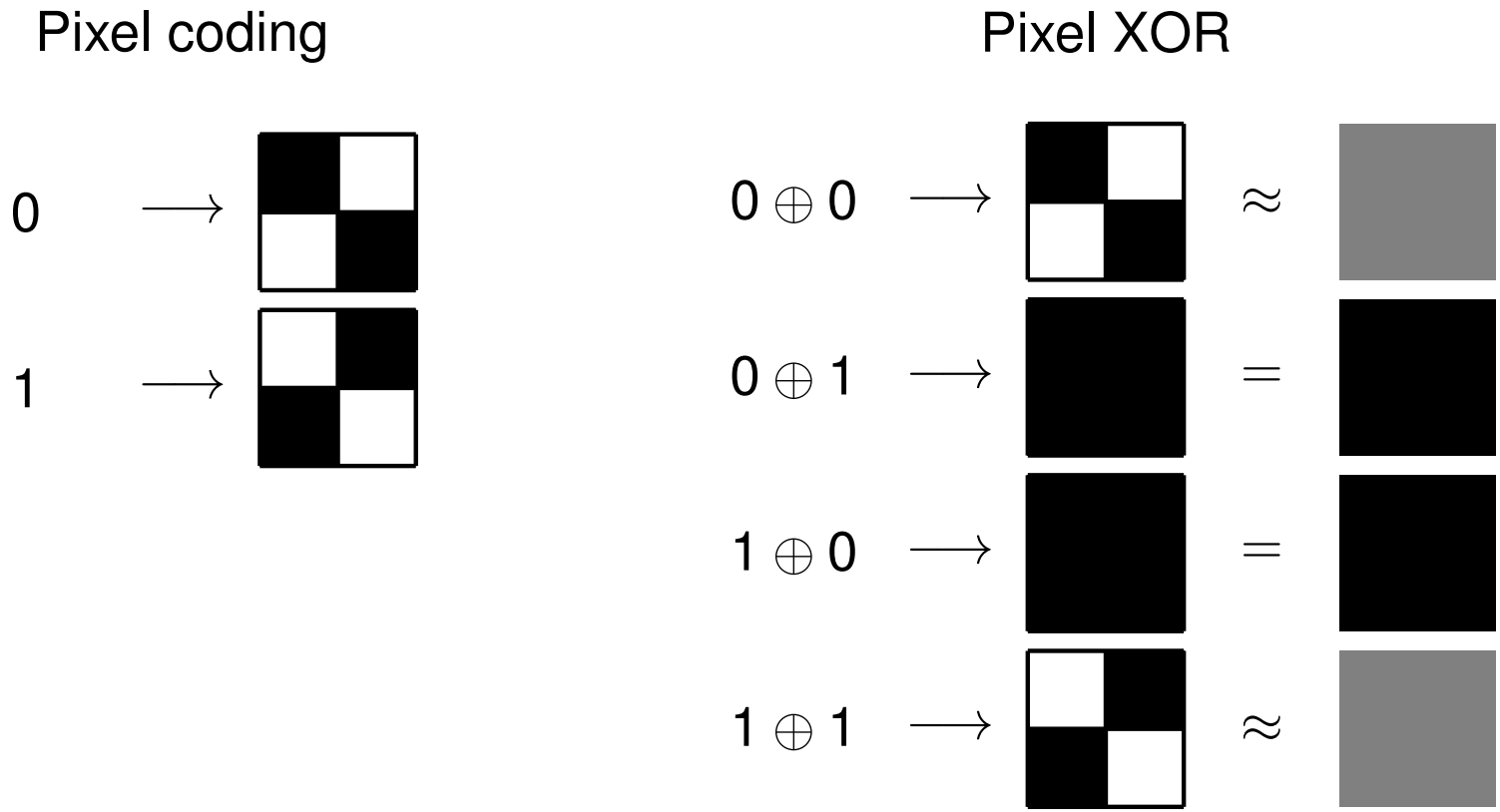
$$\begin{aligned} Y_1 &= X_1 \oplus K \\ Y_2 &= X_2 \oplus K \end{aligned}$$



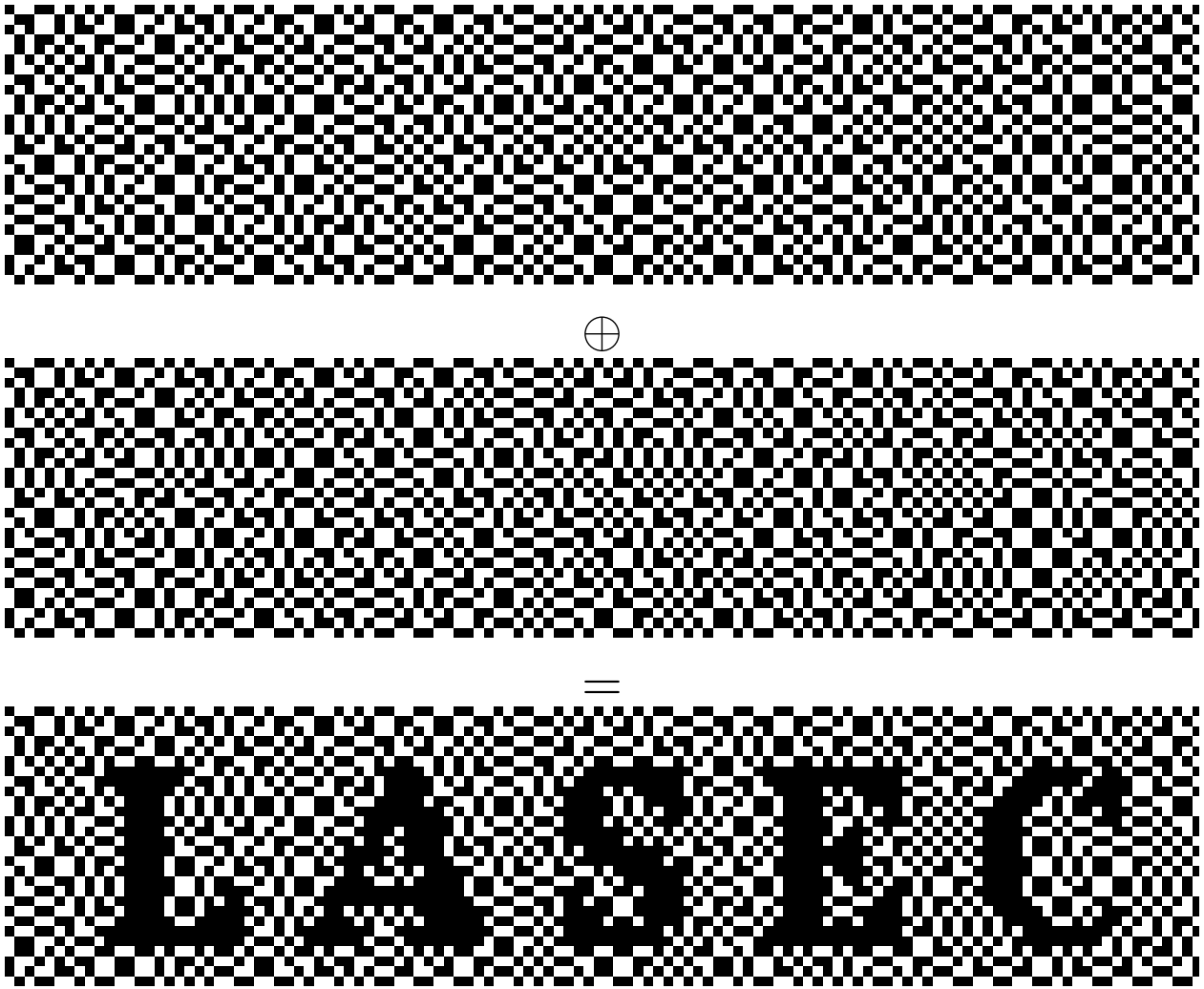
$$Y_1 \oplus Y_2 = (X_1 \oplus K) \oplus (X_2 \oplus K) = (X_1 \oplus X_2) \oplus (K \oplus K) = X_1 \oplus X_2$$

leakage of the $X_1 \oplus X_2$ value

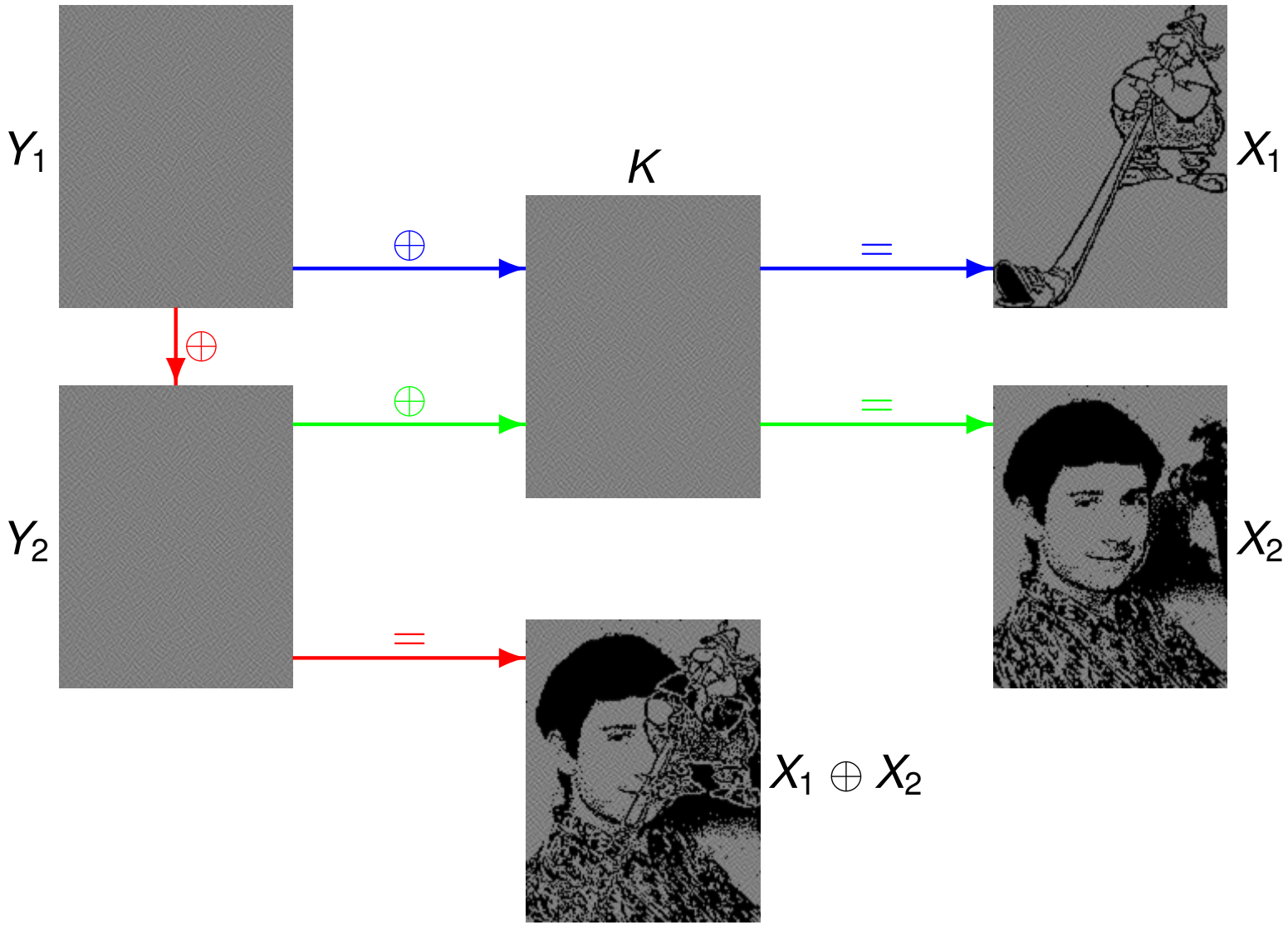
Visual Cryptography



Example



Using the Same Key Twice



Insecurity Cases in the Vernam Cipher

- if K is smaller than X

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

$\rightarrow$ insecure

- if K is not uniformly distributed

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

$\rightarrow$ insecure

- if K is used twice

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

$\rightarrow$ 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 $a + a + \cdots + a$ (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 $\oplus$
- $\{0, 1, \ldots, n-1\}$ 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.

(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)
- $H(X|Y) = H(X, Y) - H(Y)$

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

Reminder on the Shannon Entropy — ii

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

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

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

- 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 ≥ 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, $\Pr[Y = f(X)] = 1$

□

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

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

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

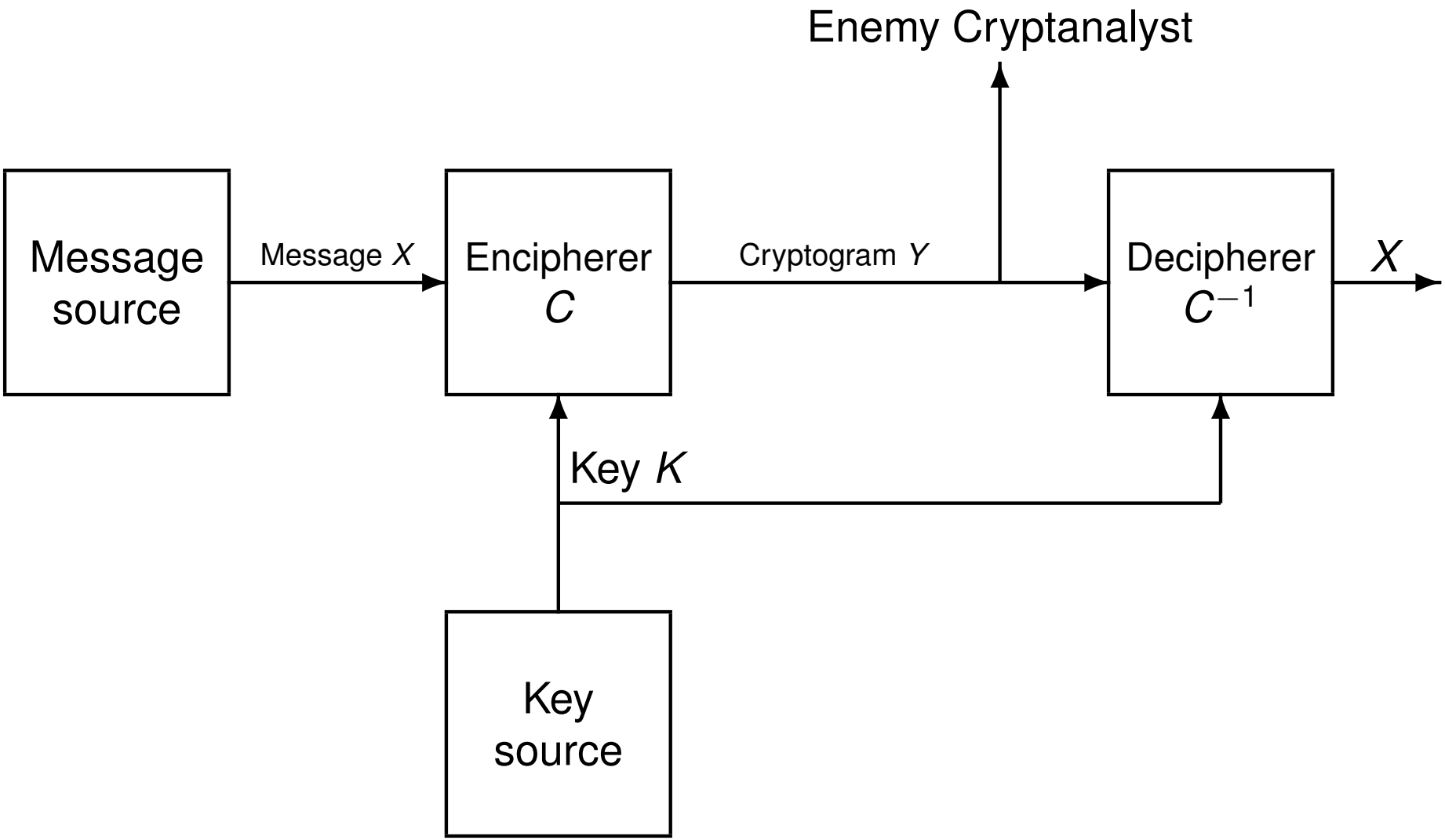
  with equality iff all t_x's for $\Pr[X = x] \neq 0$ are equal
- Applying this to $t_x = 1/\Pr[X = x]$ yields

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

  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:

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

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

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

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

Perfect Secrecy - iii (skip)

Proposition

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

Proof.

Perfect secrecy is equivalent to statistic independence of X and Y.
Statistic 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 (independent of K) such that $\text{support}(p') \subseteq \text{support}(p)$.
C_K has perfect secrecy with $p \Longrightarrow$ C_K has perfect secrecy with p'.

Proof. If $p'(x) \neq 0$ then $p(x) \neq 0$ and

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

$$\uparrow \quad p'(x) \neq 0 \qquad \uparrow \quad p(x) \neq 0 \qquad \uparrow \quad \text{perfect secrecy}$$

then

$$\Pr_{p'}[Y = y] = \sum_{x \in \text{support}(p')} \Pr_{p'}[Y = y|X = x]p'(x)$$

$$= \sum_{x \in \text{support}(p')} \Pr_{p}[Y = y]p'(x) = \Pr_{p}[Y = y]$$

□

Shannon Theorem

Theorem (Shannon 1949)

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

Proof. (skip)

- 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 (Bad News)

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 for all x such that $\Pr[X = x] \neq 0$, $\Pr[X = x, Y = y] = \Pr[X = x] \Pr[Y = y] \neq 0$
- 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's write it $k = f(x)$.
- for any x in the support of X we have $C^{-1}_{f(x)}(y) = x$. 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

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 zvffrf 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

Milestones of Modern Cryptography

- **Vigenère** (XVIth Century): secret key
- **Kerckhoffs** (1883): algorithm known by the adversary
- **Shannon** (1949): an info-theoretical approach of cryptography
- **Diffie-Hellman** (1976): public-key cryptography
- **DES** (1977): encryption standard for non-military applications

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 ex4
- ciphertext length: midterm exam 2022–23 ex1
- Enigma and perfect secrecy: midterm exam 2023–24 ex1

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

2 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
- Orders in a Group
- 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.

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots$$

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* $\alpha_1, \ldots, \alpha_r$ *are positive integers.*

Modulo n

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

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

$$8273 \bmod 143 = 122$$

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

Euclidean Division

Theorem (Euclidean Division)

For any $x \in \mathbf{Z}$ and any $n > 0$ there exists a unique pair $(q, r) \in \mathbf{Z}^2$ such that $x = qn + r$ and $0 \leq r < n$.
We denote $r = x \bmod n$ and have $q = \left\lfloor \frac{x}{n} \right\rfloor$.

Two Notations for “mod”

- **without parentheses**: $x \bmod n$
  $\rightarrow$ a two-input operator
  $=$ remainder in the Euclidean division of x by n
- **with parentheses**: $a \equiv b \pmod{n}$
  $\rightarrow$ 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

$$a = b \bmod n \quad \text{and} \quad a \equiv b \pmod{n}$$

a set to $(b \bmod n)$ — a and b are (equal modulo n)

$\mathbf{Z}_n$ for Dummies ($n > 1$)

- $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$
- 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:

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

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

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

- 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:

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

2 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
- Orders in a Group
- 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 $\varphi : \mathbf{Z} \longrightarrow G$ 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
$\forall a \in G_1 \quad f(a)$ neutral in $G_2 \Longrightarrow a$ neutral in G_1

Group Constructions: Subgroups

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

Example:

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

Subgroups of Z

Theorem

If H is a subgroup of $\mathbf{Z}$ and $H \neq \{0\}$, 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 $(G, .)$, an element g **generates/spans** a subgroup
$$\langle g \rangle = \{\ldots, g^{-2}, g^{-1}, g^0, g^1, g^2, \ldots\}$$
- If $\langle g \rangle$ is finite, of cardinality n, then $g^n = 1$ and
$$\langle g \rangle = \{g^0, g^1, \ldots, g^{n-1}\}$$
(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 $\langle g \rangle$
  - $x \mapsto \log_g x$ is the inverse isomorphism from $\langle g \rangle$ to $\mathbf{Z}_n$

Finite Groups and Orders

Definition

If $(G, .)$ 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 $i \mapsto g^i$...)
- it must be of form $n\mathbf{Z}$ where n is the smallest among all $i > 0$
- $\{1, g, g^2, \ldots, g^{n-1}\}$ is a non-repeating exhaustive list of all $\langle g \rangle$ elements

□

Consequence

if g is of order n...

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

Group Constructions: Groups Product

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

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

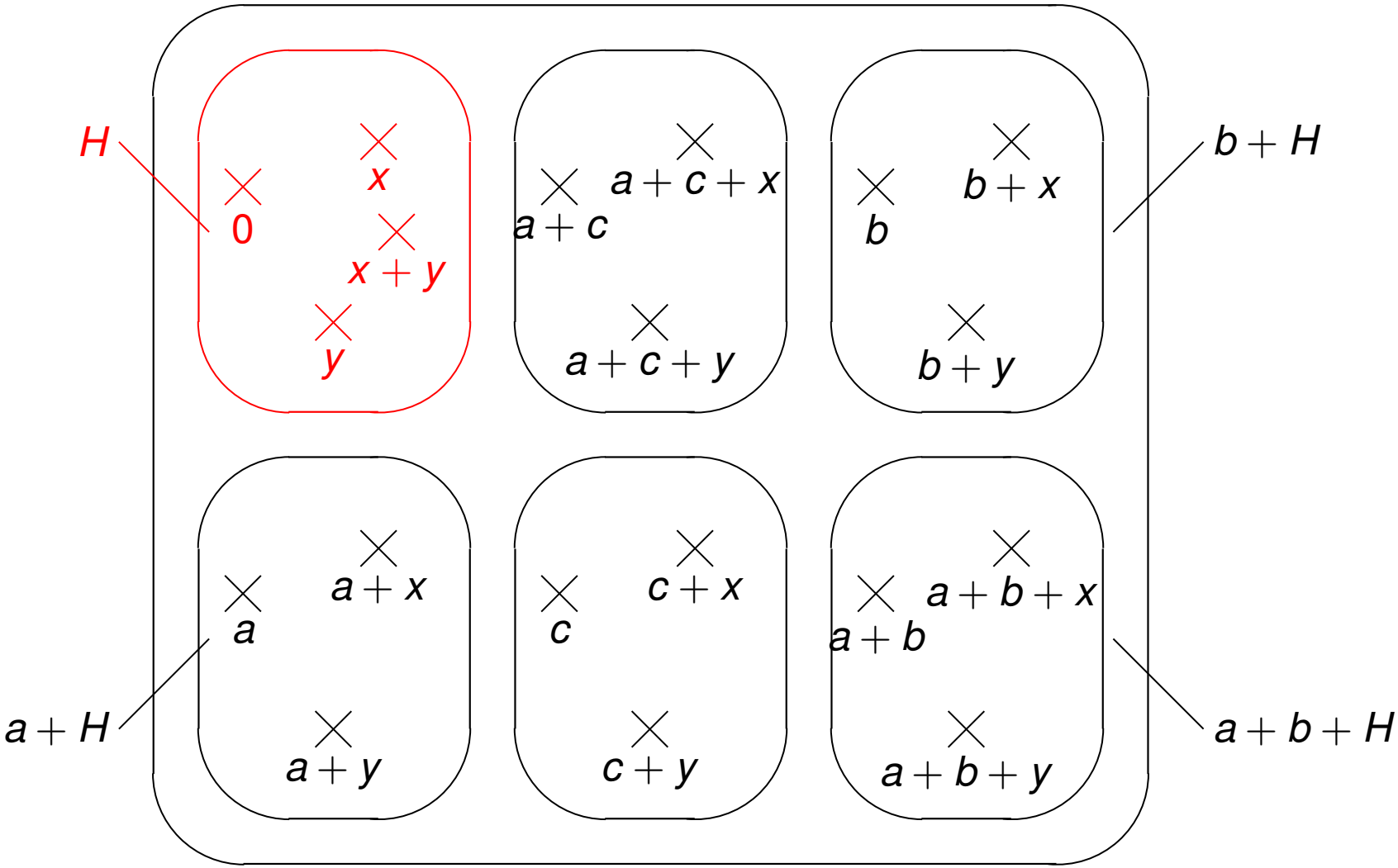
	functional notations	family notations
	function domain D	index set I
	function range R	set S
finite domain	$f : \{1, \ldots, n\} \to R$	$(x_1, \ldots, x_n)$
infinite domain	$f : D \to R$	$(x_i)_{i \in I}$
input	$x \in D$	$i \in I$
image	$f(x) \in R$	$x_i \in S$
set	R^D	S^I or S^n

Group Constructions: Quotient Group

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 $+$

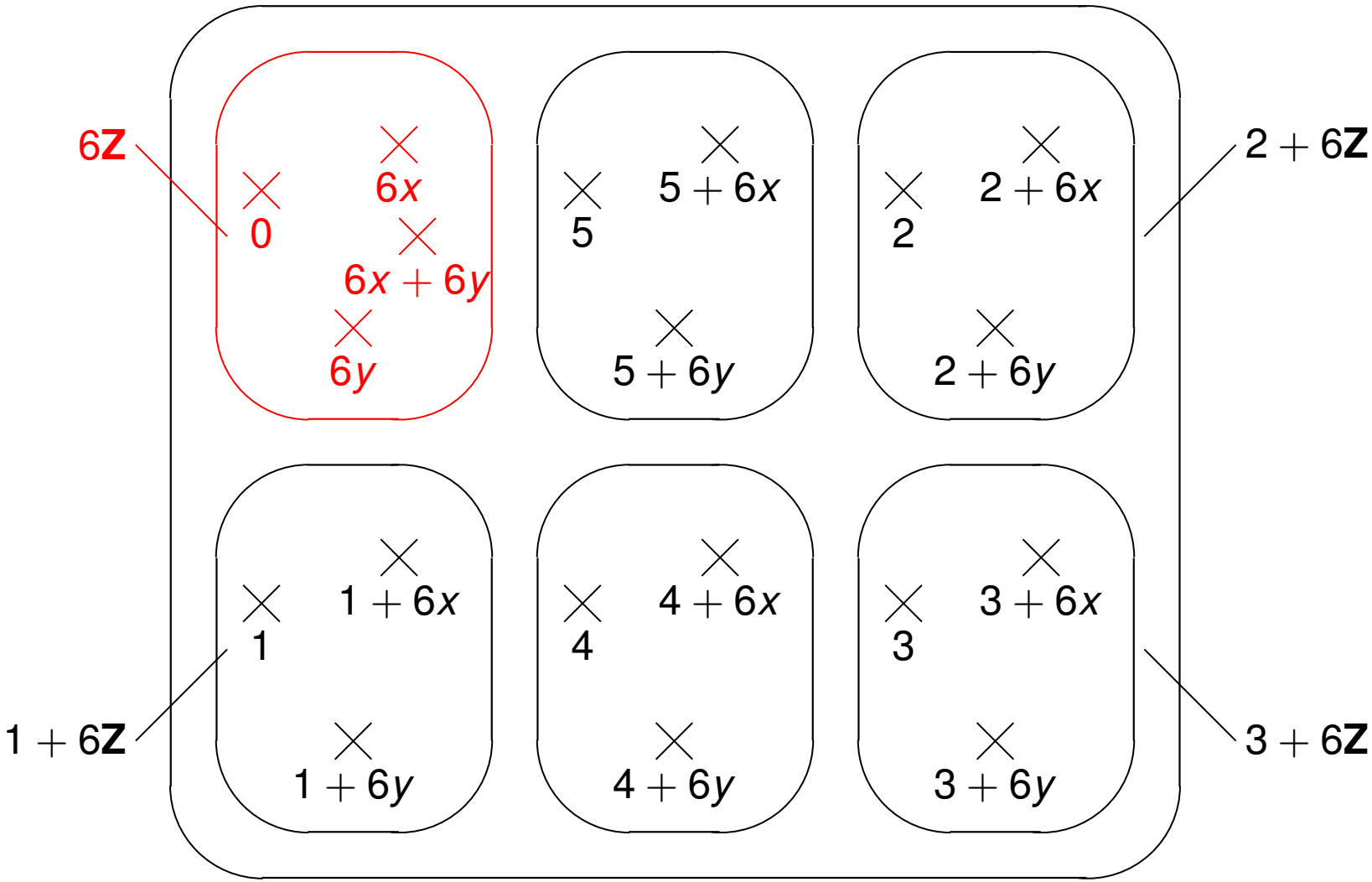
- a and b in G are said to be *congruent modulo* H if $b - a \in H$
  notation: $a \equiv b \pmod{H}$
- the relation "...is congruent to ... modulo H" is an equivalence relation (reflexive, symmetric, transitive)
- notation: for $a \in G$, $a + H$ is the set of all G elements which can be written $a + h$ 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: $\mathbb{Z}/6\mathbb{Z}$



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

Lagrange Theorem

Theorem (Lagrange)

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

Proof.
in $G/\langle g \rangle$ (set quotient), 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) □

Consequence

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

Application: Generators in a Group of Prime Order

Theorem

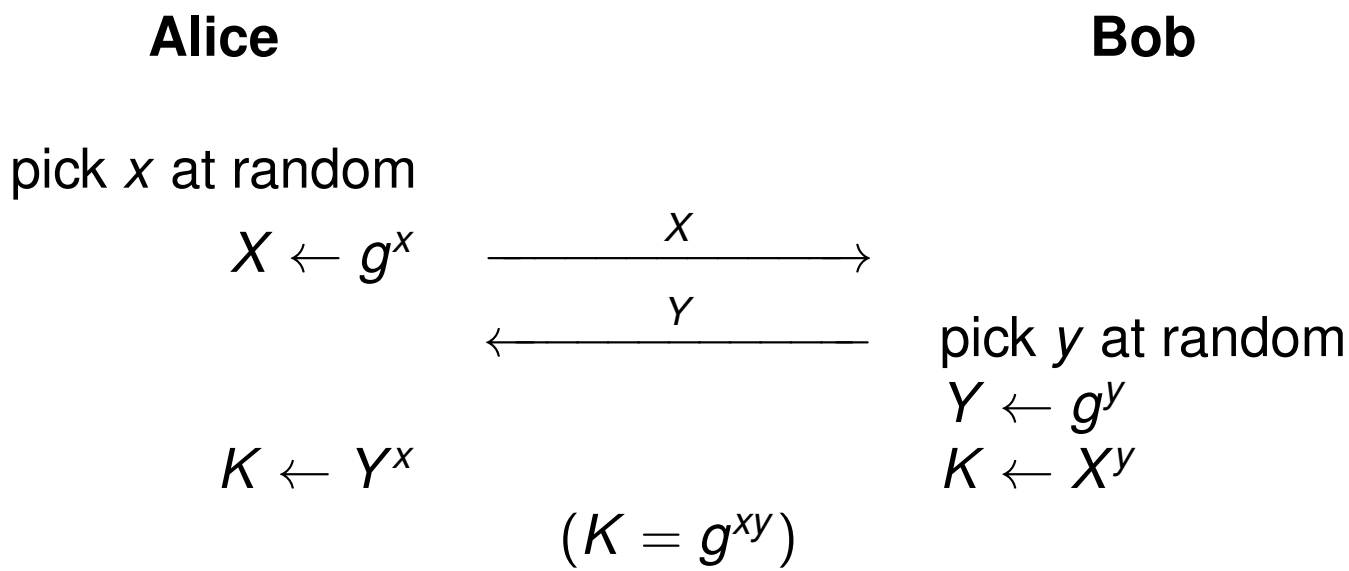
if $(G, .)$ *has prime order, all elements (except 1) are generators*

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
- $\langle x \rangle$ has cardinality p
- since $\langle x \rangle \subseteq G$ and $\#G = p$, we have $\langle x \rangle = G$ □

The Diffie-Hellman Key Agreement Protocol

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



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
  $\rightarrow$ public-key cryptography

Example of Diffie-Hellman groups:

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

2 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
- Orders in a Group
- The $\mathbf{Z}_p$ Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

Addition with Big Numbers (in Decimal)

$$\begin{array}{cr} & {\scriptstyle 1\ 1\quad\ \ 1\quad\ } \\ & 8\,427\,403 \\ + & 12\,951\,842 \\ \hline = & 21\,379\,245 \end{array}$$

Input: two integers a and b of ℓ digits
Output: one integer $c = a + b$

1: $r \leftarrow 0$
2: **for** $i = 0$ to $\ell - 1$ **do**
3: $\quad d \leftarrow a_i + b_i + r$
4: $\quad$ write $d = 10r + c_i$ with $c_i < 10$
5: **end for**
6: $c_\ell \leftarrow r$

Addition in Binary

$1 + 1 = 10$

$$\begin{array}{rr} & {\scriptstyle 1\ 1} \\ & 1\ 001\ 001 \\ + & 10\ 011\ 000 \\ \hline = & 11\ 100\ 001 \end{array}$$

Input: a and b, two integers of at most ℓ 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: $\quad d \leftarrow a_i + b_i + r$
4: $\quad$ 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)

		1	0	1	0	1	0	0	0x54	(84)
+	1	0	0	1	0	0	1	0	0x92	(146)
=	1	1	1	0	0	1	1	0	0xe6	(230)

hexadecimal = compact way to represent bistrings
(bits groupped 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):

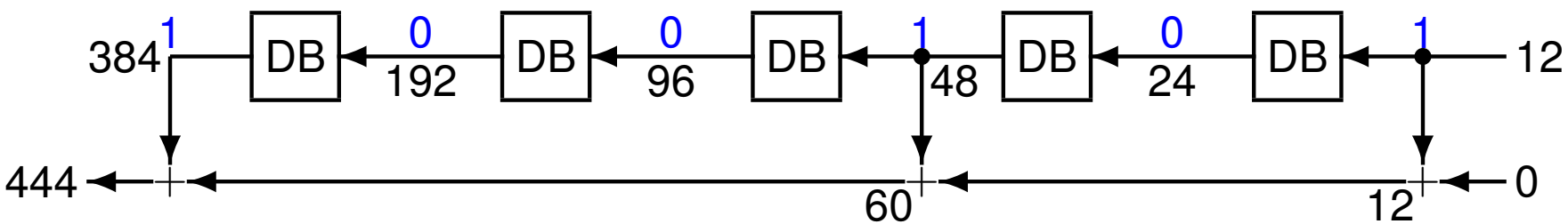
$$\begin{aligned} & 12 \times 100101 \\ = \quad & 12 \times \left(1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1\right) \\ = \quad & 12 \times \left(2^5 + 2^2 + 1\right) \\ = \quad & 12 \times 2^5 + 12 \times 2^2 + 12 \times 1 \end{aligned}$$

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)

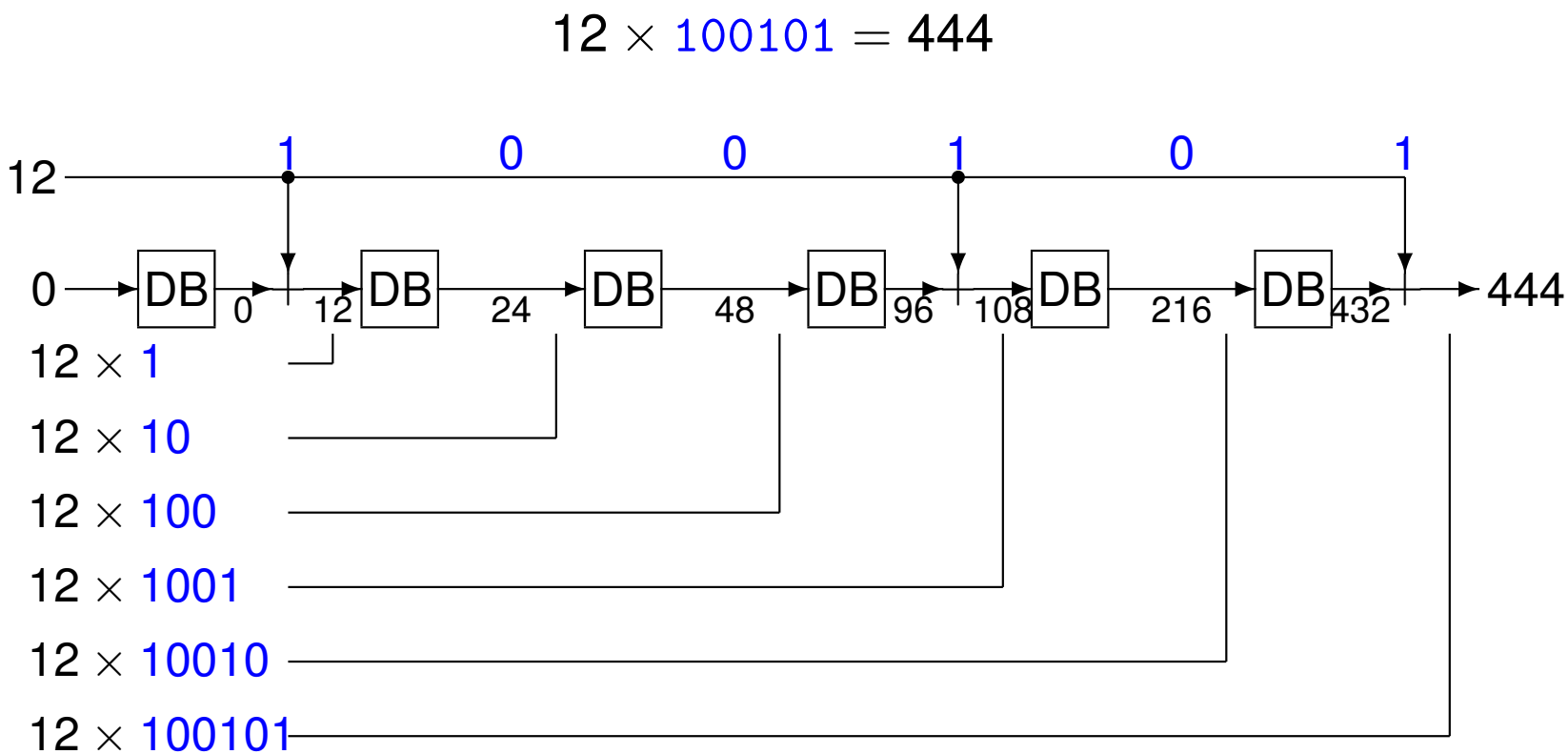


Double-and-Add From Right to Left

Input: a in monoid, n integer of at most ℓ 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



Double-and-Add From Left to Right

Input: a in monoid, n integer of at most ℓ 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: $\quad x \leftarrow x + x$
4: $\quad$ **if** $n_i = 1$ **then**
5: $\quad\quad x \leftarrow x + a$
6: $\quad$ **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 $(\mathrm{EC}, +)$: 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 $\mathcal{O}(T \log n)$

2 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**
- Orders in a Group
- 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: ~~$R^* = R - \{0\}$~~

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: Z

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

1. **Z** is closed for the addition
2. the addition is associative in **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 **Z**
6. **Z** is closed for the multiplication
7. the multiplication is associative in **Z**
8. 1 is neutral for the multiplication
9. addition is distributive for multiplication
10. the multiplication is commutative in **Z**

Z is a commutative ring of infinite size

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

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

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
$A(X)(B(X) + C(X)) = A(X)B(X) + A(X)C(X)$ 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

$$
\begin{aligned}
& 296\,527 \bmod 9 \\
= \;& (200\,000 + 90\,000 + 6\,000 + 500 + 20 + 7) \bmod 9 \\
= \;& (2 \times 100\,000 + 9 \times 10\,000 + 6 \times 1\,000 + 5 \times 100 + 2 \times 10 + 7) \bmod 9 \\
= \;& (2 \times 10^5 + 9 \times 10^4 + 6 \times 10^3 + 5 \times 10^2 + 2 \times 10 + 7) \bmod 9 \\
= \;& (2 \times (10 \bmod 9)^5 + 9 \times (10 \bmod 9)^4 + 6 \times (10 \bmod 9)^3 + \\
& + 5 \times (10 \bmod 9)^2 + 2 \times (10 \bmod 9) + 7) \bmod 9 \\
= \;& (2 \times 1^5 + 9 \times 1^4 + 6 \times 1^3 + 5 \times 1^2 + 2 \times 1 + 7) \bmod 9 \\
= \;& (2 + 9 + 6 + 5 + 2 + 7) \bmod 9 \\
= \;& 31 \bmod 9 \\
= \;& (3 + 1) \bmod 9 \\
= \;& 4 \bmod 9 \\
= \;& 4
\end{aligned}
$$

“Preuve par 9”

$$
\begin{array}{rrrrr}
 & & & 5 & 2 & 6 \\
 & & \times & & 2 & 8 \\
\hline
 & & 4 & 2 & 0 & 8 \\
+ & 1 & 0 & 5 & 2 & \\
\hline
 & 1 & 4 & 7 & 2 & 8
\end{array}
\qquad
\begin{array}{r}
\mathrm{mod}9 \\
4 \\
\times \quad 1 \\
\\
\\
\hline
4
\end{array}
$$

Example: the Ring of Residues Modulo n

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

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 distributive 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

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

- 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 "$\bmod \underline{\underline{n}}$" 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 "$\bmod \underline{\underline{m}}$" 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: $\langle 2 \rangle = \{1, 2, 4, 8\}$

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 Z_n

Input: an integer n of ℓ 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 Z_n From Left to Right

Input: an integer n of ℓ 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: $\quad x \leftarrow x + x \bmod n$
4: $\quad$ **if** $b_i = 1$ **then**
5: $\quad\quad x \leftarrow x + a \bmod n$
6: $\quad$ **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 ℓ bits, an integer e (e in binary)
Output: $x = a^e \bmod n$
Complexity: $\mathcal{O}(\ell^2 \log e)$

1: $x \leftarrow 1$
2: **for** $i = \lfloor \log_2 e \rfloor$ to 0 **do**
3: $\quad x \leftarrow x \times x \bmod n$
4: $\quad$ **if** $e_i = 1$ **then**
5: $\quad\quad x \leftarrow x \times a \bmod n$
6: $\quad$ **end if**
7: **end for**

Multiplication Easy $\Longrightarrow$ Exponentiation Easy

Theorem

We can compute g^x with $2n$ multiplications or less, for $x < 2^{n+1}$.

example: g^{54} in 8 multiplications

	54	squaring	multiplying
1		g	
2	$= 2$	$g \times g = g^2$	g^2
4	$+4$	$(g^2) \times (g^2) = g^4$	$\times g^4$
8		$(g^4) \times (g^4) = g^8$	
16	$+16$	$(g^8) \times (g^8) = g^{16}$	$\times g^{16}$
32	$+32$	$(g^{16}) \times (g^{16}) = g^{32}$	$\times g^{32}$
			$= g^{54}$

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 $0 \leq r < n$
  $q = \left\lfloor \frac{a}{n} \right\rfloor$ and $r = a \bmod n$
- algorithm runs in $\mathcal{O}(\ell^2)$
  (ℓ is the size of a, $n < a$)

Modular Inversion

Theorem

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

Proof.

$\Longrightarrow$ 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 ℓ bits
Output: $d = \gcd(a, b)$
Complexity: $\mathcal{O}(\ell^2)$

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	0
	↙	||	
1	35	22	1
	↙	||	
2	22	13	1
	↙	||	
3	13	9	1
	↙	||	
4	9	4	2
	↙	||	
5	4	1	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): runing time (complexity) is quadratic

Extended Euclid Algorithm

Input: a and b, two integers of at most ℓ 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	$(22, 1, 0)$	$-$	$(35, 0, 1)$	$\times 0$
		$\swarrow$	$\|\|$	
1	$(35, 0, 1)$	$-$	$(22, 1, 0)$	$\times 1$
		$\swarrow$	$\|\|$	
2	$(22, 1, 0)$	$-$	$(13, -1, 1)$	$\times 1$
		$\swarrow$	$\|\|$	
3	$(13, -1, 1)$	$-$	$(9, 2, -1)$	$\times 1$
		$\swarrow$	$\|\|$	
4	$(9, 2, -1)$	$-$	$(4, -3, 2)$	$\times 2$
		$\swarrow$	$\|\|$	
5	$(4, -3, 2)$	$-$	$(1, 8, -5)$	$\times 4$
		$\swarrow$	$\|\|$	
6	$(1, 8, -5)$		$(0, -35, 22)$	

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 $ux \bmod n = 1$
   (Note: we may need to reduce u modulo n)

Modular Inversion

Theorem

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

Proof:

- $\Rightarrow$: already seen (slide 169)
- $\Leftarrow$: 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 (Recap)

- 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 (Recap)

- 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

2 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
- Orders in a Group
- The $\mathbf{Z}_p$ Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

Structure Property of Z (Reminder)

(already seen, see slide 127)

Theorem

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

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

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

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

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

$$\varphi(m) \quad = \quad (p_1 - 1)p_1^{\alpha_1 - 1} \times \cdots \times (p_r - 1)p_r^{\alpha_r - 1}$$

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

Checking a Generator of a Group with Known Order Factorization

Input: an element g in an Abelian cyclic group of order with known factorization $n = p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$
Output: say if g is a generator
Complexity: $\mathcal{O}(r)$ exponentials

1: **for** $i = 1$ to r **do**
2: $\quad y \leftarrow g^{n/p_i}$
3: $\quad$ **if** $y = 1$ **then**
4: $\quad\quad$ abort: g is not a generator
5: $\quad$ **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 because (to be seen in the next chapter)

$$\Pr_g \left[g^{\frac{n}{p_i}} = 1 \right] = \frac{1}{p_i}$$

  which is small for p_i large
- if n is hard to factor, we can still find generators: find the prime factors up to some bound B
- application: generate a generator of $\mathbf{Z}_p^*$ for a prime p (we will see that it is cyclic)

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

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

Generating a Generator

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 $i = 1, \ldots, r$
- $\Pr_{g \in_U G}\left[g^{\frac{n}{p_i}} = 1\right] = \frac{1}{p_i}$ and these events are independent (to be seen in next chapter with CRT)
- 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 $i = 1, \ldots, s$

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

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

2 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
- Orders in a Group
- 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:

- $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$
- $\mathbf{Z}_p$ for p prime (next slide)
- GF(2^n) (in Chapter 5)

Z_p Properties

Theorem (Z_p structure)

Let p be a prime number.

1. $\mathbf{Z}_p^* = \{1, \ldots, p-1\}$
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*

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

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)

□

An Interesting Group

the subgroup

$$\langle g \rangle \subseteq \mathbf{Z}_p^*$$

of prime order q

$$\langle g \rangle = \{1, g, \ldots, g^{q-1}\} \leftrightarrow \{0, 1, \ldots, q-1\} = \mathbf{Z}_q$$

Example: the SSH2 Parameters

$$
\begin{aligned}
p &= 2^{1024} - 2^{960} - 1 + 2^{64} \left\lfloor 2^{894}\pi + 129093 \right\rfloor \\
g &= 2 \\
q &= \frac{p-1}{2}
\end{aligned}
$$

try it with gp:

```
allocatemem(80000000)
\p 300
p=2^1024-2^960-1+2^64*floor(2^894*Pi+129093)
g=2
q=(p-1)/2
isprime(p)
isprime(q)
Mod(g,p)^q
```

Algorithms To Be Seen Later

- we can generate large prime numbers
- we can verify the primality of a number
- 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

Definition (Discrete Logarithm (DL) Problem)

The DL problem, relative to Setup, is hard if for any PPT (probabilistic polynomial-time) algorithm $\mathcal{A}$, the probability that the following **game** returns 1 is **negl**(λ):

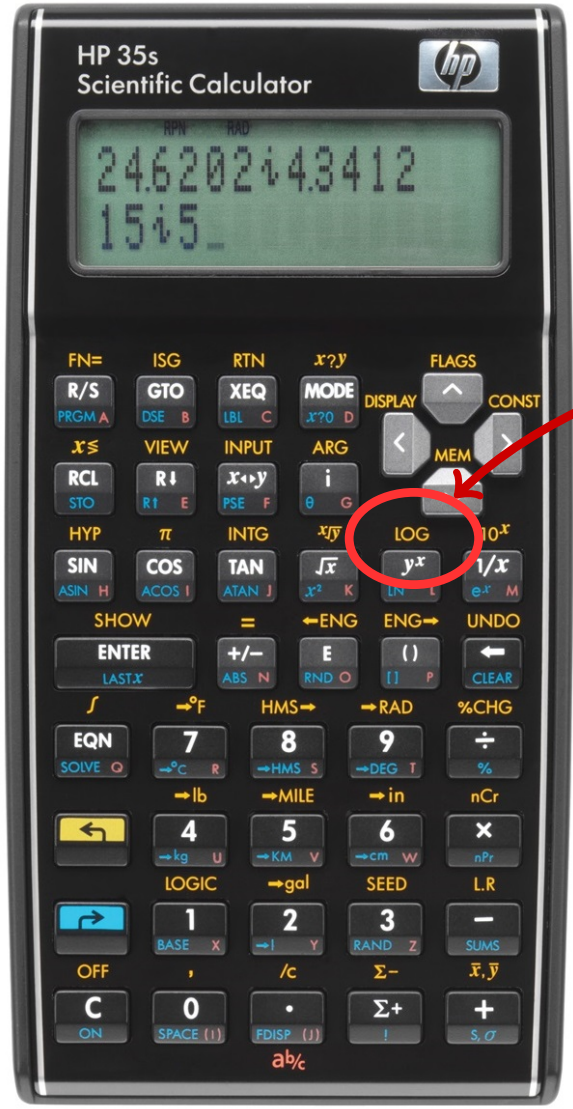
DL(λ):

1: Setup$(1^\lambda) \to (\text{group}, q, g)$
2: pick $x \in \mathbf{Z}_q$
3: $X \leftarrow g^x$
4: $\mathcal{A}(\text{group}, q, g, X) \to x'$
5: **return** $1_{X=g^{x'}}$

Examples:

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

Is Logarithme Hard?!?



We can compute log

The Notion of Game

DL(λ):
1: Setup(1^λ) $\rightarrow$ (group, q, g)
2: pick $x \in \mathbf{Z}_q$
3: $X \leftarrow g^x$
4: $\mathcal{A}$(group, q, g, X) $\rightarrow x'$
5: **return** $1_{X=g^{x'}}$

- Given a PPT adversary $\mathcal{A}$:

  Game(security parameter):
  1: setup of parameters
  2: initialization of the game
  3: $\mathcal{A}$(what he should know) $\rightarrow$ result
  4: **return** $1_{\text{winning condition}}$
- Advantage of $\mathcal{A}$:

$$\text{Adv}(\text{security parameter}) = \Pr[\text{Game} \rightarrow 1]$$

- Security:

$$\forall \text{PPT } \mathcal{A} \quad \text{Adv} = \text{negl}$$

Polynomial Function

$$f(\lambda) = \text{poly}(\lambda)$$
$$\Updownarrow$$
$$\exists n \qquad f(\lambda) = \mathcal{O}(\lambda^n)$$

as $\lambda \to +\infty$

we assume that Setup(1^λ) generates in time-complexity $\mathcal{O}(\text{poly}(\lambda))$ some parameters allowing to make group operations in time-complexity $\mathcal{O}(\text{poly}(\lambda))$

$$1^\lambda = \underbrace{11\cdots1}_{\lambda \text{ times}}$$

Negligible Function

$$f(\lambda) = \mathsf{negl}(\lambda)$$
$$\Updownarrow$$
$$\forall n \qquad f(\lambda) = \mathcal{O}(\lambda^{-n})$$

as $\lambda \to +\infty$

Example:

- $f(\lambda) = c^{-\lambda}$ is negligible (for $c > 1$)
- $f(\lambda) = \lambda^{-1\,000\,000\,000}$ is *not* negligible

Some Facts About The Discrete Logarithm Problem

in a cyclic group of order q:

- easy on a quantum computer:
  - $\rightarrow$ Shor algorithm
- easy if q has only small prime factors (e.g. $< 2^{100}$):
  - $\rightarrow$ Pohlig-Hellman algorithm
- best algorithm for a subgroup of $\mathbf{Z}_p^*$ with p and q prime:
  - $\rightarrow$ General Number Field Sieve (GNFS) with complexity

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

this is mostly precomputation (without X)

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

- generic algorithms in $\mathcal{O}(\sqrt{q})$:
  - $\rightarrow$ baby-step giant-step algorithm
  - $\rightarrow$ Pollard ρ algorithm

Baby Step - Giant Step Algorithm

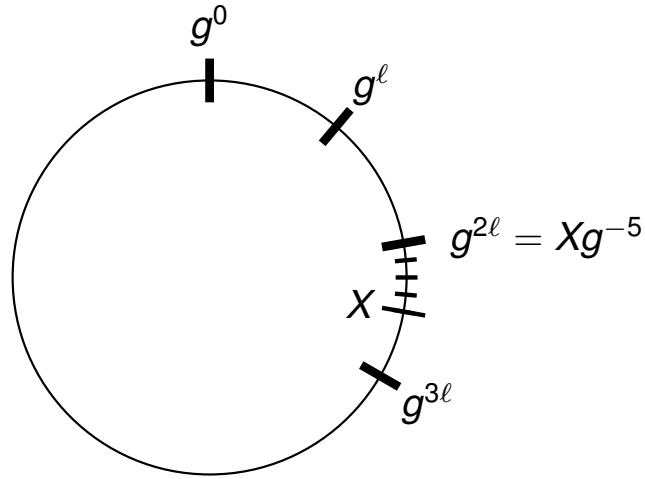
Input: g and X in a group G, B an upper bound for $\#G$
Output: the logarithm of X in base g
Complexity: $\mathcal{O}(\sqrt{B})$ group operations

Precomputation

1: let $\ell = \lceil \sqrt{B} \rceil$ be the size of a "giant step"
2: **for** $i = 0, \ldots, \ell - 1$ **do**
3: set $T\{g^{i\ell}\} \leftarrow i$
4: **end for**

Algorithm

5: **for** $j = 0, \ldots, \ell - 1$ **do**
6: compute $z = Xg^{-j}$
7: **if** $T\{z\}$ exists **then**
8: $i \leftarrow T\{z\}$
9: yield $x = i\ell + j$ and stop ▷ we get $Xg^{-j} = g^{i\ell}$
10: **end if**
11: **end for**



Attacks based on Precomputation

Adrian++; Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice; CCS 2015

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

p **length** (bits)	**precomputation** (core-time)	**attack** (core-time)
512	10.2 years	10 minutes
768	36 500 years	2 days
1024	45 000 000 years	30 days

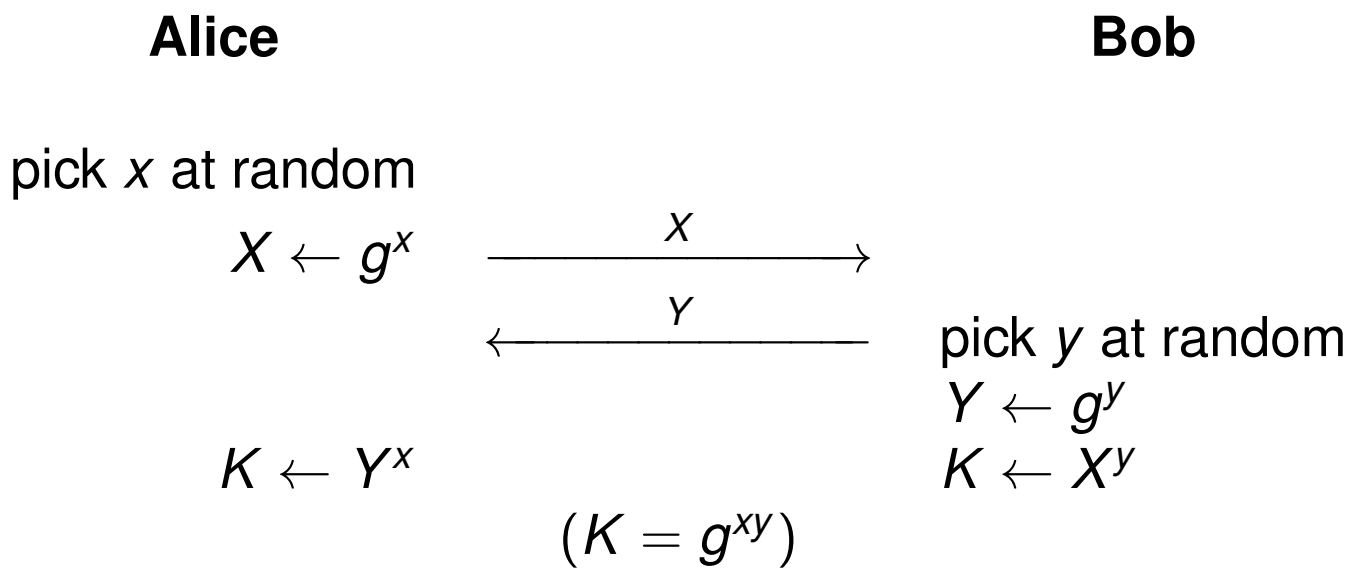
example: SSH2 uses a fixed p of 1024 bits...

2 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
- Orders in a Group
- The $\mathbf{Z}_p$ Field
- The Diffie-Hellman Key Exchange, Concretely
- The ElGamal Public-Key Cryptosystem

The Diffie-Hellman Key Agreement Protocol (again)

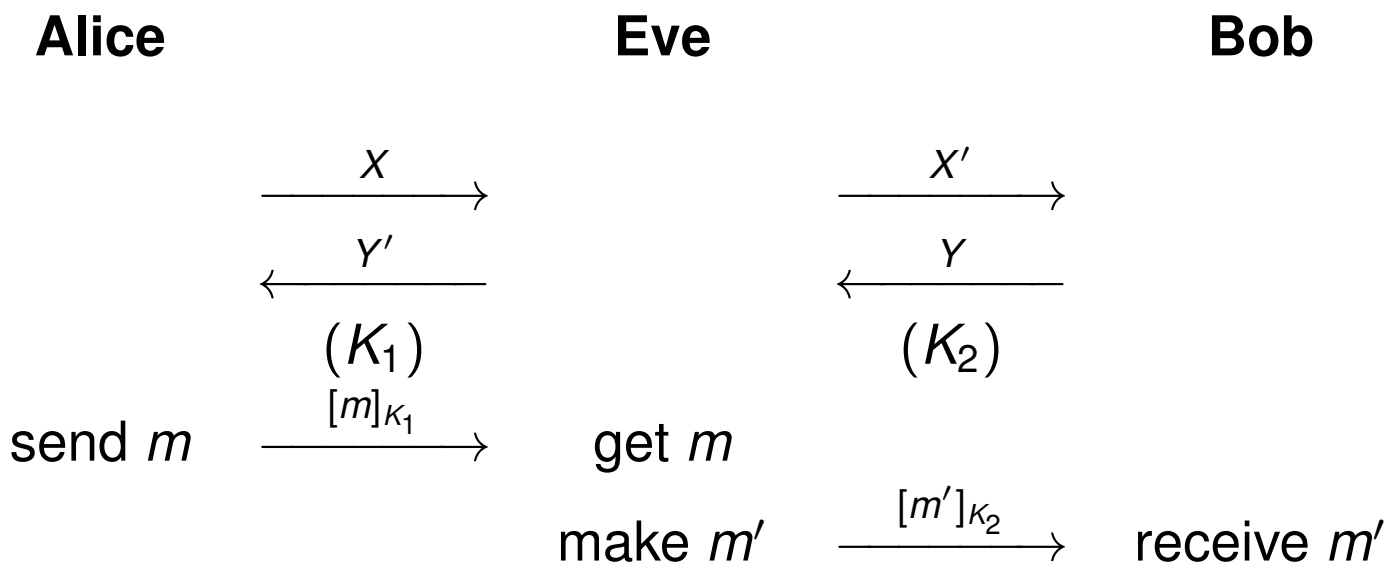
Assume a group generated by some g



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

An Unavoidable Active Attack

Man-in-the-Middle Attack



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

Definition (Computational Diffie-Hellman (CDH) Problem)

The CDH problem relative to Setup is hard if for any PPT algorithm $\mathcal{A}$, the probability that the following game returns 1 is negl(λ):

CDH(λ):

1: Setup(1^λ) $\rightarrow$ (group, q, g)
2: pick $x, y \in \mathbf{Z}_q$
3: $X \leftarrow g^x$, $Y \leftarrow g^y$
4: $\mathcal{A}(\text{group}, q, g, X, Y) \rightarrow K$
5: **return** $1_{K=g^{xy}}$

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

CDH hard $\Longrightarrow$ DL hard

The CDH Problem Reduces to the DL Problem

$\mathcal{B}(\text{group}, q, g, X, Y)$:
1: run $\mathcal{A}(\text{group}, q, g, X) \to x'$
2: compute $K = Y^{x'}$
3: **return** K

- assume CDH is hard
- to prove DL hardness, consider a DL algorithm $\mathcal{A}$
- construct $\mathcal{B}$ s.t. $\mathcal{A}$ wins DL $\Longrightarrow$ $\mathcal{B}$ wins CDH:

$$\Pr\left[\left[\begin{array}{l}\text{DL}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x\\ X \leftarrow g^x\\ \mathcal{A}(\ldots, X) \to x'\\ \textbf{return } 1_{X=g^{x'}}\end{array}\right] \to 1\right] \leq \Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{A}(\ldots, X) \to x'\\ K \leftarrow Y^{x'}\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right] = \text{negl}$$

$$X = g^{x'} \Longrightarrow K = Y^{x'} = g^{yx'} = X^y = g^{xy}$$

□

(More details on next slide.)

CDH hard $\Longrightarrow$ DL hard (details)

$\mathcal{B}(\text{group}, q, g, X, Y)$:
1: $\mathcal{A}(\text{group}, q, g, X) \to x'$
2: $K \leftarrow Y^{x'}$
3: **return** K

$$\Pr\left[\left[\begin{array}{l}\text{DL}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x\\ X \leftarrow g^x\\ \mathcal{A}(\ldots, X) \to x'\\ \textbf{return } 1_{X=g^{x'}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{A}(\ldots, X) \to x'\\ K \leftarrow Y^{x'}\\ \textbf{return } 1_{X=g^{x'}}\end{array}\right] \to 1\right]$$

$$\leq \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{A}(\ldots, X) \to x'\\ K \leftarrow Y^{x'}\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{B}(\ldots, X, Y) \to K\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right]$$

$$X = g^{x'} \Longrightarrow Y^{x'} = g^{yx'} = X^y = g^{xy}$$

DDH Problem

Definition (Decisional Diffie-Hellman (DDH) Problem)

The DDH problem relative to Setup is hard if for any PPT algorithm $\mathcal{A}$, we have

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{DDH}(\lambda, 1) \to 1] - \Pr[\mathsf{DDH}(\lambda, 0) \to 1] = \mathsf{negl}(\lambda)$$

$\mathsf{DDH}(\lambda, b)$:

1: $\mathsf{Setup}(1^\lambda) \to (\mathsf{group}, q, g)$
2: pick $x, y, z \in \mathbf{Z}_q$
3: **if** $b = 1$ **then** $z \leftarrow xy$
4: $X \leftarrow g^x$, $Y \leftarrow g^y$, $Z \leftarrow g^z$
5: $\mathcal{A}(\mathsf{group}, q, g, X, Y, Z) \to t$
6: **return** t

DDH hard $\Longrightarrow$ CDH hard — i

- assume DDH is hard
- consider

  $\mathcal{C}(\text{group}, q, g, X, Y, Z)$:

  1: pick $x' \in \mathbf{Z}_q$

  2: **return** $1_{X=g^{x'}, Z=Y^{x'}}$

  $$\text{negl} = \text{Adv}_{\mathcal{C}} = \Pr[\text{DDH}_{\mathcal{C}}(1) \to 1] - \Pr[\text{DDH}_{\mathcal{C}}(0) \to 1] = \frac{1}{q} - \frac{1}{q^2} \sim \frac{1}{q}$$

  hence $\frac{1}{q} = \text{negl}(\lambda)$

DDH hard $\Longrightarrow$ CDH hard — ii

- to prove CDH hardness, consider a CDH algorithm $\mathcal{A}$
- we define a DDH algorithm as follows

  $\mathcal{B}(\text{group}, q, g, X, Y, Z)$:
  1: run $\mathcal{A}(\text{group}, g, X, Y) \to K$
  2: **return** $1_{K=Z}$
- $\text{DDH}_{\mathcal{B}}(1) \to 1$ is equivalent to $\text{CDH}_{\mathcal{A}} \to 1$

$$\Pr[\text{DDH}_{\mathcal{B}}(1) \to 1] = \Pr[\text{CDH}_{\mathcal{A}} \to 1]$$

- in $\text{DDH}_{\mathcal{B}}(0)$, Z is uniform in $\langle g \rangle$ and independent from K

$$\Pr[\text{DDH}_{\mathcal{B}}(0) \to 1] = \frac{1}{q} = \text{negl}$$

- hence, $\text{Adv}_{\mathcal{B}}(\lambda) = \text{Adv}_{\mathcal{A}}(\lambda) - \text{negl}$
- we know that $\text{Adv}_{\mathcal{B}}(\lambda) = \text{negl}(\lambda)$ (since DDH is hard)
- hence, $\Pr[\mathcal{A} \text{ wins}] = \text{negl}(\lambda)$ □

DDH Easy Case of a Group whose Order has a Small Factor

G of order q such that $q' = \frac{q}{w}$ is small and $q' > 1$:
let $\mathcal{A}(\text{group}, g, X, Y, Z) = 1$ iff

$$\log_{g^w} Z^w = \left(\log_{g^w} X^w\right) \times \left(\log_{g^w} Y^w\right)$$

we have $\text{Adv}_{\mathcal{A}}(\lambda) = 1 - \frac{1}{q'}$
Indeed,

$$\Pr[\text{DDH}(\lambda, 0) \to 1] = \frac{1}{q'} \quad , \quad \Pr[\text{DDH}(\lambda, 1) \to 1] = 1$$

If $q' > 1$ then $q' \geq 2$ and $\text{Adv}_{\mathcal{A}}(\lambda) \geq \frac{1}{2}$ which is not negligible.

Hardness Depending on Groups

DL hard $\Longleftarrow$ CDH hard $\Longleftarrow$ DDH hard

easy if order is smooth

easy if order has a small factor (> 1)

Hard Cases

The DDH problem is believed to be hard relative to:

- large subgroup of prime order of $\mathbf{Z}_p^*$ (p prime)
  1: pick a random prime q of size 2λ
  (so that generic algorithms have complexity $> \lambda$)
  2: pick a random p of size $f(\lambda)$ such that $q|p-1$
  (so that GNFS has complexity $> \lambda$)
  3: start again until p is prime
  4: pick a random g in $\mathbf{Z}_p^*$ of order q
- large subgroup of prime order of a "regular" elliptic curve

("pick a random prime" and Step 4 to be seen later)

Problems when not Checking Group Membership

Lim-Lee CRYPTO 1997

assume:

- Bob uses a static key Y
- Bob's algorithm runs even though X does not belong to the group

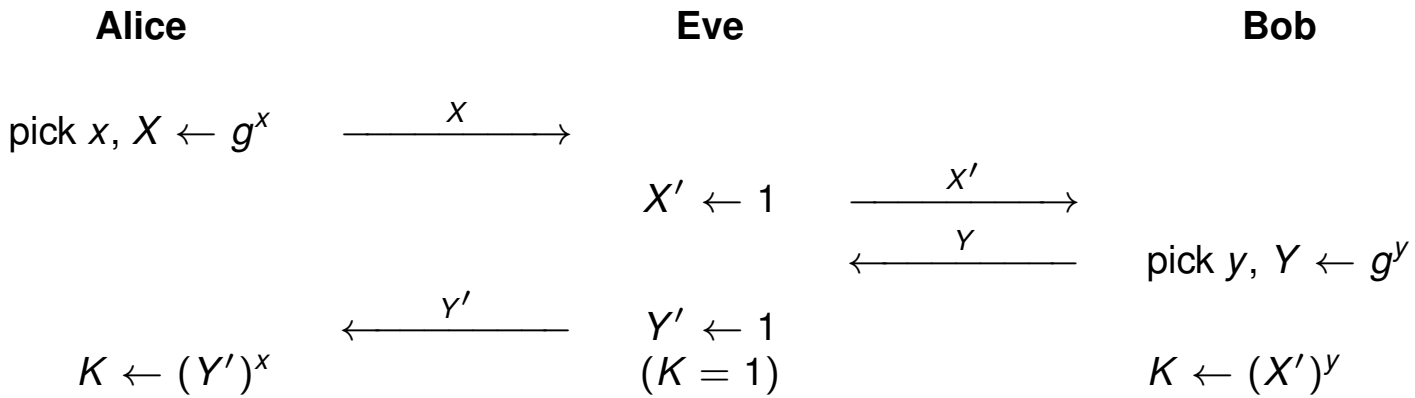
we can select X outside of the group, with a small order q'

Adversary (Bob's public key Y)		**Bob** (static y, $Y = g^y$)
pick X of small order q'	$\xrightarrow{X}$	$K = X^y$
	$\xleftarrow{\text{ct}}$	$\text{ct} = \text{Enc}_{\text{KDF}(K)}(\text{pt})$

find $y_{q'}$ such that $\text{Dec}_{\text{KDF}(X^{y_{q'}})}(\text{ct})$ makes sense
deduce $y \bmod q' = y_{q'}$
(KDF to be seen later)

Man-in-the-Middle Attack Making $K_1 = K_2$ — i

(Using any group)



after that, Eve remains passive
→ we must check $X \neq 1$ and $Y \neq 1$

Man-in-the-Middle Attack Making $K_1 = K_2$ — ii

(Using groups of order divisible by w s.t. DL is easy in $\langle g^w \rangle$)

Alice		**Eve**		**Bob**
pick x, $X \leftarrow g^x$	$\xrightarrow{X}$			
		$X' \leftarrow X^w$	$\xrightarrow{X'}$	
			$\xleftarrow{Y}$	pick y, $Y \leftarrow g^y$
	$\xleftarrow{Y'}$	$Y' \leftarrow Y^w$		
$K \leftarrow (Y')^x$		solve $X' = g^{x'w}$, $K \leftarrow Y^{x'w}$ $(K = g^{xyw})$		$K \leftarrow (X')^y$

after that, Eve remains passive

$$Y^{x'w} = g^{x'yw} = (g^{x'w})^y = X'^y = K$$

$\rightarrow$ we should use groups of prime order

Problems with Subgroups

Adrian++; Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice; CCS 2015

- we can compute $\log X$ modulo the small factors of the order

  1: set q the order of g
  2: **for** all small prime factor r of q **do**
  3: take $Z = X^{\frac{q}{r}}$ ▷ we have $Z \in \langle g^{\frac{q}{r}} \rangle$ of order r
  4: find x_r s.t. $Z = (g^{\frac{q}{r}})^{x_r}$ ▷ easy because r is small
  5: deduce $\log X \mod r = x_r$:
  ▷ if $X = g^x$ then $(g^{\frac{q}{r}})^{x_r} = Z = (g^{\frac{q}{r}})^x$ so $x \equiv x_r \pmod r$
  6: **end for**
  7: compute CRT(all x_r) (explained in next chapter)
  ▷ from $x \bmod r$ for all r, we can reconstruct x with CRT

- we can deduce $x = \log X$ when it is small
  (example: lazy servers which select a small x)
- done for 159 DH servers (out of 3.4M) on the Internet in 2015

Other issue: Weird Key Distribution

- the final key K is random in $\langle g \rangle$ which has its own representation
  E.g. $\langle g \rangle \subset \mathbf{Z}_p^*$ is a very small subset. So, the binary representation of K is far from being uniformly distributed
- we need a bitstring with a “reliable distribution”
- solution: use a Key Derivation Function (KDF)

Summary: Problems with the Original DH Protocol

- nobody is checking the group membership for X and Y
- 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 $y \in \mathbf{Z}_q^*$, $Y \leftarrow g^y$
$K \leftarrow \mathrm{KDF}(Y^x)$		$K \leftarrow \mathrm{KDF}(X^y)$
	$(K = \mathrm{KDF}(g^{xy}))$	

KDF: a Key Derivation Function

RFC 2631

Diffie-Hellman Key Agreement Method

- group parameters (p, q, g):
  p prime, q prime, q divides $p-1$ (generation algo in a next slide),
  $g = h^{\frac{p-1}{q}} \bmod p$ with h random until $1 < h < p-1$ 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: $ZZ = g^{x_A x_B} \bmod p$

Exercise

- group parameters (p, q, g):
  p prime, q prime, q divides $p - 1$,
  $g = h^{\frac{p-1}{q}} \bmod p$ with any h such that $1 < h < p - 1$ and $g > 1$

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

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: $U \leftarrow \sum_{i=0}^{m'-1} 2^{160i} \left(\text{SHA1}(\text{seed}+i) \oplus \text{SHA1}(\text{seed}+m'+i)\right) \bmod 2^m$
5: $q \leftarrow U$ OR 1 OR 2^{m-1}
6: **until** q is prime
7: $L \leftarrow$ required length for p, $L' \leftarrow \lceil \frac{L}{160} \rceil$
8: counter $\leftarrow 0$
9: **repeat**
10: $R \leftarrow \text{seed} + 2m' + (L' * \text{counter})$
11: $W \leftarrow \left(\sum_{i=0}^{L'} 2^{160i} \text{SHA1}(R+i)\right) \bmod 2^L$
12: $X \leftarrow W$ OR 2^{L-1}
13: $p \leftarrow X - (X \bmod (2q)) + 1$
14: counter $\leftarrow$ counter $+ 1$
15: **if** counter $\geq 4096 \lceil \frac{L}{1024} \rceil$ **then** abort (fail)
16: **until** $p > 2^{L-1}$ and p is prime
17: **return** p, q, seed, counter

Parameter Validation in RFC 2631 (Group Membership Verification Part)

Group parameters validation:

- p and q are prime and q divides $p - 1$
- (optional) p and q follow parameter generation algorithm from seed and counter
- $g^q \bmod p = 1$ and $1 < g < p$

Public key validation:

- check $2 \leq y \leq p - 1$ and $y^q \bmod p = 1$
  $\rightarrow$ this is enough to prove $y \in \langle g \rangle - \{1\}$! (see next slide)

Checking Group Membership

($\langle g \rangle$ 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

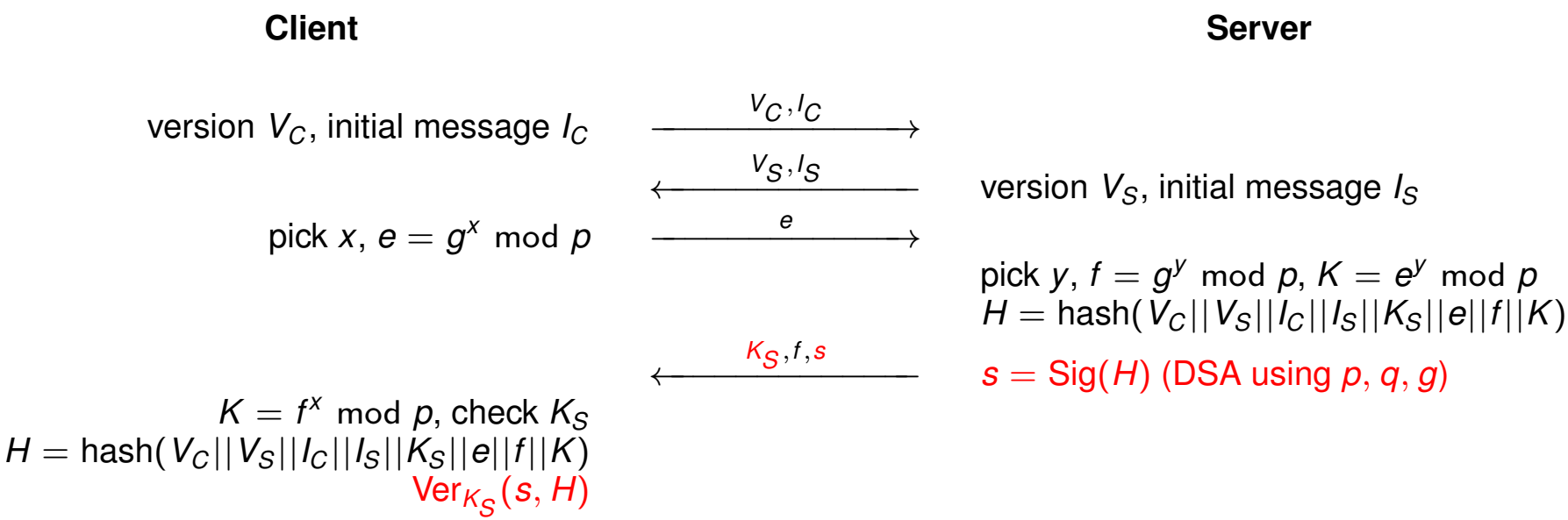
- $\langle g \rangle$ is a subgroup of $\mathbf{Z}_p^*$ of order q: clear
- $\langle g \rangle \subseteq \{y \in \mathbf{Z}_p^*; y^q \bmod p = 1\}$: clear
- $\langle g \rangle \supseteq \{y \in \mathbf{Z}_p^*; y^q \bmod p = 1\}$:
  let $y \in \mathbf{Z}_p^*$ be such that $y^q \bmod p = 1$
  - let $\theta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

Example: Semi-Authenticated Key Exchange in SSH2

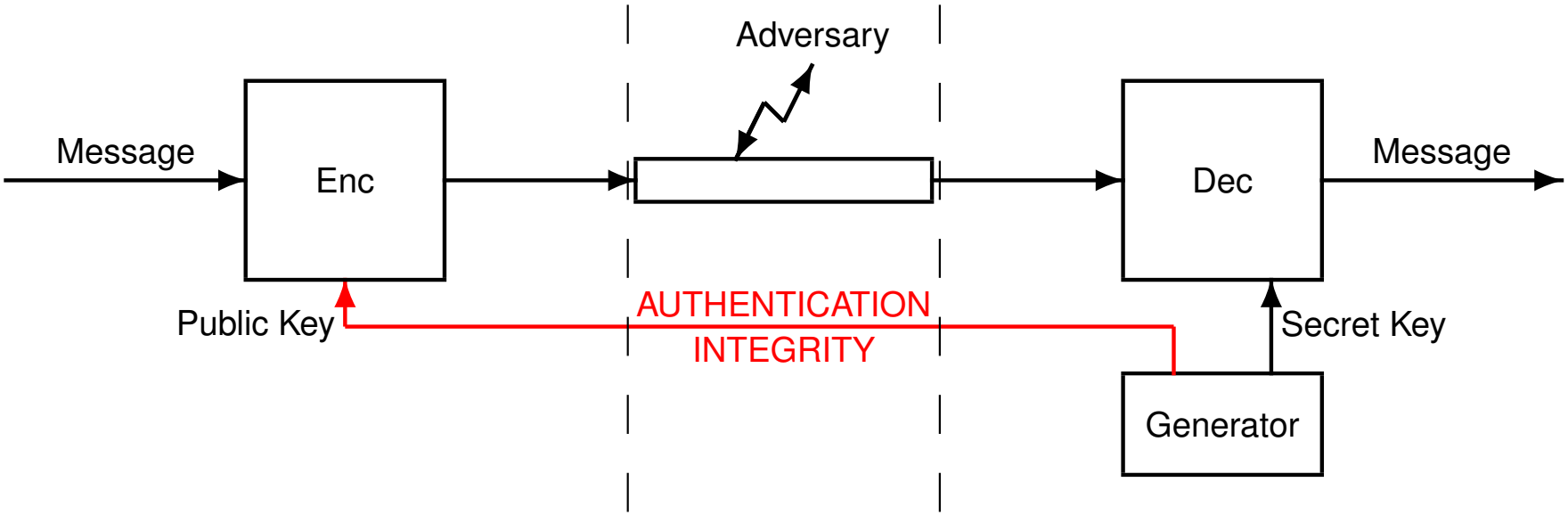
- I_C and I_S: negotiation of crypto algorithms
- K_S: public key of the server (may come with a certificate)
- for `diffie-hellman-group1-sha1` key exchange:
  $p = 2^{1024} - 2^{960} - 1 + 2^{64} \left\lfloor 2^{894}\pi + 129093 \right\rfloor$, $g = 2$, $q = \frac{p-1}{2}$



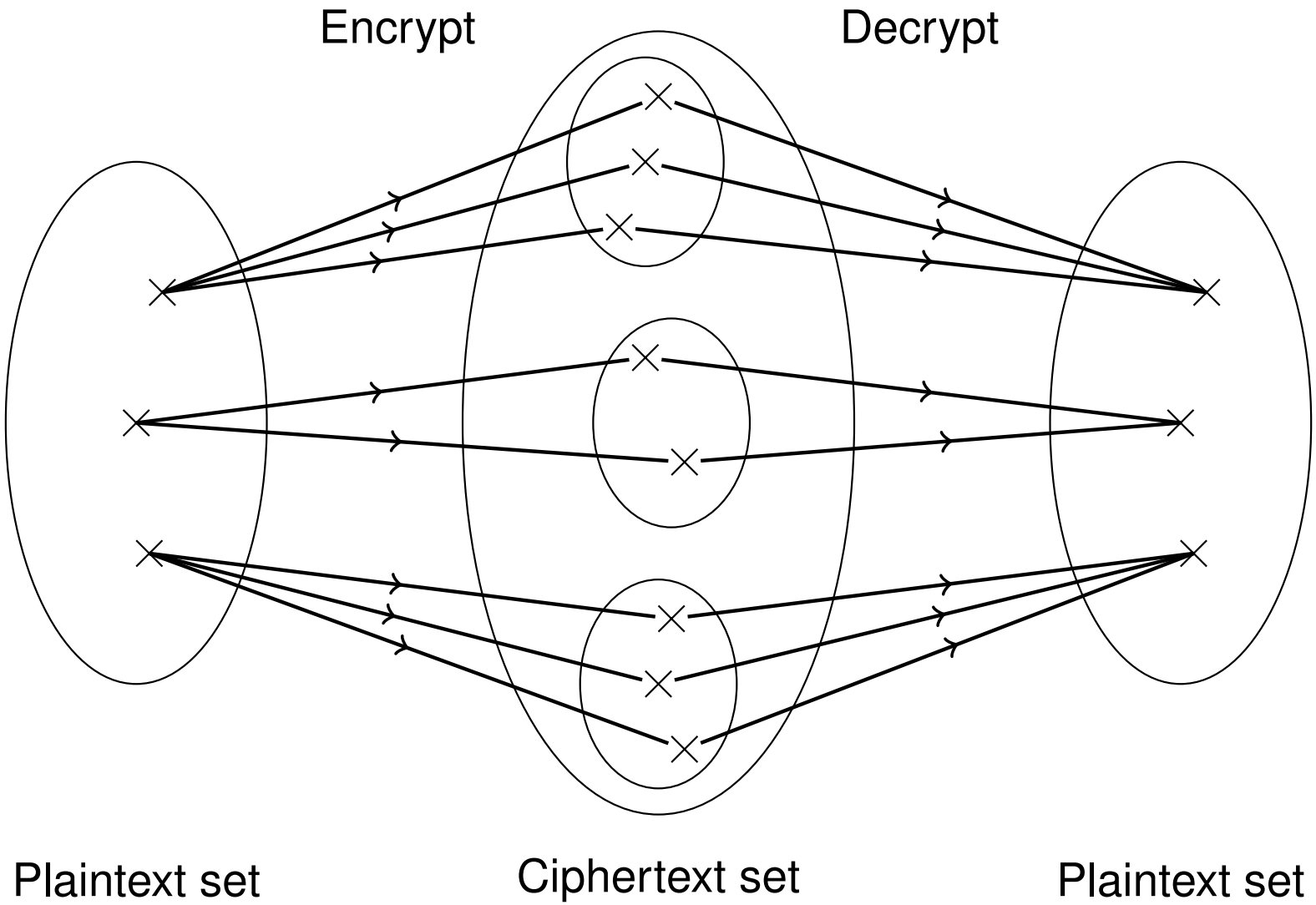
2 Diffie-Hellman Cryptography

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

Public-Key Cryptosystem

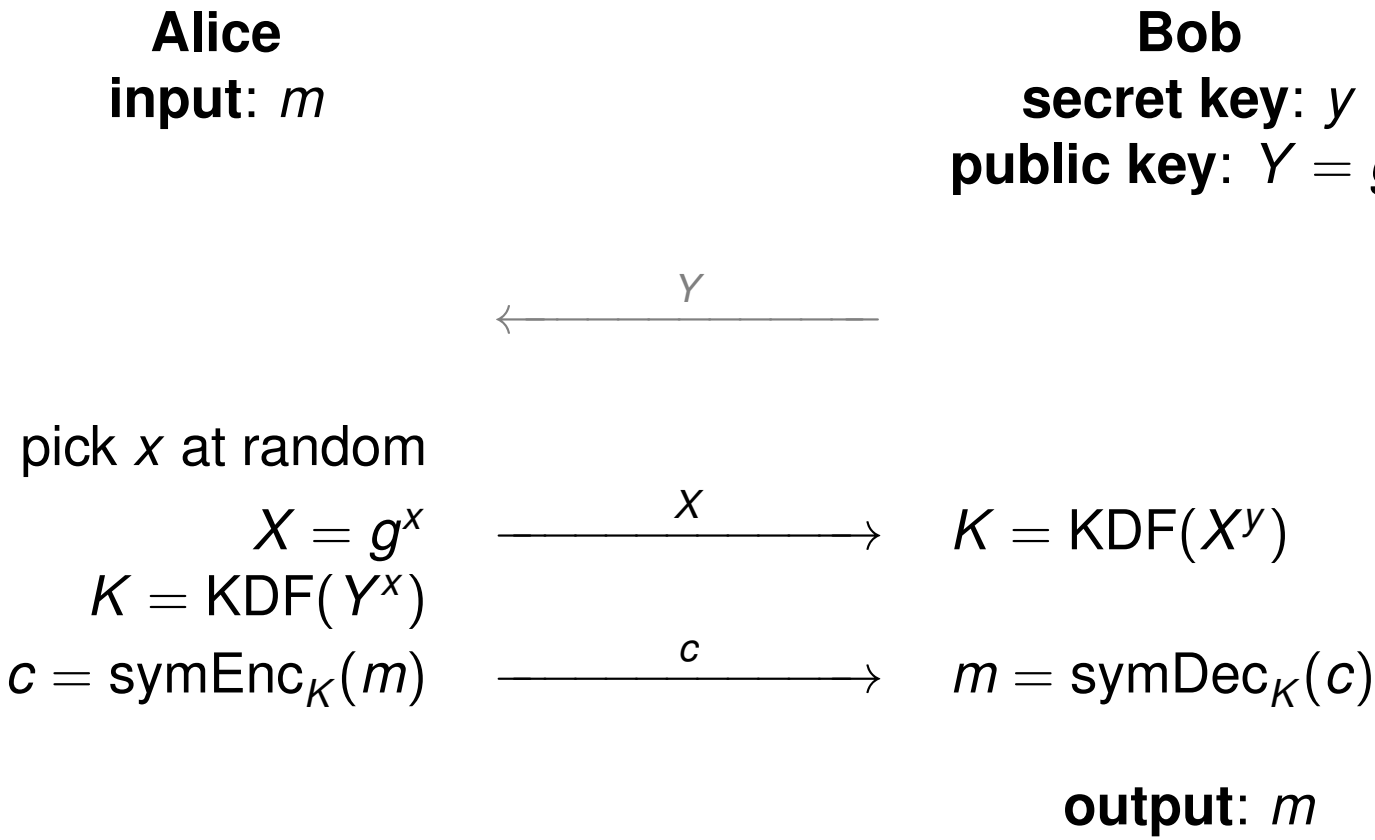


Non-Deterministic Encryption



Semi-Static-DH to Public-Key Encryption

Towards ElGamal Encryption



The Plain ElGamal Encryption Case

In the original ElGamal cryptosystem:

- in $\mathbf{Z}_p^*$, not of prime order...
- no KDF...
- symEnc is one-time-pad, adapted in the DH group

... this is all we should not do...
but wait: this is the basis of many cryptosystems...

In what follows: we work in $\langle g \rangle$ of order n
CAUTION: notation change

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: $\mathrm{pk} = y$

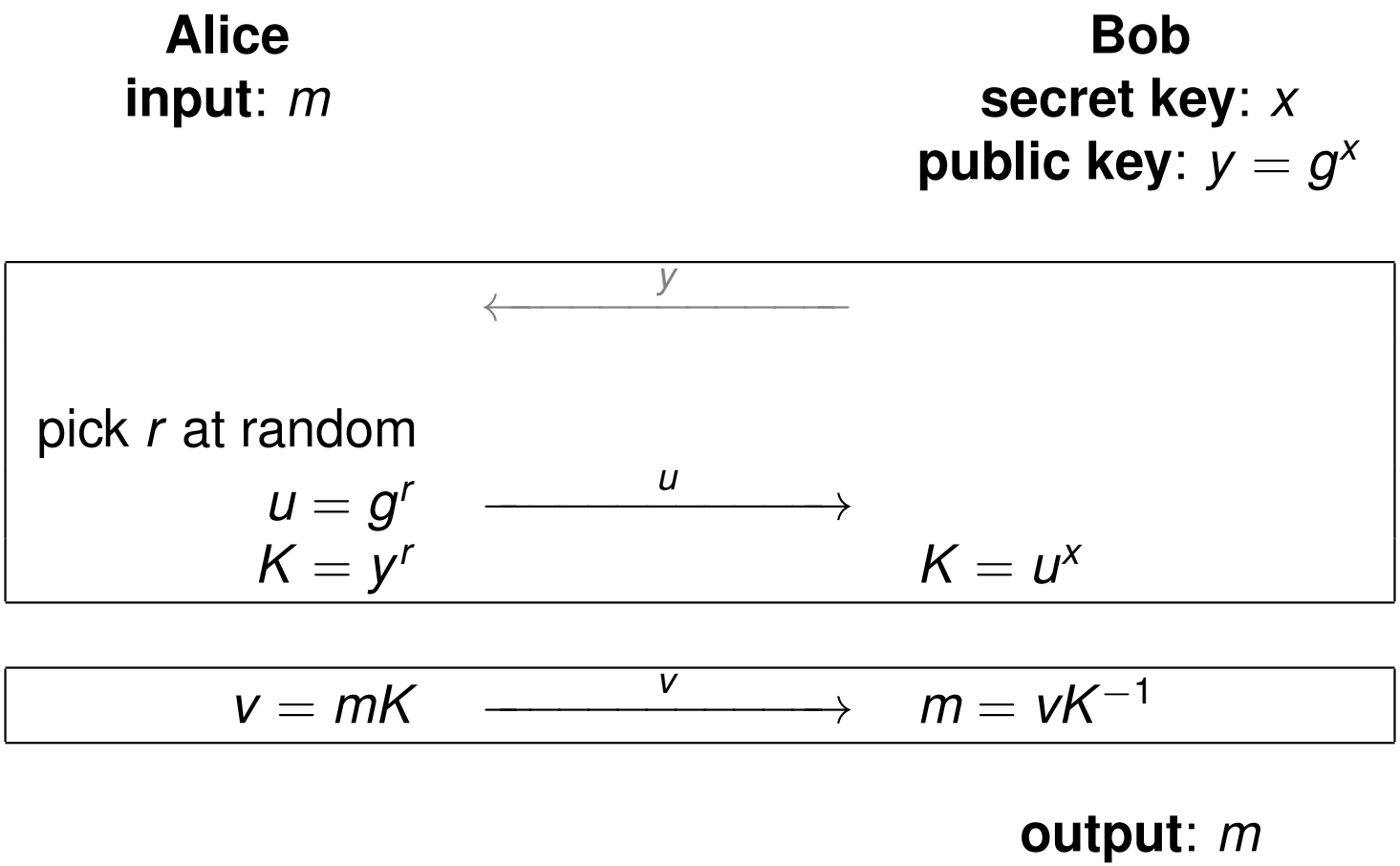
Secret key: $\mathrm{sk} = x$

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

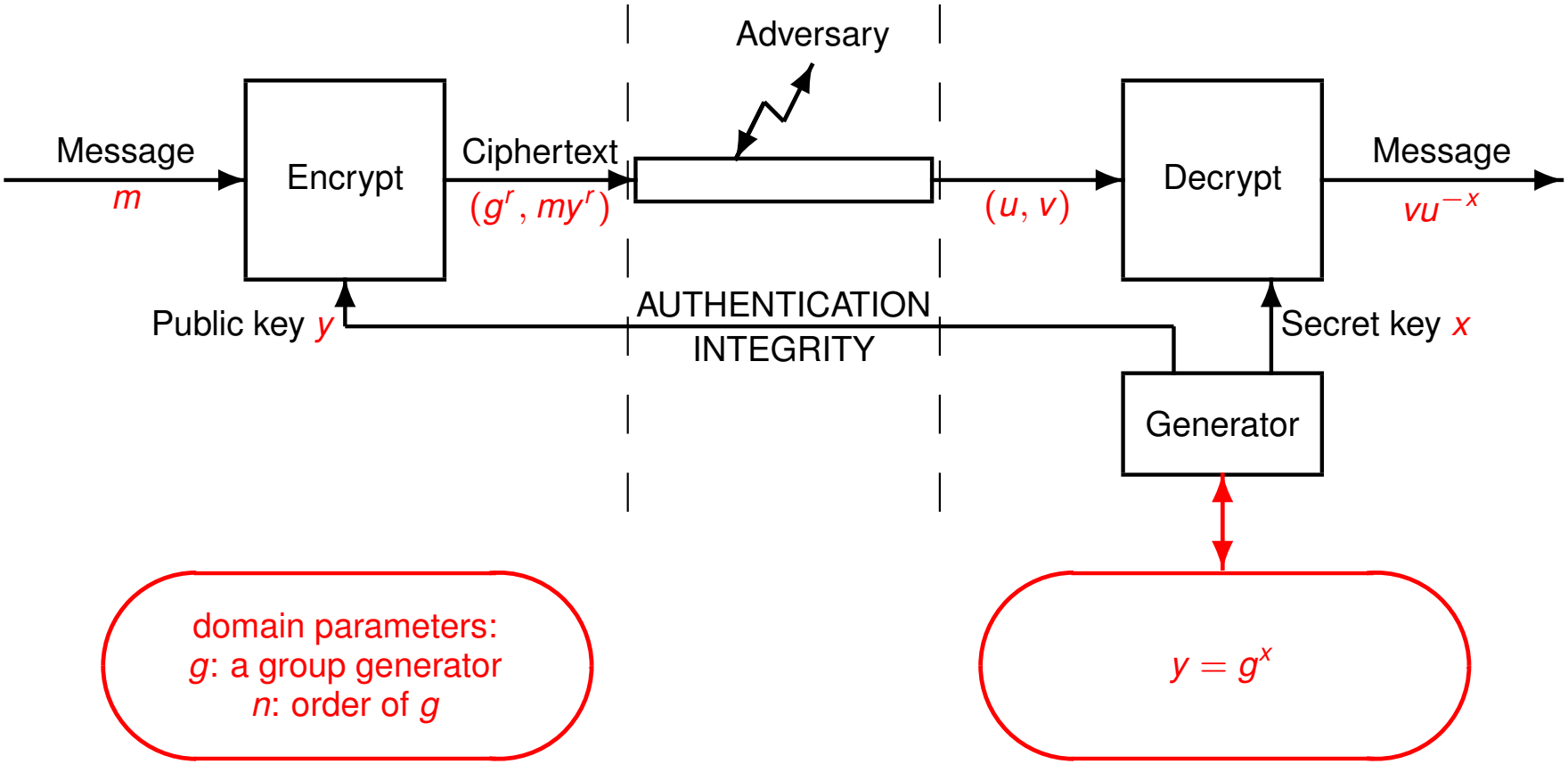
Decryption: if $u, v \in \langle g \rangle$, extract the u and v parts of the ciphertext and compute $m = vu^{-x}$

ElGamal Cryptosystem

Semi-Static DH + Vernam Generalized



Plain ElGamal Encryption



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

ElGamal Encryption Complexity

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

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

ElGamal Security: ElGamal Problems

EGKR (ElGamal Key Recovery Problem)

1: $\text{Setup}(1^\lambda) \to (\text{group}, n, g)$
2: $\text{Gen}(\text{group}, n, g) \to (y, x)$ ▷ pick $x \in \mathbf{Z}_n$, $y = g^x$
3: $\mathcal{A}(\text{group}, n, g, y) \to x'$
4: **return** $1_{x=x'}$

EGD (ElGamal Decryption Problem)

1: $\text{Setup}(1^\lambda) \to (\text{group}, n, g)$
2: $\text{Gen}(\text{group}, n, g) \to (y, x)$ ▷ pick $x \in \mathbf{Z}_n$, $y = g^x$
3: pick $\text{pt} \in \langle g \rangle$ ▷ pick $\text{pt} \in \langle g \rangle$
4: $\text{Enc}(y, \text{pt}) \to (u, v)$ ▷ pick $r \in \mathbf{Z}_n$, $u = g^r$, $v = \text{pt} \cdot y^r$
5: $\mathcal{A}(\text{group}, n, g, y, u, v) \to m$
6: **return** $1_{m=\text{pt}}$

key recovery problem $\iff$ DL problem
decryption problem $\iff$ CDH problem [next slide]

EGD hard $\Longrightarrow$ CDH hard

The EGD Problem Reduces to the CDH Problem

$\mathcal{B}(\text{group}, n, g, y, u, v)$
1: $\mathcal{A}(\ldots, g, y, u) \to K$
2: $m \leftarrow v/K$
3: **return** m

- assume EGD is hard
- to prove CDH hardness, consider a CDH algorithm $\mathcal{A}$
- construct $\mathcal{B}$ s.t. $\mathcal{A}$ wins CDH $\Longrightarrow$ $\mathcal{B}$ wins EGD:

$$\Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{A}(\ldots, X, Y) \to K\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{EGD}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick pt}\\ \text{pick } r,\ u \leftarrow g^r\\ v \leftarrow \text{pt} \cdot y^r\\ \mathcal{A}(\ldots, y, u) \to K\\ m \leftarrow v/K\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right] = \text{negl}$$

□

EGD hard $\Longrightarrow$ CDH hard (details)

$\mathcal{B}(\text{group}, n, g, y, u, v)$
1: $\mathcal{A}(\ldots, g, y, u) \to K$
2: $m \leftarrow v/K$
3: **return** m

$$\Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \mathcal{A}(\ldots, X, Y) \to K\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick } r,\ u \leftarrow g^r\\ \mathcal{A}(\ldots, y, u) \to K\\ \textbf{return } 1_{K=y^r}\end{array}\right] \to 1\right]$$

$$= \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick pt}\\ \text{pick } r,\ u \leftarrow g^r\\ v \leftarrow \text{pt} \cdot y^r\\ \mathcal{A}(\ldots, y, u) \to K\\ m \leftarrow v/K\\ \textbf{return } 1_{K=y^r}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{EGD}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick pt}\\ \text{pick } r,\ u \leftarrow g^r\\ v \leftarrow \text{pt} \cdot y^r\\ \mathcal{A}(\ldots, y, u) \to K\\ m \leftarrow v/K\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right]$$

$$K = y^r \iff \frac{\text{pt} \cdot y^r}{K} = \text{pt}$$

CDH hard $\Longrightarrow$ EGD hard

The CDH Problem Reduces to the EGD Problem

$\mathcal{B}(\text{group}, n, g, X, Y)$
1: pick v
2: $\mathcal{A}(\ldots, X, Y, v) \to m$
3: $K \leftarrow v/m$
4: **return** K

- assume CDH is hard
- to prove EGD hardness, consider a EGD algorithm $\mathcal{A}$
- construct $\mathcal{B}$ s.t. $\mathcal{A}$ wins EGD $\Longrightarrow$ $\mathcal{B}$ wins CDH:

$$\Pr\left[\left[\begin{array}{l}\text{EGD}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick pt}\\ \text{pick } r,\ u \leftarrow g^r\\ v \leftarrow \text{pt} \cdot y^r\\ \mathcal{A}(\ldots, y, u, v) \to m\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right] \leq \Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \text{pick v}\\ \mathcal{A}(\ldots, X, Y, v) \to m\\ K \leftarrow v/m\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right] = \text{negl}$$

□

CDH hard $\Longrightarrow$ EGD hard (details)

$\mathcal{B}(\text{group}, n, g, X, Y)$
1: pick v
2: $\mathcal{A}(\ldots, X, Y, v) \to m$
3: $K \leftarrow v/m$
4: **return** K

$$\Pr\left[\left[\begin{array}{l}\text{EGD}_{\mathcal{A}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x,\ y \leftarrow g^x\\ \text{pick pt}\\ \text{pick } r,\ u \leftarrow g^r\\ v \leftarrow \text{pt} \cdot y^r\\ \mathcal{A}(\ldots, y, u, v) \to m\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \text{pick pt}\\ v \leftarrow \text{pt} \cdot X^y\\ \mathcal{A}(\ldots, X, Y, v) \to m\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right]$$

$$= \Pr\left[\left[\begin{array}{l}\text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \text{pick } v\\ \mathcal{A}(\ldots, X, Y, v) \to m\\ \text{pt} \leftarrow v/X^y\\ \textbf{return } 1_{m=\text{pt}}\end{array}\right] \to 1\right] = \Pr\left[\left[\begin{array}{l}\text{CDH}_{\mathcal{B}}\text{ game:}\\ \text{Setup} \to \ldots, g\\ \text{pick } x, y\\ X \leftarrow g^x\\ Y \leftarrow g^y\\ \text{pick v}\\ \mathcal{A}(\ldots, X, Y, v) \to m\\ K \leftarrow v/m\\ \textbf{return } 1_{K=g^{xy}}\end{array}\right] \to 1\right]$$

ElGamal Encryption Security

- **key recovery** is equivalent to solving DL
- **decryption** is equivalent to the solving CDH
- **INDCPA security** equivalent to solving DDH
  (IND-CPA security defined on slide 785)

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, CDH, and DDH 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 + tricks to check/pick a generator
  - Lagrange Theorem
- **$\mathbb{Z}_n$ ring**: invertibility
- **$\mathbb{Z}_p$ field**: the multiplicative group is cyclic
- **algorithmic number theory**:
  - square-and-multiply
  - extended Euclid algorithm
- **Diffie-Hellman key exchange**:
  - resist to passive adversaries
  - better on a goup 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

Train Yourself

- subgroup issues: final exam 2016–17 ex5
- variant of DH: midterm exam 2017–18 ex3
- DDH mod pq: midterm exam 2022–23 ex2
- DLP in $\mathbf{Z}_{n^2}^*$: final exam 2019–20 ex2
- DLP in GGM: final exam 2022–23 ex3
- DH in composite group: midterm exam 2023–24 ex2
- comparing DH and ElGamal: final exam 2021–22 ex1
- DH as a group action: midterm exam 2024–25 ex2

Roadmap

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

3 RSA Cryptography

- Euler and Other Chinese
- 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* $\iff \mathbf{Z}_n^* = \mathbf{Z}_n \backslash \{0\} \iff \varphi(n) = n - 1 \iff n$ *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* $d = e^{-1} \bmod \varphi(n)$. *For all* $y \in \mathbf{Z}_n^*$, $y^d \bmod n$ *is the only eth root of y modulo n*

Proof — i

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

Proof.
$\Longrightarrow$: 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.
$\Longleftarrow$: if $\gcd(x, n) = 1$, the extended Euclid algorithm constructs an inverse of x (see slide 176) □

Proof — ii

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

Proof. By definition, $\mathbf{Z}_n$ is a field $\iff \mathbf{Z}_n^* = \mathbf{Z}_n \backslash \{0\}$.
Since $\mathbf{Z}_n^* \subseteq \mathbf{Z}_n \backslash \{0\}$, $\mathbf{Z}_n^*$ and $\mathbf{Z}_n \backslash \{0\}$ are equal iff they have the same cardinality.
We have $\#\mathbf{Z}_n^* = \varphi(n)$ and $\#\mathbf{Z}_n \backslash \{0\} = n - 1$, so we deduce $\mathbf{Z}_n^* = \mathbf{Z}_n \backslash \{0\} \iff \varphi(n) = n - 1$.

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

($\mathbf{Z}_n$ field $\impliedby$ n prime was seen on ▸ slide 190) □

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

"for $y \in \mathbf{Z}_n^*$, $x = y^d$ is the unique root of equation $y = x^e$"

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

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

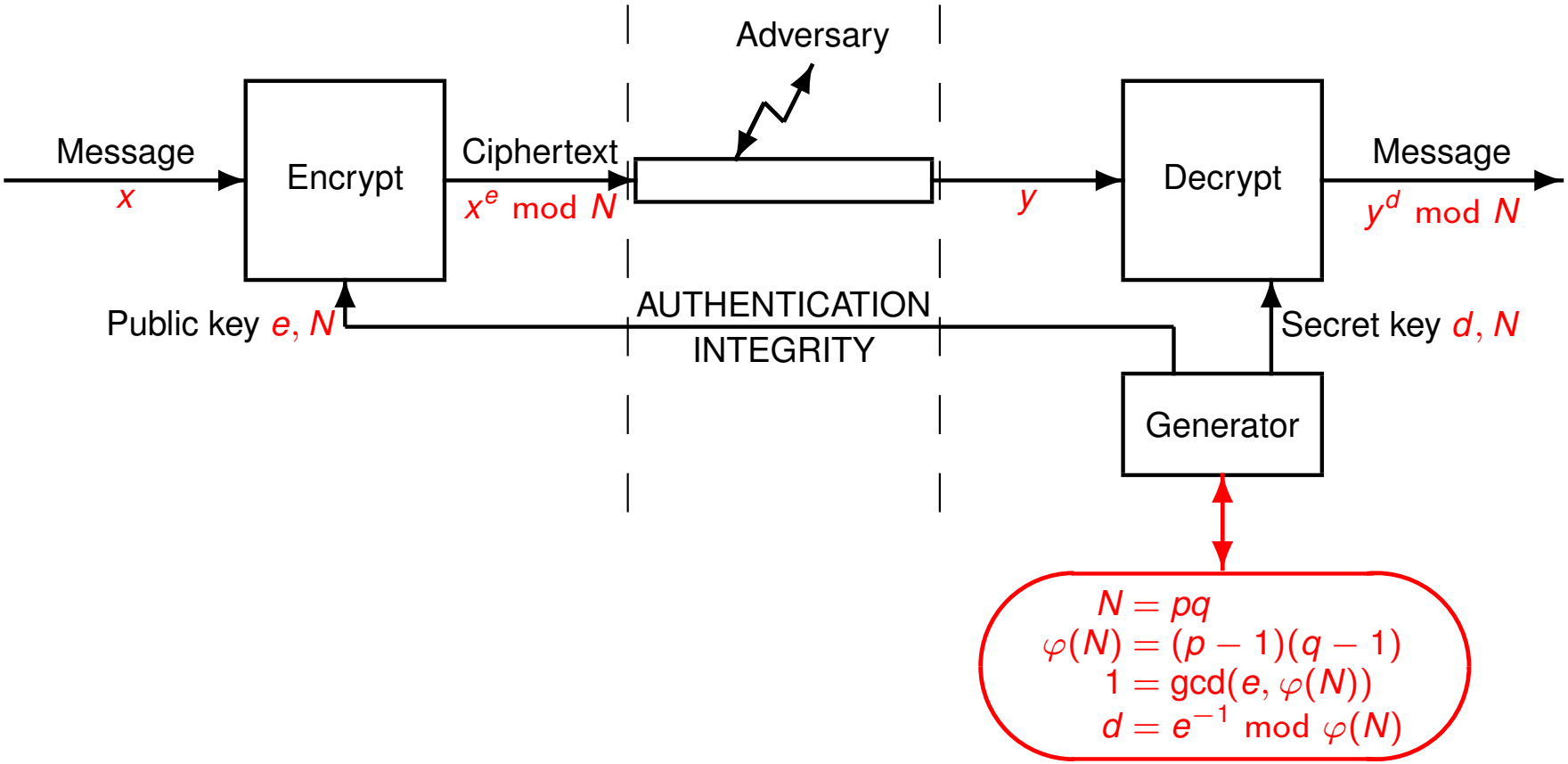
- $x \equiv y^d \Longrightarrow x^e \equiv y^{1+k\cdot\varphi(n)} \equiv y$ so $x = y^d$ is a eth root of y.
- If $y \equiv x^e$, we have $x \in \mathbf{Z}_n^*$ because

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

we have $y \equiv x^e \Longrightarrow y^d \equiv x^{1+k\cdot\varphi(n)} \equiv x$ so a eth root of y 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

$$\begin{aligned} x &\equiv a \pmod{m} \\ x &\equiv b \pmod{n} \end{aligned}$$

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*

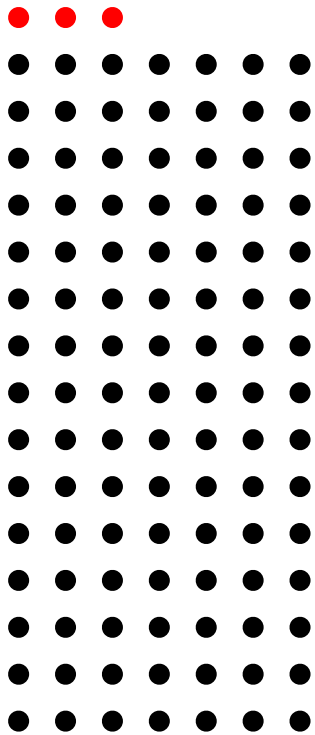
- $f : \mathbf{Z}_{mn} \to \mathbf{Z}_m \times \mathbf{Z}_n$ *defined by* $f(x) = (x \bmod m, x \bmod n)$ *is a ring isomorphism*
- $f^{-1}(a, b) \equiv an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \pmod{mn}$

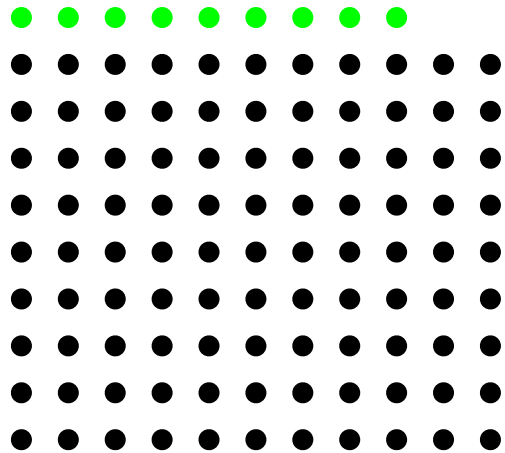
Example: $(m = 5, n = 7, mn = 35)$

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

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*

$$\left.\begin{array}{ll} a \equiv b & (\mathrm{mod}\ m) \\ a \equiv b & (\mathrm{mod}\ n) \end{array}\right\} \Longleftrightarrow a \equiv b \pmod{mn}.$$

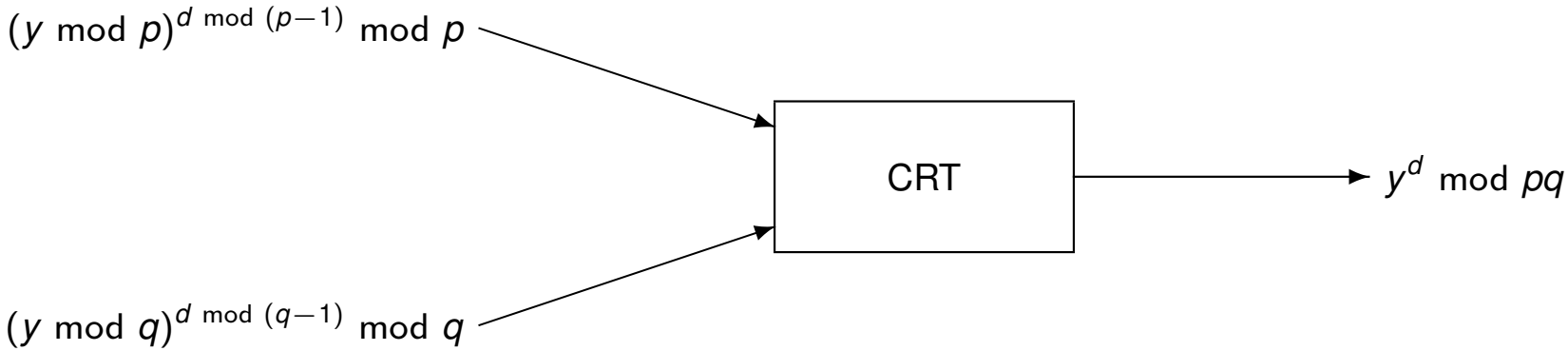
Indeed, $f(a \bmod (mn)) = f(b \bmod (mn))$ hence
$a \bmod (mn) = b \bmod (mn)$

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

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



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

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:

$$\begin{aligned} ((x+y) \bmod (mn)) \bmod m &= ((x \bmod m) + (y \bmod m)) \bmod m \\ ((x+y) \bmod (mn)) \bmod n &= ((x \bmod n) + (y \bmod n)) \bmod n \end{aligned}$$

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

Null Kernel

m and n divide x and are coprime

- let $n = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$ be the unique factorization of n into pairwise different primes p_i
- write $x = mx'$ (since m divides x)
- since n divides x, each $p_i^{\alpha_i}$ divides $x = mx'$
- since n is coprime with m, p_i does not divide m
- hence, $p_i^{\alpha_i}$ divides x'
- each $p_i^{\alpha_i}$ divides x' so n divides x'
- write $x' = nx''$
- $x = mnx''$: mn divides x

Proof of CRT — iii

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

$$\begin{aligned} an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) &\equiv a \pmod m \\ an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) &\equiv b \pmod n \end{aligned}$$

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*

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

is well defined and is a ring isomorphism.

Note: g is well defined because

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

does not depend on i or j
Consequence: $(u + v) \bmod (mn) = g(1, 1) = 1$

Proof

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

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 $a \bmod m = 0$ and $b \bmod n = 0$ so g is injective
- due to cardinality, g is bijective: so, a group isomorphism
- $g^{-1}(x) = (x \bmod m, x \bmod n)$ 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: $f(x) \in \mathbf{Z}_m^* \times \mathbf{Z}_n^*$
  so f maps $\mathbf{Z}_{mn}^*$ to $\mathbf{Z}_m^* \times \mathbf{Z}_n^*$
- 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 $x \in \mathbf{Z}_{mn}^*$
  so f from $\mathbf{Z}_{mn}^*$ to $\mathbf{Z}_m^* \times \mathbf{Z}_n^*$ is surjective
- f is an injection on $\mathbf{Z}_{mn}$ so an injection on $\mathbf{Z}_{mn}^*$ as well

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

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

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

For Generating a Generator

go to

For $g \in_U G$ in a cyclic group G of order $n = \prod_{j=1}^{r} p_j^{\alpha_j}$ with pairwise different primes p_j:

- let $h \in G$ be a generator; due to CRT, there is an isomorphism f

$$G \xrightarrow{\log_h} \mathbf{Z}_n \xrightarrow{f} \prod_{j=1}^{r} \mathbf{Z}_{p_j^{\alpha_j}} \quad \text{so} \quad g \overset{f \circ \log_h}{\longmapsto} (a_j)_{1 \le j \le r} \in_U \mathbf{Z}_{p_1^{\alpha_1}} \times \cdots \times \mathbf{Z}_{p_r^{\alpha_r}}$$

- due to isomorphism,

$$\begin{aligned} g^{\frac{n}{p_i}} = 1 &\iff \bigwedge_{j=1}^{r} \frac{n}{p_i} a_j \equiv 0 \pmod{p_j^{\alpha_j}} \\ &\iff \frac{n}{p_i} a_i \equiv 0 \pmod{p_i^{\alpha_i}} \\ &\iff a_i \bmod p_i = 0 \end{aligned}$$

- a_i's are independent and uniform

so $\Pr_{g \in_U G}\left[g^{\frac{n}{p_i}} = 1\right] = \frac{1}{p_i}$ and these events are independent

3 RSA Cryptography

- Euler and Other Chinese
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

Trial Division Algorithm

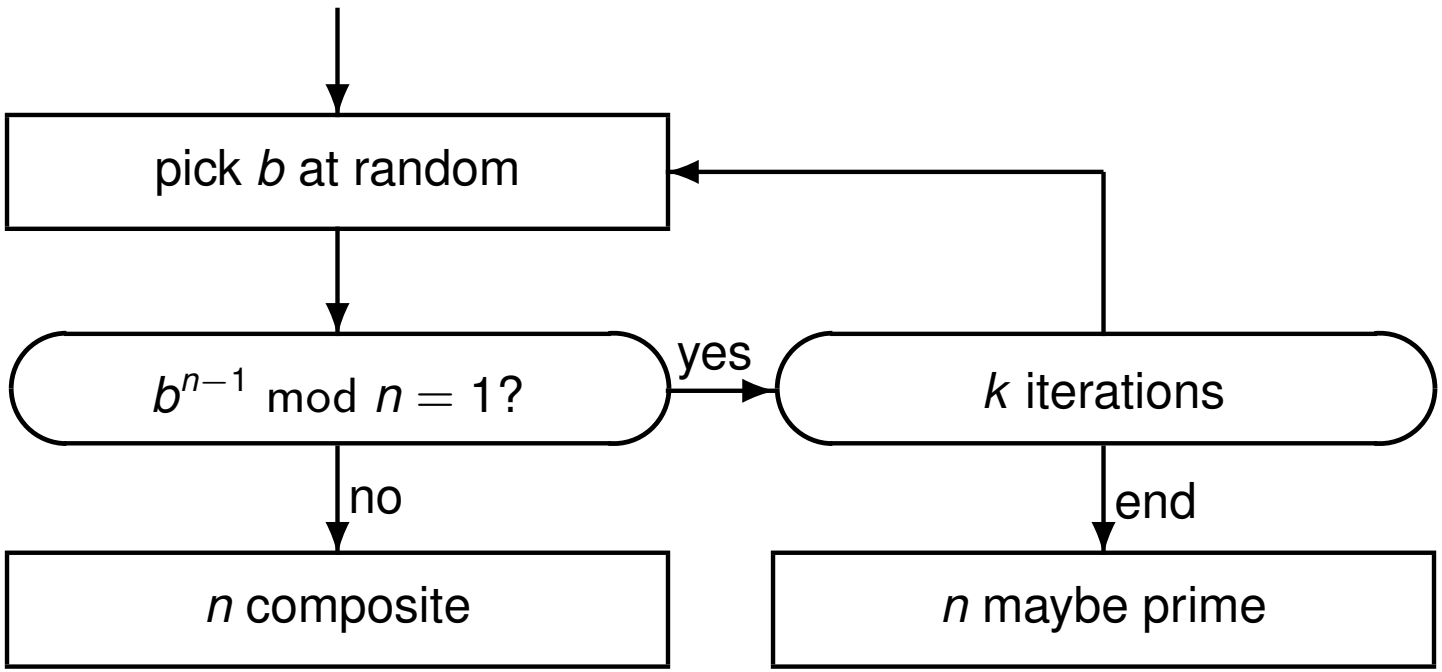
Input: an integer n
Output: a list of prime numbers whose product is n
Complexity: $\mathcal{O}(\sqrt{n})$ iterations (poly-time arithmetic operations in each iteration)

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

Fermat Test

Theorem (Little Fermat Theorem)

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



Fermat Test

Parameter: k, an integer
Input: n, an integer of ℓ 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: $\Pr[\text{output : composite} | n \text{ prime}] = 0$
- False Positive: there exist pathologic numbers n which are not prime such that $\Pr[\text{output : maybe prime} | n]$ 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

  $$\Pr[\text{output : maybe prime} | n] = \left(\frac{\varphi(n)}{n-1}\right)^k.$$

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* $b^{n-1} \equiv 1 \pmod{n}$.

Proof.

- $\Rightarrow$: get $b^{n-1} \bmod p_i = 1$ then apply CRT
- $\Leftarrow$: get $b^{n-1} \bmod p_i = 1$ for a genarator b of $\mathbf{Z}_{p_i}^*$ so $p_i - 1 | n - 1$
  PB to show that n is square-free (p_i's are pairwise different)

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 $b^{n-1} \equiv 1 \pmod{n}$.
$n - 1 = 560 = 2^4 \times 5 \times 7$ is divisible by $3 - 1$, $11 - 1$, $17 - 1$

The Fermat 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

$$\begin{aligned} 949\,631\,589\,089 &= 6917 \times 10193 \times 13469 \\ 949\,631\,589\,088 &= 2^5 \times 7^3 \times 13 \times 19 \times 37 \times 9467 \end{aligned}$$

- 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

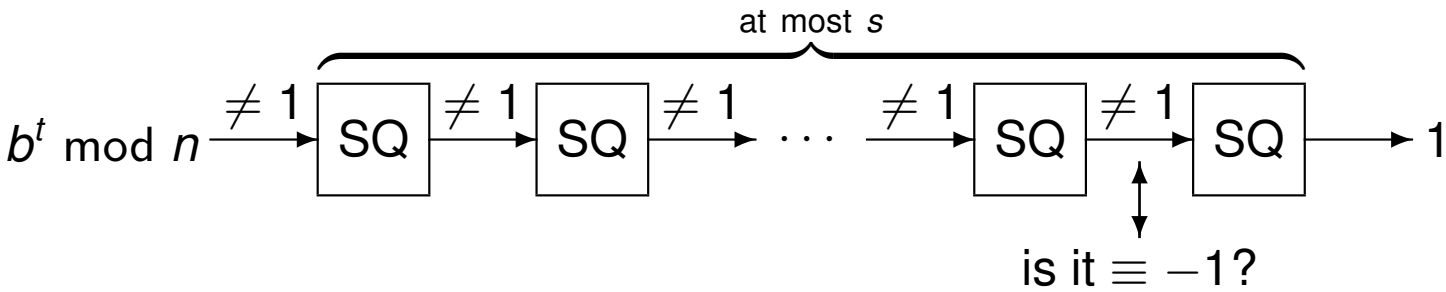
$$\left(\frac{\varphi(n)}{n-1}\right)^k = \cdots = \left(\frac{9464}{9467}\right)^k \approx (1 - 0.000317)^k$$

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 $b^{n-1} \bmod n = \left(\cdots \left((b^t)^2 \right)^2 \cdots \right)^2 \bmod n = 1$
- If n is prime, $+1$ and -1 are the only possible square roots of 1 (▸ slide 297)

The Miller-Rabin Test



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

The Miller-Rabin Primality Test

Parameter: k, an integer
Input: n, an integer of ℓ bits
Output: notification of non-primality
or pseudo-primality
Complexity: $\mathcal{O}(k\ell^3)$

if $n = 2$ **then**
    return "prime"
end if
if n is even **then**
    return "composite"
end if
write $n = 2^s t + 1$ with t odd
repeat
    pick $b \in \{1, \ldots, n-1\}$
    $x \leftarrow b^t \bmod n$
    $i \leftarrow 0$
    if $x \neq 1$ **then**
        while $x \neq n-1$ **do**
            $x \leftarrow x^2 \bmod n$
            $i \leftarrow i + 1$
            if $i = s$ or $x = 1$ **then**
                return "composite"
            end if
        end while
    end if
until k iterations are made
return "maybe prime"

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 $\square$

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:

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

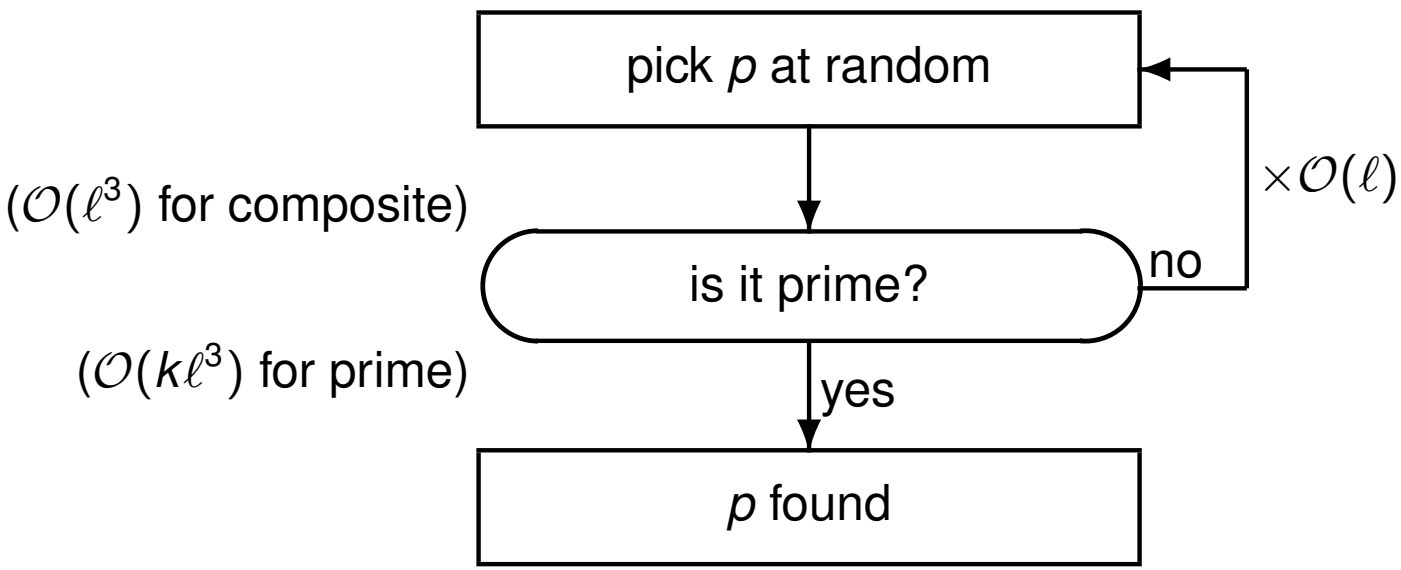
Prime Number Generation

Theorem (Prime Number Theorem)

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

$\rightarrow$ the probability that a random ℓ-bit number is prime is $\approx \frac{1}{\ell \ln 2}$
Example: a 512-bit random integer is prime with probability $\approx \frac{1}{355}$

$\rightarrow$ generating a random ℓ-bit prime number takes $\mathcal{O}(\ell^4 + k\ell^3)$



Implementation

Input: ℓ
Output: a random prime number less than 2^ℓ
Complexity: $\mathcal{O}(\ell^4)$ arithmetic operations

1: **repeat**
2: pick a random number n of ℓ 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 ε.

$$\Pr[\text{output not prime}] \leq \mathcal{O}(\ell) \times 4^{-k} = \mathcal{O}(\varepsilon)$$

(next slide)

Incorrectness Probability

- p_i: probability to make exactly i iterations and make an incorrect response at the i-th iteration
- $\Pr[\text{output not prime}] = \sum_i p_i$
- we have

$$\begin{aligned} p_i &\leq \Pr[\text{pick composite and be wrong}] \Pr[\text{pick composite}]^{i-1} \\ &\leq \Pr[\text{wrong}|\text{composite}] \Pr[\text{pick composite}]^{i-1} \\ &\leq 4^{-k} \Pr[\text{pick composite}]^{i-1} \end{aligned}$$

- hence

$$\begin{aligned} \Pr[\text{output not prime}] &= \sum_i p_i \\ &\leq 4^{-k} \sum_i \Pr[\text{pick composite}]^{i-1} \\ &= 4^{-k} / \Pr[\text{pick prime}] \\ &= \mathcal{O}(\ell) \times 4^{-k} \end{aligned}$$

3 RSA Cryptography

- Euler and Other Chinese
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

Plain RSA Cryptosystem

Public parameter: an integer ℓ.

Set up: find two random different prime numbers p and q of size $\frac{\ell}{2}$ bits. Set $N = pq$. Select e such that $\gcd(e, (p-1)(q-1)) = 1$:

- either pick a random e until it is valid
- or pick $e = 17$ or $e = 2^{16} + 1$ if valid.

Set $d = e^{-1} \bmod ((p-1)(q-1))$.

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

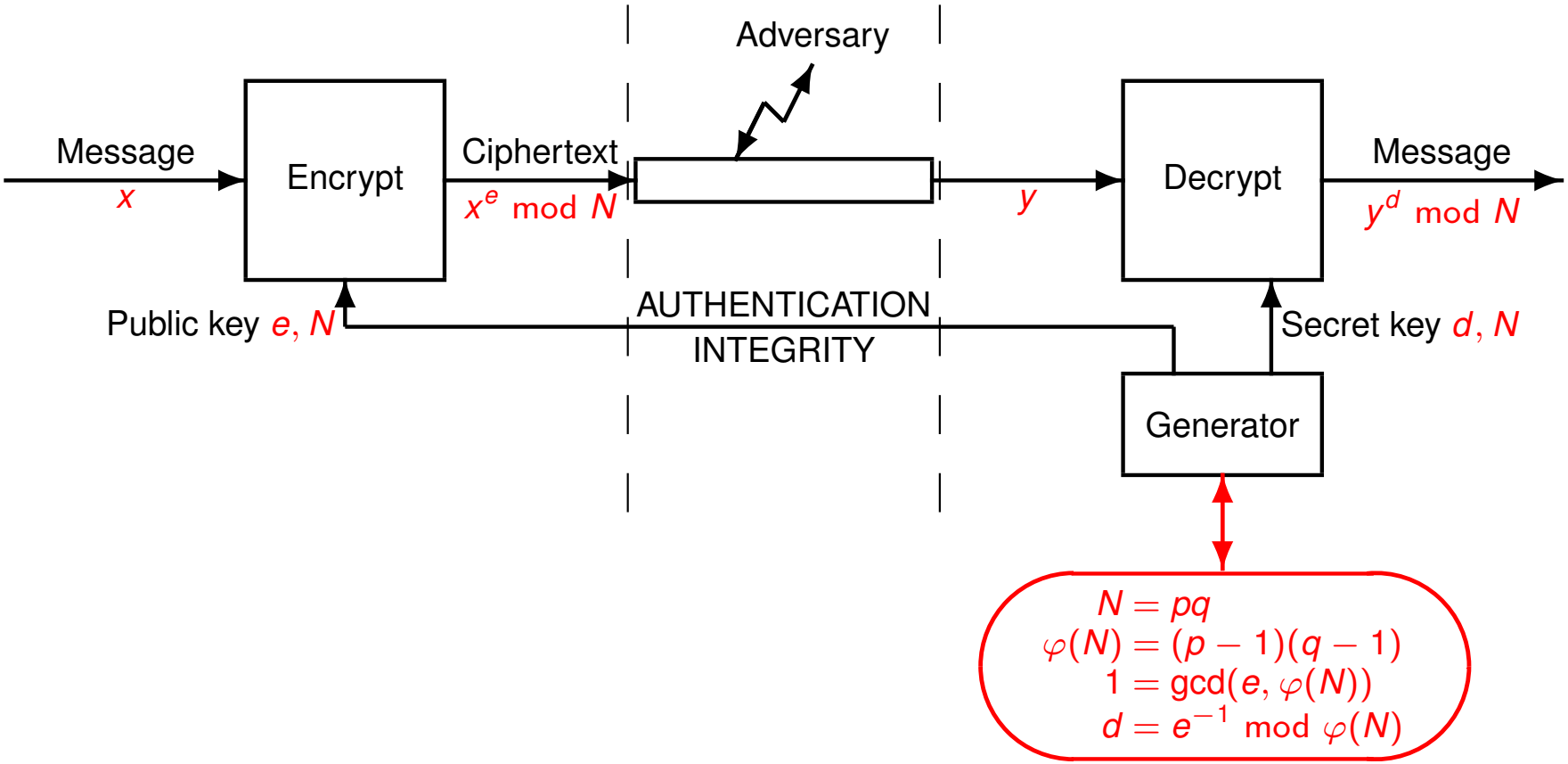
Public key: $\mathsf{pk} = (e, N)$.

Secret key: $\mathsf{sk} = (d, N)$.

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

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

Plain RSA


Adversary
Message
x
Encrypt
Ciphertext
$x^e \bmod N$
y
Decrypt
Message
$y^d \bmod N$
Public key e, N
AUTHENTICATION
INTEGRITY
Secret key d, N
Generator
$N = pq$
$\varphi(N) = (p-1)(q-1)$
$1 = \gcd(e, \varphi(N))$
$d = e^{-1} \bmod \varphi(N)$

RSA Correctness

Theorem (Euler)

Let p, q be two different primes and $N = p \times q$.
If $ed \equiv 1 \pmod{\varphi(N)}$, then, for any $x \in \{0, \ldots, N-1\}$,
$x^{ed} \bmod N = x$.

Consequence: RSA decryption works!

Proof. see application of CRT...

RSA Complexity

RSA with a modulus of ℓ 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 ℓ 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

- No common setup for RSA
- Complexity of Gen is much lower for ElGamal (ElGamal is better) (once common ElGamal setup is done)
- Complexity of Enc is lower for RSA (constant e) (RSA is better)
- Problem: ElGamal encryption is length-increasing (RSA is better)
- ElGamal can be easily adapted to other groups (e.g. elliptic curves) (ElGamal is better)
- ElGamal is probabilistic, (plain) RSA is deterministic

3 RSA Cryptography

- Euler and Other Chinese
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

Square Roots in Fields

Lemma

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

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

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 field

Existence of Square Roots in Z_p

Theorem

Let p be an odd prime number.
$b \in \mathbf{Z}_p^*$ has a square root if and only if $b^{\frac{p-1}{2}} \bmod p = 1$.
*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 $b \equiv g^{2e'} \equiv \left(g^{e'}\right)^2$ so b has a square root $g^{e'}$

□

Note: there exists an algorithm in $\mathcal{O}((\log p)^2)$ to check QR

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

$\frac{p+1}{4}$ is integer!

Lemma

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

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

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

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

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

$$\begin{aligned} x &\equiv a^2 \pmod{p} \\ x &\equiv b^2 \pmod{q} \end{aligned}$$

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

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

$$\left.\begin{aligned} x &\equiv y^2 \pmod{p} \\ x &\equiv y^2 \pmod{q} \end{aligned}\right\} \Leftrightarrow \left\{\begin{aligned} a^2 &\equiv y^2 \pmod{p} \\ b^2 &\equiv y^2 \pmod{q} \end{aligned}\right\} \Leftrightarrow \left\{\begin{aligned} y &\equiv \pm a \pmod{p} \\ y &\equiv \pm b \pmod{q} \end{aligned}\right.$$

□

3 RSA Cryptography

- Euler and Other Chinese
- Primality Testing
- RSA Basics
- Quadratic Residuosity
- The Factoring Problem

Factoring Problem

Gen-Factoring Problem

Factoring(λ):

1: $\text{Gen}(1^\lambda) \to n$
2: $\mathcal{A}(n) \to (p, q)$
3: **return** $1_{p \times q = n \wedge p,q \in \{2,\ldots,n-1\}}$

Example: Gen generates an RSA modulus
(Note: this is the splitting problem, not the full factoring problem.)

Record using the Number Field Sieve Algorithm

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

RSA250 (829 bits)

= 21403246502407449612644230728393335630086147151447550177977549208814180234471401366433455190958046796109928518724709145876873962619215573630474547705208051190564931066876915900197594056934574522305893259766974716817380693648946998715784949759374979 37

= 64135289477071580278790190170577389084825014742943447208116859632024532344630238623598752668347708737661925585694639798853367

$\times$

33372027594978156556226010605355114227940760344767554666784520987023841729210037080257448673296881877565718986258036932062711

factored in 2020 by an equivalent of 2700 years of computation on one core 2.1GHz Intel Xeon Gold 6130.

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

Factoring Algorithms on Classical Computers

- GNFS: factor n

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

  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 Z_{pq}

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

RSA-Factoring

Factoring(λ):

1: RSA-Gen(1^λ) $\rightarrow n$
2: $\mathcal{A}(n) \rightarrow (p, q)$
3: **return** $1_{p \times q = n \wedge p,q \in \{2,\ldots,n-1\}}$

RSA-Square roots

Factoring(λ):

1: RSA-Gen(1^λ) $\rightarrow n$
2: pick $x \in \mathrm{QR}_n$
3: $\mathcal{A}(n, x) \rightarrow y$
4: **return** $1_{y^2 \bmod n = x}$

Factoring $n \Longrightarrow$ Computing Square Roots in Z_n

Input: n, x
Output: y such that $y^2 \bmod n = x$
Complexity: $\mathcal{O}((\log n)^3)$ + factoring n
$\mathcal{B}(n, x)$:

1: $\mathcal{A}(n) \to (p, q)$ ▷ $\mathcal{A}$ playing the factoring game
2: find y_p, a square roots of x modulo p by using efficient algorithms
(e.g. for $p \bmod 4 = 3$ compute $x^{\frac{p+1}{4}} \bmod p$)
3: find y_q, a square roots of x modulo q
4: **return** $y = \mathrm{CRT}_{p,q}(y_p, y_q)$

$$\Pr[\mathcal{B} \text{ wins SQRT}] \geq \Pr[\mathcal{A} \text{ wins FACT}]$$

Computing Square Roots in $\mathbf{Z}_n \Longrightarrow$ Factoring n

Input: n
Output: p, q prime such that $n = pq$
Complexity: $\mathcal{O}((\log n)^2) + |\text{SQRT}|$

$\mathcal{B}(n)$:

1: pick $y_0 \in \{1, \ldots, n-1\}$
2: **if** $y_0 \notin \mathbf{Z}_n^*$ **then** factor...
3: $x \leftarrow y_0^2 \bmod n$
4: $y \leftarrow \mathcal{A}(n, x)$ ▷ SQRT
5: **if** $y = y_0$ or $y = -y_0 \bmod n$ **then** abort
6: $p \leftarrow \gcd(y - y_0, n)$
7: $q \leftarrow n/p$
8: **return** (p, q)

- 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

$$\Pr[\mathcal{B} \text{ wins FACT}] \geq \frac{1}{2} \Pr[\mathcal{A} \text{ wins SQRT}]$$

Note

Lemma

For $y_0, y_1 \in \mathbf{Z}_n$

$$\left.\begin{array}{rcll} y_0^2 & \equiv & y_1^2 & (\text{mod } n) \\ y_0 & \not\equiv & y_1 & (\text{mod } n) \\ y_0 & \not\equiv & -y_1 & (\text{mod } n) \end{array}\right\} \Longrightarrow \gcd(y_0 - y_1, n) \notin \{1, n\}$$

Proof.

- $y_0 - y_1 \not\equiv 0 \Longrightarrow \gcd(y_0 - y_1, n) \neq n$.
- $y_0^2 - y_1^2 \equiv 0 \Longrightarrow n | (y_0 - y_1)(y_0 + y_1)$.
- If $\gcd(y_0 - y_1, n) = 1$ then $n | y_0 + y_1$ which contradicts $y_0 + y_1 \not\equiv 0$.
- Hence $\gcd(y_0 - y_1, n) \notin \{1, n\}$. □

RSA Security: RSA Problems

(implicit: n is product of two different large primes and $\gcd(e, \varphi(n)) = 1$)

RSADP (RSA Decryption Problem)

1: $\text{RSA.Gen}(1^\lambda) \to (n, e, d)$
2: pick $x \in \mathbf{Z}_n$
3: $y \leftarrow x^e \bmod n$
4: $\mathcal{A}(n, e, y) \to z$
5: **return** $1_{x=z}$

GOP (Group Order Problem)

1: $\text{RSA.Gen}(1^\lambda) \to (n, e, d)$
2: $\mathcal{A}(n) \to z$
3: **return** $1_{\varphi(n)=z}$

RSAKRP (RSA Key Recovery Problem)

1: $\text{RSA.Gen}(1^\lambda) \to (n, e, d)$
2: $\mathcal{A}(n, e) \to z$
3: **return** $1_{z=d}$

RSAFP (RSA Factorization Problem)

1: $\text{RSA.Gen}(1^\lambda) \to (n, e, d)$
2: $\mathcal{A}(n) \to (p, q)$
3: **return** $1_{pq=n, 1<p,q<n}$

OMP (Order Multiple Problem)

1: $\text{RSA.Gen}(1^\lambda) \to (n, e, d)$
2: $\mathcal{A}(n) \to z$
3: **return** $1_{\varphi(n) \text{ divides } z \text{ and } z \neq 0}$

$$\begin{array}{ccccc} \text{RSADP} & \Leftarrow & \text{RSAKRP} & \Leftarrow & \text{GOP} \\ & & \Downarrow & & \Updownarrow \\ & & \text{OMP} & \Rightarrow & \text{RSAFP} \end{array}$$

RSAKRP $\Longrightarrow$ RSADP

$$\Pr\left[\begin{array}{l}\textbf{RSAKRP}_{\mathcal{A}}: \\ \text{Gen} \rightarrow (n, e, d) \\ \mathcal{A}(n, e) \rightarrow d' \\ \textbf{return } 1_{d=d'} \\ \downarrow \\ 1\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{RSADP}_{\mathcal{B}}: \\ \text{Gen} \rightarrow (n, e, d) \\ \text{pick } x \\ y \leftarrow x^e \bmod n \\ // \ \mathcal{B}(n, e, y) \rightarrow x': \\ \mathcal{A}(n, e) \rightarrow d' \\ x' \leftarrow y^{d'} \bmod n \\ \textbf{return } 1_{x=x'} \\ \downarrow \\ 1\end{array}\right]$$

because $x = y^d \bmod n$ due to correctness

Important Note

the $\Longrightarrow$ symbol is not your friend

- in these slides, $A \Longrightarrow B$ means

  "solving A implies solving B"

- in other slides, it could have meant

  "A hard implies B hard"

GOP $\Longrightarrow$ RSAKRP

$$\Pr\left[\begin{array}{l}\textbf{GOP}_{\mathcal{A}}: \\ \text{Gen} \rightarrow (n, e, d) \\ \mathcal{A}(n) \rightarrow z \\ \textbf{return } 1_{\varphi(n)=z} \\ \downarrow \\ 1\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{RSAKRP}_{\mathcal{B}}: \\ \text{Gen} \rightarrow (n, e, d) \\ // \ \mathcal{B}(n, e) \rightarrow d': \\ \mathcal{A}(n) \rightarrow z \\ d' \leftarrow e^{-1} \bmod z \\ \textbf{return } 1_{d=d'} \\ \downarrow \\ 1\end{array}\right]$$

because $d = e^{-1} \bmod \varphi(n)$

RSAKRP $\Longrightarrow$ OMP

$$\Pr\left[\begin{array}{l}\textbf{RSAKRP}_{\mathcal{A}}: \\ \text{Gen} \to (n,e,d) \\ \mathcal{A}(n,e) \to d' \\ \textbf{return } 1_{d=d'} \\ \qquad\downarrow \\ \qquad 1\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{OMP}_{\mathcal{B}}: \\ \text{Gen} \to (n,e,d) \\ //\ \mathcal{B}(n) \to z: \\ \mathcal{A}(n,e) \to d' \\ z \leftarrow ed' - 1 \\ \textbf{return } 1_{\varphi(n) \text{ divides } z \text{ and } z \neq 0} \\ \qquad\qquad\downarrow \\ \qquad\qquad 1\end{array}\right]$$

because $e > 1$, $d \geq 0$, $ed \equiv 1 \pmod{\varphi(n)}$

RSAFP $\Longrightarrow$ GOP

$$\Pr\left[\begin{array}{l}\textbf{RSAFP}_{\mathcal{A}}: \\ \text{Gen} \rightarrow (n,e,d) \\ \mathcal{A}(n) \rightarrow (p,q): \\ \textbf{return } 1_{pq=n,1<p,q<n} \\ \qquad\qquad \downarrow \\ \qquad\qquad 1\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{GOP}_{\mathcal{B}}: \\ \text{Gen} \rightarrow (n,e,d) \\ //\ \mathcal{B}(n) \rightarrow z: \\ \mathcal{A}(n) \rightarrow (p',q') \\ z \leftarrow (p'-1)\times(q'-1) \\ \textbf{return } 1_{\varphi(n)=z} \\ \qquad\qquad \downarrow \\ \qquad\qquad 1\end{array}\right]$$

because $\varphi(n) = (p-1)\times(q-1)$

GOP $\Longrightarrow$ RSAFP

$$\Pr\left[\begin{array}{l}\textbf{GOP}_{\mathcal{A}}:\\ \text{Gen}\rightarrow(n,e,d)\\ \mathcal{A}(n)\rightarrow z\\ \textbf{return } 1_{\varphi(n)=z}\\ \qquad\downarrow\\ \qquad 1\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{RSAFP}_{\mathcal{B}}:\\ \text{Gen}\rightarrow(n,e,d)\\ //\ \mathcal{B}(n)\rightarrow(p,q):\\ \mathcal{A}(n)\rightarrow z\\ p,q\leftarrow \text{ roots in } \mathbf{Z} \text{ of}\\ X^2-(n-z+1)X+n=0\\ \textbf{return } 1_{pq=n,1<p,q<n}\\ \qquad\downarrow\\ \qquad 1\end{array}\right]$$

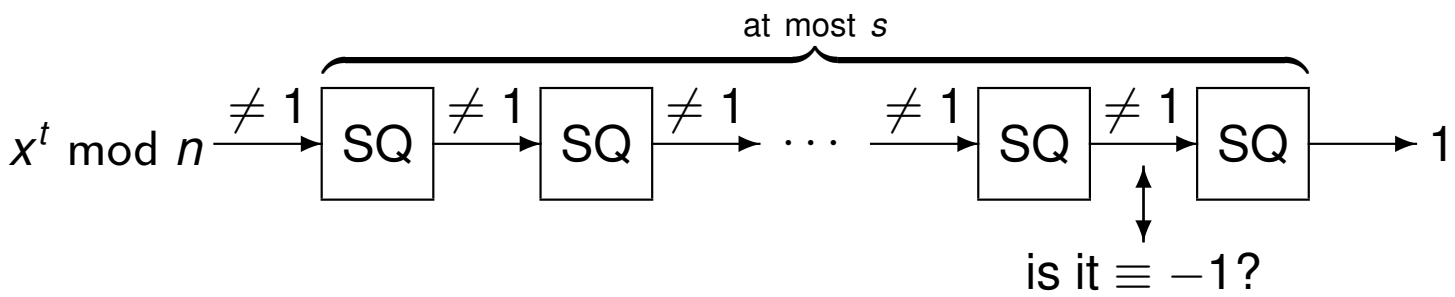
because $p+q=n-\varphi(n)+1$ and $p\times q=n$

OMP $\Longrightarrow$ RSAFP

$$\Pr\left[\begin{array}{l}\textbf{OMP}_{\mathcal{A}}: \\ \text{Gen} \to (n,e,d) \\ \mathcal{A}(n) \to z: \\ \textbf{return } 1_{\varphi(n) \text{ divides } z \text{ and } z \neq 0}\end{array}\right] \leq \Pr\left[\begin{array}{l}\textbf{RSAFP}_{\mathcal{B}}: \\ \text{Gen} \to (n,e,d) \\ // \ \mathcal{B}(n) \to (p,q): \\ \mathcal{A}(n) \to z \\ \text{run “factor using } \varphi(n)\text{”} \\ \text{algo using } z \\ \textbf{return } 1_{pq=n, 1<p,q<n}\end{array}\right]$$

factorization using $\varphi(n)$: next slide

Factorization using $\varphi(n)$



- write $z = 2^s t$ with t odd
- pick a random x, replace x by $x^t \bmod N$
- iteratively square x, get the last x which is not 1
- if $x \equiv -1$, try again, otherwise, output $\gcd(x - 1, N)$

Factorization using $\varphi(n)$ Multiple

Input: n odd, z multiple of $\varphi(n)$
Output: a non trivial factor of n
$\mathcal{B}(n)$:

1: write $z = 2^s t$ 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: $f \leftarrow \gcd(y-1, n)$
12: **return** $(f, n/f)$

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

RSA Security

- key recovery is equivalent to factoring n
- decryption is the RSA problem
  (not known to be equivalent to factoring)
- knowing pk and sk in RSA implies factoring n

Conclusion

- **Euler φ 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 modulo n
- square root mod n when factorization of n is known
- checking primality
- finding a generator when group order is known
- computing order when factorization of group order is known

hard (maybe)

- factoring
- discrete logarithm (sometimes)
- square root mod n
- computing $\varphi(n)$
- 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 ϕ function**: formula, properties
- **Chinese Remainder Theorem**: how to use it
- **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 2018–19 ex2 (RSA with Carmichael numbers)
  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
  midterm exam 2017–18 ex1
- arithmetic modulo 99 991: midterm exam 2023–24 ex3
- square roots modulo a prime $p \equiv 5 \pmod 8$: midterm exam 2024–25 ex3

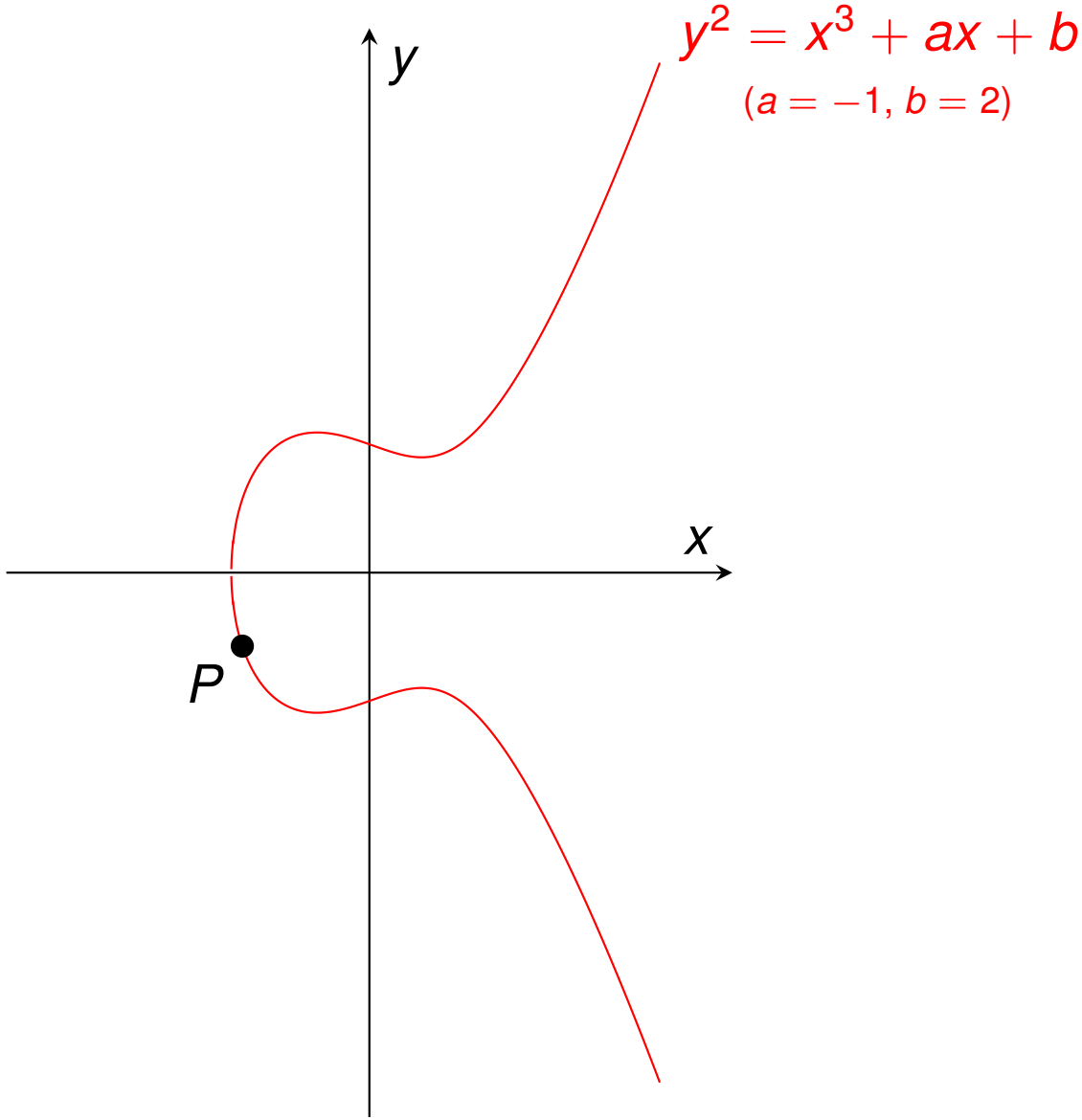
Roadmap

- elliptic curves over $\mathbf{Z}_p$
- using standard curves
- Diffie-Hellman over elliptic curves
- ElGamal over elliptic curves
- pairing-based cryptography

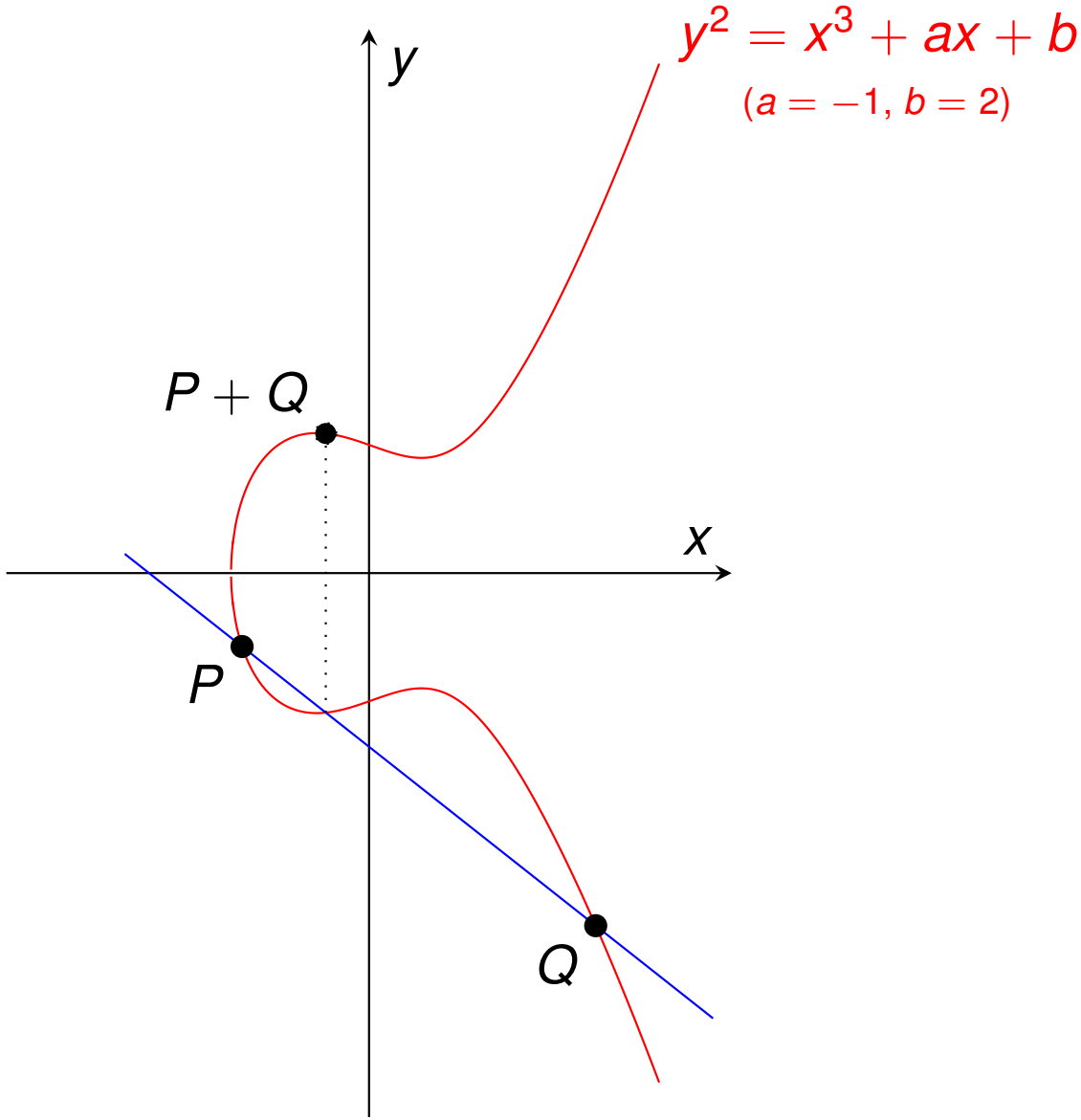
4 Elliptic Curve Cryptography

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

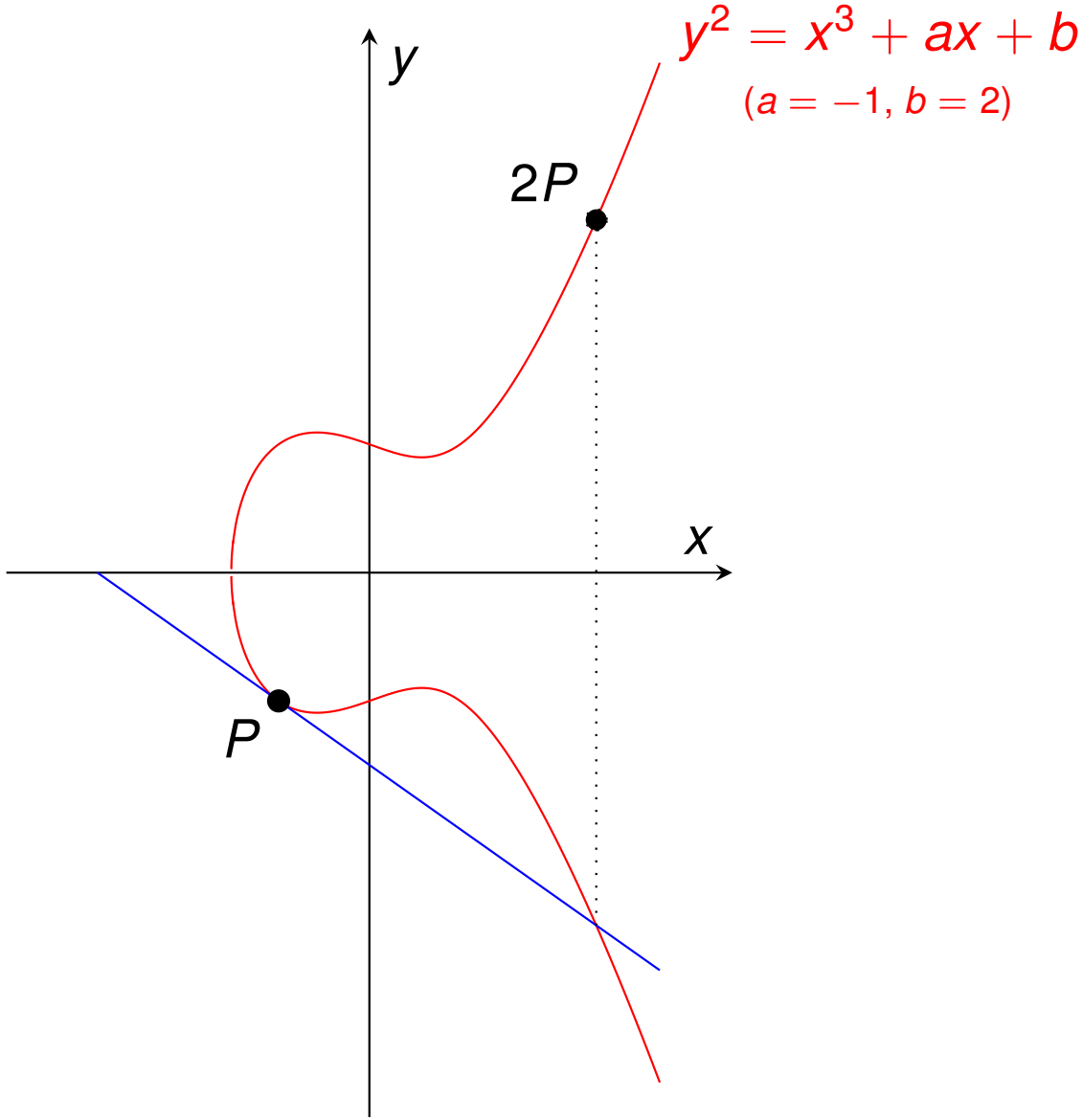
Elliptic Curves



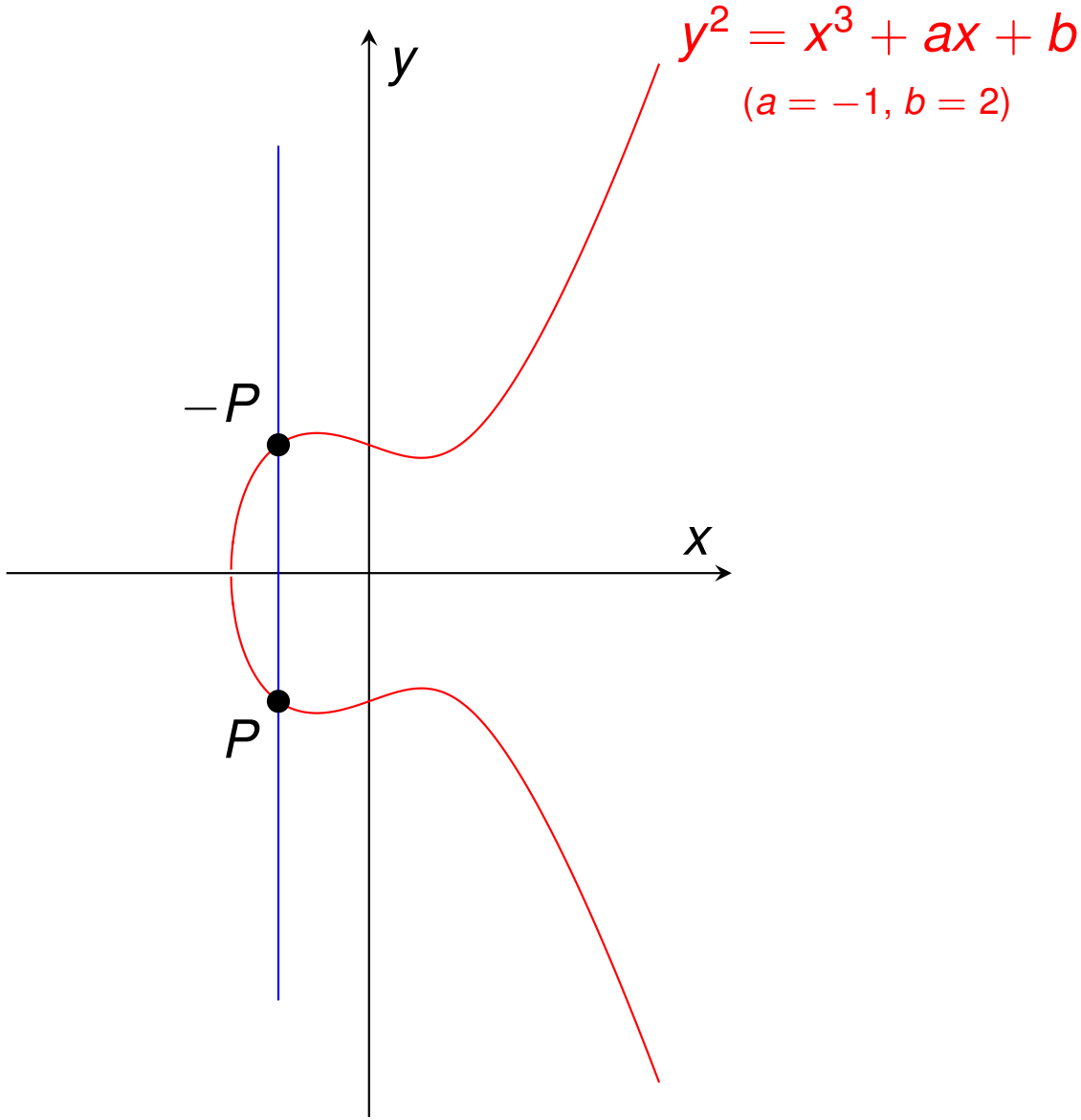
Elliptic Curves - Point Addition



Elliptic Curves - Point Doubling



Elliptic Curves - Point Symmetric



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 multiple root of $x^3 + ax + b = 0$
  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

$$\begin{aligned} \lambda &= \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } x_P \neq x_Q \\ \frac{3x_P^2 + a}{2y_P} & \text{if } x_P = x_Q \end{cases} \\ x_R &= \lambda^2 - x_P - x_Q \\ y_R &= (x_P - x_R)\lambda - y_P \end{aligned}$$

  $R = (x_R, y_R)$ 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: $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathrm{GF}(p^k)$)

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)

- order-2 points in elliptic curves:

$$\begin{aligned} P = (x, y) \text{ has order } 2 &\iff P = -P \text{ and } P \neq \mathcal{O} \\ &\iff y = 0 \text{ and } x^3 + ax + b = 0 \end{aligned}$$

  So, the number of points of order 2 is the number of roots of $x^3 + ax + b$ in $\mathbf{K}$

- order-2 elements in cyclic groups:
  being cyclic is equivalent to being isomorphic to some $\mathbf{Z}_n$
  in $\mathbf{Z}_n$, we have one (n even) or no (n odd) element of order 2
- conclusion:
  the group is not cyclic if $x^3 + ax + b$ has two distinct roots in $\mathbf{K}$

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 Δ)
- 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
  $\rightarrow$ can multiply a point by an integer
  $\rightarrow$ some curves can be isomorphic
  $\rightarrow$ contrarily to $\mathbf{Z}_p^*$, EC are not always cyclic
  (but we can work on a cyclic subgroup)

4 Elliptic Curve Cryptography

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- 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

$$\begin{aligned} \lambda &= \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } x_P \neq x_Q \\ \frac{3x_P^2 + a}{2y_P} & \text{if } x_P = x_Q \end{cases} \\ x_R &= \lambda^2 - x_P - x_Q \\ y_R &= (x_P - x_R)\lambda - y_P \end{aligned}$$

  $R = (x_R, y_R)$ 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 ($p > 3$) — i

Hypothesis: field **K** of characteristic $p > 3$ and $\Delta \neq 0$

- $E_{a,b}$ and E_{u^4a,u^6b} are isomorphic (by $(x, y) \mapsto (u^2x, u^3y)$)

$$y^2 = x^3 + ax + b \iff (u^3y)^2 = (u^2x)^3 + (u^4a)(u^2x) + (u^6b)$$

and addition is homomorphic:

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} \\ \frac{3x_P^2 + a}{2y_P} \end{cases} \iff (u\lambda) = \begin{cases} \frac{(u^3y_Q) - (u^3y_P)}{(u^2x_Q) - (u^2x_P)} \\ \frac{3(u^2x_P)^2 + (u^4a)}{2(u^3y_P)} \end{cases}$$

$$x_R = \lambda^2 - x_P - x_Q \iff (u^2x_R) = (u\lambda)^2 - (u^2x_P) - (u^2x_Q)$$

$$y_R = (x_P - x_R)\lambda - y_P \iff (u^3y_R) = ((u^2x_P) - (u^2x_R))(u\lambda) - (u^3y_P)$$

- $E_{a,b}$ and E_{v^2a,v^3b} are **twist** of each other if v is not a square
  Remark: they become isomorphic in $\mathbf{K}[\theta]/(\theta^2 - v)$: an extension of **K** where v becomes a square ($v = \theta^2$)

Maybe Useful to Know ($p > 3$) — ii

Hypothesis: field **K** of characteristic $p > 3$ and $\Delta \neq 0$

- $\#E_{a,b}$ is between $q + 1 - 2\sqrt{q}$ and $q + 1 + 2\sqrt{q}$ where q is the cardinality of **K** (Hasse Theorem)
  Remark: for two twists, the average of $\#E_{a,b}$ is $q + 1$
  indeed, if v is not a square, if we write $a' = v^2 a$, $b' = v^3 b$,
  $x' = vx$, for any x, we have

$$\#\{y; y^2 = x^3 + ax + b\} + \#\{y'; y'^2 = x'^3 + a'x' + b'\} = 2$$

  so $\#E_{a,b} + \#E_{a',b'} = 2q + 2$

Maybe Useful to Know ($p > 3$) — iii

Hypothesis: field **K** of characteristic $p > 3$ and $\Delta \neq 0$

- **j-invariant**: $j = 1728 \frac{4a^3}{4a^3+27b^2}$
  (case a and b nonzero)

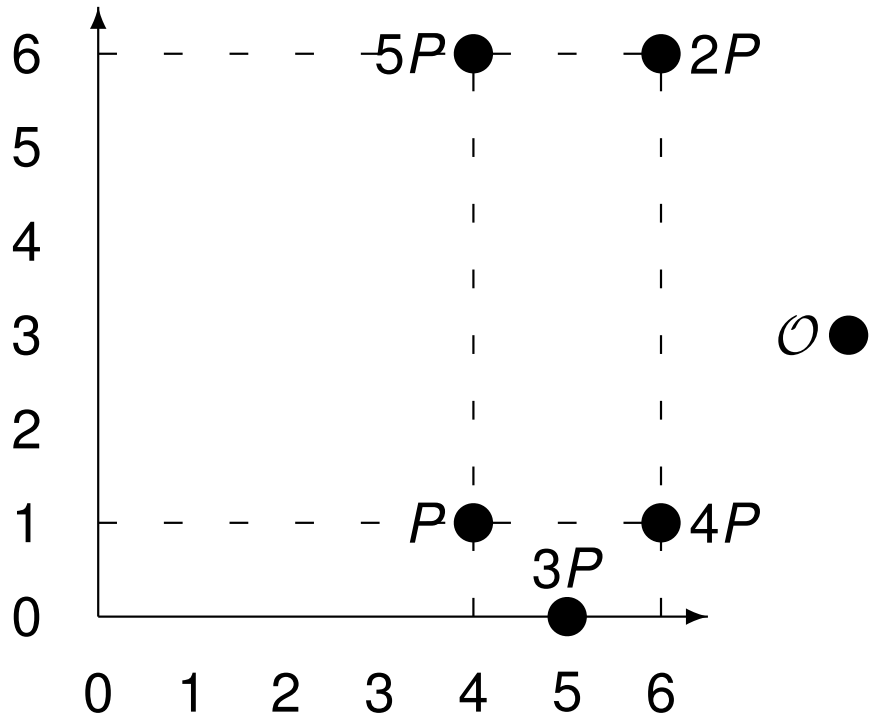
  same j-invariant $\iff$ same $a^3/b^2 \iff \exists v \quad a' = v^2 a, b' = v^3 b$
  $\implies$ isomorphic groups (over **K** or $\mathbf{K}[\theta]/(\theta^2 - v)$)

  (converse is true as well)
  to find v, write (Bezout) $3\alpha + 2\beta = 1$ ($\alpha = 1$, $\beta = -1$) then
  $v = \left(\frac{b'}{b}\right)^{\alpha} \left(\frac{a'}{a}\right)^{\beta}$
  we have: $v^2 = \left(\frac{b'}{b}\right)^{2\alpha} \left(\frac{a'}{a}\right)^{2\beta} = \left(\frac{a'}{a}\right)^{3\alpha} \left(\frac{a'}{a}\right)^{2\beta} = \frac{a'}{a}$ and
  $v^3 = \left(\frac{b'}{b}\right)^{3\alpha} \left(\frac{a'}{a}\right)^{3\beta} = \left(\frac{b'}{b}\right)^{3\alpha} \left(\frac{b'}{b}\right)^{2\beta} = \frac{b'}{b}$

Other Example

$E_{1,3}$ over $\mathbf{Z}_7$ 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

4 Elliptic Curve Cryptography

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

Pollard $p-1$ Factorization Algorithm

Input: n s.t. there exists a prime factor p of n s.t. $p-1$ is smooth (the largest prime fact r of $p-1$ is at most B)
Output: a nontrivial factor of n
Complexity: $\mathcal{O}(B)$ arithmetic operations

1: pick x at random in $\{2,\ldots,n-1\}$
2: **if** $\gcd(x,n) \neq 1$ **then**
3: output this gcd and stop
4: **end if**
5: $i \leftarrow 1$
6: **while** $\gcd(x-1,n) = 1$ **do**
7: $x \leftarrow x^i \bmod n$ ▷ $x_1^{i!} \bmod n$
8: $i \leftarrow i+1$
9: **end while**
10: **if** $x = 1$ **then**
11: fail
12: **else**
13: output $\gcd(x-1,n)$ and stop
14: **end if**

trick: if $p-1 | i!$ then $x_1^{i!} \equiv 1 \pmod{p}$

Pollard $p-1$ Factorization with $n = 18923$

initial $x = 2347$

$$
\begin{array}{lllllll}
i & = & 1 & x & = & 2347 & \text{gcd} \quad =1 \\
i & = & 2 & x & = & 1816 & \text{gcd} \quad =1 \\
i & = & 3 & x & = & 4072 & \text{gcd} \quad =1 \\
i & = & 4 & x & = & 14891 & \text{gcd} \quad =1 \\
i & = & 5 & x & = & 18431 & \text{gcd} \quad =1 \\
i & = & 6 & x & = & 7247 & \text{gcd} \quad =1 \\
i & = & 7 & x & = & 13590 & \text{gcd} \quad =127
\end{array}
$$

$\longrightarrow 18923 = 127 \times 149$
(Note that $p-1 = 2 \times 3^2 \times 7$ and $q-1 = 2^2 \times 37$)

Potential Problem

The algorithm is essentially doing:

1: pick $x \in \mathbf{Z}_p^*$
2: $i \leftarrow 1$
3: **while** $x \neq 1$ **do**
4: $\quad x \leftarrow x^i$ in $\mathbf{Z}_p^*$
5: $\quad i \leftarrow i + 1$
6: **end while**
7: deduce something about n

- computation in $\mathbf{Z}_p^*$ is done in $\mathbf{Z}_n^*$
- $x = 1$ test is done by $\gcd(x - 1, n)$

Wish: there is a factor p s.t. $\mathbf{Z}_p^*$ has a smooth order (so that #iterations is small)

If not, we would like to "randomize" the group $\mathbf{Z}_p^*$

ECM Factorization

- same algorithm as the $p-1$ algorithm, but on a "random elliptic curve" over $\mathbf{Z}_p$ instead of $\mathbf{Z}_p^*$
- we use the probability that an elliptic curve over $\mathbf{Z}_p$ has a smooth order
- Complexity: $\mathcal{O}\left(e^{\sqrt{(1+o(1))\log p \log\log p}}\right)$
- pretty good to find a small factor p!

ECM Factorization with $n = 44023$

pick $a = 13$ and $X = (23482, 9274)$, deduce $b = 21375$ from $y^2 \equiv x^3 + ax + b$

$$\begin{aligned} X_1 &= 1.X = (23482, 9274) \\ X_2 &= 2.X_1 = (18935, 21838) \\ X_3 &= 3.X_2 = (2.X_2) + X_2 = (15187, 29168) \\ X_4 &= 4.X_3 = 2.(2.X_3) = (10532, 5412) \\ X_5 &= 5.X_4 = (2.(2.X_4)) + X_4 = \quad \ldots \quad \text{error} \end{aligned}$$

$(2.(2.X_4)) + X_4 = (27556, 42335) + (10532, 5412)$, but this requires computing

$$\lambda = \frac{42335 - 5412}{27556 - 10532} \bmod n$$

and $27556 - 10532 = 17024$ is not invertible modulo n:
$\gcd(17024, n) = 133... \longrightarrow n = 133 \cdot 331$

ECM Factorization Algorithm

Input: n
Output: a nontrivial factor p of n
Complexity: $\mathcal{O}(B)$ arithmetic operations where

$$B \approx \underset{N \in [p-2\sqrt{p}, p+2\sqrt{p}]}{E} (\max\{\text{prime factors of } N\})$$

1: pick a and $X = (x, y)$ at random in $\mathbf{Z}_n$
2: let b such that $y^2 \equiv x^3 + ax + b \pmod{n}$
3: $i \leftarrow 1$
4: **repeat**
5: $\quad i \leftarrow i + 1$
6: $\quad X \leftarrow i \cdot X$ over the curve (modulo n)
7: **until** division error modulo n
8: if divisor multiple of n then fail
9: output gcd(divisor, n)

4 Elliptic Curve Cryptography

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- 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 $\mathcal{O}(n^{\frac{1}{3}})$
  (people tend to use the very same curves...)
- in general, DL is harder than in $\mathbf{Z}_p^*$ with similar size

Note: there are curves with easy DH problem and hard DL which may be useful (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.

$\rightarrow$ *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)

Manipulating Elliptic Curves in Practice

A representation problem:

- bit strings
- byte strings
- integers
- 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 GF(2) of degree $\log_2 q$)
- field elements defining an elliptic curve E (coefficients)
- 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)

ECDSA Parameters Generation

(ECDSA to be seen later)

1. Choose the finite field $\mathbf{Z}_p$.
2. Pseudo-randomly generate a c from seed. Take an elliptic curve defined by some a and b such that the j-invariant is $j = 1728\frac{4c}{4c+27}$ (i.e. $c = a^3/b^2$).
3. Check that $4a^3 + 27b^2 \bmod p \neq 0$. 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 $n \leq 4\sqrt{p}$, 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 $(p, a, b, n, G, \text{seed})$.

ECDSA Parameters Validation

Parameters: $(p, a, b, n, G, \text{seed})$.

1. Check that p is an odd prime of appropriate size.
2. Check that a, b, x_G, y_G (where $G = (x_G, y_G)$) lie in $\mathbf{Z}_p$.
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. Check that $4a^3 + 27b^2 \bmod p \neq 0$. Check that G lies in the elliptic curve. Check that n is a prime greater than both 2^{160} and $4\sqrt{p}$. 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

Standard Curves

- pseudorandom curves over $\mathbf{Z}_p$
  - $y^2 = x^3 + ax + b$
  - provide seed to generate j

  $\rightarrow$ 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 (2010)

SEC2: Recommended Elliptic Curve Domain Parameters
`http://www.secg.org/sec2-v2.pdf`

- pseudorandom over $\mathbf{Z}_p$: secp192r1, secp224r1, secp256r1, secp384r1, secp521r1
- special over $\mathbf{Z}_p$: secp192k1, secp224k1, secp256k1 (called **generalized Koblitz curves**)

Other Standards

- ANSI X9.62
- IEEE P1363

Example: secp192r1 = P192

$$\text{secp192r1} = \{\mathcal{O}\} \cup \{(x, y) \in \mathbf{Z}_p; y^2 = x^3 + ax + b\}$$

$$\begin{aligned}
p &= 6277101735386680763835789423207666416083908700390324961279 \\
&= \texttt{ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff} \\
a &= p - 3 \\
&= 6277101735386680763835789423207666416083908700390324961276 \\
&= \texttt{ffffffff ffffffff ffffffff fffffffe ffffffff fffffffc} \\
b &= 2455155546008943817740293915197451784769108058161191238065 \\
n &= 6277101735386680763835789423176059013767194773182842284081 \\
G &= \texttt{03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012} \\
&= 03 : 602046282375688656758213480587526111916698976636884684818 \\
\text{seed} &= \texttt{3045ae6f c8422f64 ed579528 d38120ea e12196d5}
\end{aligned}$$

note that $p = 2^{192} - 2^{64} - 1$, $2^{192} - 2^{95} < n < 2^{192}$, 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

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

Elliptic Curve Cryptography

- **key agreement**: ECDH
- **digital signature scheme**: ECDSA
- **public-key cryptosystem**: ECIES

ECDH: Elliptic Curve Diffie-Hellman

- specified in SEC1 (`http://www.secg.org/sec1-v2.pdf`) and IEEE1363
- used in Bluetooth 2.1

▸ in Bluetooth slide 1057

- used in the PKI of Swiss biometric passports

▸ see slide 369

- used in EAC for epassports

▸ see EAC slide 1094

ECDH

$\mathrm{GF}(2^m) = \mathbf{Z}_2[x]/(f(x))$

Participants: U and V

- U and V agree on domain parameters $T = (p, a, b, G, n, h)$ or $T = (m, f(x), a, b, G, n, h)$ + make sure that T is valid
  (h is the cofactor $\frac{1}{n}\#E(\mathrm{GF}(q))$ 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 $Q_U = d_U.G$ resp. $Q_V = d_V.G$
- U and V exchange their public keys
- both check $Q \in E(\mathrm{GF}(p))$, $Q \neq \mathcal{O}$, $n.Q = \mathcal{O}$
- both compute $P = d_U.Q_V = d_V.Q_U$
- set $z = x_P$
- convert the field element z into a byte string Z
- use a KDF as agreed to derive a key K

U		V
pick $d_U \in \mathbf{Z}_n^*$, $Q_U \leftarrow d_U.G$	$\xrightarrow{Q_U}$	if $Q_U = \mathcal{O}$ or $n.Q_U \neq \mathcal{O}$, abort
if $Q_V = \mathcal{O}$ or $n.Q_V \neq \mathcal{O}$, abort	$\xleftarrow{Q_V}$	pick $d_V \in \mathbf{Z}_n^*$, $Q_V \leftarrow d_V.G$
$Z \leftarrow \mathrm{xcoord}(d_U.Q_V)$		$Z \leftarrow \mathrm{xcoord}(d_V.Q_U)$
	$(Z = \mathrm{xcoord}(d_U d_V.G))$	

Checking Subgroup Membership

Lemma

Let G be an element of order n in a group of order hn with neutral element $\mathcal{O}$. If n is prime and is coprime with h, then

$$\langle G \rangle = \{Q \in \text{group}; n.Q = \mathcal{O}\}$$

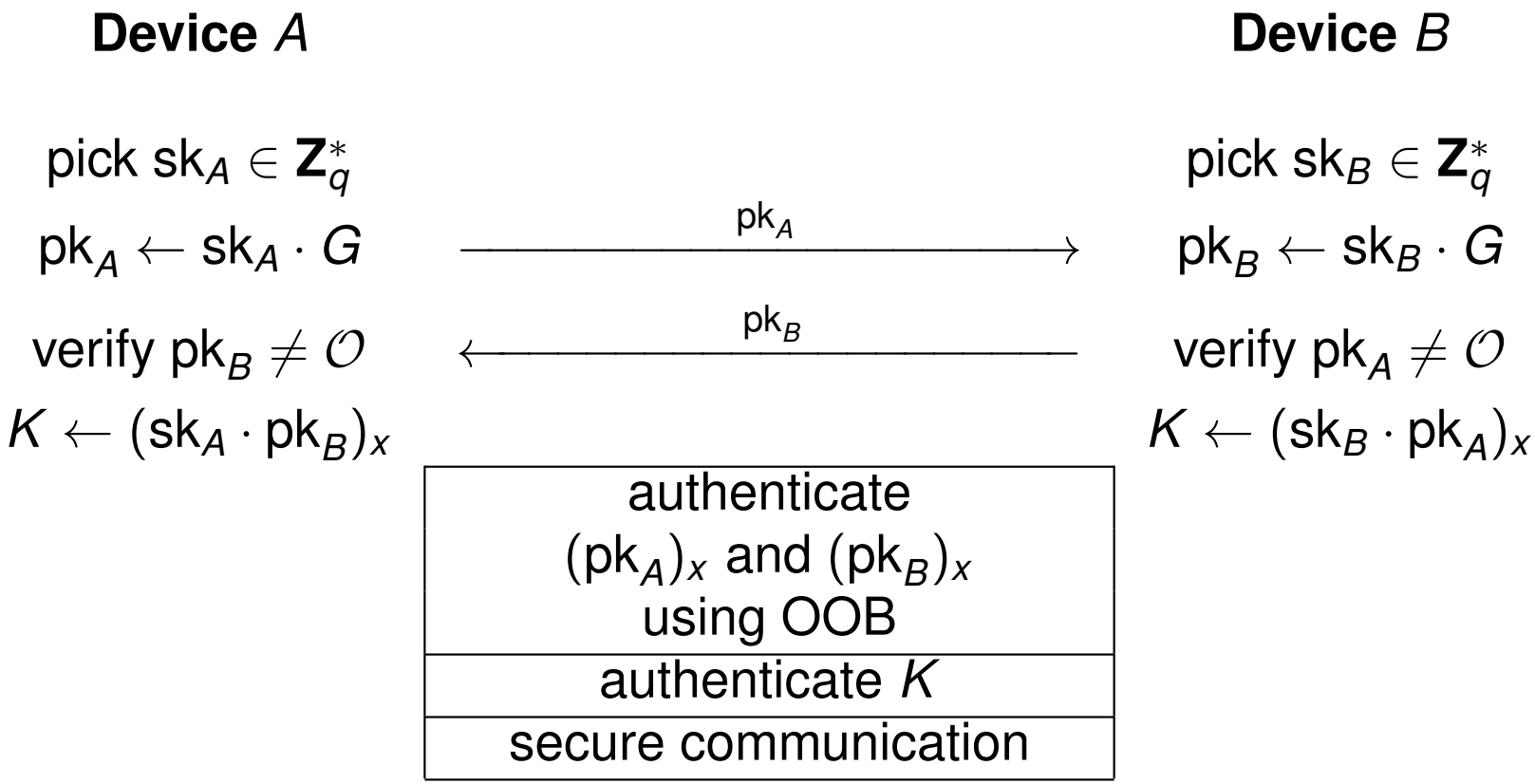
Proof. $\subseteq$ is trivial
for $\supseteq$:

- assume that $Q \in$ group and $nQ = \mathcal{O}$
- the mapping $f : \mathbf{Z}_n^2 \to$ group defined by $f(u, v) = uG + vQ$ is a group homomophism
- f injective would imply that $\mathbf{Z}_n^2$ is isomorphic to $f(\mathbf{Z}_n^2)$ which is a subgroup of group
- since $\#\mathbf{Z}_n^2$ does not divide #group, this cannot be injective
- so, there exists a nonzero $(u, v) \in \mathbf{Z}_n^2$ such that $uG + vQ = \mathcal{O}$
- we must have $v \neq 0$ since G has order n, so $v \in \mathbf{Z}_n^*$ and $Q = (-uv^{-1} \bmod n).G$
- hence, $Q \in \langle G \rangle$

□

Problems without Membership Verification

The Case of Secure Simple Pairing (SSP) in Bluetooth

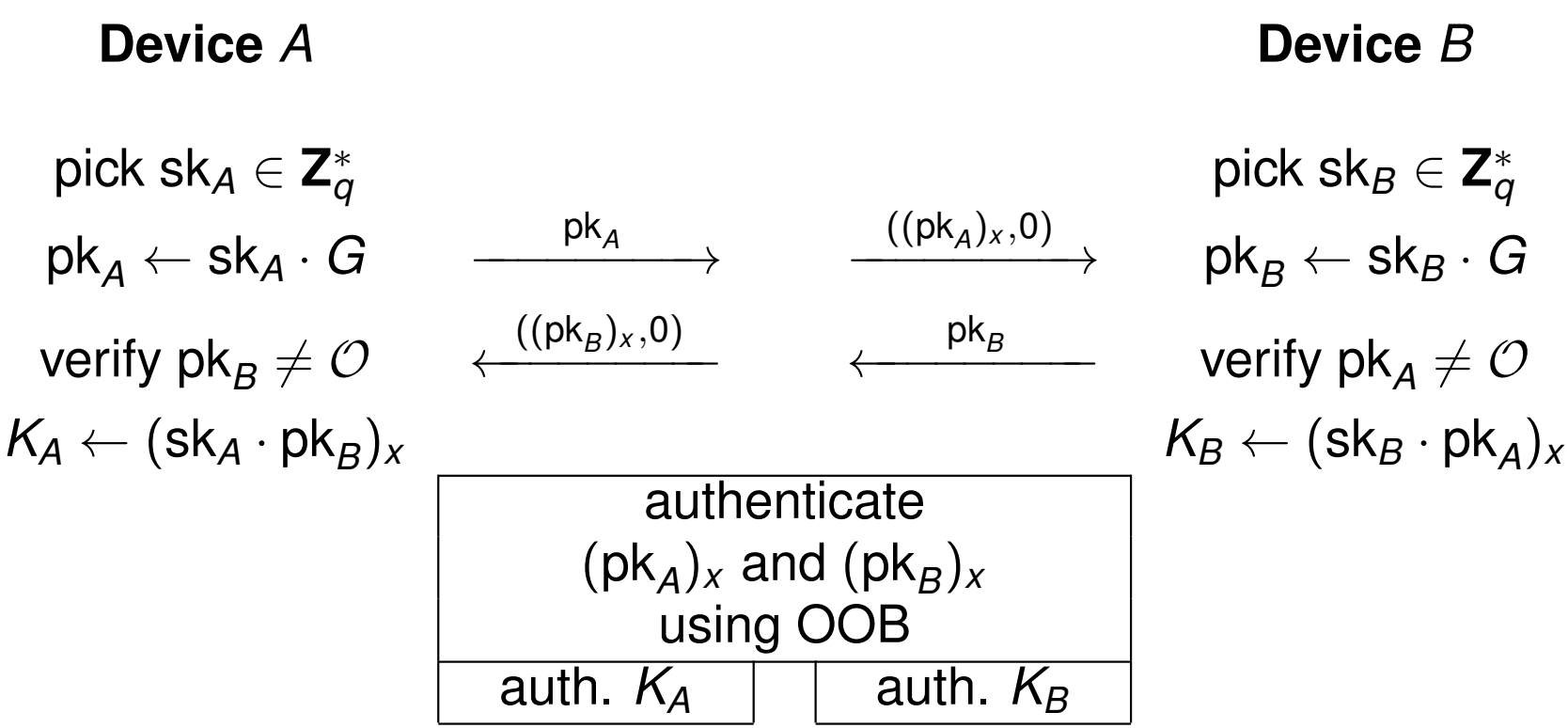


scalar multiplication is done by the double-and-add algorithm, using point addition

The Invalid Curve Attack on Bluetooth

Biham-Neumann; Breaking the BT Pairing: Fixed Coordinate Invalid Curve Attack

- note: point addition does not depend on the curve equation
- note: only the x-coordinates are authenticated
- the adversary can replace y to make pk have degree 2
- with probability $\frac{1}{4}$...

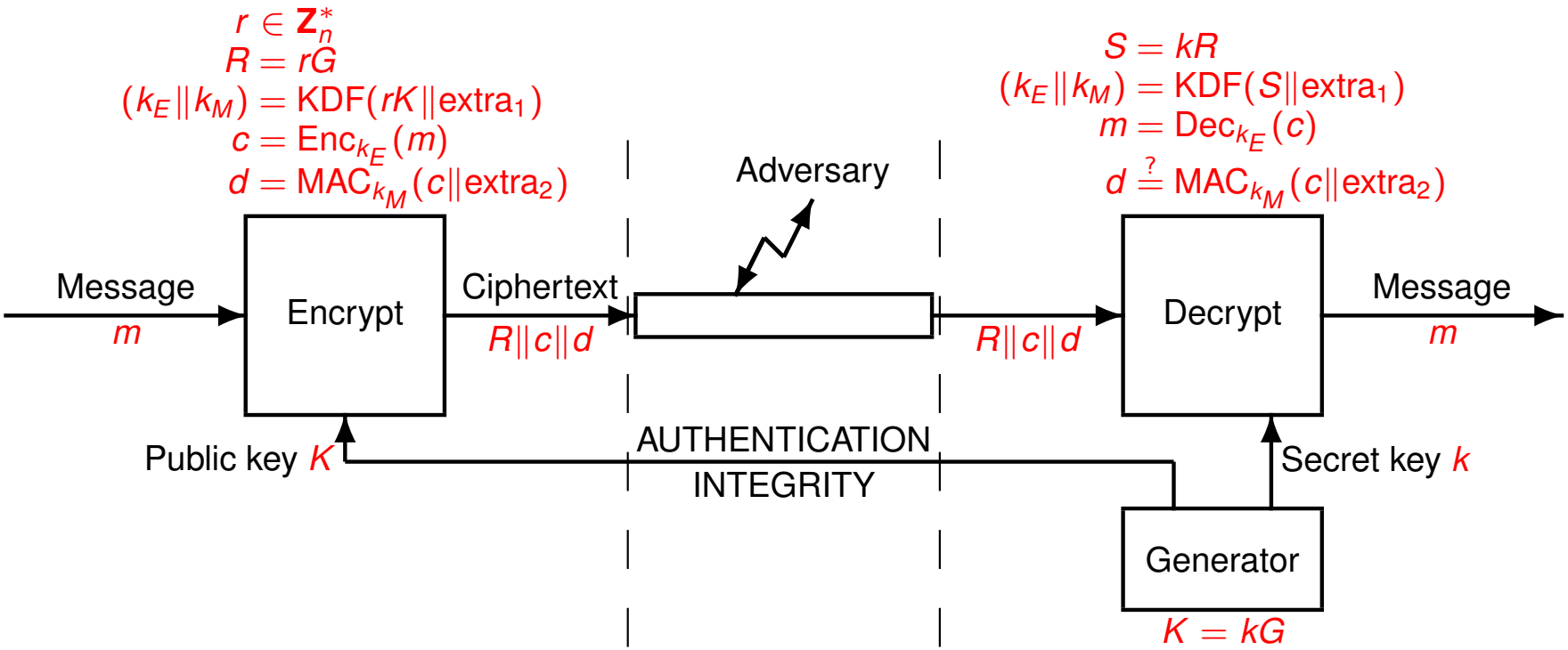


Principles of ECIES

- proposed by Victor Shoup in 2001
- in SEC1, IEEE1363a, ANSI X9.63, ISO/IEC 18033-2
- 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

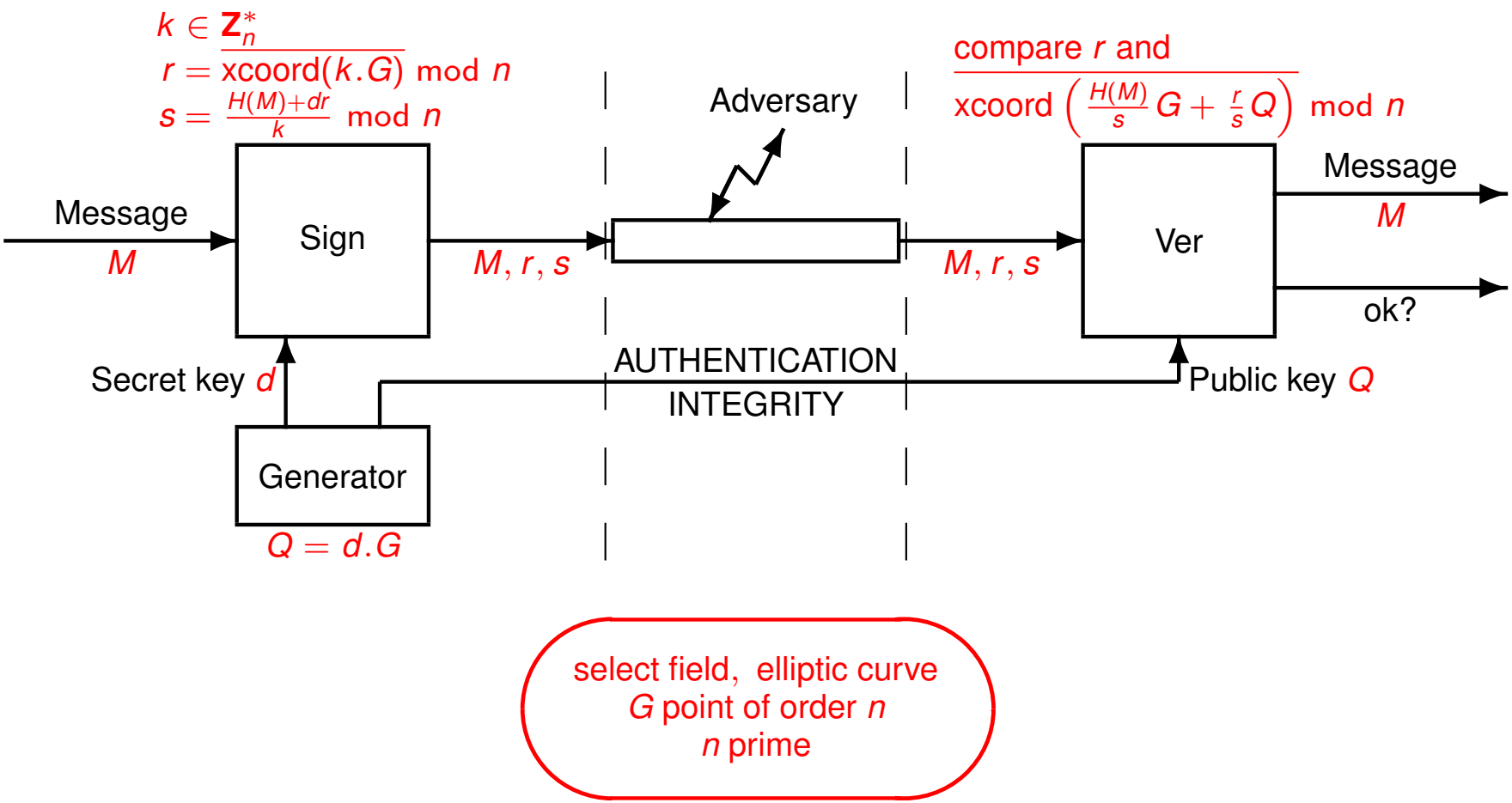
ECIES (EC Integrated Encryption Scheme)



select field, elliptic curve
G point of order n
n prime

extra is context-based information (public)

ECDSA Signature



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...

4 Elliptic Curve Cryptography

- Elliptic Curves
- Elliptic Curves over a Prime Field
- Elliptic Curve and Factoring
- Using Elliptic Curves
- Elliptic Curve Cryptography
- Pairing-Based Cryptography

Pairing of Elliptic Curves

for some pairs of elliptic curves $\mathcal{G}_1$ and $\mathcal{G}_2$ we can construct a function

$$e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$$

to a group $\mathcal{G}_T$ (with multiplicative notations) such that

- e is **bilinear**: $e(aP, bQ) = e(P, Q)^{ab}$ for $a, b \in \mathbf{Z}$, $P \in \mathcal{G}_1$, $Q \in \mathcal{G}_2$
  $\to e(P + P', Q) = e(P, Q)e(P', Q)$
  $\to e(P, Q + Q') = e(P, Q)e(P, Q')$
- e is **non-degenerate**: $e(P, Q) \neq 1$ for some $P \in \mathcal{G}_1$ and $Q \in \mathcal{G}_2$

(pairing is not available for all curves)

Types of Pairing

- **Type-1 pairing**: we have $\mathcal{G}_1 = \mathcal{G}_2$
  (common on supersingular elliptic curves)
- **Type-2 pairing**: we have $\mathcal{G}_1 \neq \mathcal{G}_2$ and there exists an efficiently computable (non-degenerate) homomorphism from $\mathcal{G}_2$ to $\mathcal{G}_1$
- **Type-3 pairing**: we have $\mathcal{G}_1 \neq \mathcal{G}_2$ and there exists no efficiently computable (non-degenerate) homomorphism between $\mathcal{G}_1$ and $\mathcal{G}_2$
- **Type-4 pairing**: we have same as Type-2 with efficient hashing into $\mathcal{G}_2$
  (those pairings are usually not efficient)

Type-1 and Type-3 are most common

Pairing-Friendly Elliptic Curves

- q: cardinality of the field
- r: (**large prime**) order of G
- such that there exists a **small** k such that r divides $q^k - 1$ (embedding degree)
- μ_r subgroup of all $z \in \mathrm{GF}(q^k)$ such that $z^r = 1$
- we have Type-1 pairing with $\mathcal{G}_1 = \mathcal{G}_2 = \langle G \rangle$ and $\mathcal{G}_T = \mu_r$

Pairing of Elliptic Curves

consequences:

- this may be bad for DDH-security in $\mathcal{G}_1 = \mathcal{G}_2$ as we can distinguish (P, xP, yP, xyP) from (P, xP, yP, zP) by checking $e(xP, yP) = e(P, xyP)$
  we call $\mathcal{G}_1 = \mathcal{G}_2$ a **gap group** because the *computational Diffie-Hellman problem* may remain hard even though the *decisional Diffie-Hellman problem* is easy
- this may be bad for DL-security in $\mathcal{G}_1 = \mathcal{G}_2$
  DL in $\mathcal{G}_1$ reduces to DL in $\mathcal{G}_T$ (MOV): $\log_g(h) = \log_{e(g,g)}(e(g, h))$
- good thing: 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 $\mathcal{G}_1 = \mathcal{G}_2$ such that $e(G, G) \neq 1$

- Alice picks $a \in \mathbf{Z}_p^*$ and broacasts $A = aG$
- Bob picks $b \in \mathbf{Z}_p^*$ and broacasts $B = bG$
- Charly picks $c \in \mathbf{Z}_p^*$ and broacasts $C = cG$
- 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

slide 952

- Boneh-Lynn-Shacham 2003: a signature scheme (short)

slide 772

- Boneh-Boyen 2004: a signature scheme (with no H)

slide 773

- Sahai-Water 2004: attribute-based encryption
  secret keys have attributes (e.g. membership)
  we can encrypt for sets of people who own some given attributes

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
- ECDSA: final exam 2016–17 ex1
- pairing: midterm exam 2016 ex3
- invalid curve attack: midterm exam 2018–19 ex3

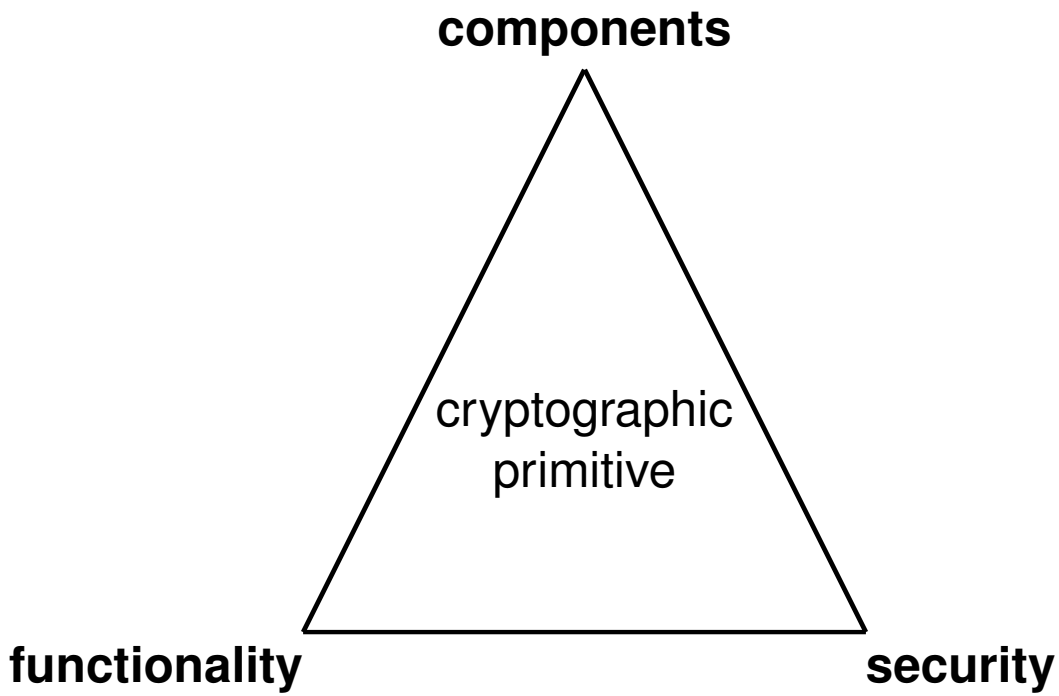
Roadmap

- Galois fields
- block ciphers: DES, triple-DES, AES
- modes of operations: ECB, CBC, OFB, CTR, XTS
- stream ciphers: RC4, A5/1
- exhaustive search and tradeoffs
- meet-in-the-middle attack

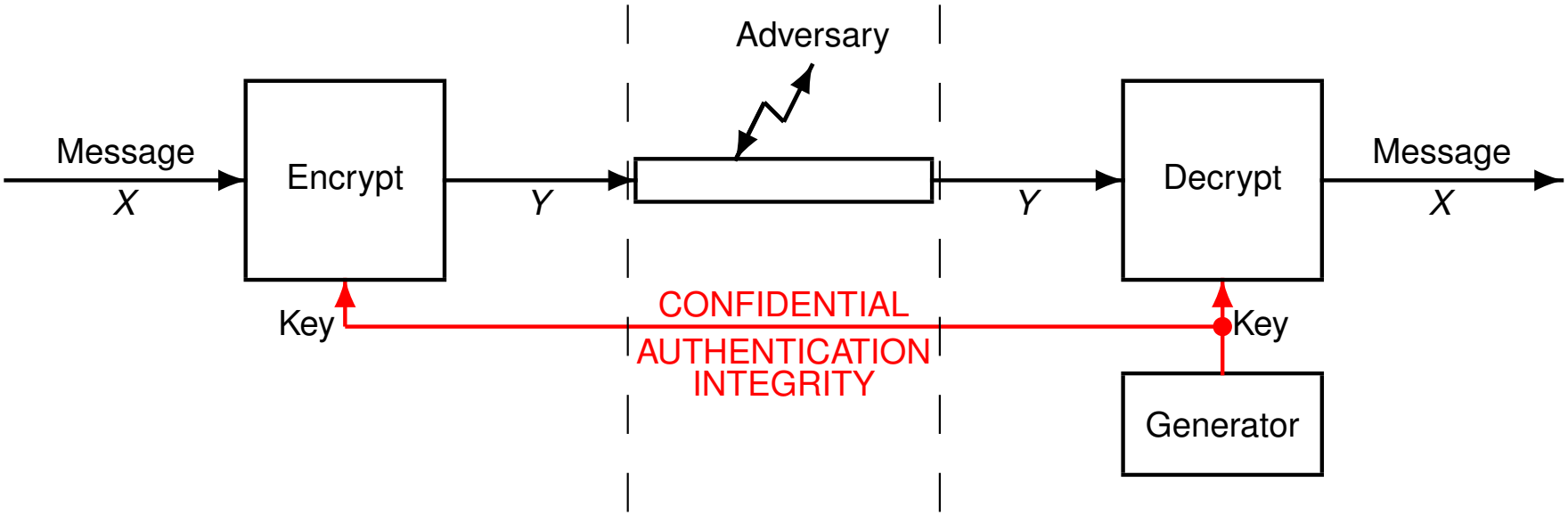
5 Symmetric Encryption

- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

Cryptographic Primitive (Reminder)



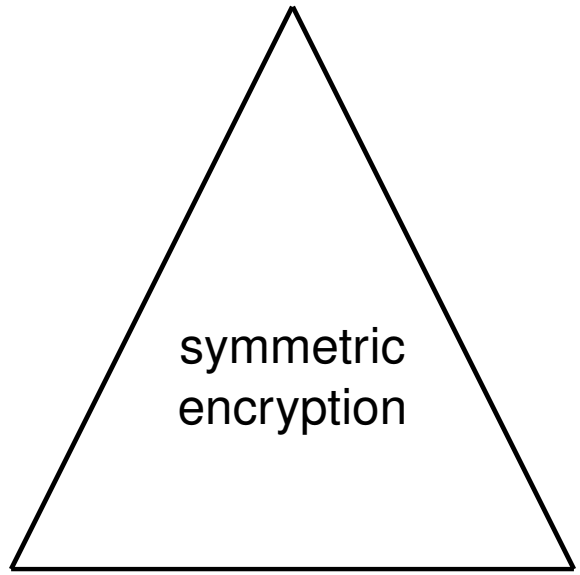
Symmetric Encryption



Symmetric Encryption (Informal)

Alice and Bob, Generator, Encrypt, Decrypt

components



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
RC4	**DES**
GSM–A5/1	**3DES**
Bluetooth–E0	IDEA
CSS	BLOWFISH
...	RC5
	AES
	KASUMI
	SAFER
	CS-Cipher
	FOX
	...

5 Symmetric Encryption

- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

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 $\deg(R) < \deg(P)$.
  $R(x) = A(x) \bmod P(x)$ is the remainder of $A(x)$ in the division by $P(x)$.
- 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 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 that 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 (mod 2) polynomial $P(x) = x^3 + x + 1$ of degree 3
- construct

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

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 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 $\mathrm{GF}(p^k) = \mathbf{Z}_p[x]/(P(x))$ 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* $\mathrm{GF}(p^k)$ *as* **Galois field** *of cardinality p^k.*
- $\mathrm{GF}(p^k)$ *is isomorphic to a subfield of* $\mathrm{GF}(p^{k\times\ell})$.
- $\mathrm{GF}(p^k)$ *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)

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

$+$	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

$\times$	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

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

Example: GF(4)

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

$+$	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

$\times$	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

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

$P(x) = x^2 + x + 1$ irreducible in $\mathbf{Z}_2[x]$, $\mathrm{GF}(4) = \mathbf{Z}_2[x]/(P(x))$

Example: GF(2^8)

Arithmetics in AES

A byte $a = a_7 \ldots a_1 a_0$ represents an element of the finite field GF(2^8) as a polynomial $a_0 + a_1.x + \ldots + a_7.x^7$ modulo $x^8 + x^4 + x^3 + x + 1$ and modulo 2

byte	polynomial
0x00	0
0x01	1
0x02	x
0x03	$x + 1$
0x1b	$x^4 + x^3 + x + 1$

The set of all bytes is given a field structure, where we can add and multiply.

Most Important Finite Fields

- "prime field": $\mathbf{Z}_p$ for a large prime p
- "binary field": GF(2^k)

	$\mathbf{Z}_p$	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

Characteristic 2 Tips

In 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: $\mathrm{Tr}(a) = a + a^2 + a^{2^2} + \cdots + a^{2^{k-1}} \in \{0, 1\}$
  (traces are roots of $z^2 = z$)
  Fact: Tr is linear: $\mathrm{Tr}(a + b) = \mathrm{Tr}(a) + \mathrm{Tr}(b)$
  Fact: for all a in GF(2^k) we have $\mathrm{Tr}(a^2) = \mathrm{Tr}(a)$

5 Symmetric Encryption

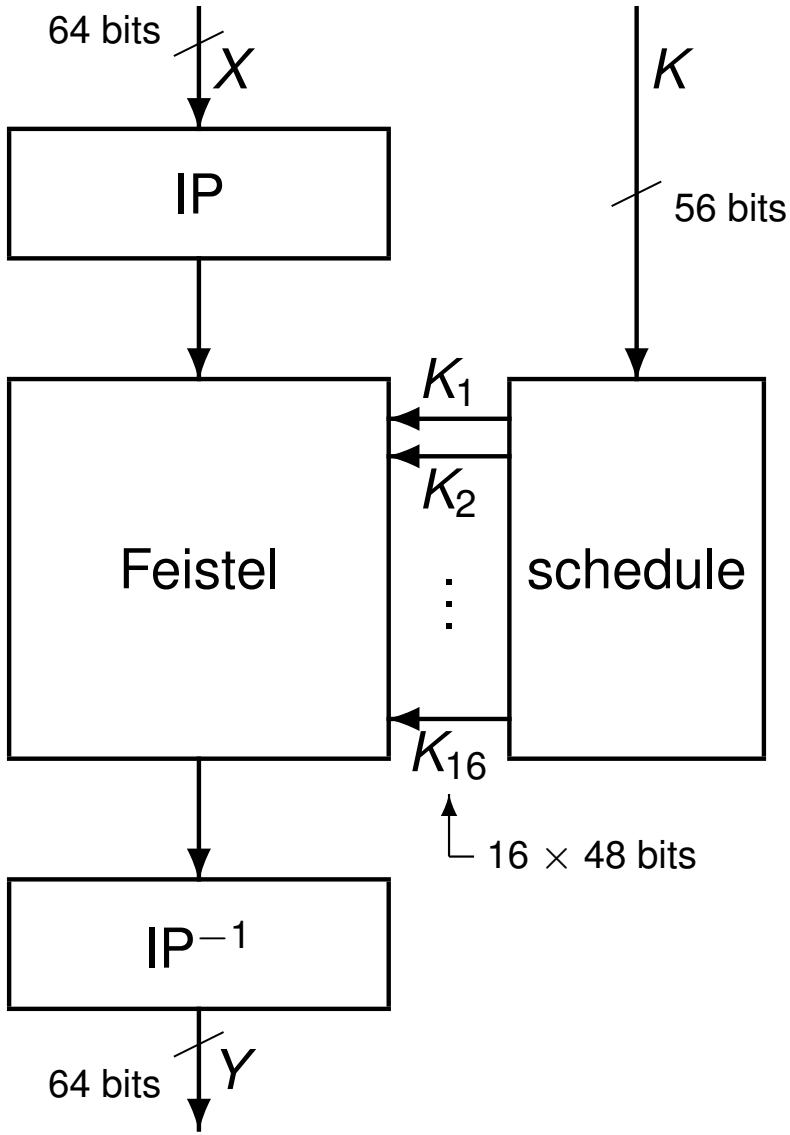
- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

DES: the Data Encryption Standard

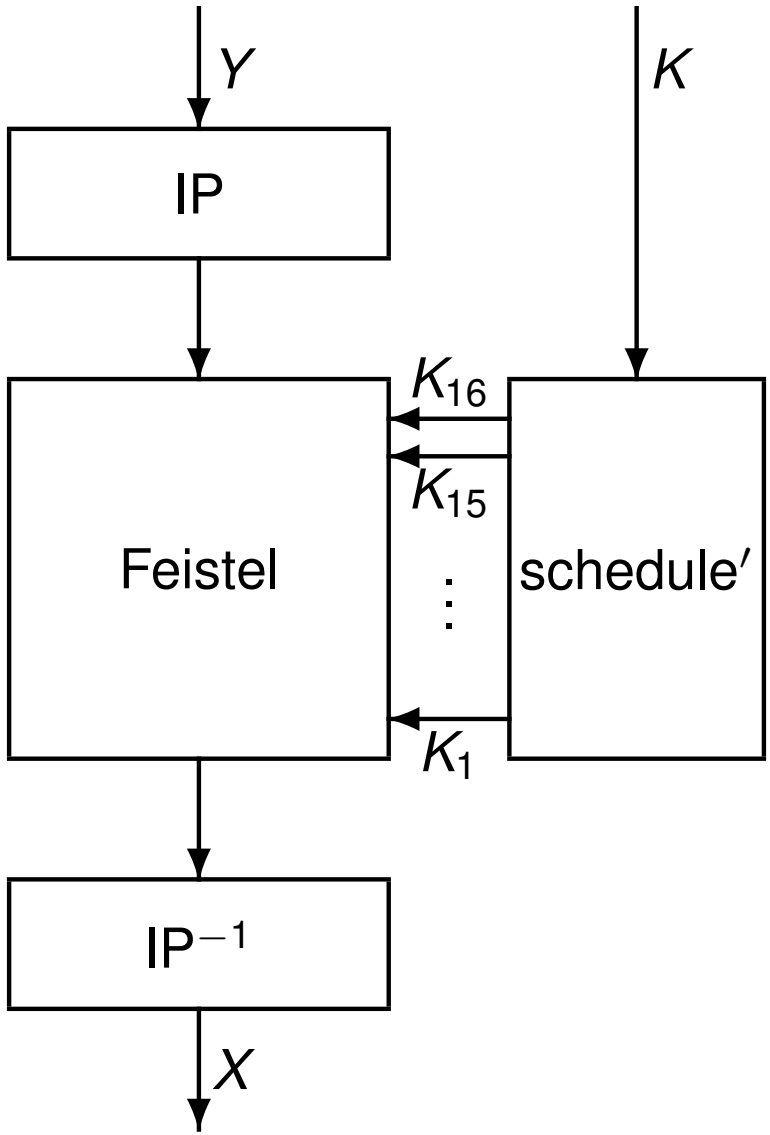
- US Standard from NBS (now NIST), branch of the Department of Commerce in 1977
- secret design by IBM based on a call for proposal
- based on LUCIFER by Horst Feistel (from IBM)
- design influenced by the NSA
- rationales of the design published by Don Coppersmith in 1994

- dedicated to hardware implementation
- block cipher with 64-bit blocks
- key of 56 effective bits

DES


64 bits
X
IP
K
56 bits
K1
K2
Feistel
schedule
K16
16 × 48 bits
IP−1
64 bits
Y

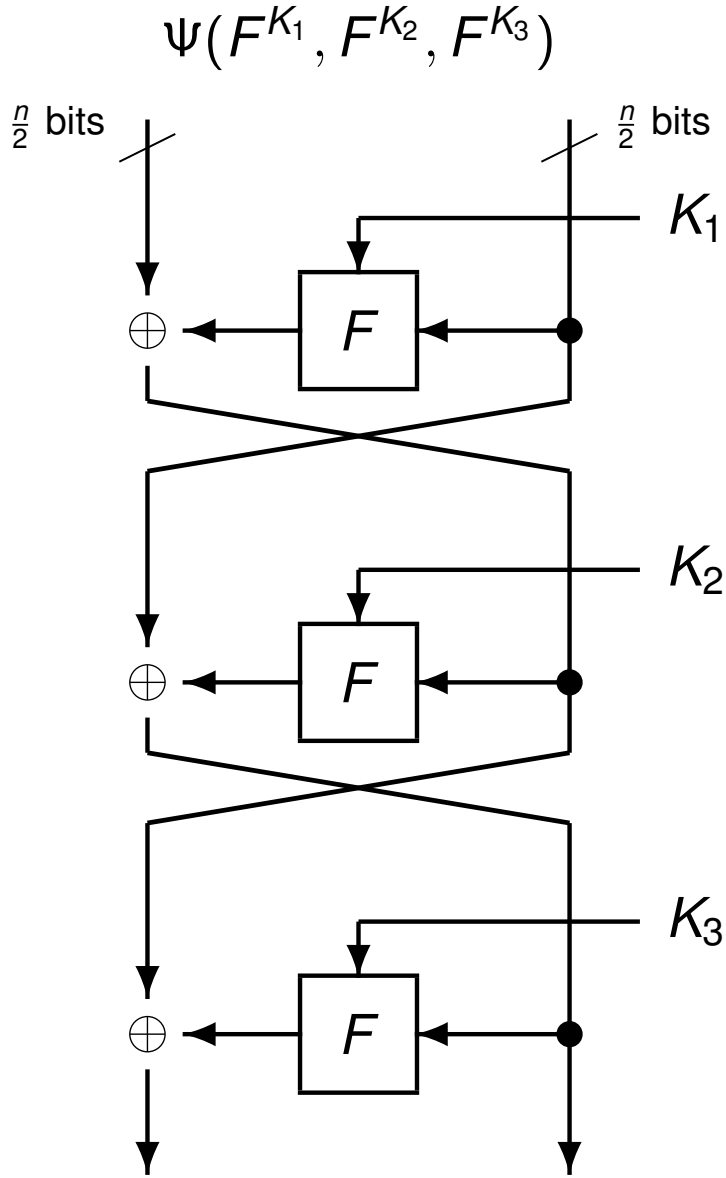
$\mathbf{DES}^{-1}$



Feistel Scheme

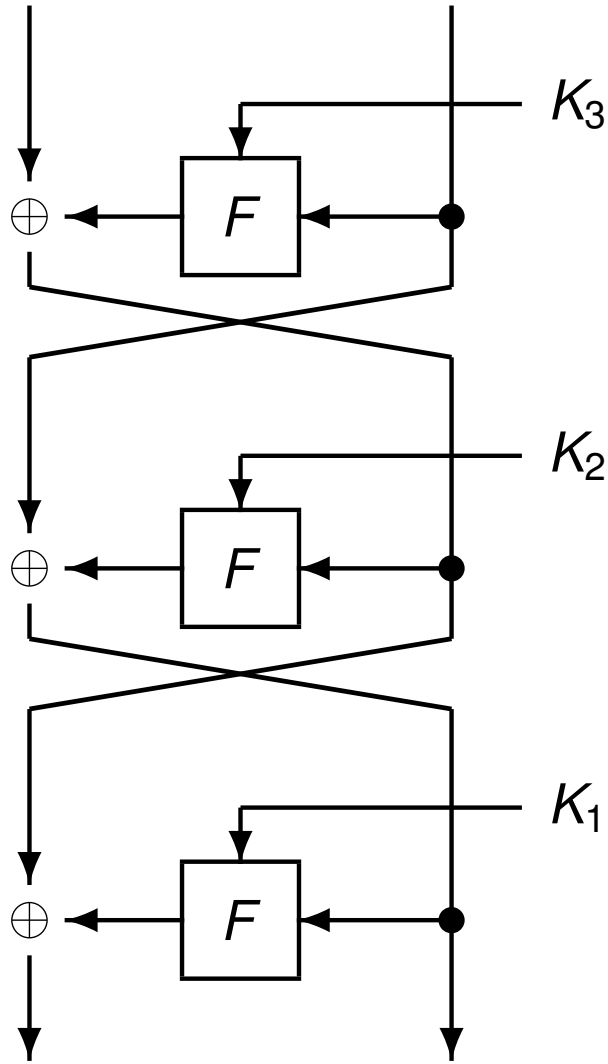
- transform function over $\{0,1\}^{\frac{n}{2}}$ into permutations over $\{0,1\}^n$
- inverse permutations have same structure
- alternate round functions and halve swaps
- final halve swap omitted

(Direct) Feistel Scheme

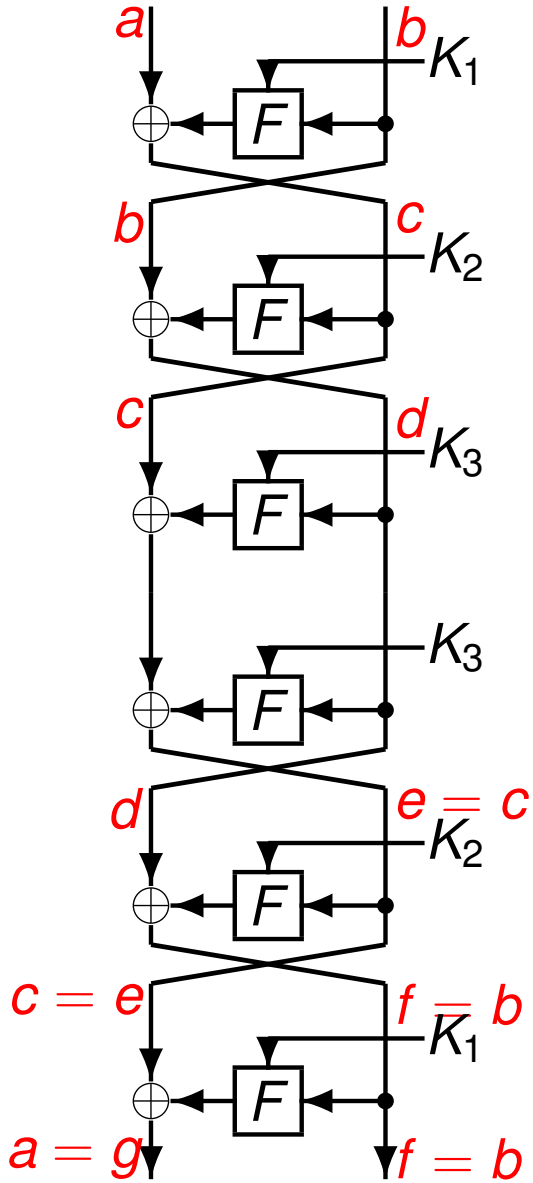


(Inverse) Feistel Scheme

$$\Psi^{-1}(F^{K_1}, F^{K_2}, F^{K_3}) = \Psi(F^{K_3}, F^{K_2}, F^{K_1})$$



(Direct + Inverse) Feistel Scheme

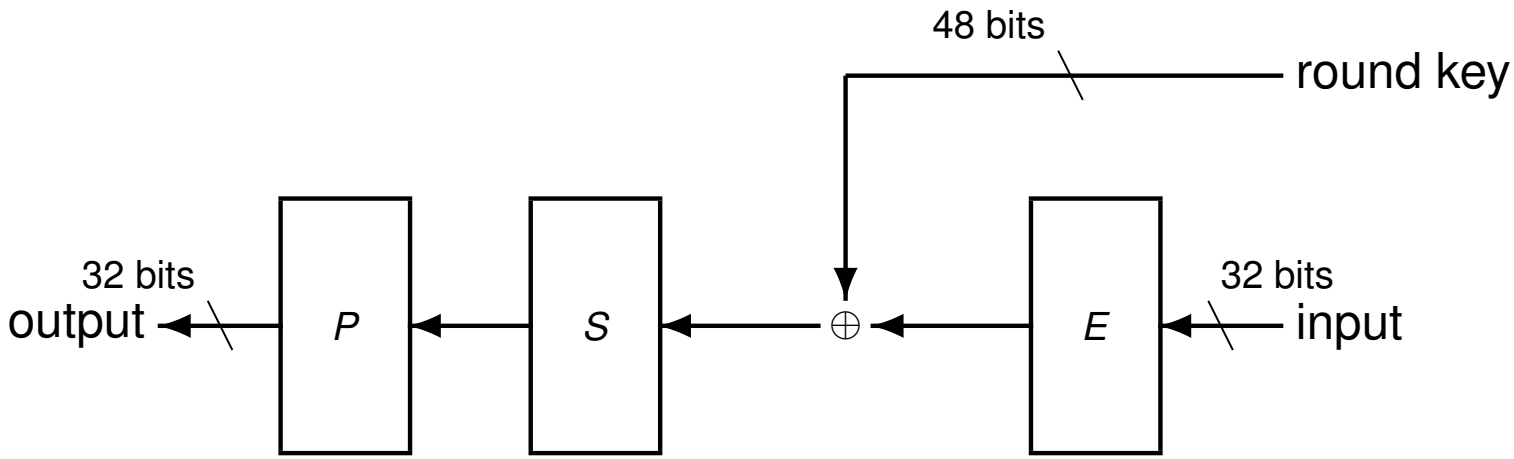


- $e = c \oplus F^{K_3}(d) \oplus F^{K_3}(d) = c$
- $f = d \oplus F^{K_2}(e) =$
  $(b \oplus F^{K_2}(c)) \oplus F^{K_2}(c) = b$
- $g = e \oplus F^{K_1}(f) = c \oplus F^{K_1}(b) =$
  $(a \oplus F^{K_1}(b)) \oplus F^{K_1}(b) = a$

DES: the Gory Details

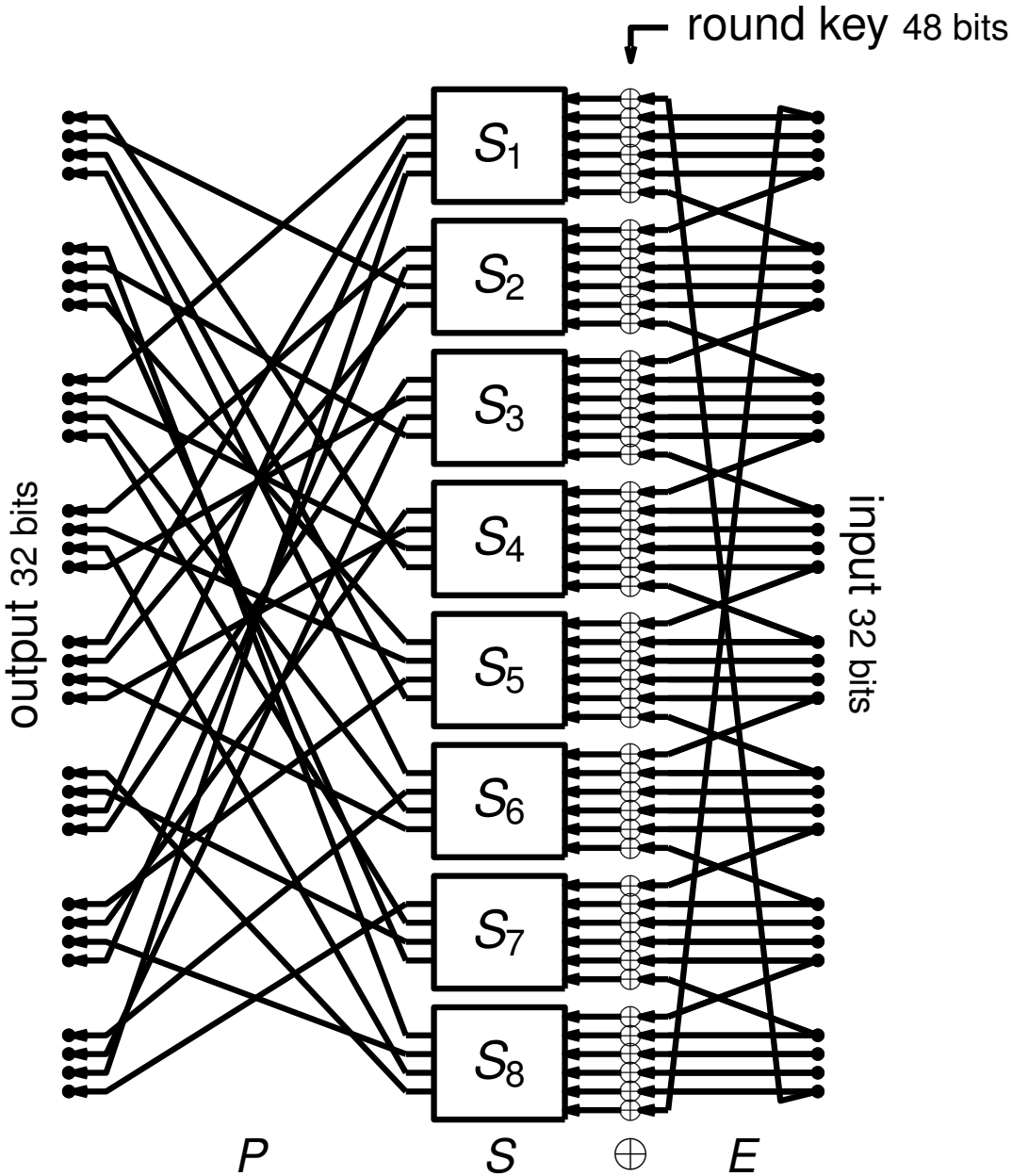
skip

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



S_3

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

Example: $S_3(111000) = 0101$:

$$\begin{aligned} 1\,1100\,0 &= 56 \\ 1100 &= 12 \\ 10 &= 2 \\ 0101 &= 5 \end{aligned}$$

DES Key Schedule

schedule(K)

1: $K \xrightarrow{\text{PC1}} (C, D)$
2: **for** $i = 1$ to 16 **do**
3: $\quad C \leftarrow \text{ROL}r_i(C)$
4: $\quad D \leftarrow \text{ROL}r_i(D)$
5: $\quad K_i \leftarrow \text{PC2}(C, D)$
6: **end for**

K: 56-bit register
C, D: two 28-bit registers
$K_1, \ldots, K_{16}$: 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: $\quad K_i \leftarrow \text{PC2}(C, D)$
4: $\quad C \leftarrow \text{ROR}r_i(C)$
5: $\quad D \leftarrow \text{ROR}r_i(D)$
6: **end for**

K: 56-bit register
C, D: two 28-bit registers
$K_1, \ldots, K_{16}$: 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

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)
- 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)

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`

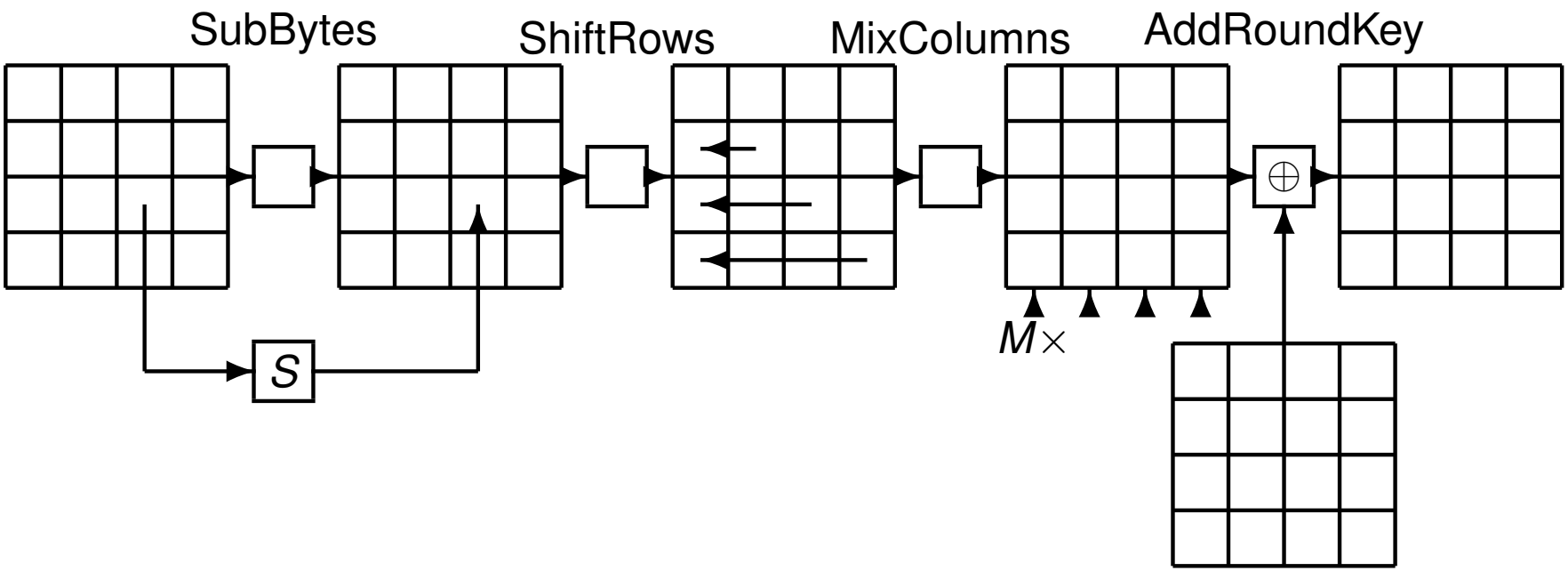
Rijndael Skeleton

- 128-bit block $\longrightarrow$ 4×4 square matrix of bytes
- $\mathrm{Nr} = 10$, 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 $\mathrm{Nr} - 1$ **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_{Nr})

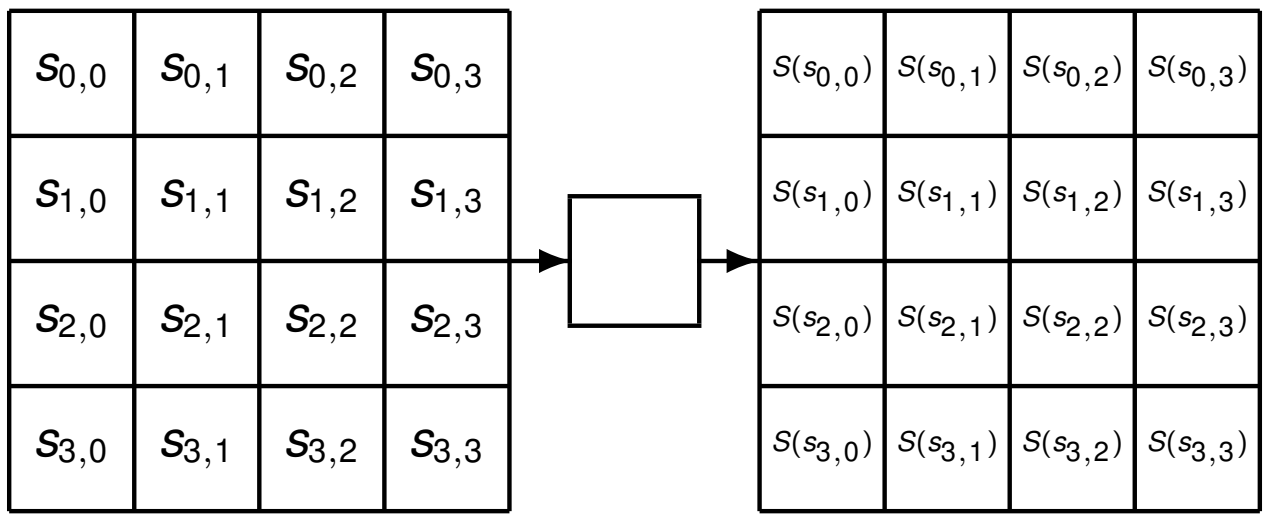
One Non-Terminal Round of Rijndael



SubBytes

SubBytes(s)

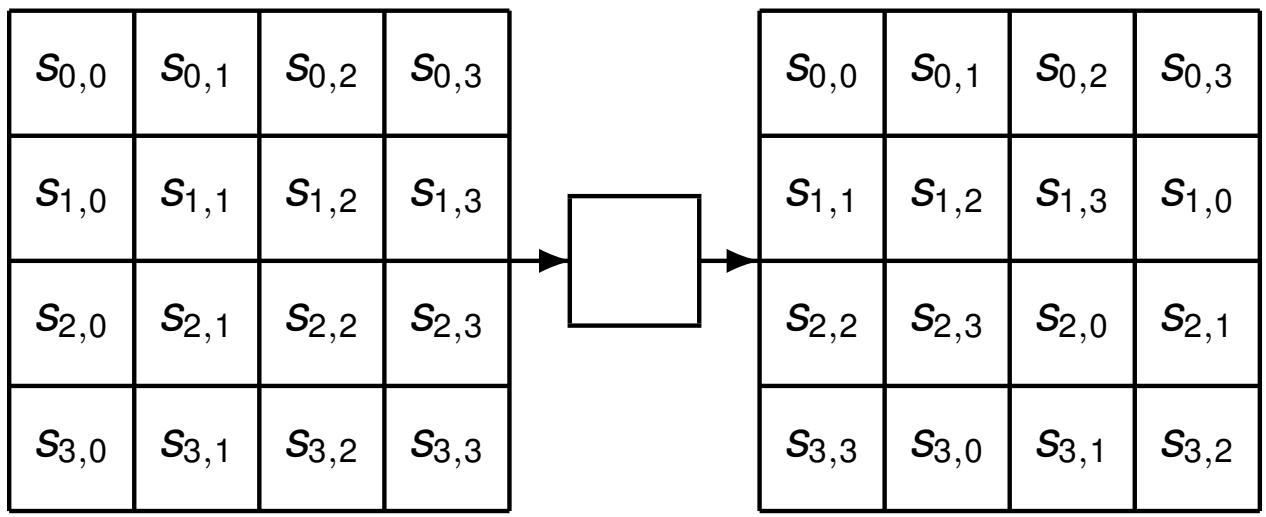
1: **for** $i = 0$ to 3 **do**
2: **for** $j = 0$ to 3 **do**
3: $s_{i,j} \leftarrow$ S-box$(s_{i,j})$
4: **end for**
5: **end for**



ShiftRows

ShiftRows(s)

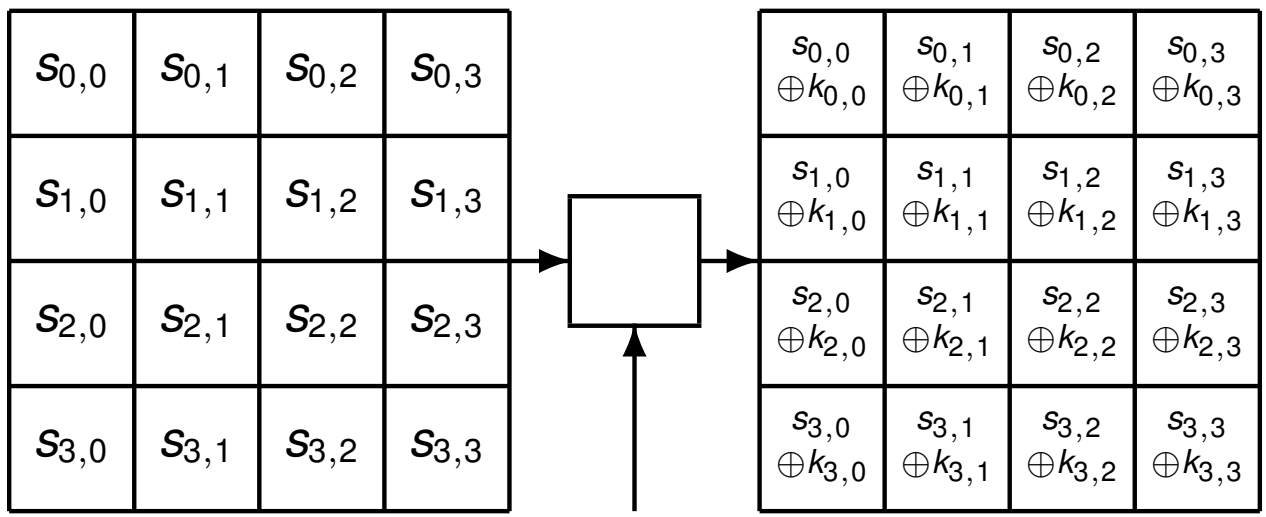
1: replace $[s_{1,0}, s_{1,1}, s_{1,2}, s_{1,3}]$ by $[s_{1,1}, s_{1,2}, s_{1,3}, s_{1,0}]$
2: replace $[s_{2,0}, s_{2,1}, s_{2,2}, s_{2,3}]$ by $[s_{2,2}, s_{2,3}, s_{2,0}, s_{2,1}]$
3: replace $[s_{3,0}, s_{3,1}, s_{3,2}, s_{3,3}]$ by $[s_{3,3}, s_{3,0}, s_{3,1}, s_{3,2}]$



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 408

- we use the following representation rule

byte	bit string	polynomial
B	$b_7 \cdots b_2 b_1 b_0$	$b_7.x^7 + \cdots + b_2.x^2 + b_1.x + b_0$

- we reduce everything modulo 2
  $\rightarrow$ monomial coefficients are binary
- we reduce everything modulo $x^8 + x^4 + x^3 + x + 1$
  $x^8 = x^4 + x^3 + x + 1$,
  $x^9 = x^8 \times x = (x^4 + x^3 + x + 1) \times x = x^5 + x^4 + x^2 + x$,
  ...
  $\rightarrow$ 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	$x^5 + x^3 + x$
=			$x^6 + x^5 + x^4 + 2.x^3 + x^2 + x$
=	0x76	01110110	$x^6 + x^5 + x^4 + x^2 + x$

- 0x9e × 0x02 = 0x27

	byte	bit string	polynomial
	0x9e	10011110	$x^7 + x^4 + x^3 + x^2 + x$
×	0x02	00000010	x
=			$x^8 + x^5 + x^4 + x^3 + x^2$
=			$x^5 + 2.x^4 + 2.x^3 + x^2 + x + 1$
=	0x27	00100111	$x^5 + x^2 + x + 1$

GF Arithmetics

A byte $a = a_7 \ldots a_1 a_0$ represents an element of the finite field GF(2^8) as a polynomial $a_0 + a_1.x + \ldots + a_7.x^7$ 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`

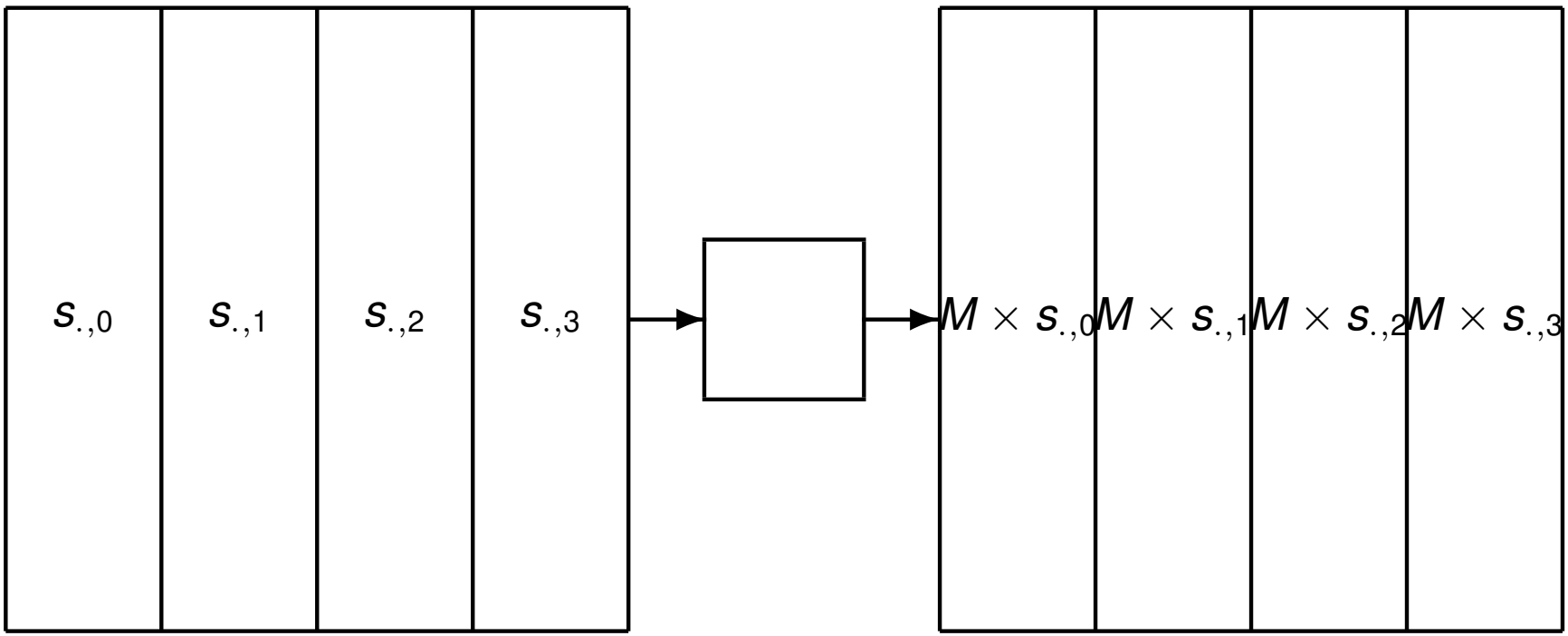
MixColumns

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**

$$M = \begin{pmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x02} \end{pmatrix}.$$

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**

$$M^{-1} = \begin{pmatrix} \texttt{0x0e} & \texttt{0x0b} & \texttt{0x0d} & \texttt{0x09} \\ \texttt{0x09} & \texttt{0x0e} & \texttt{0x0b} & \texttt{0x0d} \\ \texttt{0x0d} & \texttt{0x09} & \texttt{0x0e} & \texttt{0x0b} \\ \texttt{0x0b} & \texttt{0x0d} & \texttt{0x09} & \texttt{0x0e} \end{pmatrix}.$$

AES Decryption

AES decryption(s, W)

1: **AddRoundKey**(s, W_{Nr})
2: **for** $r = \text{Nr} - 1$ down to 1 **do**
3: **InvSubBytes**(s)
4: **InvShiftRows**(s)
5: **AddRoundKey**(s, W_r)
6: **InvMixColumns**(s)
7: **end for**
8: **InvSubBytes**(s)
9: **InvShiftRows**(s)
10: **AddRoundKey**(s, W_0)

Key Expansion

- we consider W as a sequence of $4(\mathrm{Nr}+1) = 44$ (resp. 52, 60) rows (32-bit words) w
- we consider the key as a sequence of $\mathrm{Nk} = 4$ (resp. 6, 8) rows
- the w_i are iteratively loaded:
  - the first w_i are loaded with the key
  - w_i is loaded with $w_{i-\mathrm{Nk}} \oplus w_{i-1}$
  - every Nk iterations, the w_i is modified before the XOR
  - for $\mathrm{Nk} = 8$, we add an extra modification

Key Expansion

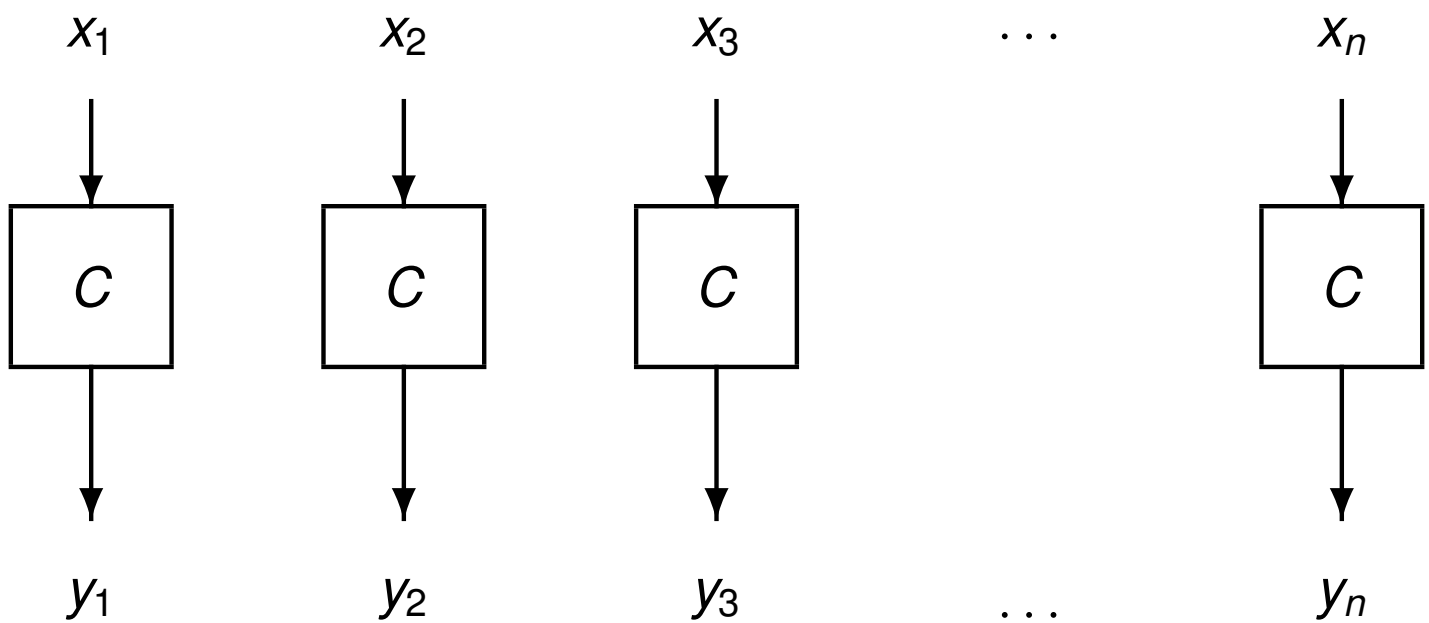
KeyExpansion(key, Nk)

1: **for** $i = 0$ to $\mathrm{Nk} - 1$ **do**
2: $\quad w_i \leftarrow \mathrm{key}_i$
3: **end for**
4: **for** $i = \mathrm{Nk}$ to $4(\mathrm{Nr} + 1) - 1$ **do**
5: $\quad t \leftarrow w_{i-1}$
6: $\quad$ **if** $i \bmod \mathrm{Nk} = 0$ **then**
7: $\quad\quad$ replace $[t_1, t_2, t_3, t_4]$ by $[t_2, t_3, t_4, t_1]$ in t
8: $\quad\quad$ apply S-box to the four bytes of t
9: $\quad\quad$ XOR $x^{i/\mathrm{Nk}-1}$ (in GF) onto the first byte of t
10: $\quad$ **else if** $\mathrm{Nk} = 8$ and $i \bmod \mathrm{Nk} = 4$ **then**
11: $\quad\quad$ apply S-box to the four bytes of t
12: $\quad$ **end if**
13: $\quad w_i \leftarrow w_{i-\mathrm{Nk}} \oplus t$
14: **end for**

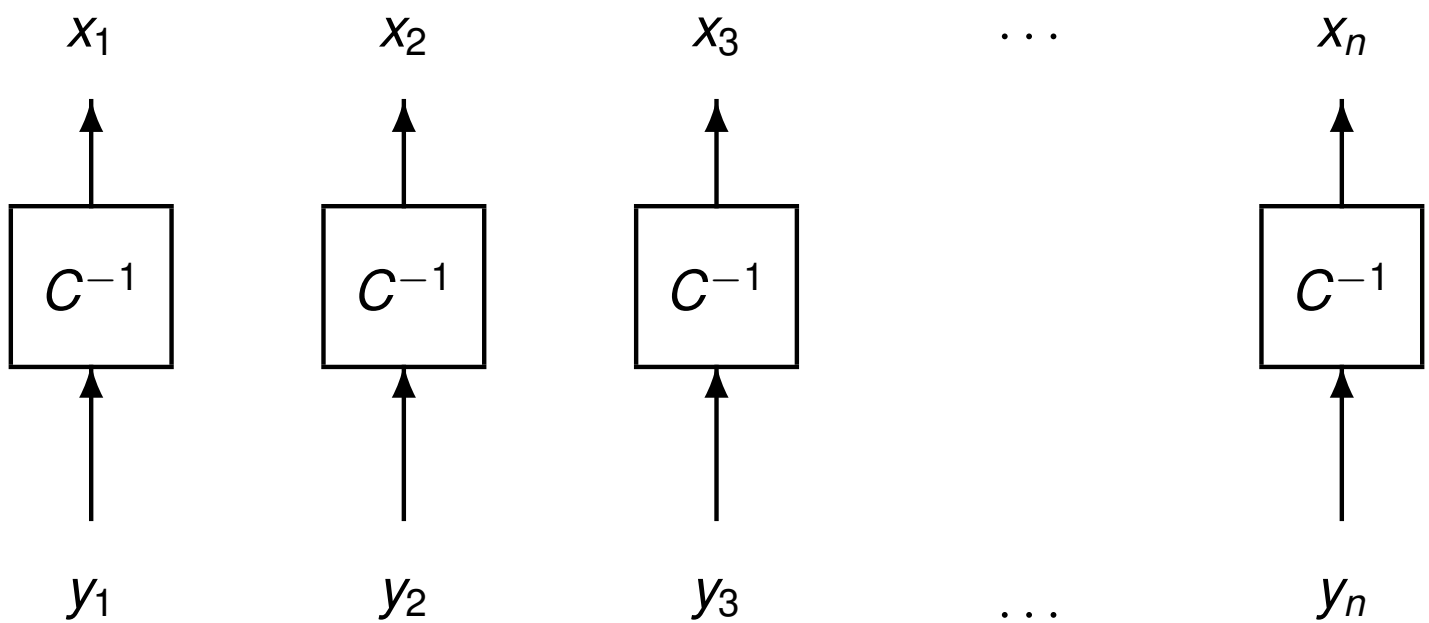
Modes of Operation

- 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



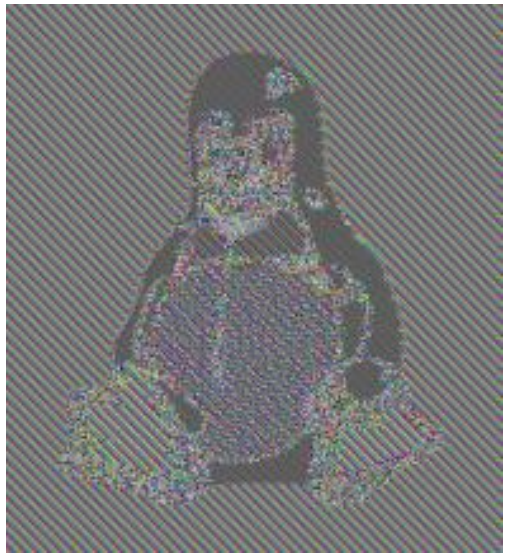
ECB Decryption



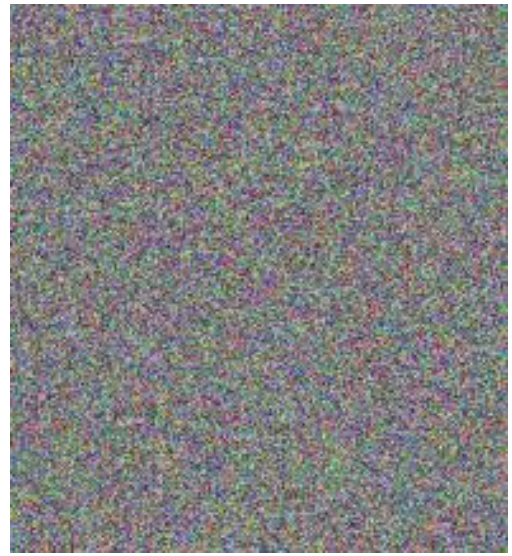
ECB vs CBC

original

ECB

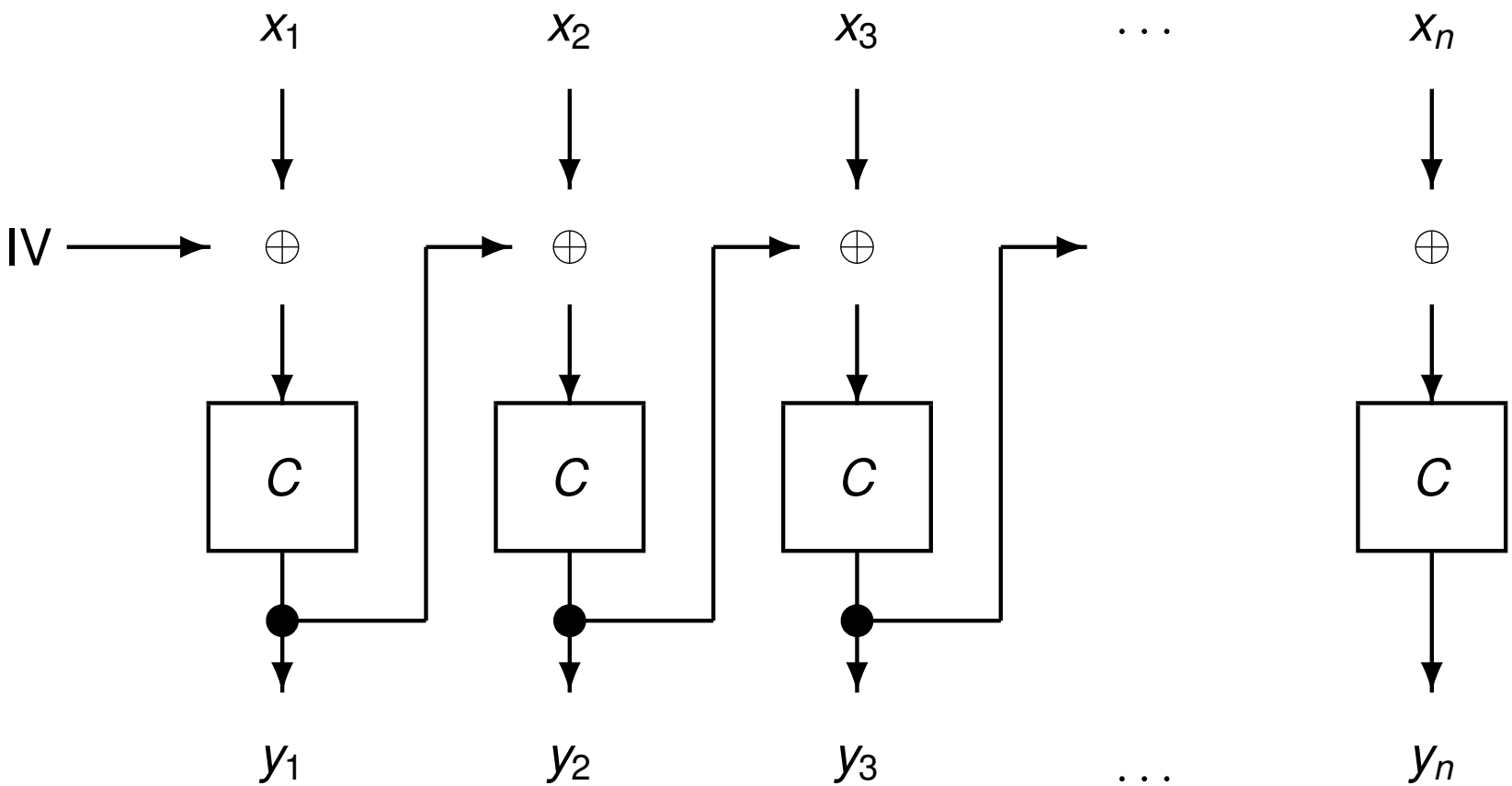


CBC

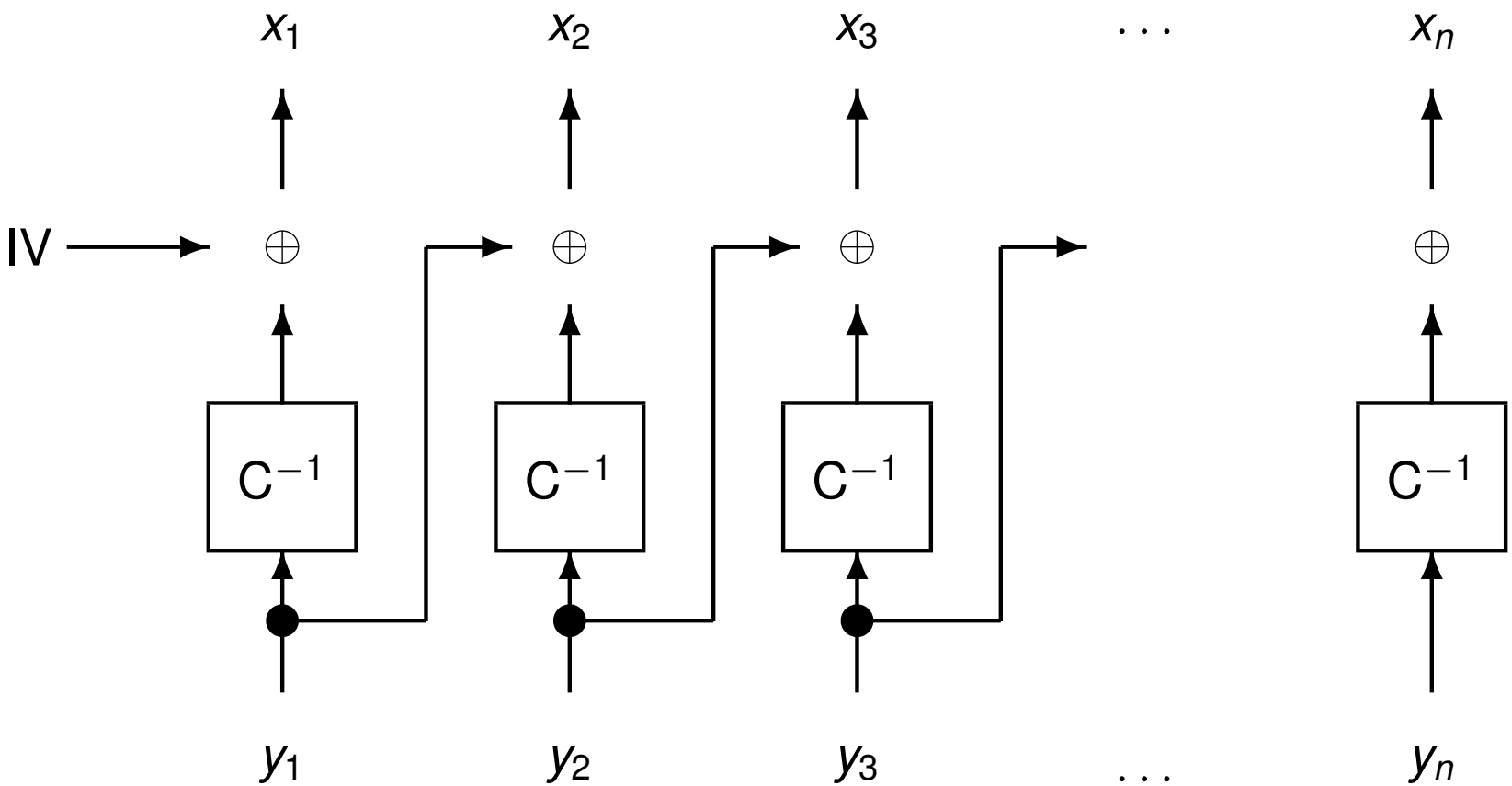


en.wikipedia.org

CBC Mode



CBC Decryption



Note on the CBC Mode

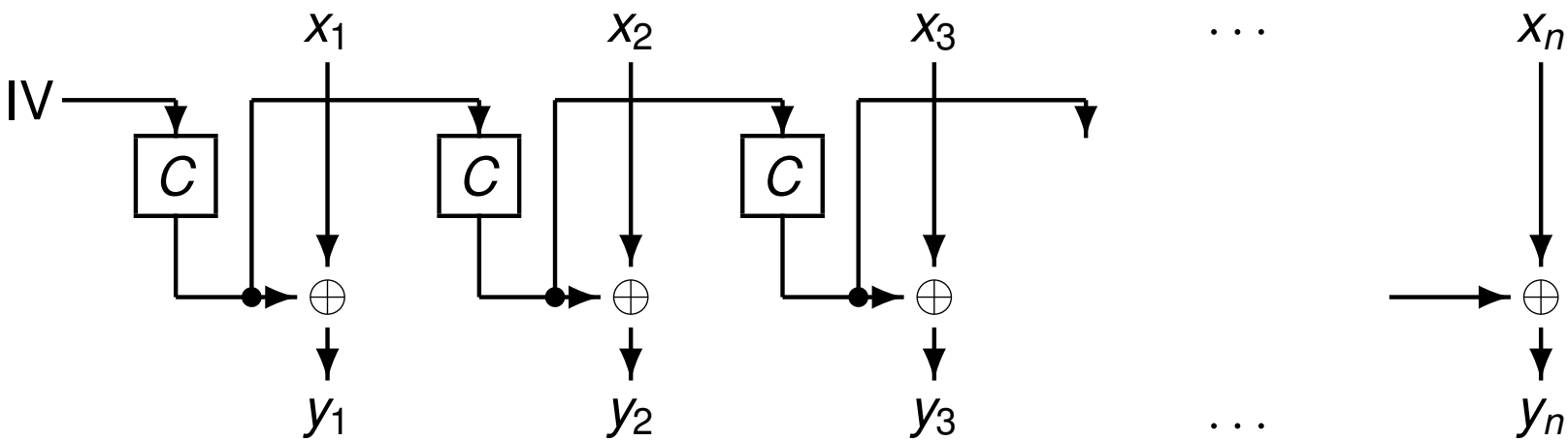
Three possibilities for dealing with IV

- Using a (non secret) constant IV
  example: **MRTD** (IV$=0$)
  (not a good idea)
- Using a secret IV which is part of the key
  example: **TLS**
  (ok if used only once like in TLS)
- Using a random IV which is sent in clear with the ciphertext

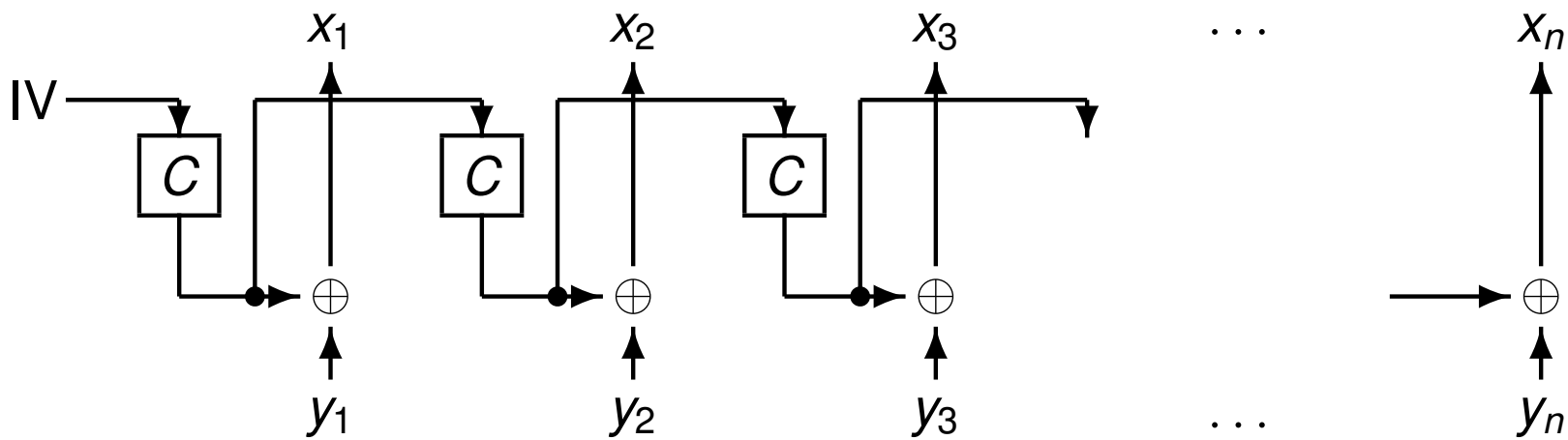
Property

If y_i is corrupted, only x_i and x_{i+1} are badly decrypted.
If y_i is lost, only x_i is incorrect (and one block is missing).

OFB Mode



OFB Decryption

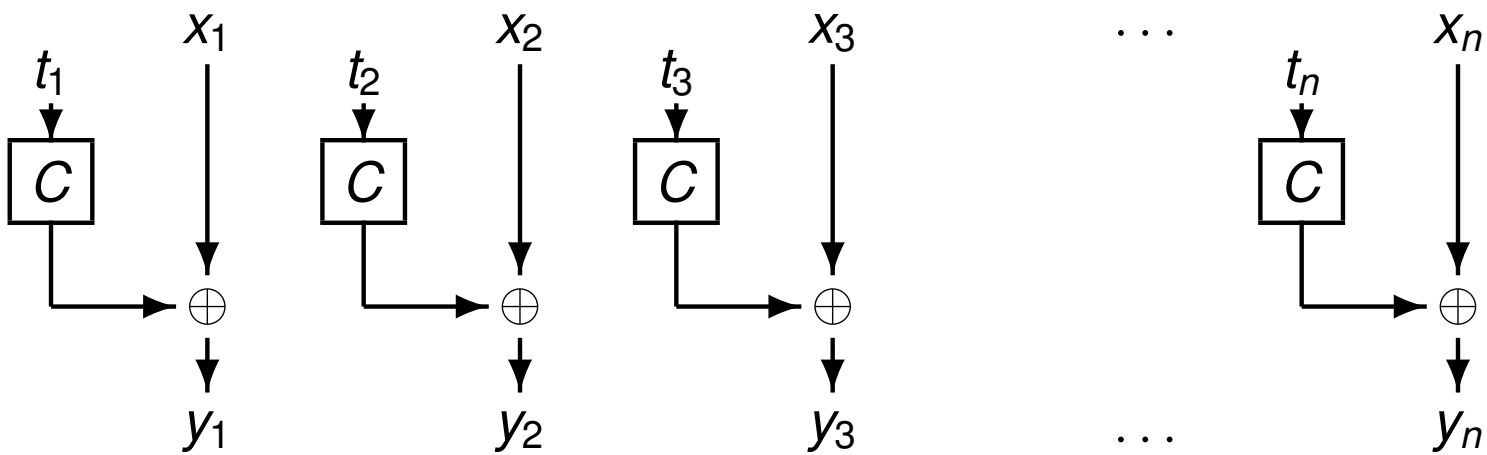


Note on the OFB Mode

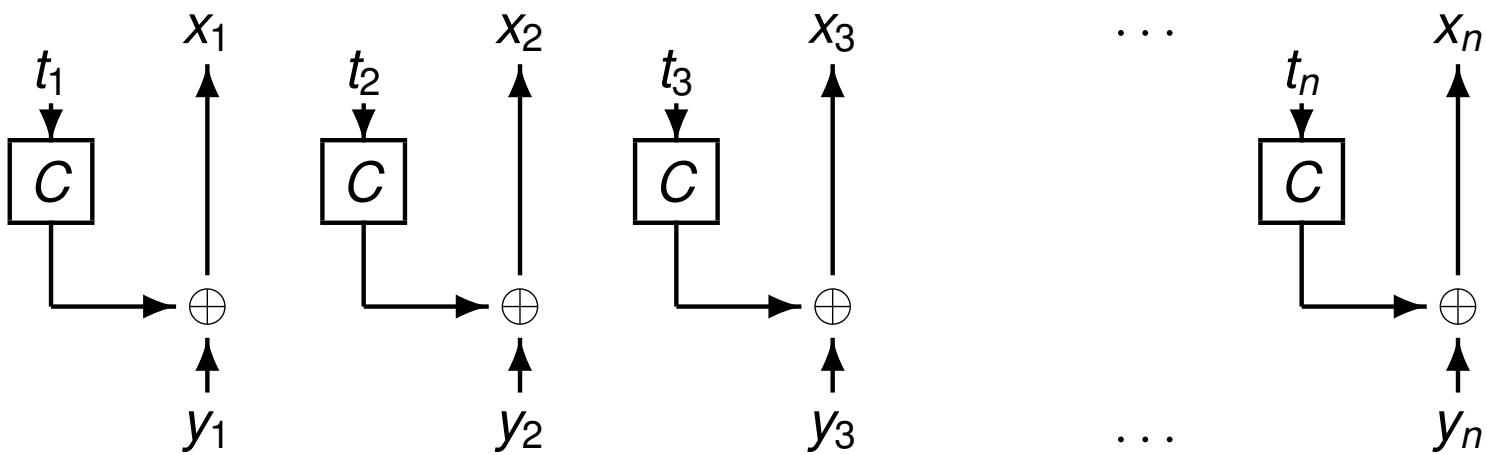
- IV must be new for every plaintext!
- two possibilities:
  - Use a random one which is sent in clear...
  - ... or use a counter-based IV
- Can easily handle message length not multiple of the blocklength
- These are not only specific properties of the OFB mode: properties of stream ciphers
- OFB actually transforms a block cipher into a stream cipher
- Interesting property: can encrypt incomplete blocks

IV is used as a **nonce** (number used once)

CTR Mode



CTR Decryption



Note on the CTR Mode

- t_i must be new for every block! (a nonce)
  Example 1: $t_i = \text{nonce} \| \text{blk\_counter}$
  with nonce = msg_counter or random
  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
- Can easily handle message length not multiple of the blocklength
- CTR also transforms a block cipher into a stream cipher
- advantage over OFB:
  can be parallelized
  random restart on any t_i

Example from the GCM mode: $\text{GCTR}_K((\text{nonce} \| 0^{31}1), \text{msg})$ with

$$\text{GCTR}_K(\text{ct}, X) = \text{trunc}_{\text{length}(X)}\left(C_K(\text{ct}) \| C_K(\text{ct}+1) \| C_K(\text{ct}+2) \cdots\right) \oplus X$$

(do not encrypt 2^{32} blocks...)

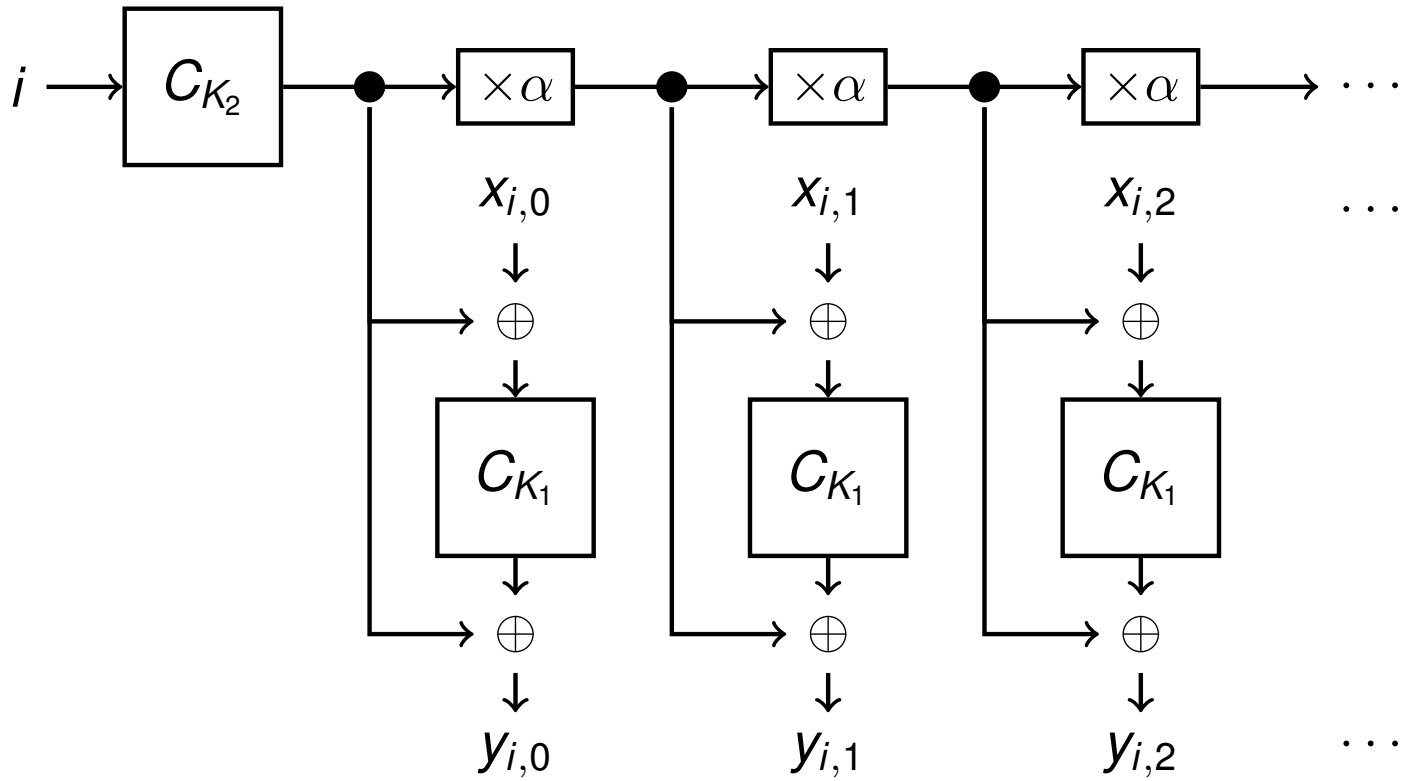
XTS Mode

- used to encrypt a hard disk
- hard disks are made of “sectors” of various lengths
  lengh 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:

$$y_{i,j} = \mathrm{Enc}_{i,j}(x_{i,j}) = C_{K_1}(x_{i,j} \oplus t_{i,j}) \oplus t_{i,j} \quad t_{i,j} = \alpha^j \times C_{K_2}(i)$$

  in a GF structure, with a constant α
- use **ciphertext stealing** for the last two blocks

XTS


i
C_{K_2}
$\times\alpha$
$\times\alpha$
$\times\alpha$
$\cdots$
$x_{i,0}$
$x_{i,1}$
$x_{i,2}$
$\cdots$
C_{K_1}
C_{K_1}
C_{K_1}
$y_{i,0}$
$y_{i,1}$
$y_{i,2}$
$\cdots$

Ciphertext Stealing

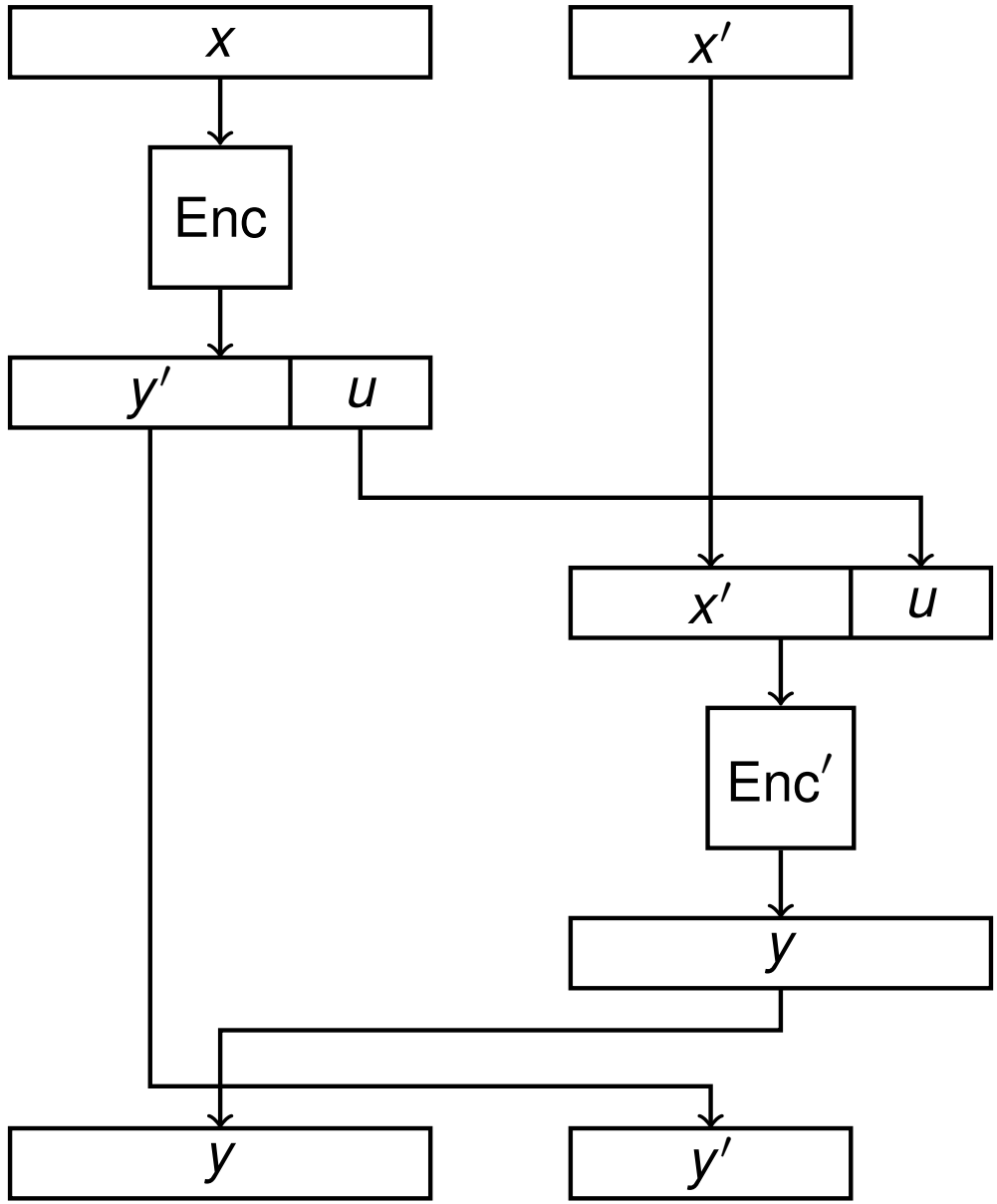
- used to encrypt two blocks x and x' (typically, the last two) with Enc and Enc$'$ respectively
- Case 1 (easy): if x and x' have regular length, encrypt normally $y = \mathrm{Enc}(x)$, $y' = \mathrm{Enc}'(x')$
- Case 2: if x' is shorter than usual.

  1: split $\mathrm{Enc}(x) = y' \| u$ with y' of same length as x'
  2: $y = \mathrm{Enc}'(x' \| u)$
  3: give y and y'

  to decrypt y and y':

  1: split $\mathrm{Dec}'(y) = x' \| u$ with x' of same length as y'
  2: $x = \mathrm{Dec}(y' \| u)$
  3: give x and x'

Ciphertext Stealing



To Be Known About Modes of Operation

- ECB should be avoided
- CBC 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

The Symmetric Encryption Zoo

- **fauna**: ARMADILLO BEAR BLOWFISH DRAGON FOX FROG LION MOSQUITO RABBIT SERPENT SHACAL SHARK TWOFISH
- **flora**: CAMELLIA LILY SEED
- **pantheon**: ANUBIS MARS KHAFRE KHUFU LUCIFER MICKEY SHANNON TURING
- **the gastronomics**: COCONUT GRANDCRU KFC MILENAGE PEANUT WALNUT
- **the elements**: CRYPTON ICE ICEBERG RAINBOW SNOW
- **the originals**: ABC ACHTERBAHN AKELARRE CAST DEAL DECIM EDON FEAL FUBUKI GOST HELIX HIEROCRYPT IDEA KASUMI KATAN KHAZAD KTANTAN LEX LEVIATHAN LOKI MACGUFFIN MADRYGA MAGENTA MIR MISTY NIMBUS NOEKEON NUSH PHELIX PRESENT PY QUAD REDOC RIJNDAEL SAFER SALSA SCREAM SFINKS SKIPJACK SMS4 SQUARE SOBER SOSEMANUK XTEA 3-WAY YAMB
- **the uninspired**: A5 AES BMGL C2 CJCSG CMEA CS-CIPHER DES DFC E0 E2 FCSR HPC MMB Q RC2 RC4 RC5 RC6 SC TSC WG

The Symmetric Encryption Zoo ...in practice

BLOWFISH

MILENAGE

IDEA KASUMI

SAFER

A5 AES DES

E0 RC4

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	

5 Symmetric Encryption

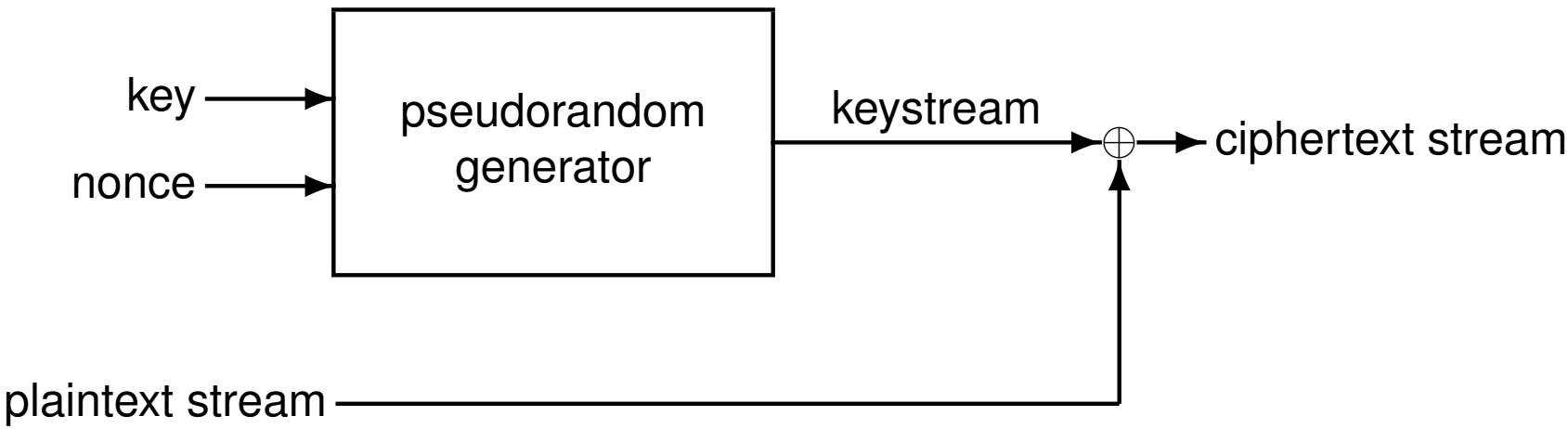
- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

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)
  Problem: stateful
- variant 2: the encrypting nonce is sent in clear with the ciphertext (asynchronous)
  Problem: nonce becomes under the control of the adversary (at least for decryption)

Stream Ciphers from a High Level

(= Vernam cipher with a pseudorandom key)



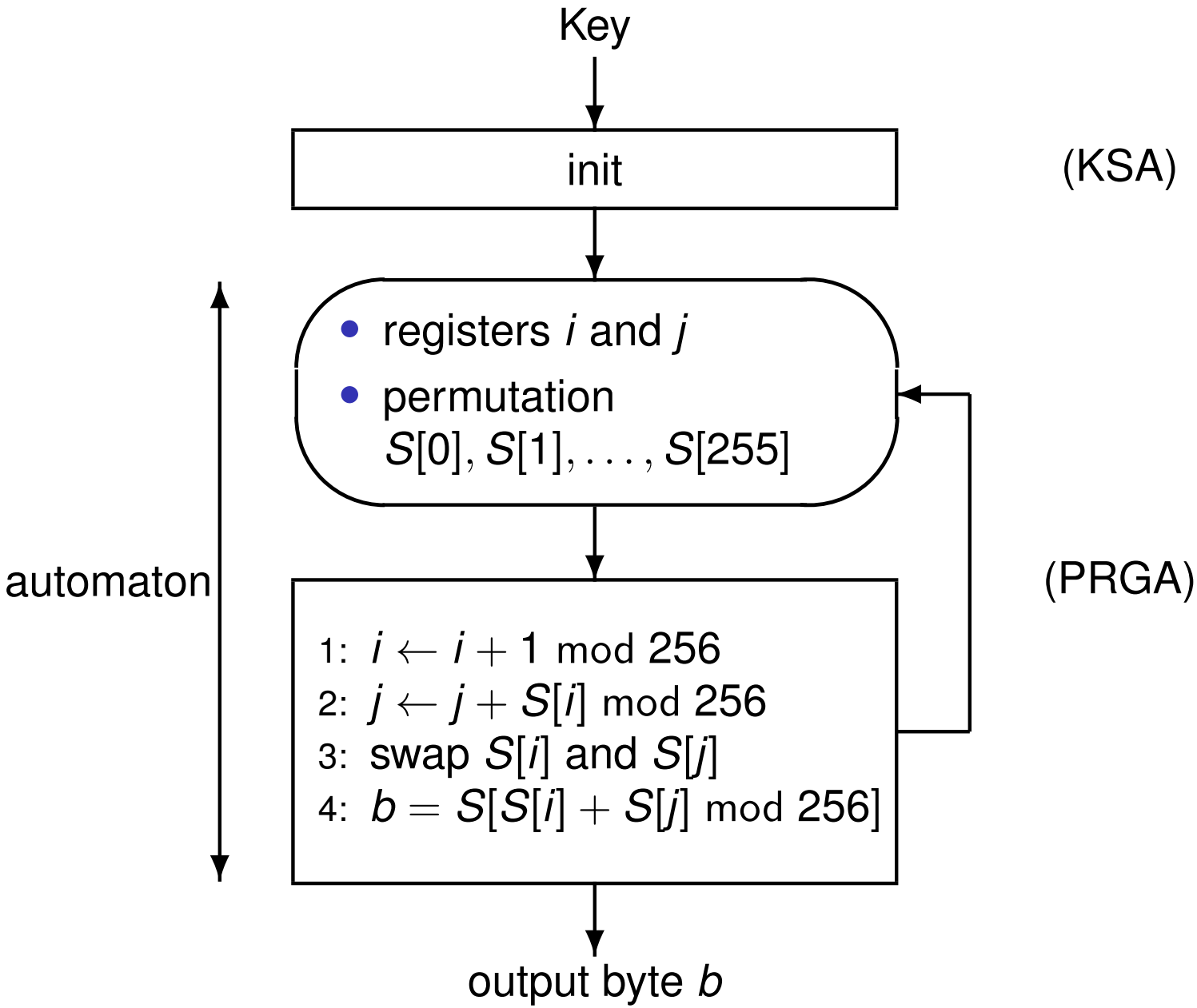
nonce = **n**umber which can be used **once**
(necessary to avoid re-using a keystream)

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)


Key
init
(KSA)
registers i and j
permutation
$S[0], S[1], \ldots, S[255]$
automaton
(PRGA)
1: $i \leftarrow i + 1 \bmod 256$
2: $j \leftarrow j + S[i] \bmod 256$
3: swap $S[i]$ and $S[j]$
4: $b = S[S[i] + S[j] \bmod 256]$
output byte b

RC4 Key Schedule (KSA)

1: $j \leftarrow 0$
2: **for** $i = 0$ to 255 **do**
3: $\quad S[i] \leftarrow i$
4: **end for**
5: **for** $i = 0$ to 255 **do**
6: $\quad j \leftarrow j + S[i] + K[i \bmod \ell] \bmod 256$
7: $\quad$ 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 or 13-byte key

Known Weaknesses

- some correlations between some output bytes and key bytes when the nonce is known
  $\rightarrow$ (passive) key recovery attack in WEP after seeing 22500 packets
- output bytes are not uniformly distributed
  $\rightarrow$ 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)

Case Study: WiFI: WEP/WPA/WPA2

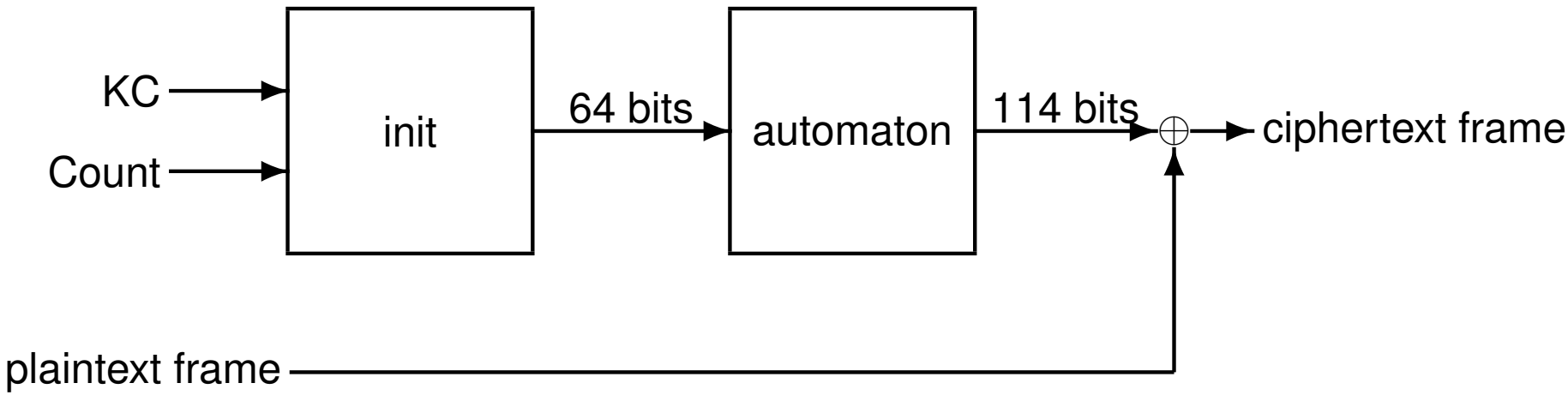
▸ case study

GSM A5/1

- Designed at ETSI by the SAGE group
- Trade secret of the GSM consortium
- reverse engineered

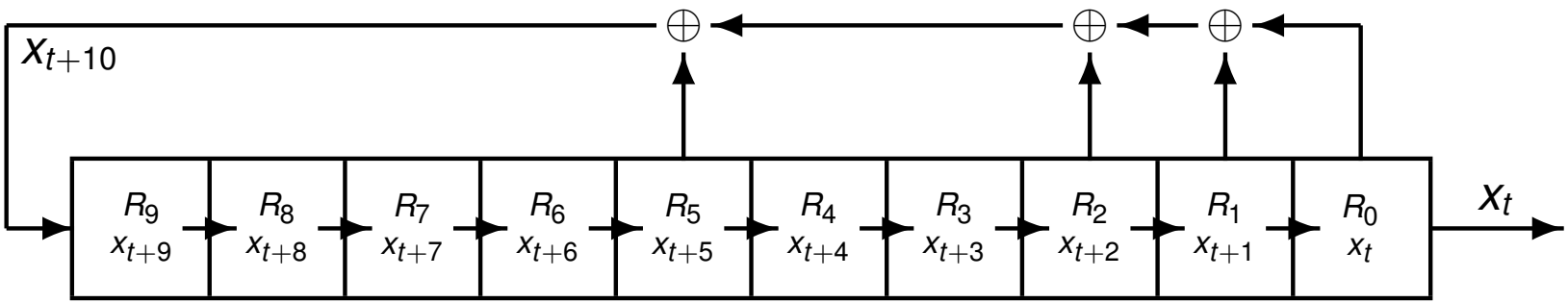
- dedicated to lightweight hardware
- stream cipher with bit streams
- 64-bit key and 22-bit counter

A5/1 from a High Level



Linear Feedback Shift Register (LFSR)

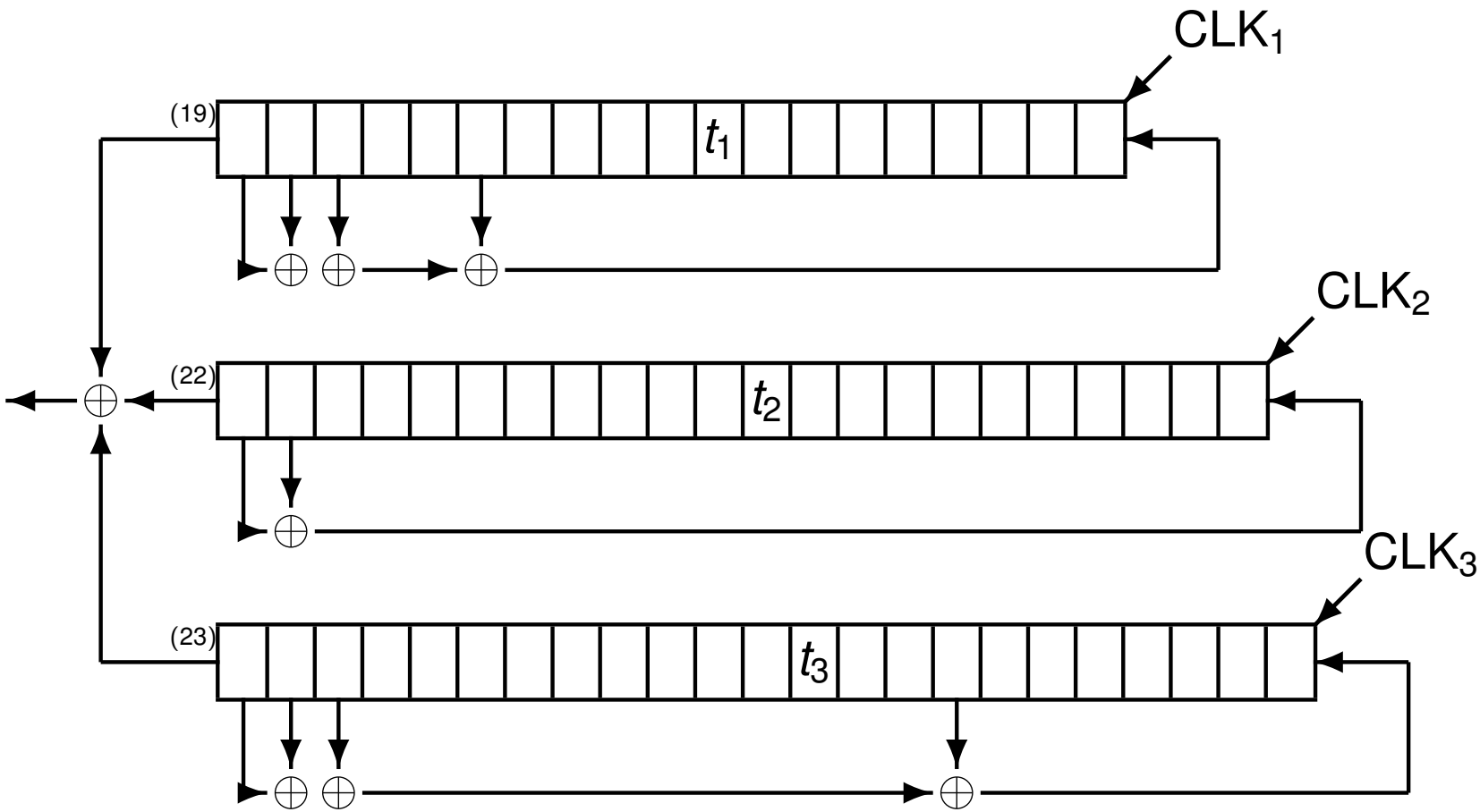
- when CLK $= 1$, increment t, load R_i with R_{i+1} and R_{d-1} with a XOR of some R_i's



- at time t, $R_i = x_{t+i}$
- $x_{t+d} = a_{d-1}x_{t+d-1} \oplus \cdots \oplus a_0 x_t$ for any t (linear recursion)
- $a_d x_{t+d} \oplus \cdots \oplus a_1 x_{t+1} \oplus a_0 x_t = 0$ for any t ($a_d = 1$)
- connection polynomial: $a_d x^d + \cdots + a_1 x + a_0$
  example: $x^{10} + x^5 + x^2 + x + 1$
- maximal period $\iff$ primitive polynomial $\implies$ irreducible polynomial

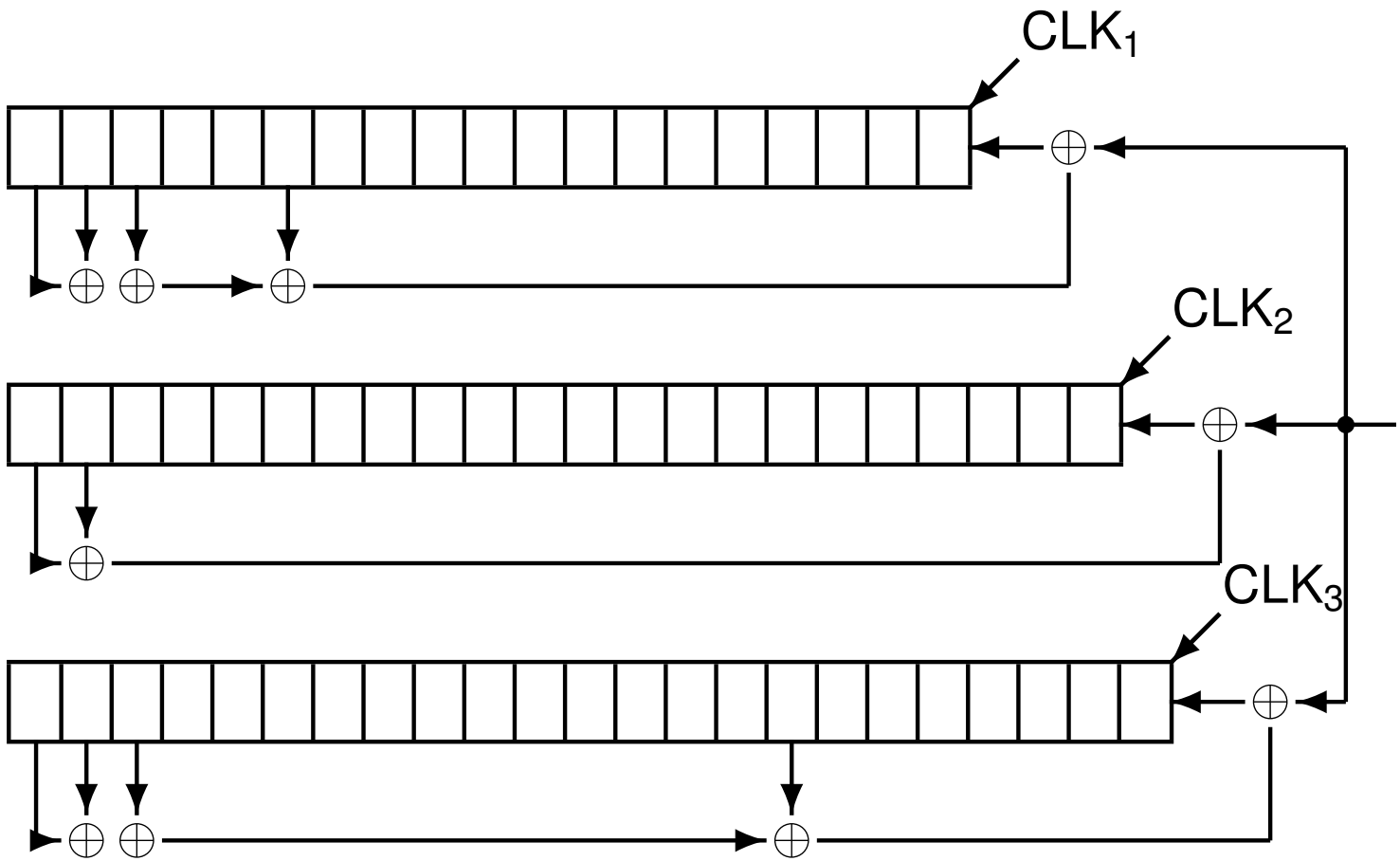
A5/1 Automaton

$$x^{19} + x^{18} + x^{17} + x^{14} + 1$$
$$x^{22} + x^{21} + 1$$
$$x^{23} + x^{22} + x^{21} + x^{8} + 1$$



asynchronous: $\mathrm{CLK}_i = \mathrm{CLK}$ if $t_i = \mathrm{majority}(t_1, t_2, t_3)$, 0 otherwise

A5/1 Initialization



synchronous: $CLK_1 = CLK_2 = CLK_3 = CLK$

A5/1 Initialization

1: set all registers to zero
2: **for** $i = 0$ to 63 **do**
3: $\quad R_1[0] \leftarrow R_1[0] \oplus \mathrm{KC}[i]$
4: $\quad R_2[0] \leftarrow R_2[0] \oplus \mathrm{KC}[i]$
5: $\quad R_3[0] \leftarrow R_3[0] \oplus \mathrm{KC}[i]$
6: $\quad$ clock registers (synchronous)
7: **end for**
8: **for** $i = 0$ to 21 **do**
9: $\quad R_1[0] \leftarrow R_1[0] \oplus \mathrm{Count}[i]$
10: $\quad R_2[0] \leftarrow R_2[0] \oplus \mathrm{Count}[i]$
11: $\quad R_3[0] \leftarrow R_3[0] \oplus \mathrm{Count}[i]$
12: $\quad$ clock registers (synchronous)
13: **end for**
14: **for** $i = 0$ to 99 **do**
15: $\quad$ clock registers (asynchronous)
16: **end for**

Known Weaknesses

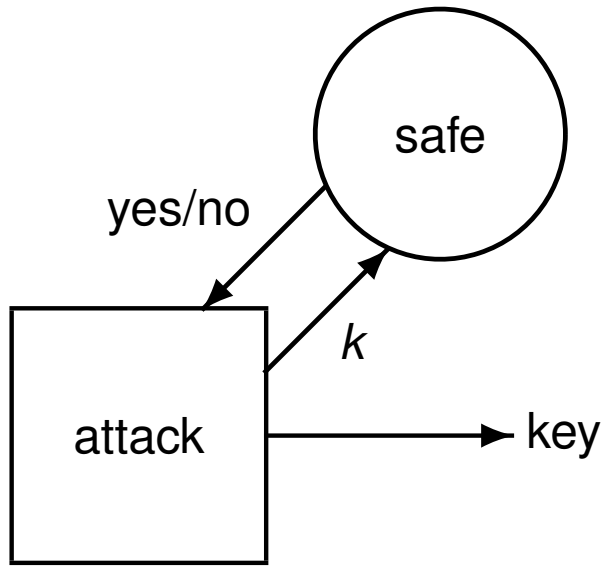
- key recovery known plaintext attack
  (kind of time-memory tradeoff)
- active attacks on GSM (chosen cipher attack)
- ciphertext-only key recovery attack
  (optimized bruteforce)

5 Symmetric Encryption

- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

Example: Opening a Safe

For any k, we can ask the safe whether the key K is equal to k



This attack makes online queries to test k

Key Recovery Game - Online (with no Clue)

"online with a stop-test oracle $\mathcal{O}$"

Game:

1: pick $K \in_D \mathcal{K}$
2: $\mathcal{A}^{\mathcal{O}} \to k$
3: **return** $1_{k=K}$

$\mathcal{O}$(query):

4: **return** $1_{K=\text{query}}$

notation $\mathcal{A}^{\mathcal{O}}$: algorithm $\mathcal{A}$ who can query oracle $\mathcal{O}$ (independent subroutine)

Variations

- K is uniform
- K follows a known distribution D
- K follows an unknown distribution D

Exhaustive Search Algorithm (Uniform Case)

(online, with no clue, D **uniform**)

Input: a set of possible keys $\mathcal{K} = \{k_1, \ldots, k_N\}$
Challenger interface: input is an element of $\mathcal{K}$,
output is Boolean

1: **for** all $i = 1$ to N **do**
2: query k_i
3: **if** answer is yes **then**
4: yield k_i and stop
5: **end if**
6: **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, D **known**)

Input: a set of possible keys $\mathcal{K} = \{k_1, \ldots, k_N\}$
Challenger interface: input is an element of $\mathcal{K}$, output is Boolean

take the permutation σ of $\{1, \ldots, N\}$ sorting $k_{\sigma(i)}$ by decreasing order of likelihood
for all $i = 1$ to N **do**
    query $k_{\sigma(i)}$
    if answer is yes **then**
        yield $k_{\sigma(i)}$ and stop
    end if
end for

$$E(\#\text{iterations}) = \min_{\sigma} \left(\sum_{i=1}^{N} \Pr[K = k_{\sigma(i)}] i \right)$$

which is sometimes called the **guesswork entropy** of D

Exhaustive Search Algorithm (Any Case)

(online, with no clue)

Input: a set of possible keys $\mathcal{K} = \{k_1, \ldots, k_N\}$
Challenger interface: input is an element of $\mathcal{K}$,
output is Boolean

1: pick a random permutation σ of $\{1, \ldots, N\}$
2: **for** all $i = 1$ to N **do**
3: query $k_{\sigma(i)}$
4: **if** answer is yes **then**
5: yield $k_{\sigma(i)}$ and stop
6: **end if**
7: **end for**

$$E(\#\text{iterations}) = \sum_{i=1}^{N} E(\Pr[K = k_{\sigma(i)}])i$$

since σ is random we have $E(\Pr[K = k_{\sigma(i)}]) = \frac{1}{N}$ for all i:

$$E(\#\text{iterations}) = \sum_{i=1}^{N} \frac{1}{N} i = \frac{N+1}{2}$$

Complexity Analysis (All Cases)

key of distribution D in a set of N elements

worst case complexity		N iterations
average complexity	D uniform	$\frac{N+1}{2}$ iterations
	D known	smaller
	D unknown	$\frac{N+1}{2}$ iterations
memory complexity		constant
success probability		1

Key Recovery Game - With Clue

Online Key Recovery with Clue:

1: pick $K \in_D \mathcal{K}$
2: $W \leftarrow$ clue about K
3: $\mathcal{A}^{\mathcal{O}}(W) \rightarrow k$
4: **return** $1_{k=K}$

$\mathcal{O}$(query):

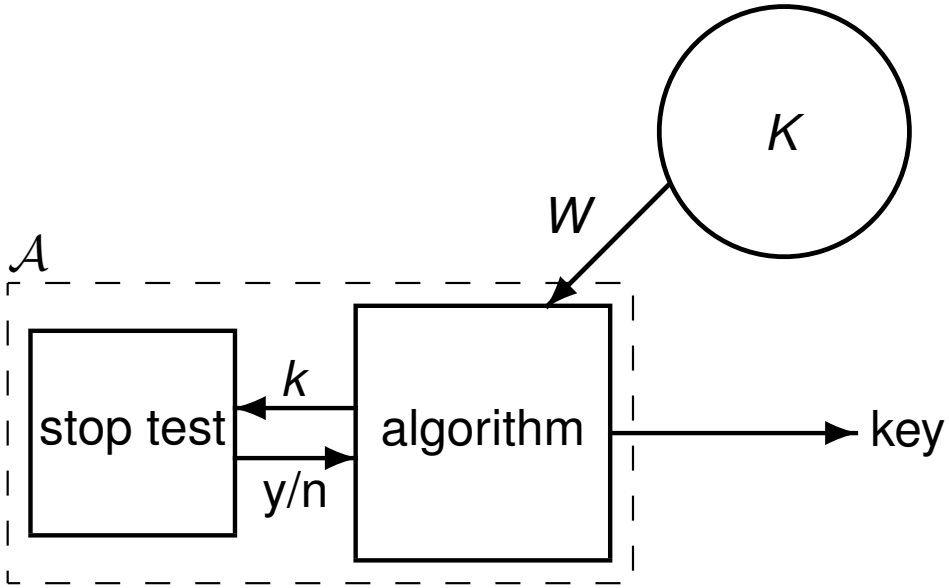
5: **return** $1_{K=\text{query}}$

Offline Key Recovery with Clue:

1: pick $K \in_D \mathcal{K}$
2: $W \leftarrow$ clue about K
3: $\mathcal{A}(W) \rightarrow k$
4: **return** $1_{k=K}$

Offline Attack with Clue

$\mathcal{A}$ uses W to emulate a stop-test oracle



Examples:

	witness	stop test
known plaintext attack	$W = (x, C_K(x))$	$C_k(W_1) = W_2$
ciphertext only attack	$W =$ ciphertext	$C_k^{-1}(W)$ meaningful
salted key hash	$W = (F(K, \text{salt}), \text{salt})$	$F(k, W_2) = W_1$

(salt to be seen on ▸ slide 507)

Cases for Deterministic Clues

- **chosen plaintext attack**:
  get $W = C_K(x)$ for some fixed x chosen by the adversary
- **fixed plaintext attack** / **deterministic hash**:
  get $W = C_K(x_0)$ for some constant x_0 (e.g. $x_0 = 0$)

Cases for Non-Deterministic Clues

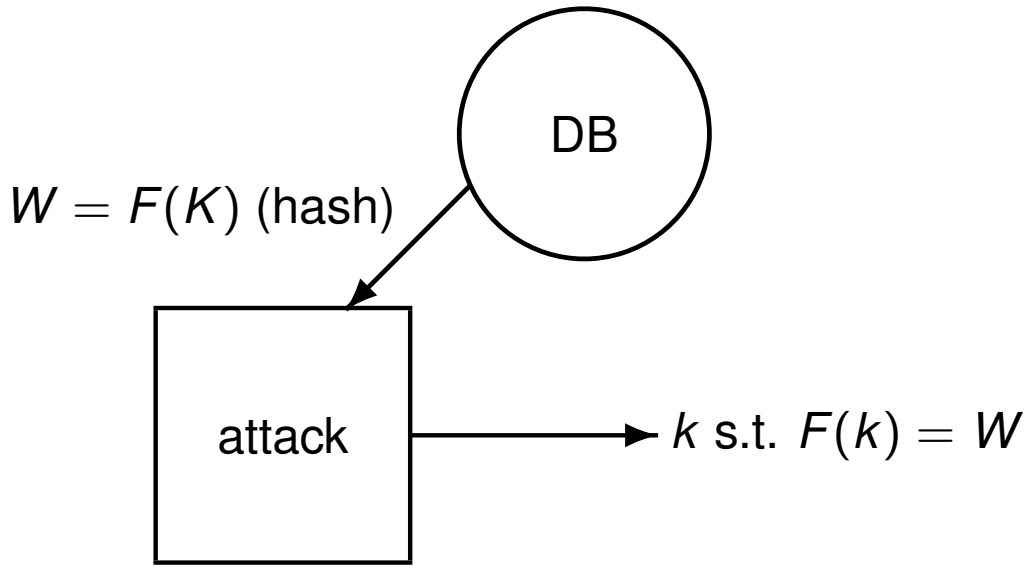
- no chosen plaintext attack:
  known plaintext attack with random $W = (x, C_K(x))$ pair
  ciphertext only attack with redundant plaintexts
- **randomized key hash**:
  leak $W = (F(K, \text{salt}), \text{salt})$ with **salt** randomly selected by the challenger
  (salt to be seen on ▸ slide 507)

Access Control

- **Enrolment**
  Enter user ID and password
  Register user ID and a clue about password
- **Access Control**
  Enter user ID and password
  User ID and password are verified using the clue

→ *any* key passing the test is enough to break
example: clue = password hash

Password Hash Inversion



the adversary has to find **one** password with correct hash
(the problem is to **invert** F, i.e. to find a preimage)

Game: Key Recovery vs Inversion

Offline key recovery:

1: pick $K \in_D \mathcal{K}$
2: $W \leftarrow F(K)$
3: $\mathcal{A}(W) \to k$
4: **return** $1_{k=K}$

Offline inversion:

1: pick $K \in_D \mathcal{K}$
2: $W \leftarrow F(K)$
3: $\mathcal{A}(W) \to k$
4: **return** $1_{F(k)=W}$

Two Forms of Bruteforce Attack with Clue

- **key recovery** in symmetric encryption
  - chosen plaintext attack ($\rightarrow$ clue)
  - known plaintext attack ($\rightarrow$ clue)
  - ciphertext only attack ($\rightarrow$ clue)

  we want to find K given a clue
- **preimage recovery** from a password hash ($\rightarrow$ clue)
  - deterministic hash
  - salted hash

  we want to find *any* password consistent with a clue

Inversion by Exhaustive Search

Goal: find any preimage of w (but preimages may not exist!)

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 $F : \mathcal{K} \to \mathcal{Y}$ is a uniformly distributed random function, $\#\mathcal{K} = N$, $\#\mathcal{Y} = M$:
for any $w \in_U \mathcal{Y}$

$$E_F\left[\Pr[\text{complexity} > i]\right] = \left(1 - \frac{1}{M}\right)^i$$

Complexity of an Inversion Attack

$$
\begin{aligned}
E(\text{complexity}) &= \sum_{i=0}^{N} i \Pr[\text{complexity} = i] \qquad (\textstyle\sum \text{ rows}) \\
&= \sum_{i=0}^{N-1} \Pr[\text{complexity} > i] \qquad (\textstyle\sum \text{ columns}) \\
&= \sum_{i=0}^{N-1} x^i \qquad \text{with } x = 1 - \frac{1}{M} \\
&= \frac{1 - x^N}{1 - x} \\
&\sim \frac{1 - e^{-\frac{N}{M}}}{1 - x} \qquad \text{as } \frac{N}{M} \to +\infty \\
&= M\left(1 - e^{-\frac{N}{M}}\right) \\
&\approx M \text{ for } N \gg M
\end{aligned}
$$

$$
\begin{array}{ccccc}
\Pr[>0] & \Pr[>1] & \Pr[>2] & & \\
\hline
\Pr[1] & & & & \\
\Pr[2] & + & \Pr[2] & & \\
\Pr[3] & + & \Pr[3] & + & \Pr[3] \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

Dictionary Inversion Attack (Full Book)

(assume a deterministic function F)

Game:

1: setup F
2: $\mathcal{A}_1^F \to \text{dict}$
3: pick $K \in_D \mathcal{K}$
4: $w \leftarrow F(K)$
5: $\mathcal{A}_2^F(\text{dict}, w) \to k$
6: **return** $1_{F(k)=w}$

$\mathcal{A}_1^F$: (preprocessing)

1: **for** all candidates k **do**
2: compute $F(k)$
3: $\text{dict}\{F(k)\} \leftarrow k$
4: **end for**
5: **return** dict

$\mathcal{A}_2^F(\text{dict}, w)$: (attack)

6: **return** $\text{dict}\{w\}$

Dictionary Inversion Attack (Smaller Dictionary)

(assume a deterministic function F)

Game:

1: setup F
2: $\mathcal{A}_1^F \to$ dict
3: pick $K \in_D \mathcal{K}$
4: $w \leftarrow F(K)$
5: $\mathcal{A}_2^F(\text{dict}, w) \to k$
6: **return** $1_{F(k)=w}$

$\mathcal{A}_1^F$: (preprocessing)

1: **for** M candidates k **do**
2: compute $F(k)$
3: dict$\{F(k)\} \leftarrow k$
4: **end for**
5: **return** dict

$\mathcal{A}_2^F(\text{dict}, w)$: (attack)

6: **return** dict$\{w\}$

Metrics of Algorithms

for comparing algorithms, we must look at:

- precomputation time
- memory complexity
- time complexity
- number of online queries
- probability of success

Complexity Analysis

Precomputation time M

Memory complexity M

Time complexity ≈ 1

Probability of success (with randomly selected dictionary keys)

$$M/N$$

Summary of Single-Target Brute Force Attacks

key search: $N = \#$ key space
inversion: $N = \#$ output range

strategy	preprocessing	memory	time	success proba.
exhaustive search	0	1	N	1
dictionary attack	N	N	1	1
tradeoffs	N	$N^{\frac{2}{3}}$	$N^{\frac{2}{3}}$	cte
partial ex. search	0	1	M	M/N
dictionary attack	M	M	1	M/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}

$\rightarrow$ 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 $\{\mathtt{a}, \ldots, \mathtt{z}, \mathtt{A}, \ldots, \mathtt{Z}, \mathtt{0}, \ldots, \mathtt{9}\}$ 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)
  $\longrightarrow$ 256*d* to find a password with a PC
- time-memory tradeoffs (complexity $N^{\frac{2}{3}}$ + precomputation N)
  [cracked a 36-bit entropy password within a few seconds]
  $\longrightarrow$ 1*h* to find a password (+ a year of precomputation)
- special purpose hardwares
  [cracked 56-bit keys within a day]
  $\longrightarrow$ 5 *min* to find a password
- `distributed.net`
  [cracked 64-bit keys in 2002 after 1757 days]
  $\longrightarrow$ 40 *min* to find a password

Extension: Multi-Target Dictionary Inversion Attack

(assume a deterministic function F)

Game:

1: setup F
2: $\mathcal{A}_1^F \to$ dict
3: pick $K_1, \ldots, K_T \in_D \mathcal{K}$
4: $w_i \leftarrow F(K_i)$ for $i = 1, \ldots, T$
5: $\mathcal{A}_2^F(\text{dict}, w_1, \ldots, w_T) \to (i, k)$
6: **return** $1_{F(k)=w_i}$

$\mathcal{A}_1^F$: (preprocessing)

1: **for** M candidates k **do**
2: compute $F(k)$
3: dict$\{F(k)\} \leftarrow k$
4: **end for**
5: **return** dict

$\mathcal{A}_2^F(\text{dict}, w_1, \ldots, w_T)$: (attack)

6: **for** $i = 1$ to T **do**
7: **if** dict$\{w_i\}$ exists **then**
8: **return** dict$\{w_i\}$
9: **end if**
10: **end for**
11: **return** $\perp$

Complexity Analysis

Precomputation time M

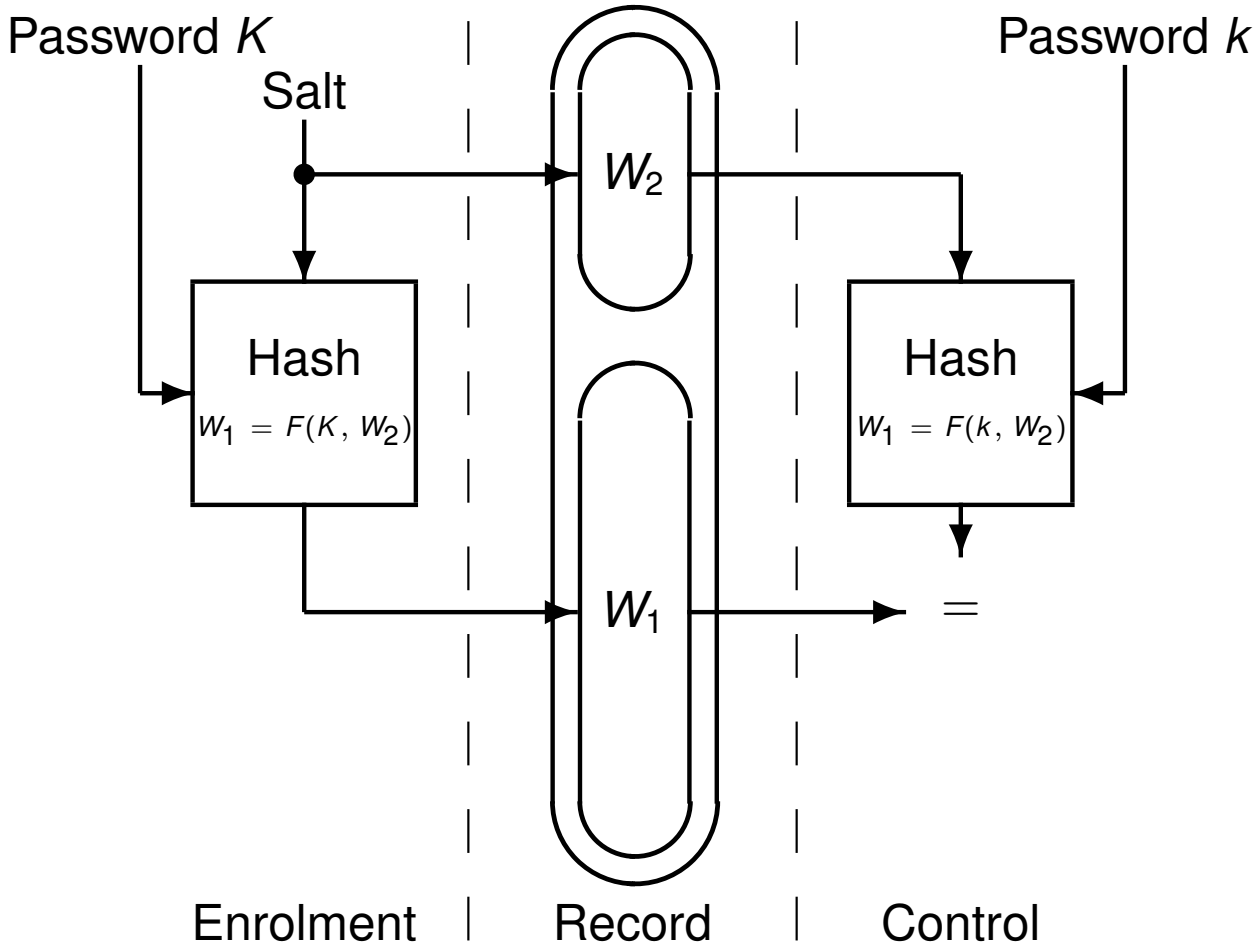
Memory complexity M

Time complexity T

Probability of success $1 - \left(1 - \frac{M}{N}\right)^T \approx 1 - e^{-\frac{MT}{N}}$

This is quite interesting when $M \approx T \approx \sqrt{N}$...

Password Recovery from a Salted Password Hash



Offline Inversion Attack with Salt

Input: a set of possible keys $\mathcal{K} = \{k_1, \ldots, k_N\}$, 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: shuffle $\mathcal{K}$ with a random permutation
2: **for** all $i = 1$ to N **do**
3: **if** $F(k_i, W_2) = W_1$ **then**
4: yield k_i and stop
5: **end if**
6: **end for**
7: search failed

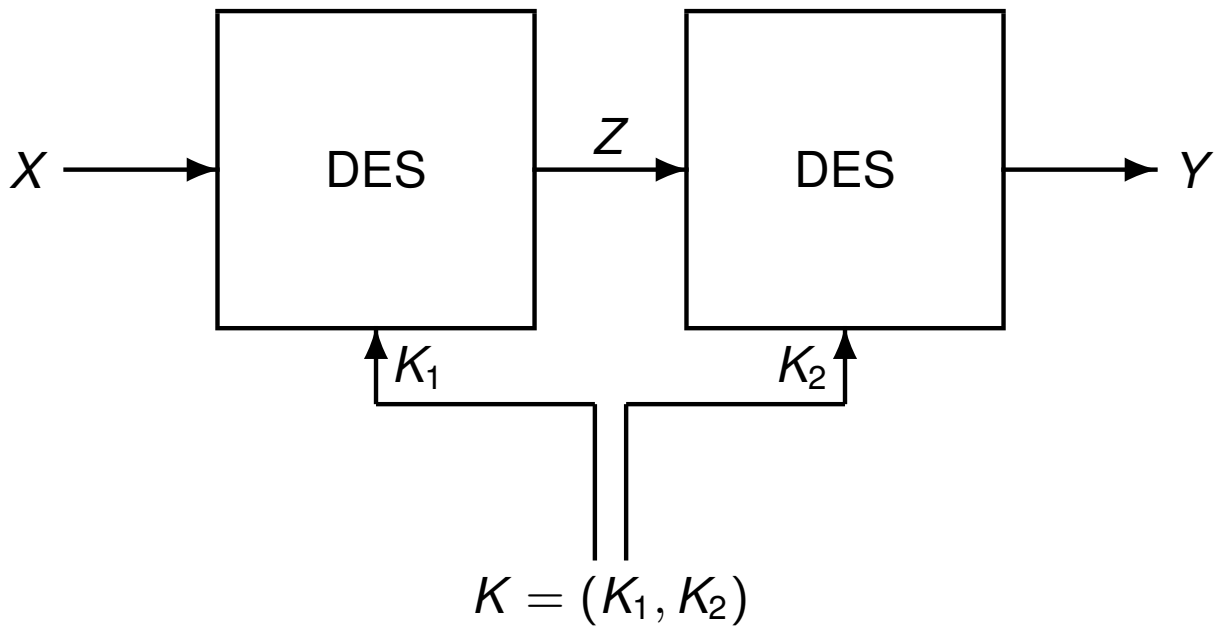
The Role of Salt

- mitigates dictionary attacks (or dictionaries are bigger)
- mitigates tradeoffs
- mitigates multitarget attacks

5 Symmetric Encryption

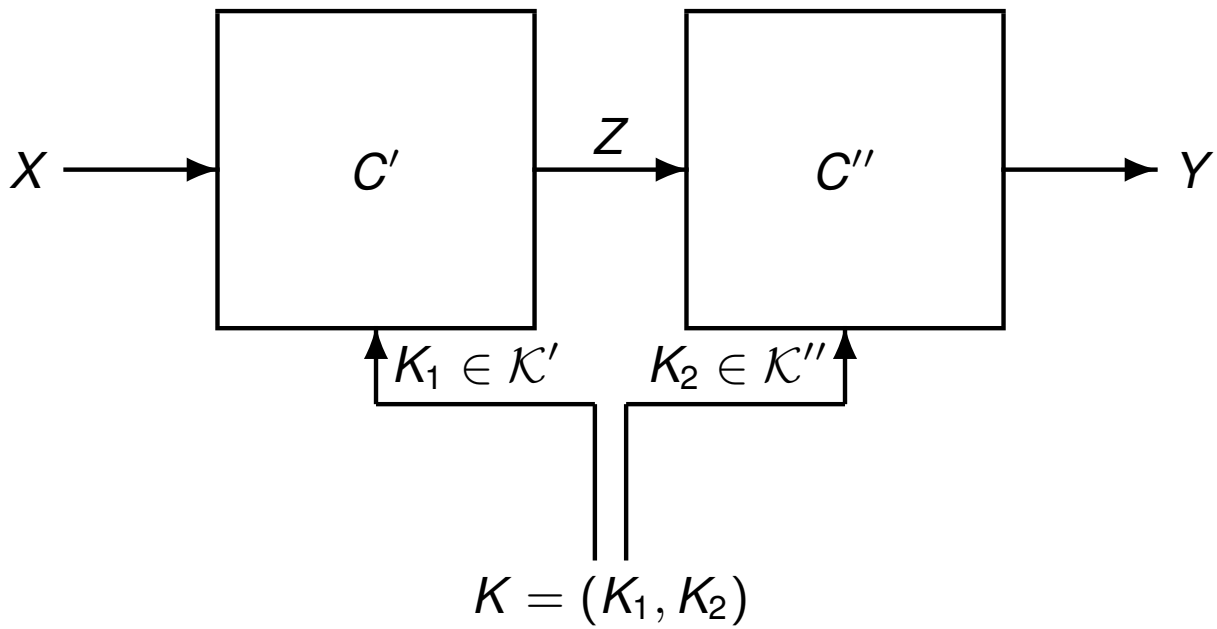
- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

Double DES



this is not much more secure than single DES

Double Encryption



this is not much more secure than single encryption

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 k_1 in dict$\{z\}$
4: **end for**
5: **for** all $k_2 \in \mathcal{K}''$ **do**
6: compute $z = {C''}^{-1}_{k_2}(y)$
7: **for** all k_1 in dict$\{z\}$ **do**
8: yield (k_1, k_2) as a possible key
9: **end for**
10: **end for**

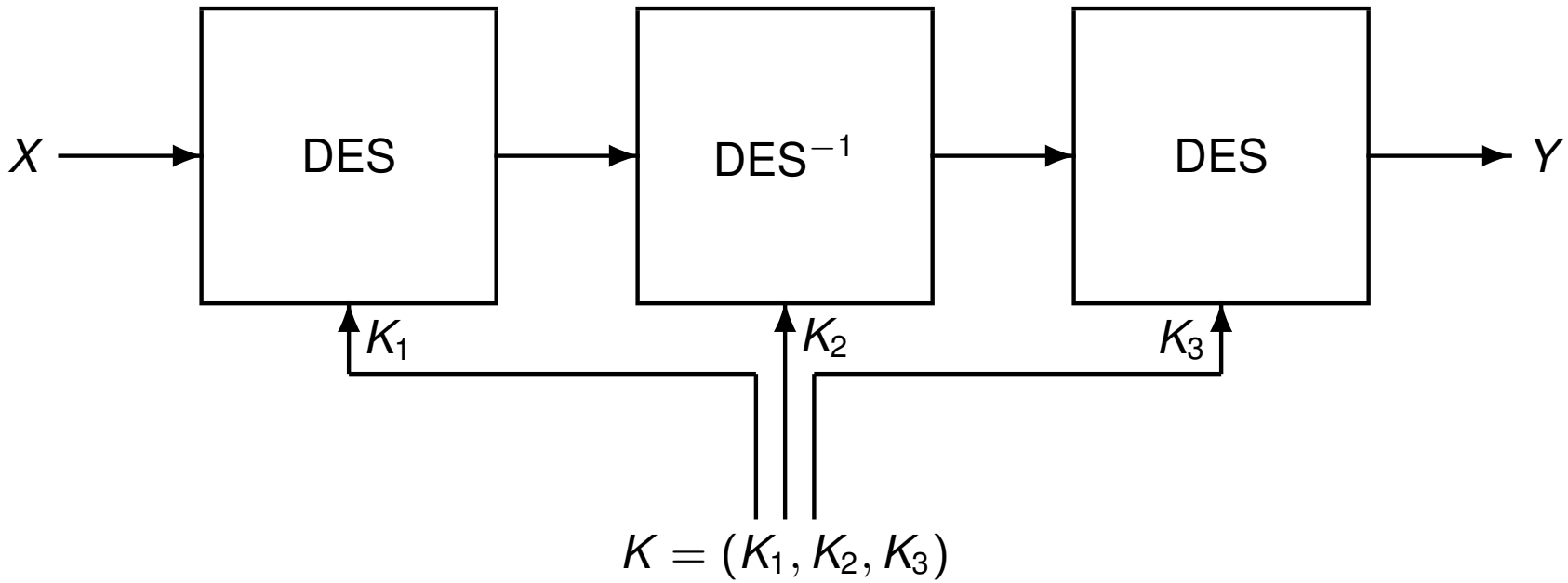
Complexity Analysis

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



a 3DES chip can do

- 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: ℓ, m, t
Preprocessing

1: **for** $s = 1$ to ℓ **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: $T_s\{k\} \leftarrow k'$
10: **end for**
11: **end for**

Precomputed Tables

$$
T_1 : \begin{bmatrix} k^1_{1,0} \\ k^1_{2,0} \\ k^1_{3,0} \\ \vdots \\ k^1_{m,0} \end{bmatrix}
\overset{f_1}{\mapsto}
\begin{bmatrix} k^1_{1,1} \\ k^1_{2,1} \\ k^1_{3,1} \\ \vdots \\ k^1_{m,1} \end{bmatrix}
\overset{f_1}{\mapsto}
\begin{bmatrix} k^1_{1,2} \\ k^1_{2,2} \\ k^1_{3,2} \\ \vdots \\ k^1_{m,2} \end{bmatrix}
\overset{f_1}{\mapsto}
\begin{bmatrix} k^1_{1,3} \\ k^1_{2,3} \\ k^1_{3,3} \\ \vdots \\ k^1_{m,3} \end{bmatrix}
\overset{f_1}{\mapsto} \cdots \overset{f_1}{\mapsto}
\begin{bmatrix} k^1_{1,t-1} \\ k^1_{2,t-1} \\ k^1_{3,t-1} \\ \vdots \\ k^1_{3,t-1} \end{bmatrix}
\overset{f_1}{\mapsto}
\begin{bmatrix} k^1_{1,t} \\ k^1_{2,t} \\ k^1_{3,t} \\ \vdots \\ k^1_{m,t} \end{bmatrix}
\Rightarrow
\begin{matrix} (k^1_{1,t}, k^1_{1,0}) \\ (k^1_{2,t}, k^1_{2,0}) \\ (k^1_{3,t}, k^1_{3,0}) \\ \vdots \\ (k^1_{m,t}, k^1_{m,0}) \end{matrix}
$$

$$
\vdots
$$

$$
T_\ell : \begin{bmatrix} k^\ell_{1,0} \\ k^\ell_{2,0} \\ k^\ell_{3,0} \\ \vdots \\ k^\ell_{m,0} \end{bmatrix}
\overset{f_\ell}{\mapsto}
\begin{bmatrix} k^\ell_{1,1} \\ k^\ell_{2,1} \\ k^\ell_{3,1} \\ \vdots \\ k^\ell_{m,1} \end{bmatrix}
\overset{f_\ell}{\mapsto}
\begin{bmatrix} k^\ell_{1,2} \\ k^\ell_{2,2} \\ k^\ell_{3,2} \\ \vdots \\ k^\ell_{m,2} \end{bmatrix}
\overset{f_\ell}{\mapsto}
\begin{bmatrix} k^\ell_{1,3} \\ k^\ell_{2,3} \\ k^\ell_{3,3} \\ \vdots \\ k^\ell_{m,3} \end{bmatrix}
\overset{f_\ell}{\mapsto} \cdots \overset{f_\ell}{\mapsto}
\begin{bmatrix} k^\ell_{1,t-1} \\ k^\ell_{2,t-1} \\ k^\ell_{3,t-1} \\ \vdots \\ k^\ell_{3,t-1} \end{bmatrix}
\overset{f_\ell}{\mapsto}
\begin{bmatrix} k^\ell_{1,t} \\ k^\ell_{2,t} \\ k^\ell_{3,t} \\ \vdots \\ k^\ell_{m,t} \end{bmatrix}
\Rightarrow
\begin{matrix} (k^\ell_{1,t}, k^\ell_{1,0}) \\ (k^\ell_{2,t}, k^\ell_{2,0}) \\ (k^\ell_{3,t}, k^\ell_{3,0}) \\ \vdots \\ (k^\ell_{m,t}, k^\ell_{m,0}) \end{matrix}
$$

Time-Memory Tradeoffs — ii

Attack

Attack input: $W = F(K)$

```
 1: for s = 1 to ℓ do
 2:     set i to 0
 3:     set k to R_s(W)
 4:     while T_s{k} does not exist and i < t do
 5:         increment i
 6:         k ← f_s(k)
 7:     end while
 8:     if T_s{k} exists then
 9:         k' ← T_s{k}
10:         while F(k') ≠ W and i < t do
11:             increment i
12:             k' ← f_s(k')
13:         end while
14:         if F(k') = W then
15:             yield k' as a possible key
16:         end if
17:     end if
18: end for
```

Complexity Analysis

$N = \#$ output range of F

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 $N^{\frac{2}{3}}$

5 Symmetric Encryption

- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle Bruteforce Inversion Algorithms
- Pushing the Physical Limits
- Formalism

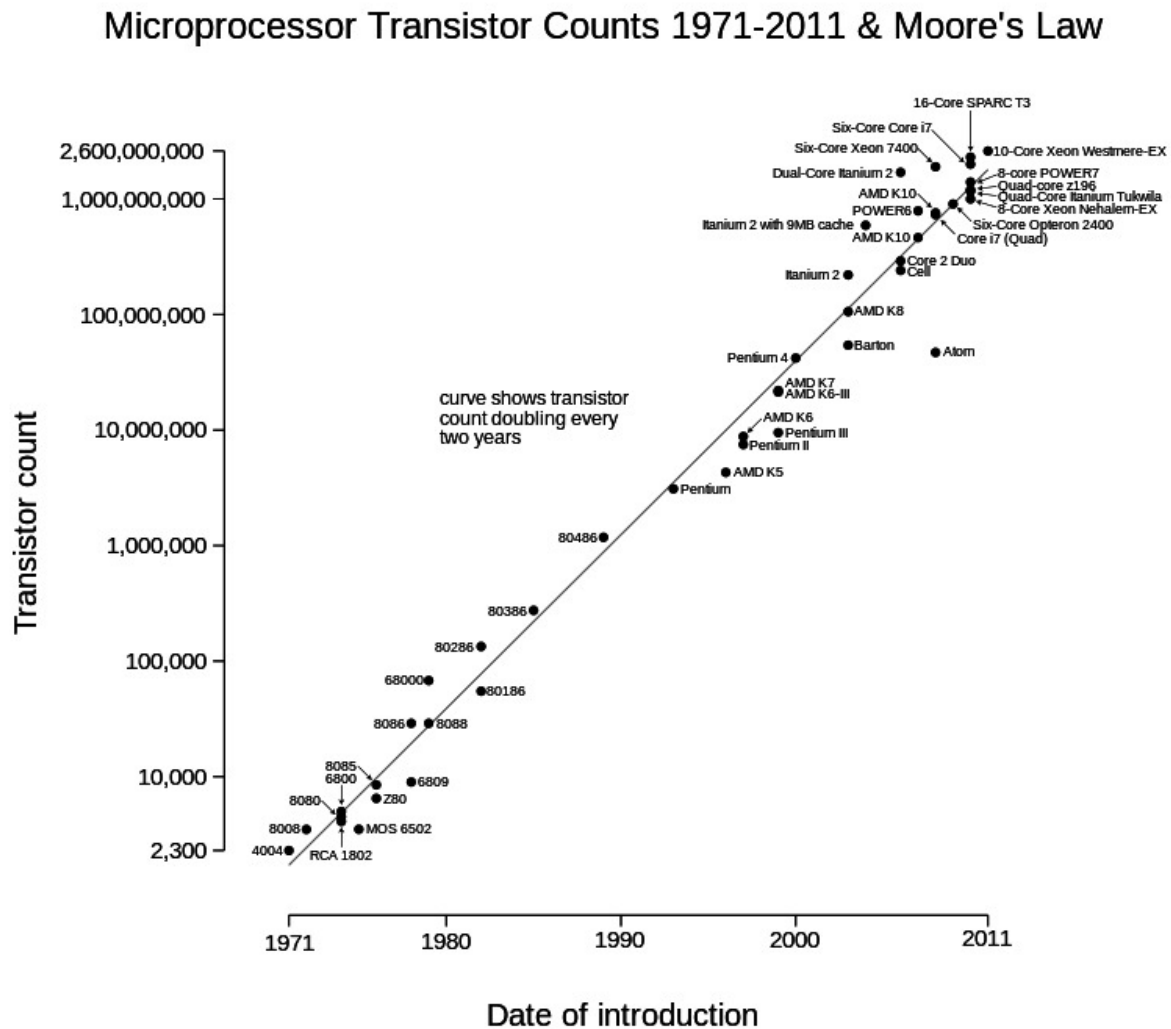
Order of Magnitudes

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: 14BY$= 14 \times 10^9$Y$\approx 440 \times 10^{15}$s
- # machines to do the exhaustive search within 14BY: 115×10^{12}

Moore's Law

- Gordon Moore 1965: # transistors / IC doubles every year
- Gordon Moore 1975: # transistors / IC doubles every 2 years
- popular version: speed of CPU's doubles every 18 months



Better Strategy (of Metaphysical Interest)

remember: slide 67

create the universe then take 14BY of vacations

humankind will create itself, invent computers, and solve the problem

Energy Bill

- 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
- we can compute without burning energy! [Bennett 1973]
  need supraconductors and invertible computation gates

  but all computations must be invertible!

  exhaustive search must keep lots of garbage in memory

Fully Reversible Exhaustive Search

- define

$$\begin{array}{lrcl} \text{INC:} & \langle x, y, k, z, t\rangle & \mapsto & \langle x, y, k+1, z, t\rangle \\ \text{ENC:} & \langle x, y, k, z, t\rangle & \mapsto & \langle x, y, k, z \oplus \text{Enc}(k, x), t\rangle \\ \text{CMP:} & \langle x, y, k, z, t\rangle & \mapsto & \langle x, y, k, z, t \oplus k \cdot 1_{y=z}\rangle \end{array}$$

- The sequence ENC, CMP, ENC, INC does

$$\begin{array}{rcl} \langle x, y, k, z, t\rangle & \mapsto & \langle x, y, k+1, z, t \oplus k \cdot 1_{y=z\oplus\text{Enc}(k,x)}\rangle \\ \langle x, y, k, 0, t\rangle & \mapsto & \langle x, y, k+1, 0, t \oplus k \cdot 1_{y=\text{Enc}(k,x)}\rangle \end{array}$$

- If we do it 2^{128} times on $\langle x, y, 0, 0, 0\rangle$, we obtain the XOR of all keys such that $y = \text{Enc}(k, x)$.
- assuming K is unique, we get $\langle x, y, 0, 0, K\rangle$

Grover Algorithm

- if the input domain of F has size N and y has a unique preimage
- the Grover algorithm finds $F^{-1}(y)$ in complexity $\mathcal{O}(\sqrt{N})$ F-equivalent evaluations
- it only runs on a quantum computer
- this may be a motivation to use 256-bit AES

5 Symmetric Encryption

- A Cryptographic Primitive
- Galois Fields
- Block Ciphers
- Stream Ciphers
- Bruteforce Inversion Algorithms
- Subttle 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 $C_K(\cdot) = \text{Enc}(K, .)$ and $C_K^{-1}(\cdot) = \text{Dec}(K, .)$.

(operate on bitstrings)

Remark: for all K, $X \mapsto \text{Enc}(K, X)$ is a permutation of $\{0,1\}^n$

Leaking the Length

- it is ok to leak the length $|X|$ of X (better be aware)
- ideally, encryption should be length-preserving: $|Y| = |X|$

Variable-Length Symmetric Encryption

Definition

A **(variable-length, length-preserving) symmetric encryption scheme** is a tuple $(\{0,1\}^k, \mathcal{D}, \mathrm{Enc}, \mathrm{Dec})$ 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

$$\forall K \in \{0,1\}^k \quad \forall X \in \mathcal{D} \quad \begin{cases} \mathrm{Dec}(K, \mathrm{Enc}(K, X)) &= X \\ |\mathrm{Enc}(K, X)| &= |X| \end{cases}$$

Write $C_K(\cdot) = \mathrm{Enc}(K, .)$ and $C_K^{-1}(\cdot) = \mathrm{Dec}(K, .)$.

$\rightarrow$ 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 $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathsf{Enc}, \mathsf{Dec})$ 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

$$\forall K \in \{0,1\}^k \quad \forall X \in \mathcal{D} \quad \forall N \in \mathcal{N} \quad \begin{cases} \mathsf{Dec}(K,N,\mathsf{Enc}(K,N,X)) = X \\ |\mathsf{Enc}(K,N,X)| = |X| \end{cases}$$

N is supposed to be used only once for encryption
random nonce (beware of random repetitions), counter, sent in clear or synchronized
$\rightarrow$ 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 **secure against key recovery under chosen plaintext attacks (CPA)** if for any algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns } 1]$$

Game

1: $K \xleftarrow{\$} \{0,1\}^k$
2: Used $\leftarrow \emptyset$
3: $\mathcal{A}^{\text{OEnc}} \rightarrow K'$
4: **return** $1_{K=K'}$

Oracle OEnc(N, X):

5: **if** $N \in$ Used **then return** $\perp$
6: Used $\leftarrow$ Used $\cup \{N\}$
7: **return** Enc(K, N, X)

CPCA Security against Key Recovery

Definition

A symmetric encryption scheme $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$ is **secure against key recovery under chosen plaintext/ciphertext attacks (CPCA)** if for any algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

1: $K \xleftarrow{\$} \{0,1\}^k$
2: Used $\leftarrow \emptyset$
3: $\mathcal{A}^{\text{OEnc},\text{ODec}} \rightarrow K'$
4: **return** $1_{K=K'}$

Oracle OEnc(N, X):

5: **if** $N \in$ Used **then return** $\perp$
6: Used $\leftarrow$ Used $\cup \{N\}$
7: **return** Enc(K, N, X)

Oracle ODec(N, Y):

8: **return** Dec(K, N, Y)

About Nonce Reuse in Decryption

- the sender who chooses the nonce can make sure it does not repeat
  example: a counter (stateful)
  example: time
  example: a random nonce (large enough)
- the receiver cannot enforce non-reuse of nonces

CPCA Security is Stronger than CPA Security

- assume we have CPCA security
- to prove CPA security, consider a CPA adversary $\mathcal{A}$
- we define a CPCA adversary $\mathcal{B} = \mathcal{A}$
  (same adversary who just never use decryption queries)
- $\mathcal{B}$ and $\mathcal{A}$ have the same advantage
- since the one of $\mathcal{B}$ is bounded by ε, the one of $\mathcal{A}$ as well $\square$

$$\begin{aligned} \text{CPA-breaking} &\implies \text{CPCA-breaking} \\ \text{CPCA-secure} &\implies \text{CPA-secure} \end{aligned}$$

Not Good Enough Security

Not good enough security: what follows is correct and secure!

- $\mathrm{Enc}(K, N, X) = X$
- $\mathrm{Dec}(K, N, Y) = Y$

Security against Decryption

Definition

A symmetric encryption scheme $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathrm{Enc}, \mathrm{Dec})$ is (q, t, ε)-**secure against decryption** under CPA resp. CPCA if for any algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε.

$$\mathrm{Adv} = \Pr[\text{game returns 1}]$$

Game

1: $K \xleftarrow{\$} \{0,1\}^k$
2: $X_0 \xleftarrow{\$} \mathcal{D}$, $N_0 \xleftarrow{\$} \mathcal{N}$
3: $\mathrm{Used} \leftarrow \{N_0\}$
4: $Y_0 \leftarrow \mathrm{Enc}(K, N_0, X_0)$
5: $\mathcal{A}^{\mathrm{OEnc},\mathrm{ODec}}(N_0, Y_0) \to X$
6: **return** $1_{X=X_0}$

Oracle $\mathrm{OEnc}(N, X)$:

1: **if** $N \in \mathrm{Used}$ **then return** $\perp$
2: $\mathrm{Used} \leftarrow \mathrm{Used} \cup \{N\}$
3: **return** $\mathrm{Enc}(K, N, X)$

Oracle $\mathrm{ODec}(N, Y)$:

4: **if** $(N, Y) = (N_0, Y_0)$ **then return** $\perp$
5: **return** $\mathrm{Dec}(K, N, Y)$

Example: Vernam-Based Ciphers

$$\mathrm{Enc}(K, N, X) = X \oplus \mathrm{PRNG}(K, N)$$

CPCA decryption attack:

$\mathcal{A}(N_0, Y_0)$:

1: pick Y' of same length as Y_0
2: query $\mathrm{ODec}(N_0, Y') \to X'$
3: $X \leftarrow Y_0 \oplus Y' \oplus X'$
4: **return** X

Decryption Security is Stronger than Key Recovery Security

$\mathcal{B}^{\cdots}(Y_0)$
1: $\mathcal{A}^{\cdots} \to K'$
2: $X \leftarrow \mathrm{Dec}(K', Y_0)$
3: **return** X

(nonce-less cipher here)

- assume decryption security
- consider a key recovery adversary $\mathcal{A}$
- define a decryption adversary $\mathcal{B}$...

$$\Pr\left[\left[\begin{array}{l}\textbf{KR}*_{\mathcal{A}}\text{ game:}\\ \text{pick } K\\ \mathcal{A}^{\cdots} \to K'\\ \textbf{return } 1_{K=K'}\end{array}\right] \to 1\right] \leq \Pr\left[\left[\begin{array}{l}\textbf{OW}*_{\mathcal{B}}\text{ game:}\\ \text{pick } K\\ \text{pick } X_0\\ Y_0 \leftarrow \mathrm{Enc}(K, X_0)\\ // \ \mathcal{B}^{\cdots}(Y_0) \to X:\\ \mathcal{A}^{\cdots} \to K'\\ X \leftarrow \mathrm{Dec}(K', Y_0)\\ \textbf{return } 1_{X=X_0}\end{array}\right] \to 1\right] \leq \varepsilon$$

□

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

$$\begin{aligned} \mathrm{Enc}(K, N, X) &= \mathrm{Enc}_0(K, N, \mathrm{lefthalf}(X)) \| \mathrm{righthalf}(X) \\ \mathrm{Dec}(K, N, Y) &= \mathrm{Dec}_0(K, N, \mathrm{lefthalf}(Y)) \| \mathrm{righthalf}(Y) \end{aligned}$$

Exercise:
If $(\{0,1\}^k, \mathcal{D}_0, \mathcal{N}, \mathrm{Enc}_0, \mathrm{Dec}_0)$ is secure, then
$(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathrm{Enc}, \mathrm{Dec})$ is secure.

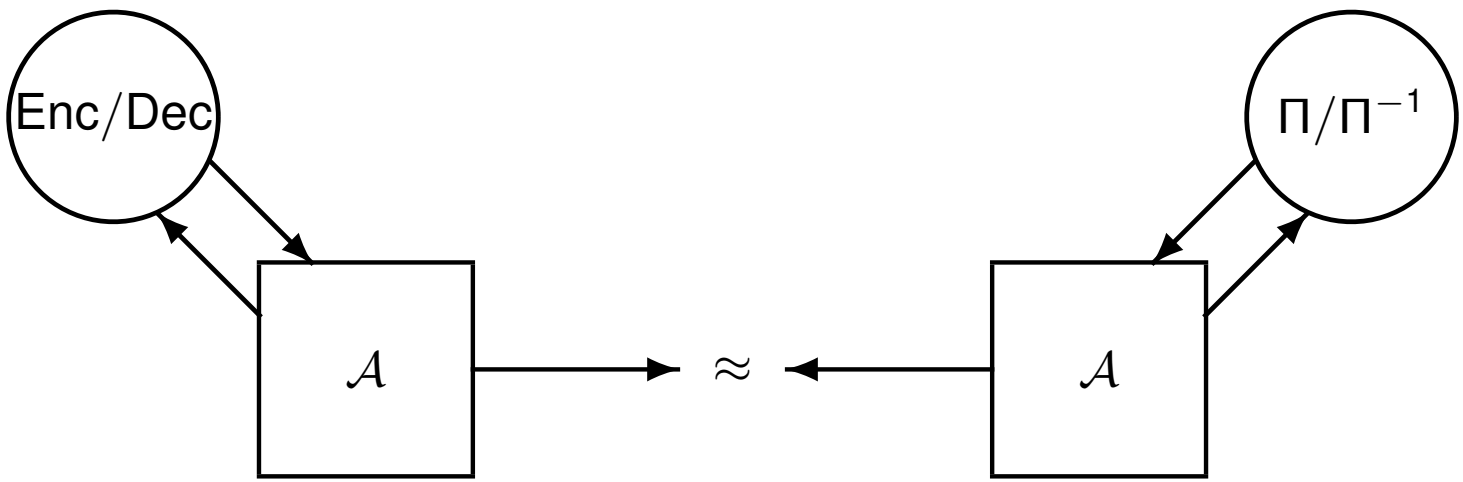
The Ideal Cipher

(assume Enc : $\{0,1\}^k \times \mathcal{N} \times \mathcal{D} \to \mathcal{D}$)

- the "ideal cipher": taking K random is equivalent to picking a random length-preserving permutation Π_N over $\mathcal{D}$ for every N

$$\begin{aligned} \mathrm{Enc}(K,N,X) &= \Pi_N(X) \\ \mathrm{Dec}(K,N,Y) &= \Pi_N^{-1}(Y) \end{aligned}$$

- 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 $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathrm{Enc}, \mathrm{Dec})$ is (q, t, ε)-**secure against distinguishers** under CPA resp. CPCA if for any algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε.

$$\mathrm{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game Γ_b

1: $K \xleftarrow{\$} \{0,1\}^k$
2: for every N, pick a length-preserving permutation Π_N over $\mathcal{D}$
3: Used $\leftarrow \emptyset$
4: $\mathcal{A}^{\mathrm{OEnc},\mathrm{ODec}} \to z$
5: **return** z

Oracle OEnc(N, X):

1: **if** $N \in$ Used **then return** $\perp$
2: Used $\leftarrow$ Used $\cup \{N\}$
3: **if** $b = 0$ **then return** $\Pi_N(X)$
4: **return** $\mathrm{Enc}(K, N, X)$

Oracle ODec(N, Y):

5: **if** $b = 0$ **then return** $\Pi_N^{-1}(Y)$
6: **return** $\mathrm{Dec}(K, N, Y)$

Security against Distinguisher (Equivalent Form)

Definition

A symmetric encryption scheme $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \mathrm{Enc}, \mathrm{Dec})$ is (q, t, ε)-**secure against distinguishers** under CPA resp. CPCA if for any algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv' is bounded by ε.

$$\mathrm{Adv}' = \Pr[\Gamma \text{ returns } 1] - \frac{1}{2}$$

Game Γ

1: pick $b \in \{0,1\}$
2: $K \xleftarrow{\$} \{0,1\}^k$
3: for every N, pick a length-preserving permutation Π_N over $\mathcal{D}$
4: $\mathrm{Used} \leftarrow \emptyset$
5: $\mathcal{A}^{\mathrm{OEnc},\mathrm{ODec}} \to z$
6: **return** $1_{z=b}$

Oracle $\mathrm{OEnc}(N, X)$:

1: **if** $N \in \mathrm{Used}$ **then return** $\perp$
2: $\mathrm{Used} \leftarrow \mathrm{Used} \cup \{N\}$
3: **if** $b = 0$ **then return** $\Pi_N(X)$
4: **return** $\mathrm{Enc}(K, N, X)$

Oracle $\mathrm{ODec}(N, Y)$:

5: **if** $b = 0$ **then return** $\Pi_N^{-1}(Y)$
6: **return** $\mathrm{Dec}(K, N, Y)$

Equivalence

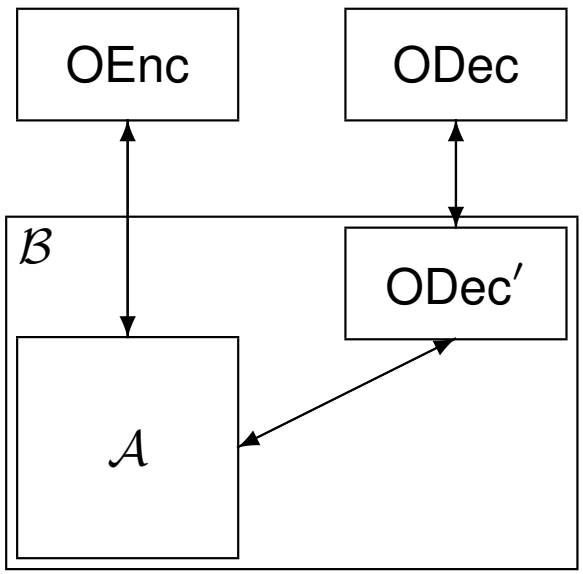
$$
\begin{aligned}
\mathsf{Adv}' &= \Pr[\Gamma \to 1] - \frac{1}{2} \\
&= \frac{1}{2}\Pr[\Gamma \to 1 | b = 1] + \frac{1}{2}\Pr[\Gamma \to 1 | b = 0] - \frac{1}{2} \\
&= \frac{1}{2}\Pr[\Gamma_1 \to 1] + \frac{1}{2}\Pr[\Gamma_0 \to 0] - \frac{1}{2} \\
&= \frac{1}{2}\Pr[\Gamma_1 \to 1] + \frac{1}{2}\left(1 - \Pr[\Gamma_0 \to 1]\right) - \frac{1}{2} \\
&= \frac{1}{2}\Pr[\Gamma_1 \to 1] - \frac{1}{2}\Pr[\Gamma_0 \to 1] \\
&= \frac{1}{2}\mathsf{Adv}
\end{aligned}
$$

Distinguisher Sec is Stronger than Decryption Sec

- assume distinguisher security
- consider a decryption adversary $\mathcal{A}$
- we define a distinguisher $\mathcal{B}$ as follows

$\mathcal{B}$:
1: $X_0 \xleftarrow{\$} \mathcal{D}$, $N_0 \xleftarrow{\$} \mathcal{N}$
2: $Y_0 \leftarrow \mathrm{OEnc}(N_0, X_0)$
3: run $\mathcal{A}^{\mathrm{OEnc},\mathrm{ODec}'}(N_0, Y_0) \to X$
4: **return** $1_{X_0=X}$

$\mathrm{ODec}'(N, Y)$:
5: **if** $(N, Y) = (N_0, Y_0)$ **then**
return $\perp$
6: **return** $\mathrm{ODec}(N, Y)$



- $\Pr[\Gamma_1^{\mathcal{B}} \to 1] = \Pr[\mathcal{A}_{\text{real}}$ wins decryption$]$
- $\Pr[\Gamma_0^{\mathcal{B}} \to 1] = \Pr[\mathcal{A}_{\text{ideal}}$ wins decryption$]$
- $\Pr[\mathcal{A}_{\text{ideal}}$ wins decryption$] \leq \frac{q+1}{\#\mathcal{D}-q}$ (believe me...)
- $\mathrm{Adv}_{\mathcal{B}} \leq \varepsilon$ hence $\mathrm{Adv}_{\mathcal{A}} \leq \varepsilon + \frac{q+1}{\#\mathcal{D}-q}$

□

Security Notions

	key recovery	decryption	distinguisher
CPA	weakest security		
CPCA			strongest 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

Conclusion

- **symmetric encryption**: stream ciphers (RC4, A5/1), block ciphers (DES, AES), modes of operation (ECB, CBC, OFB, 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
• small granularity (encrypt bits or bytes) • based on the Vernam cipher, requires a *nonce* (number to be used only once) • very high speed rate, very cheap on hardware • low confidence on security	• large granularity (encrypt blocks of 64 or 128 bits), require padding techniques for messages with arbitrary length • high rate, nice for software implementation, can be adapted to various platforms (8-bit, 32-bit, or 64-bit microprocessors) • well established security

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.

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

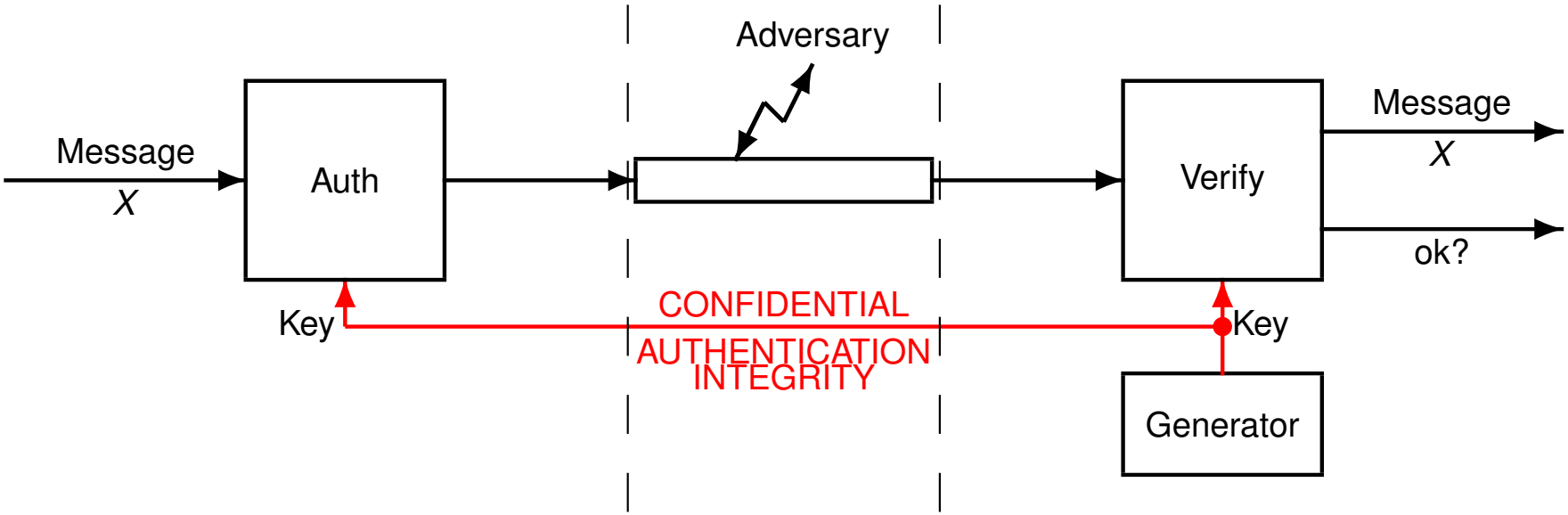
Train Yourself

- encryption:
  midterm exam 2011–12 ex3
  midterm exam 2012–13 ex3
  final exam 2013–14 ex1
- modes of operation:
  midterm exam 2009–10 ex3
  midterm exam 2011–12 ex1
- Moore's law: final exam 2008–09 ex1
- multitarget password recovery: final exam 2014–15 ex3
- meet-in-the-middle:
  midterm exam 2016–17 ex1
  midterm exam 2017–18 ex2
- security:
  midterm exam 2016–17 ex2
  final exam 2021–22 ex1
- design challenge: midterm exam 2018–19 ex1
- discrete logarithm in a small set: final exam 2021–22 ex3
- perfect secrecy except message length: midterm exam 2024–25 ex1

Roadmap

- hash functions: SHA-2, SHA-3
- message authentication codes: HMAC, KMAC, CBCMAC, WC-MAC
- other primitives: commitment, key derivation
- birthday paradox

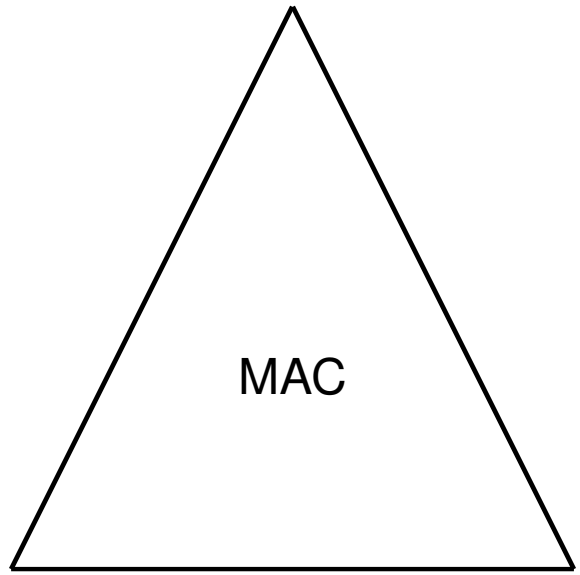
Message Authentication Code



Message Authentication Code (Informal)

Alice and Bob, Generator, Auth, Verify

components



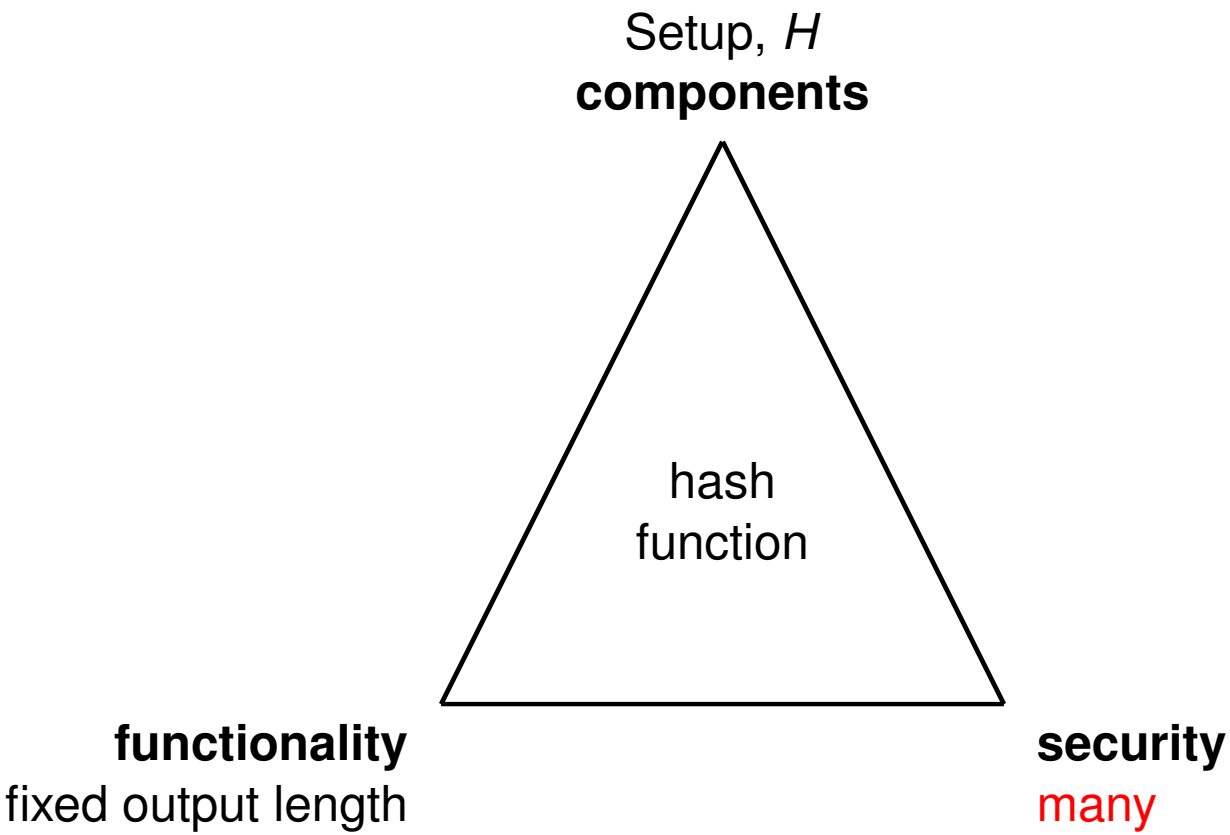
functionality

$\text{Verify}_K(\text{Auth}_K(X)) = (X, \text{ok})$

security

cannot forge

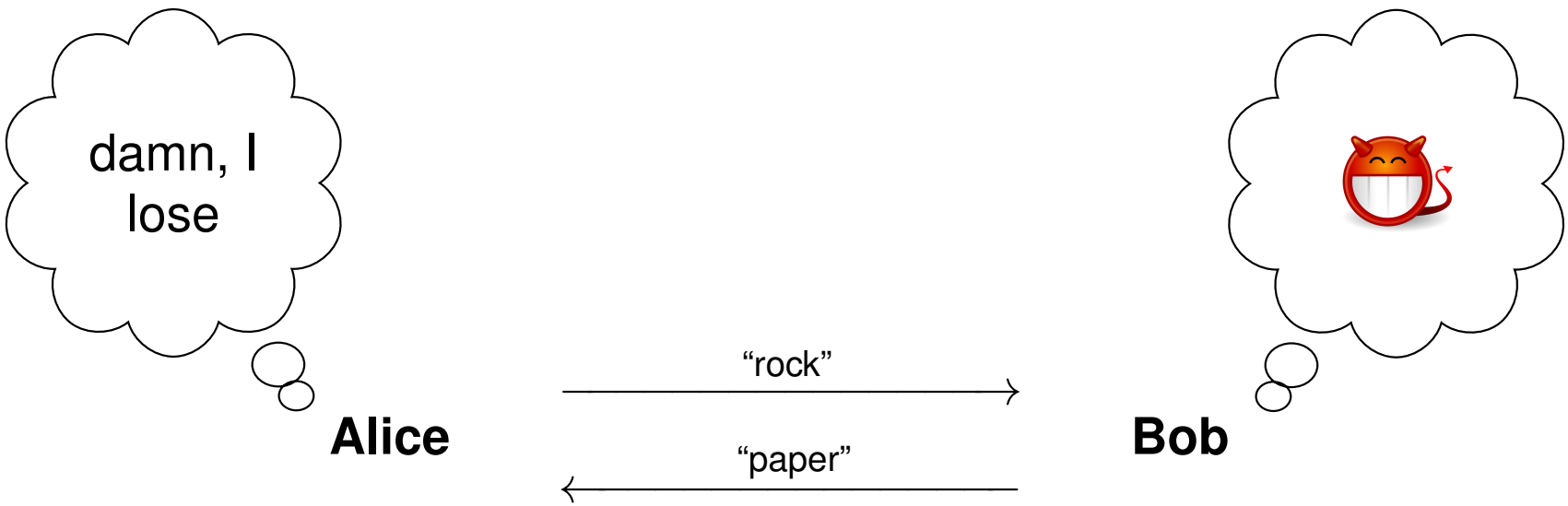
Hash Function (Informal)



6 Integrity and Authentication

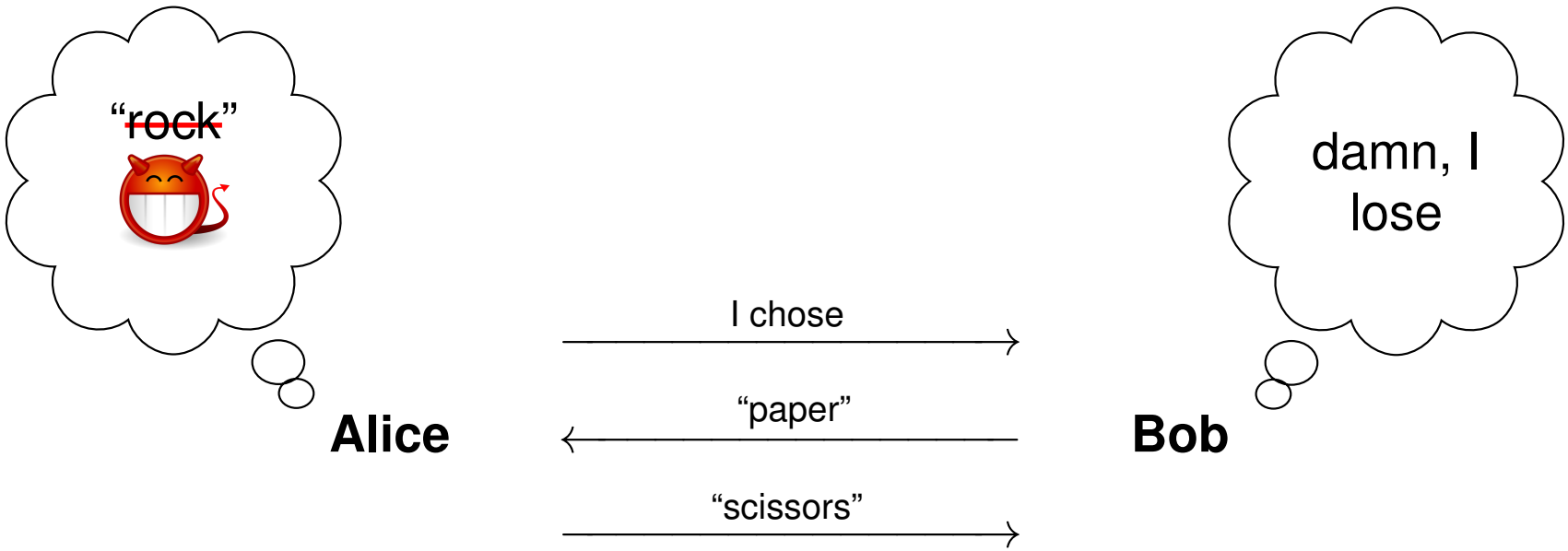
- Commitment Scheme
- Key Derivation Function and Pseudorandom Generator
- Cryptographic Hash Function
- Message Authentication Codes
- Formalism
- Bruteforce Collision Search Algorithms
- How to Select Security Parameters?
- Other Reasons why Security Collapses

Playing Rock-Paper-Scissors



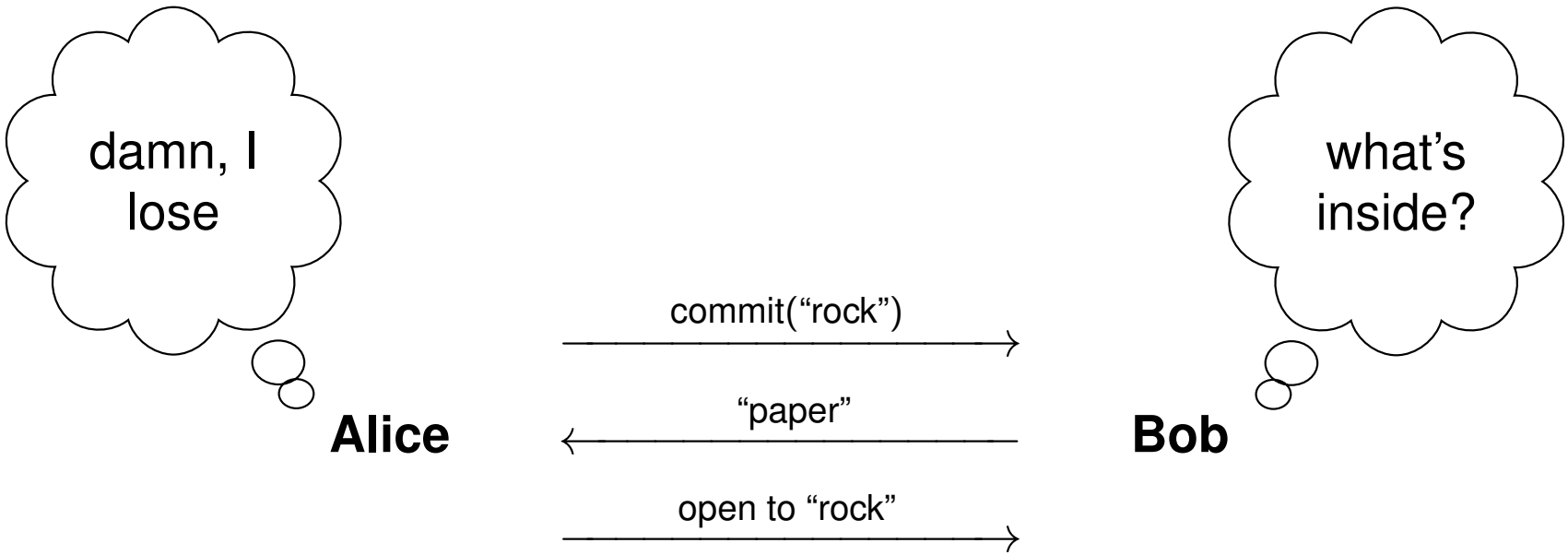
Bob should not see Alice’s move before making his choice

Playing Rock-Paper-Scissors



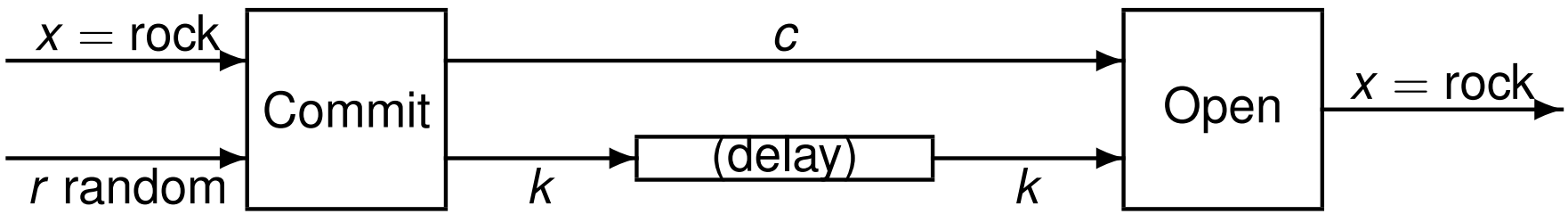
Alice should not be able to change her mind after Bob made his choice

Commitment to Play Rock-Paper-Scissors

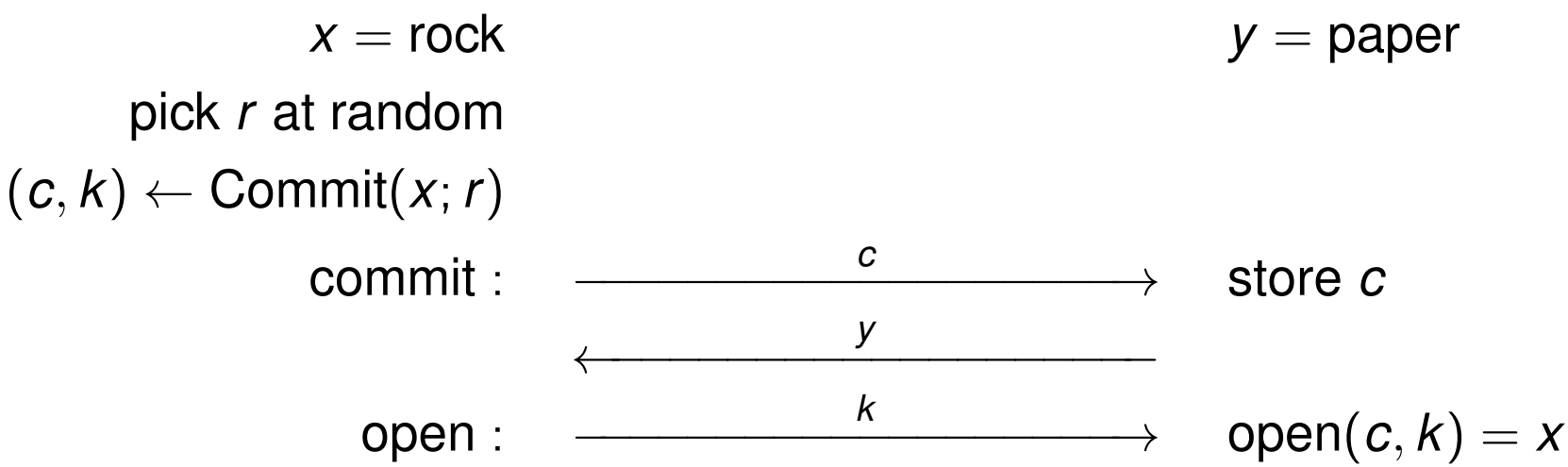


- cheat 1: Bob guesses Alice's play and adapts his own play to win
- cheat 2: Alice changes her play after seeing Bob's play

Commitment



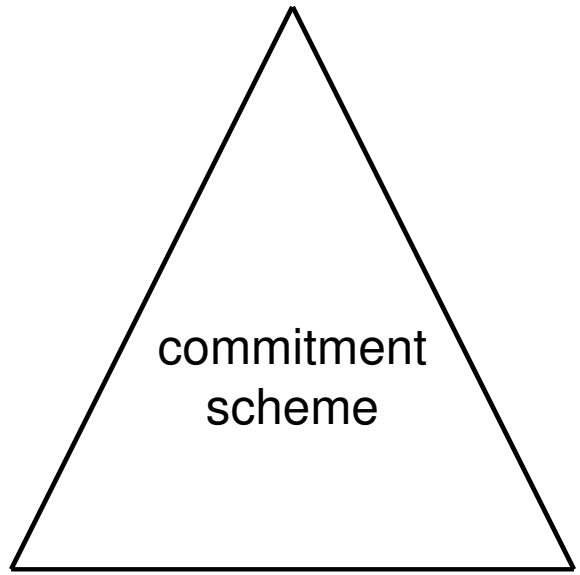
Using a Commitment Scheme



Commitment Scheme (Informal)

Alice and Bob, Setup, Commit, Open

components



functionality

if Commit$(X; r) = (c, k)$
then Open$(c, k) = X$

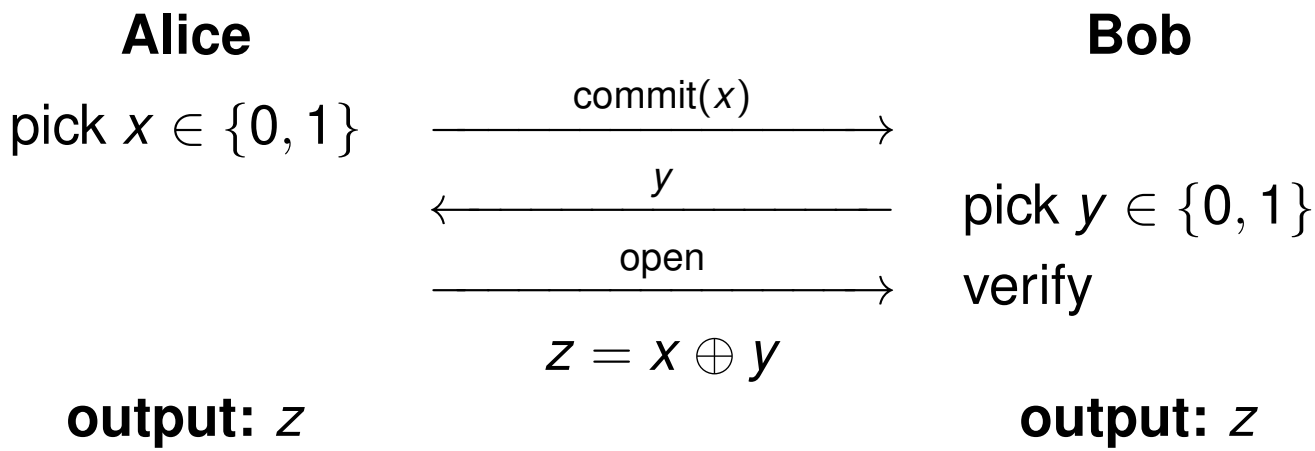
security

hiding, binding

- hiding: Bob does not get a clue on X from c
- binding: Alice cannot produce c, k, k' such that Open$(c, k) \neq$ Open(c, k')

Application Example: Tossing a Coin

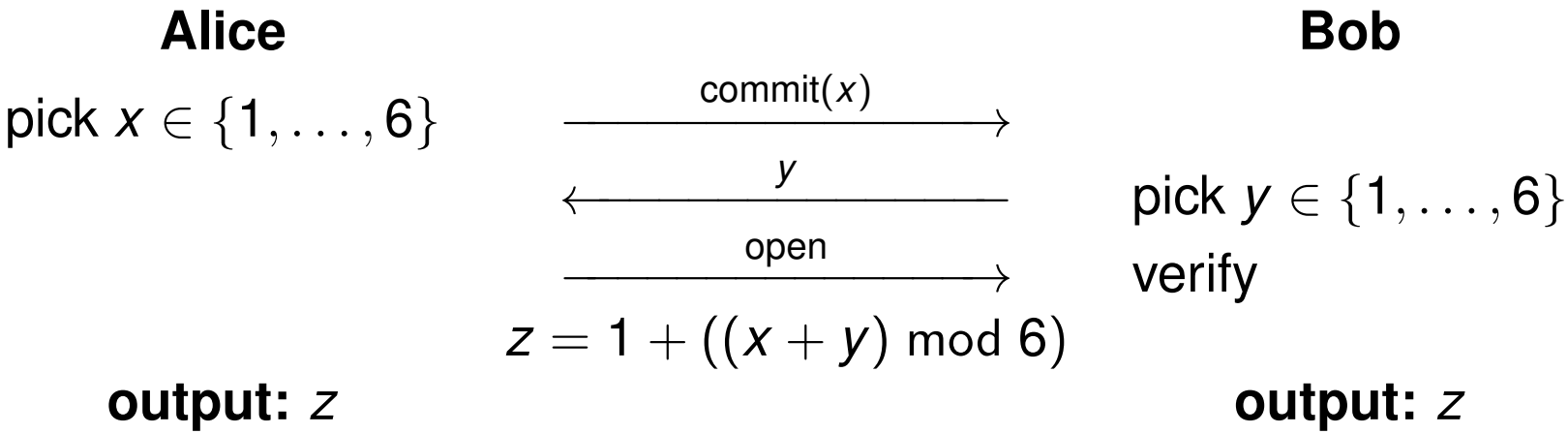
how to toss a coin:



z is the outcome of the tossed coin

Application Example: Playing Dice

how to throw a 6-face die:



z is the outcome of the thrown die

Examples

- a BAD one: $\text{Commit}(x; r) = (\text{Enc}_r(x), r)$
  (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)
  (the length of r must be fixed to avoid ambiguous encoding)

Pedersen Commitment (Based on DL)

setup generates two large primes p and q s.t. $q|(p-1)$, 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 $\text{Commit}(X; r) = g^X h^r \bmod p$ for $r \in_U \mathbf{Z}_q$

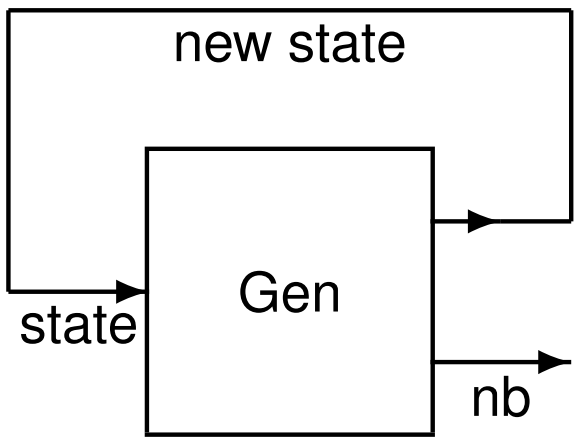
unconditionally hiding $\text{Commit}(X; r) = g^{X+ar}$ is uniformly distributed in $\langle g \rangle$ and independent of X

computationally binding commiting 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

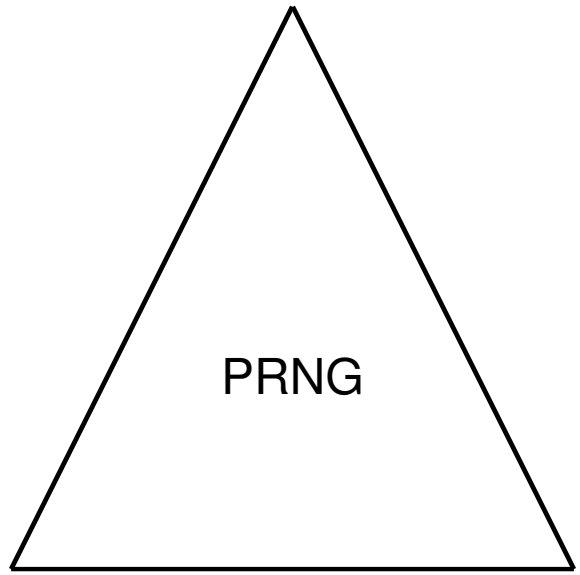
6 Integrity and Authentication

- 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

Pseudorandom Number Generator (PRNG)



PRNG (Informal)



functionality

$\text{Gen}(\text{state}) = (\text{nb}, \text{new state})$

security

indistinguishable from truly random

PRNG Examples

- stream ciphers: RC4, A5/1...
- block ciphers with mode of operation
  - OFB: state $=$ (key, block)
  - CTR: state $=$ (key, key)

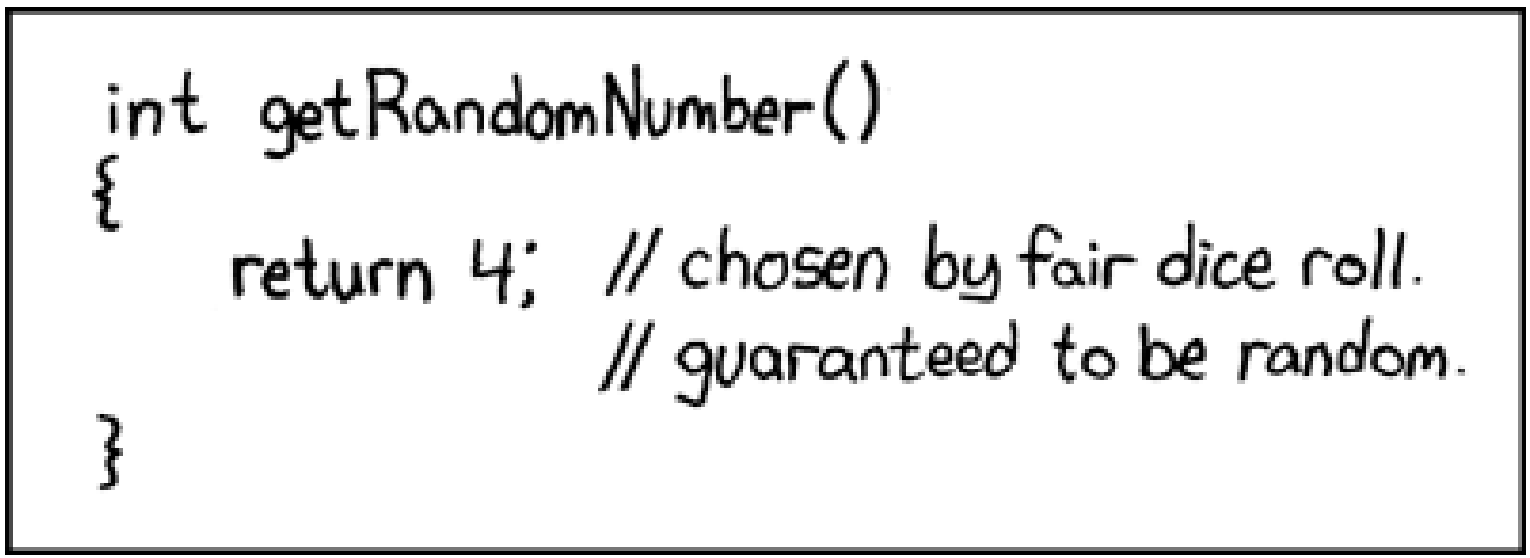
Famous Failure Cases

- 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/`

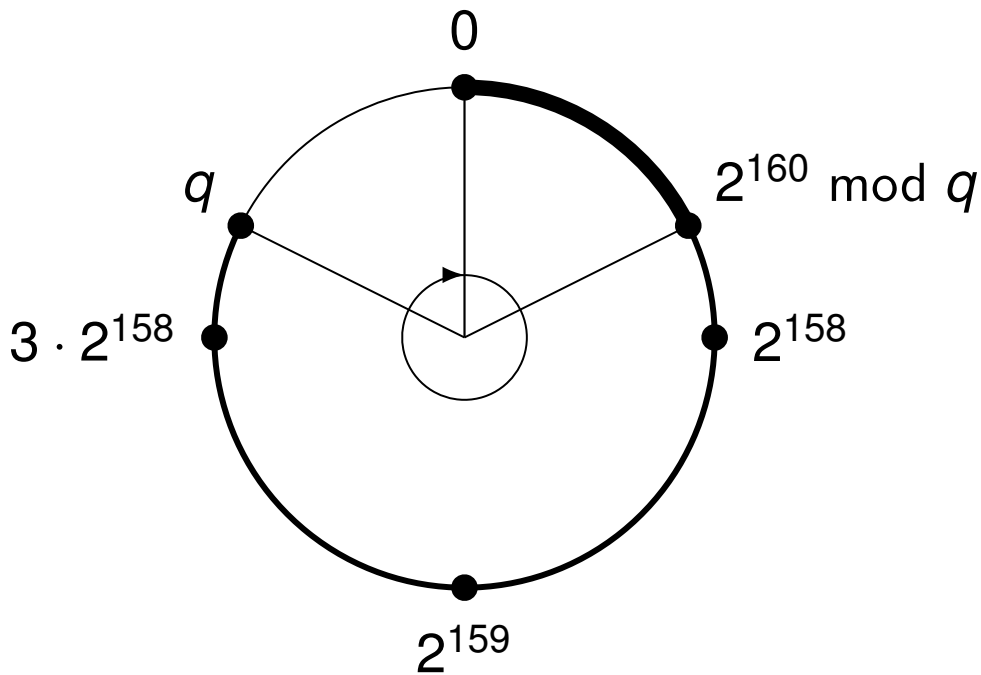


Possible Threats

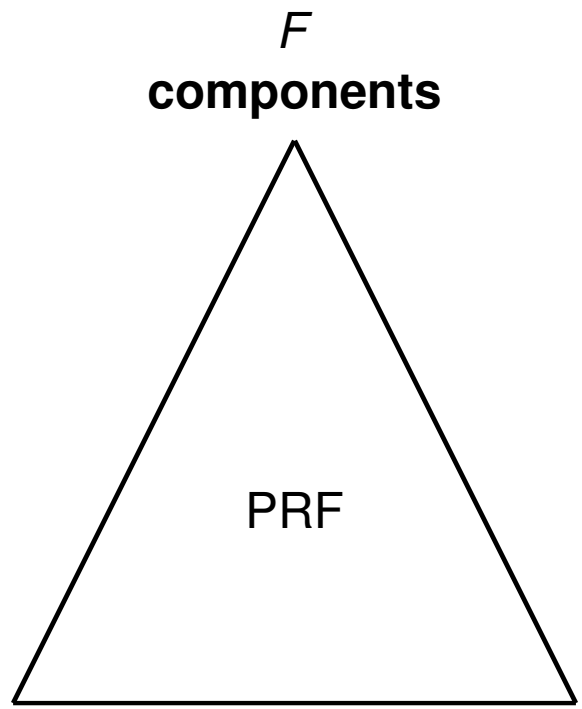
- bad coins during key generation makes the secret key guessable
- bad coins during signing may expose the secret key
  example: DSA, ECDSA
  stealing an ECDSA key may mean stealing all bitcoins...

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)

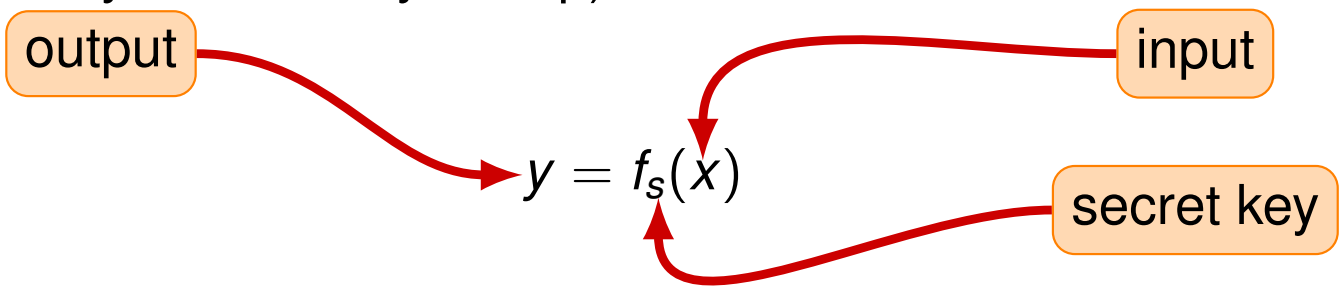


functionality
$F_K(\cdot)$

security
indistinguishable from truly random function (as a black-box)

PRF: PseudoRandom Function

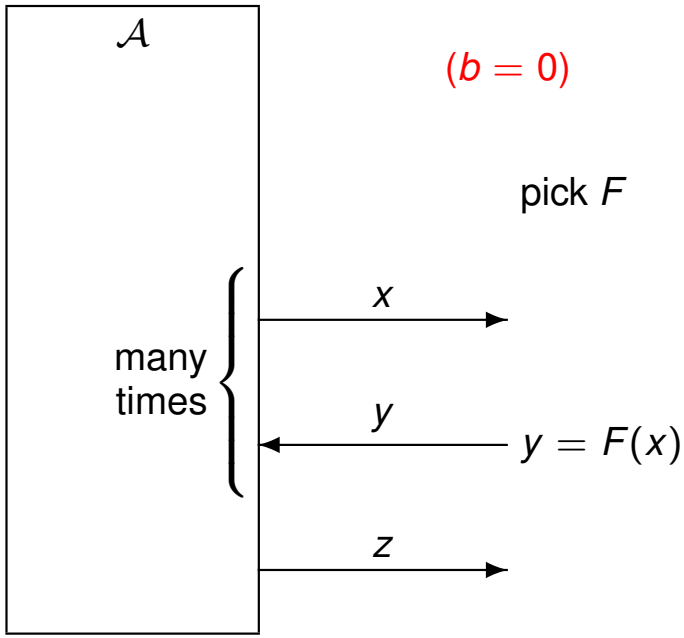
PRF: a deterministic function with a secret looking like random (when the key is randomly set up)

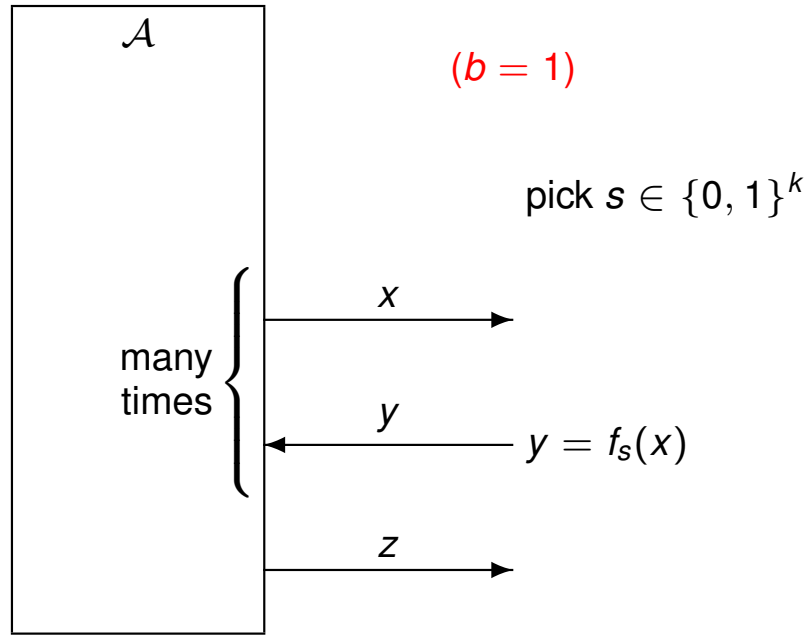


Application:

- **pseudorandom generator**: generation $= f_s(\text{counter})$
- **key generation**: key $= f_s(\text{nonces}, \text{params})$
- **encryption**: ct $=$ pt $\oplus f_s(\text{nonce})$
- **message authentication**: tag $= f_s(\text{message})$
- **peer authentication**: response $= f_s(\text{challenge})$
- ...

PRF Security Definition





$$\mathrm{Adv}(\mathcal{A}) = \Pr[z = 1 | b = 1] - \Pr[z = 1 | b = 0]$$

advantage

q queries, complexity t

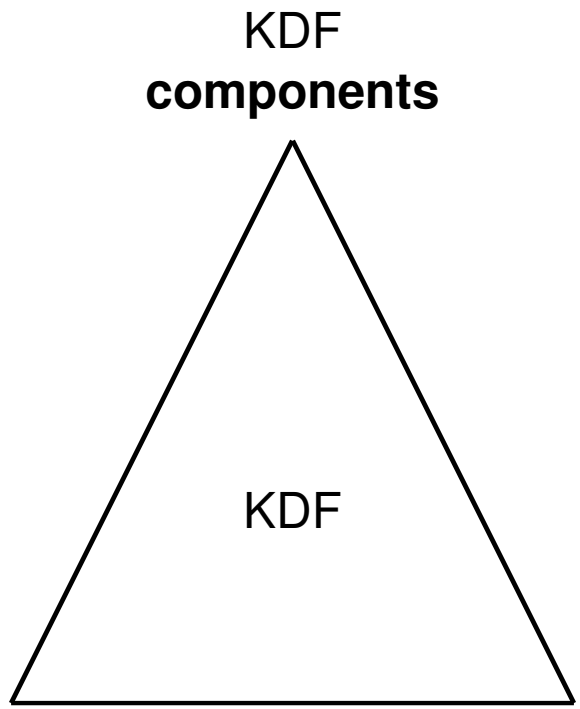
Definition

The PRF is (q, t, ε)-secure if for all bounded $\mathcal{A}$, $\mathrm{Adv}(\mathcal{A})$ is negligible.

less than ε

(game-based definition: see slide 674)

Key Derivation Function (KDF)



Generate some random key from some secret (password, output from key agreement protocols) and non-secret objects (salt, domain parameters, exchange messages)

KDF Examples

- typically: a hash function
- PKCS#5/RFC 2898
  example:

$$\text{PBKDF1}(\text{password}, \text{salt}, c, \ell) = \text{trunc}_\ell(H^c(\text{password} \| \text{salt}))$$

  where H^c is H iterated c times
  NB: ℓ shall not be larger than the H length
- HKDF (RFC 5869)

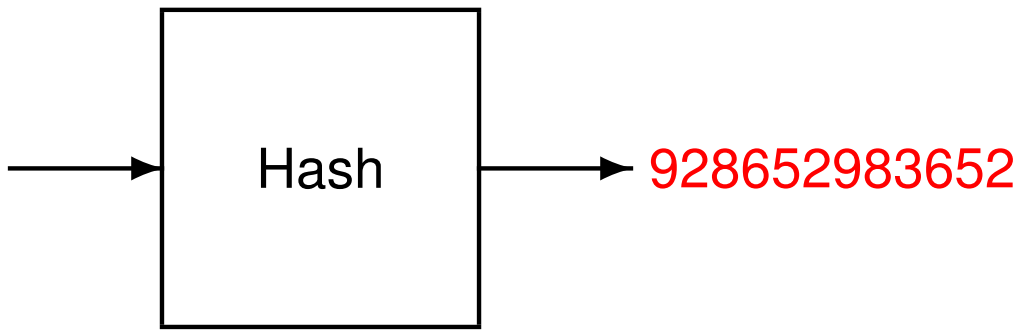
$$\begin{aligned}
\text{HKDF}(\text{salt}, \text{input}, \text{extra}, L) &= \text{trunc}_L\left(K_1 \| K_2 \| \cdots \| K_{\lceil \frac{L}{\text{HMAC_length}} \rceil}\right) \\
\text{PRK} &= \text{HMAC}_{\text{salt}}(\text{input}) \\
K_1 &= \text{HMAC}_{\text{PRK}}(\text{extra} \| 0) \\
K_{i+1} &= \text{HMAC}_{\text{PRK}}(K_i \| \text{extra} \| i)
\end{aligned}$$

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

Cryptographic Hashing

La cigale ayant chanté tout l'été se trouva fort dépourvue quand la bise fut venue pas un seul petit morçeau 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. 256 bits)

A Swiss Army Knife Cryptographic Primitive

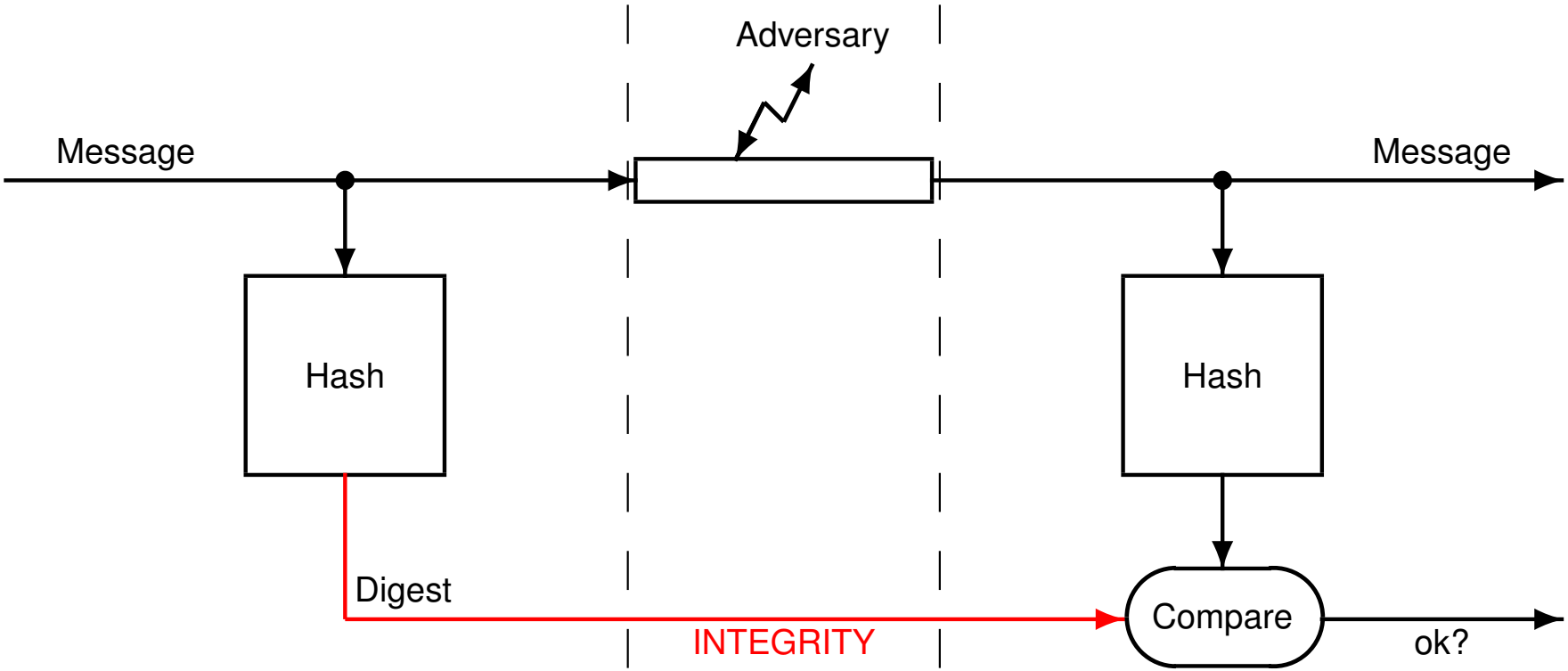
Domain expander: hash bitstrings of arbitrary length into bitstrings of fixed length.
Application: instead of specifying digital signature algorithms on set of bitstring with arbitrary length, we specify them with bitstrings of fixed length and use the hash-and-sign paradigm.

Unique indexing: "uniquely" characterizes a bitstring without revealing information on it.
Application:

- commitment which is binding and hiding.
- integrity check

Pseudorandom generator: generate bitstrings from seeds which are unpredictable.
Application: generation of cryptographic keys from a seed.

Enforcing Integrity by Hash Function



Integrity Does Not Come From Encryption

- symmetric encryption is sometimes used for message authentication but this is a BAD practice
  e.g. Enc(message||redundancy)
- but there are some encryption not enforcing integrity
  example: problem in GSM/WEP/Bluetooth/... (see slide 974)
- in practice: integrity comes with authentication
  (MAC enforces authentication *and* integrity)

Constructing Other Primitives with Hash Functions

- commitment:

$$\begin{aligned} \text{Commit}(X; \text{random}) &= (H(\text{Key}), \text{Key} = X \| \text{random}) \\ \text{Open}(c, X \| \text{random}) &= \begin{cases} X & \text{if } H(X \| \text{random}) = c \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

- PRNG:

$$(\text{seed} \| \text{counter}) \longrightarrow (\text{seed} \| \text{counter} + 1)\ ,\ H(\text{seed} \| \text{counter})$$

- KDF:

$$\text{seed} \longrightarrow \text{trunc}\,(H(\text{seed} \| 1) \| H(\text{seed} \| 2) \| H(\text{seed} \| 3) \| \cdots)$$

- domain expander for authentication (MAC or signature):

$$\text{Authenticate}(H(X))$$

Security for Hash Functions: Wishlist

Collision resistance: hash function h for which it is hard to find x and x' such that $h(x) = h(x')$ and $x \neq x'$.

$\rightarrow$ *digital fingerprint of the bitstring*

One-wayness: hash function h for which given y it is hard to find even one x such that $y = h(x)$.

$\rightarrow$ *witness for a password*

Pseudo-randomness: output "looks like random" (how to formalize?)

$\rightarrow$ *pseudo-random generation*

Popular Threat Models for Hash Functions

Collision attack: find x and x' s.t. $x \neq x'$ and $h(x) = h(x')$.
example: substitution in commitment/signature

1st preimage attack: given y find x s.t. $y = h(x)$.
example: password search based on hash

2nd preimage attack: given x find x' s.t. $x \neq x'$ and $h(x) = h(x')$.
example: substitution in the integrity check process

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 $x_1, x_2, \ldots$
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

complexity: $\mathcal{O}(|\text{output domain}|)$

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 $h(x) = h(x')$

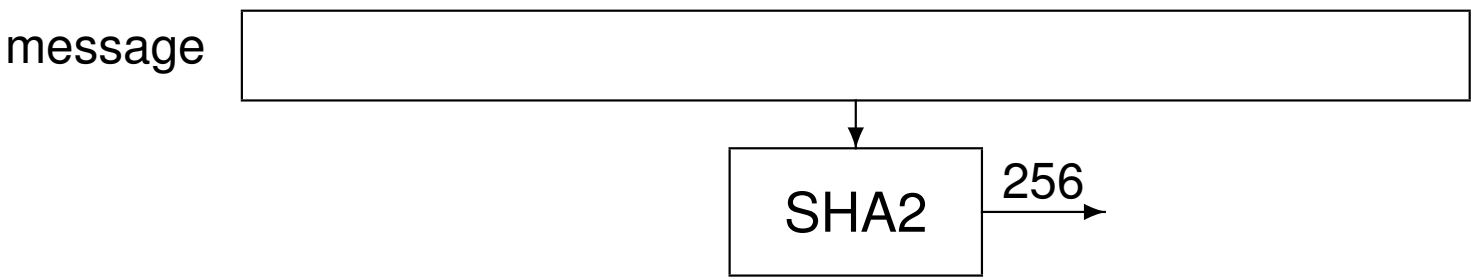
1: compute $h(x)$
2: pick a random ordering of all inputs $x_1, x_2, \ldots$
3: **for all** i such that $x_i \neq x$ **do**
4: compute $h(x_i)$
5: **if** $h(x_i) = h(x)$ **then**
6: yield $x' = x_i$ and stop
7: **end if**
8: **end for**
9: search failed

complexity: $\mathcal{O}(|\text{output domain}|)$

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)
- theoretical attack on SHA0 (Chabaud-Joux 1998)
- collision found on MD4 (Dobbertin 1996)
- preimage attack on MD4 (Dobbertin 1997)
- SHA-2 in 2002: SHA256, SHA384, SHA512 (size of digest)
- collision found on SHA0 (Joux+ 2004)
- collision found on MD5 (Wang+ 2004)
- theoretical attack on SHA1 (Wang+ 2005)
- SHA-3 in 2015
- collision found on SHA1 (Stevens+ 2017)

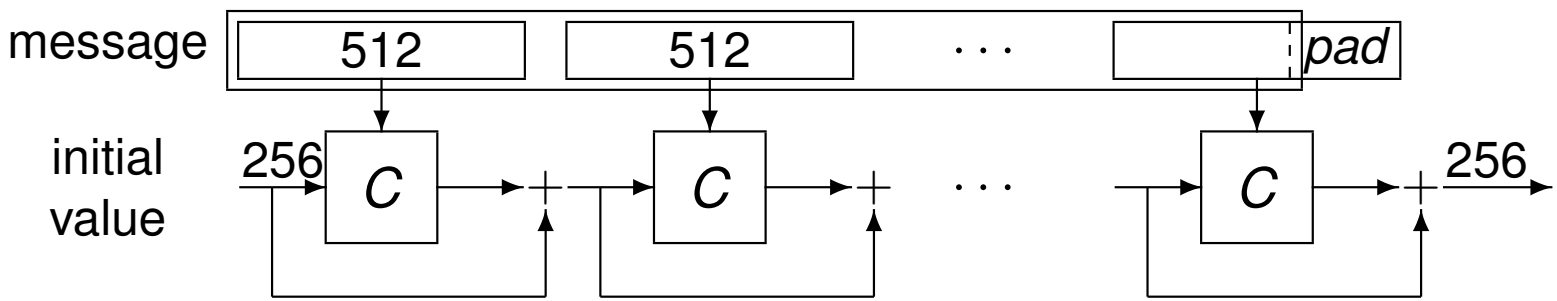
Cryptographic Hashing



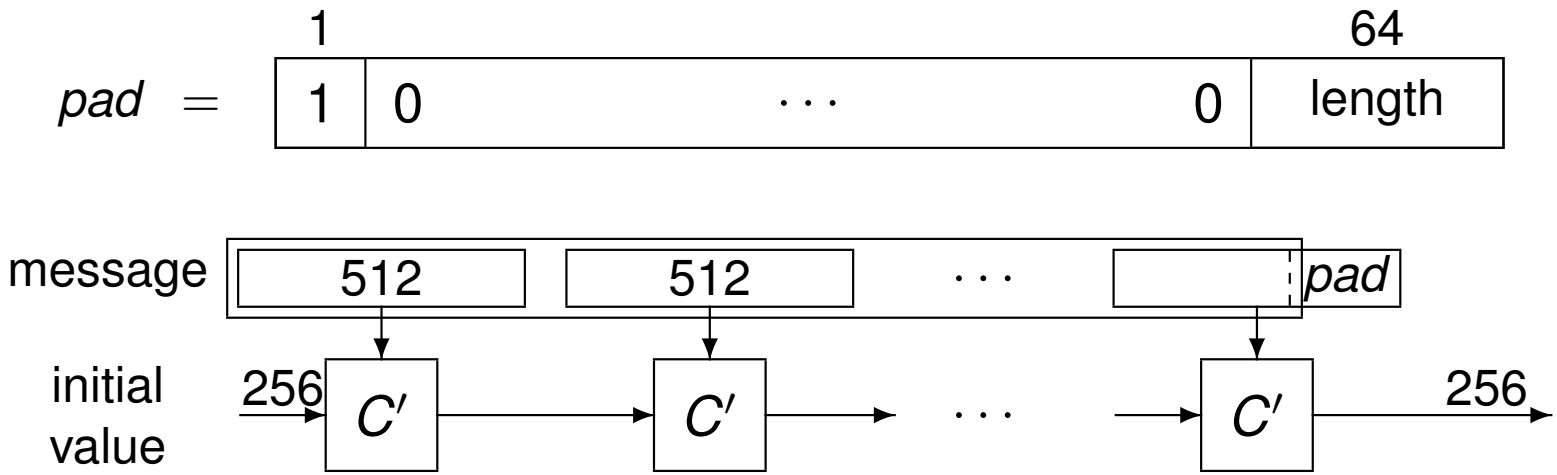
Encryption to Hashing

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

$$\text{SHA256} : \bigcup_{\ell=0}^{2^{64}-1} \{0,1\}^{\ell} \longrightarrow \{0,1\}^{256}$$

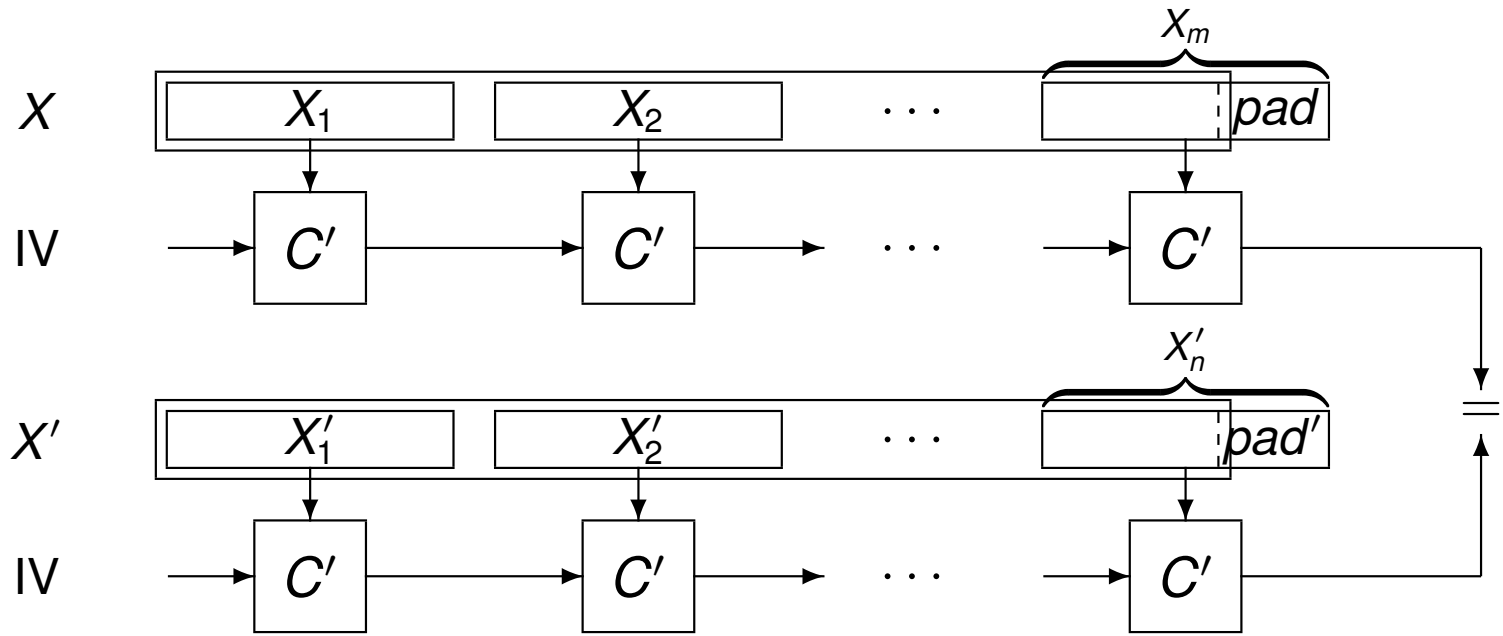
Merkle-Damgård Theorem

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.

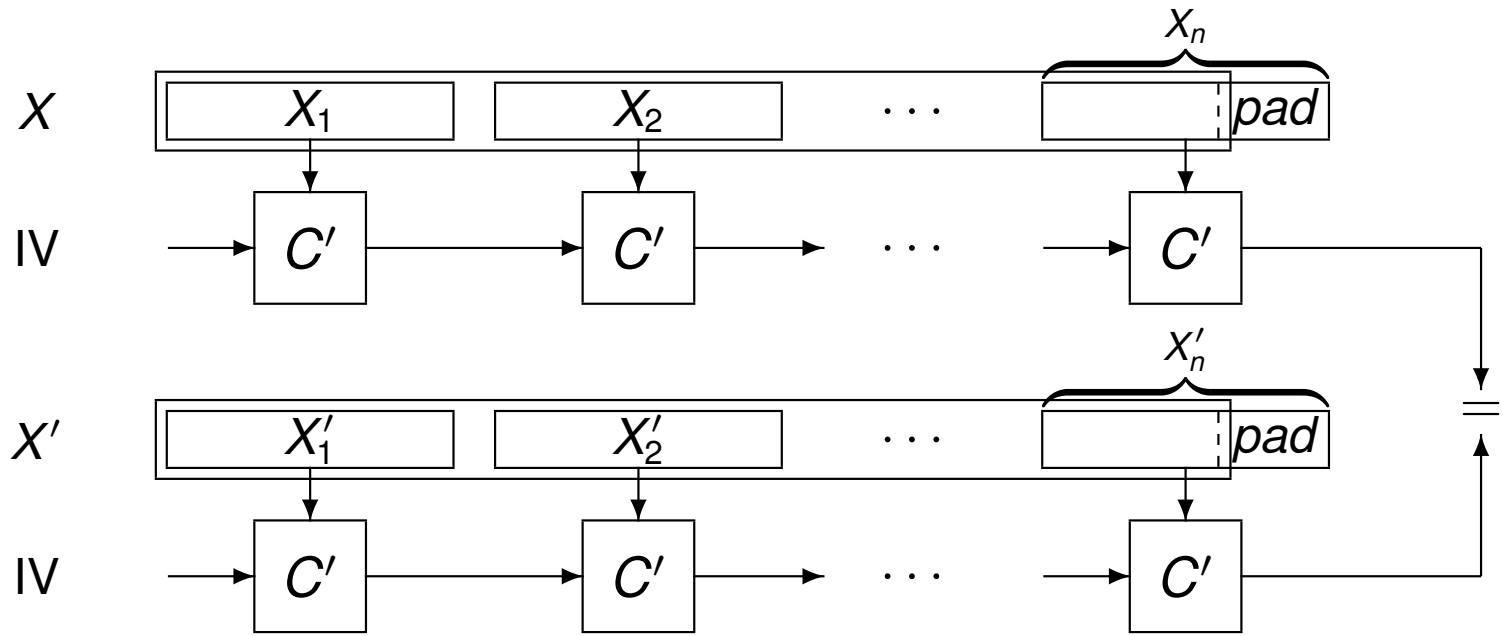
Proof. Assume a colision $h(X) = h(X')$
Case 1: messages of different length
Case 2: messages of same length

Proof of Merkle-Damgård Theorem - Case 1



$$C'(H_m, X_m) = C'(H'_n, X'_n)$$

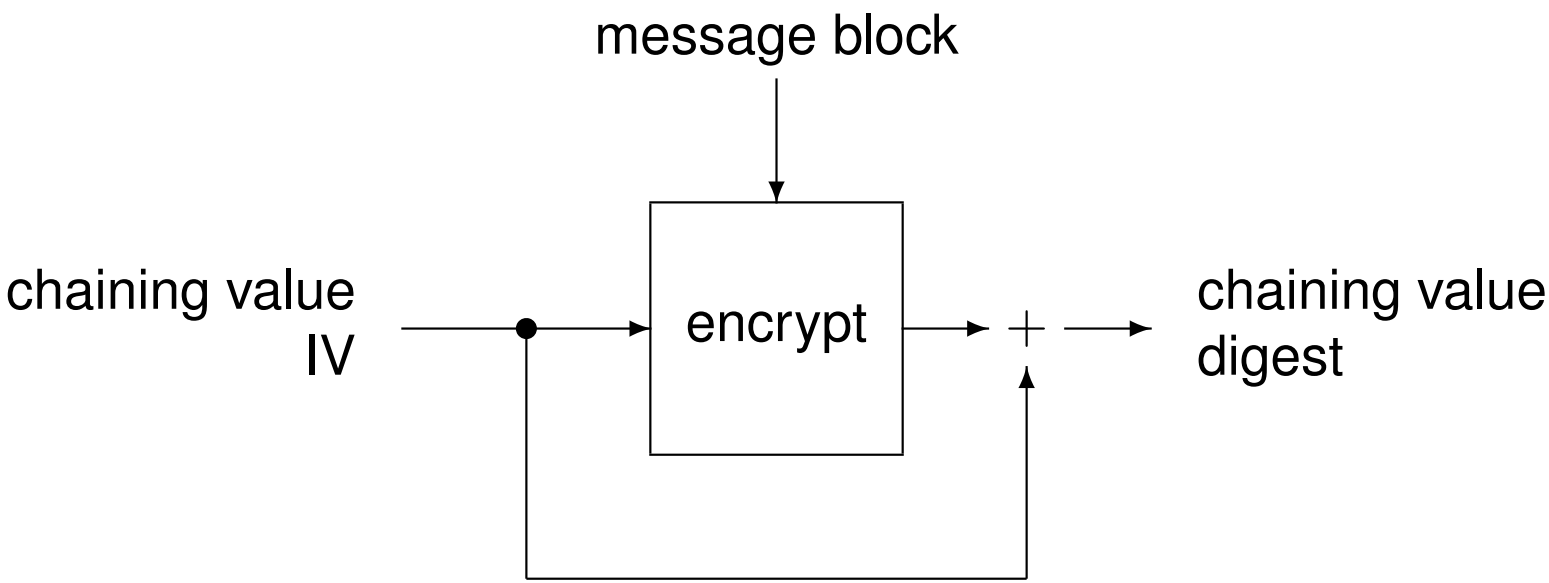
Proof of Merkle-Damgård Theorem - Case 2



$$C'(H_i, X_i) = C'(H'_i, X'_i)$$

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 $x_0, \ldots, x_{15}$
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

- (Keccak has many possible instances, SHA-3 only kept four)
- 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

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- θ is a linear diffusion layer using the parity of columns

$$\theta(a)_{x,y,z} = a_{x,y,z} \oplus \bigoplus_{j=0}^{4} a_{x-1,j,z} \oplus \bigoplus_{j=0}^{4} a_{x+1,j,z-1}$$

- ρ permutes some lanes

$$\rho(a)_{x,y,z} = a_{x,y,z-\frac{(t+1)(t+2)}{2}} \quad \text{with} \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

for $t = 0, \ldots, 23$ (+ use $\rho(a)_{0,0,z} = a_{0,0,z}$)

One Round of f — ii

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- π permutes the slices

$$\pi(a)_{X,Y,z} = a_{x,y,z} \quad \text{with} \quad \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- χ has degree two

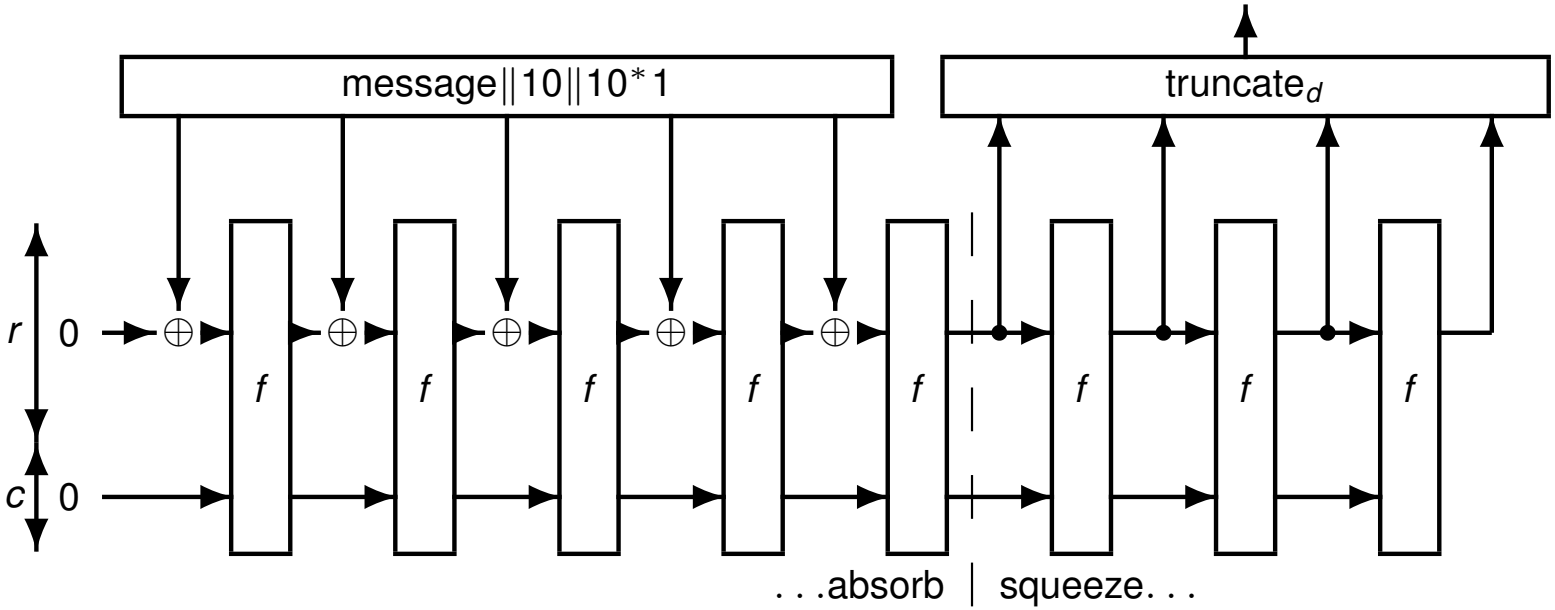
$$\chi(a)_{x,y,z} = a_{x,y,z} \oplus (a_{x+1,y,z} \oplus 1) a_{x+2,y,z}$$

- ι adds a constant for $x = y = 0$

$$\iota(a)_{x,y,z} = \begin{cases} a_{0,0,z} \oplus \mathrm{RC}[i_r]_z & \text{if } x = y = 0 \\ a_{x,y,z} & \text{otherwise} \end{cases}$$

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

$r + c = b$: state s is split into two values of r and c bits

Hash Functions to Remember

algorithm	release	digest	comment
MD5	1991	128	broken
SHA1	1995	160	broken
SHA2	2001	224, 256, 384, 512	
SHA3	2015	224, 256, 384, 512	

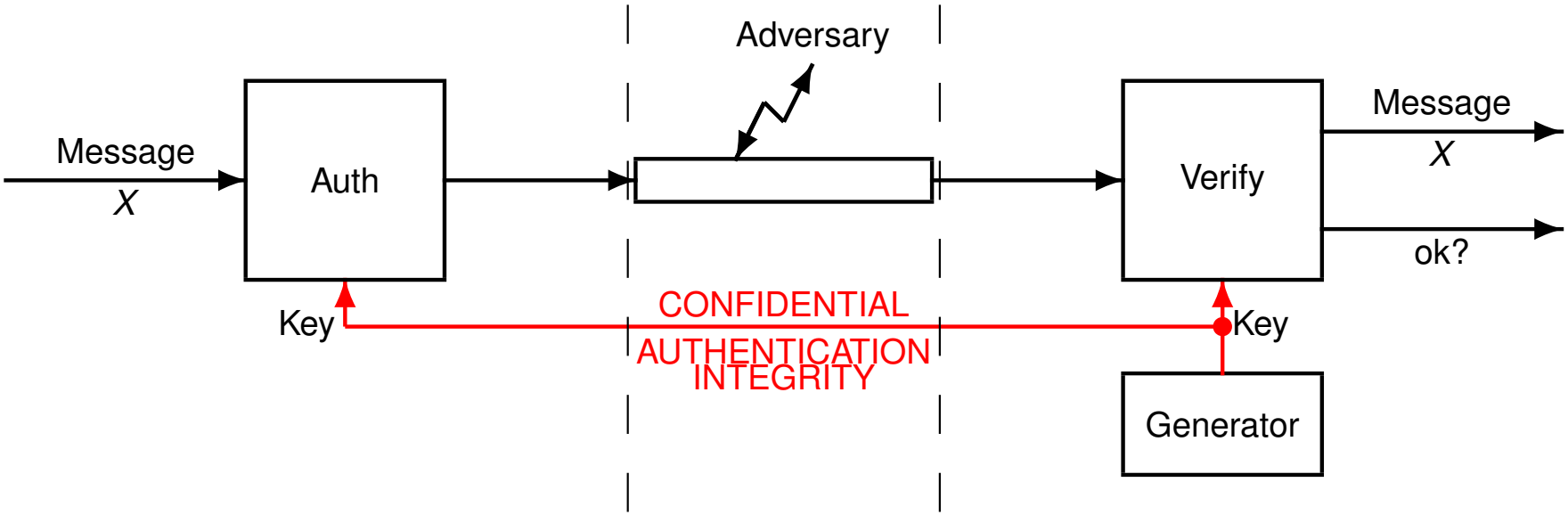
Case Study: Block Chains

case study

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

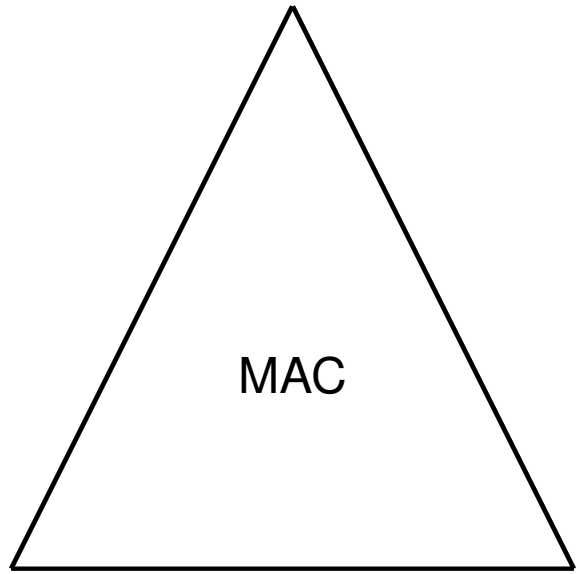
MAC


Adversary
Message
X
Auth
Verify
Message
X
ok?
Key
CONFIDENTIAL
AUTHENTICATION
INTEGRITY
Key
Generator

MAC Primitive

Alice and Bob, Gen, Auth, Verify

components



functionality

$\text{Verify}_K(\text{Auth}_K(X)) = (X, \text{ok})$

Typically:

Auth: compute $c = \text{MAC}_K(X)$ and send (X, c)

Verify: parse (X, c) and check $c = \text{MAC}_K(X)$

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]

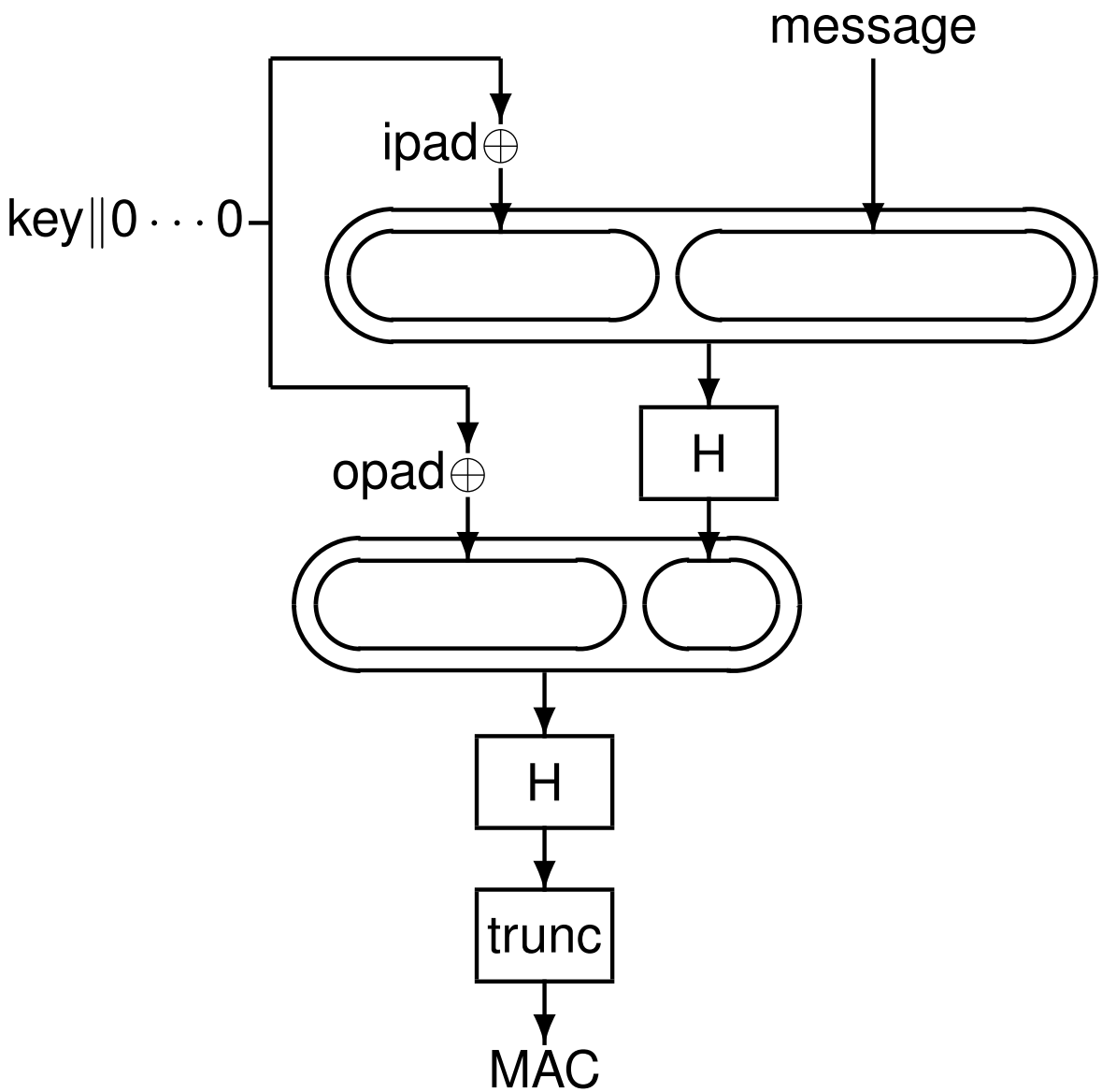
Computing the MAC of t bytes for a message X with a key K using a Merkle-Damgård hash function with block size B bytes, digest size L bytes. ($t = L$ by default.) E.g. $H = \text{SHA256}$, $B = 64$, $L = 32$.

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 ipad and opad are two fixed bitstrings of B bytes. The ipad consists of B bytes equal to `0x36` in hexadecimal. The 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
`SHA256`	SHA256	64	32	32
`SHA384`	SHA384	128	48	48
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

HMAC Security

- (Bellare 2006) If the compression function is a PRF, then HMAC is a PRF
- (Kim et al. 2006) Distinguishing attack between a random function and HMAC with HAVAL, MD4, SHA-0, or a reduced version of MD5 or SHA-1
- (Wang et al. 2009) Distinguishing attack between a random function and HMAC with MD5 (needs 2^{97} queries)

KMAC: Keccack Message Authentication Code

[NIST SP 800-185]

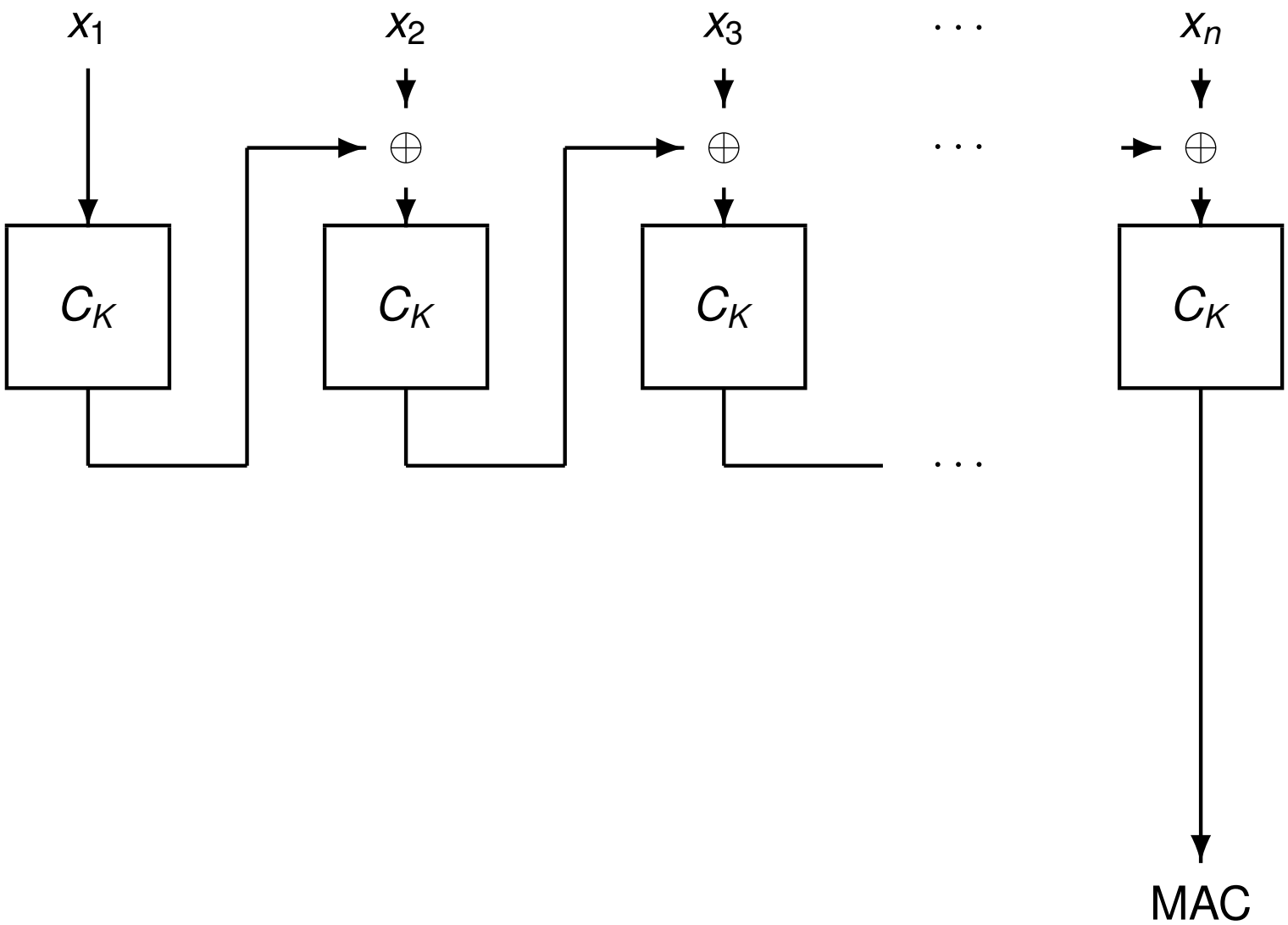
MAC based on SHA3
Inputs:

- K: a key of any length ($\ell = |K|$)
- X: input bitstring (message)
- L: output length (in bits)
- S: empty string

Output: L bits computed as

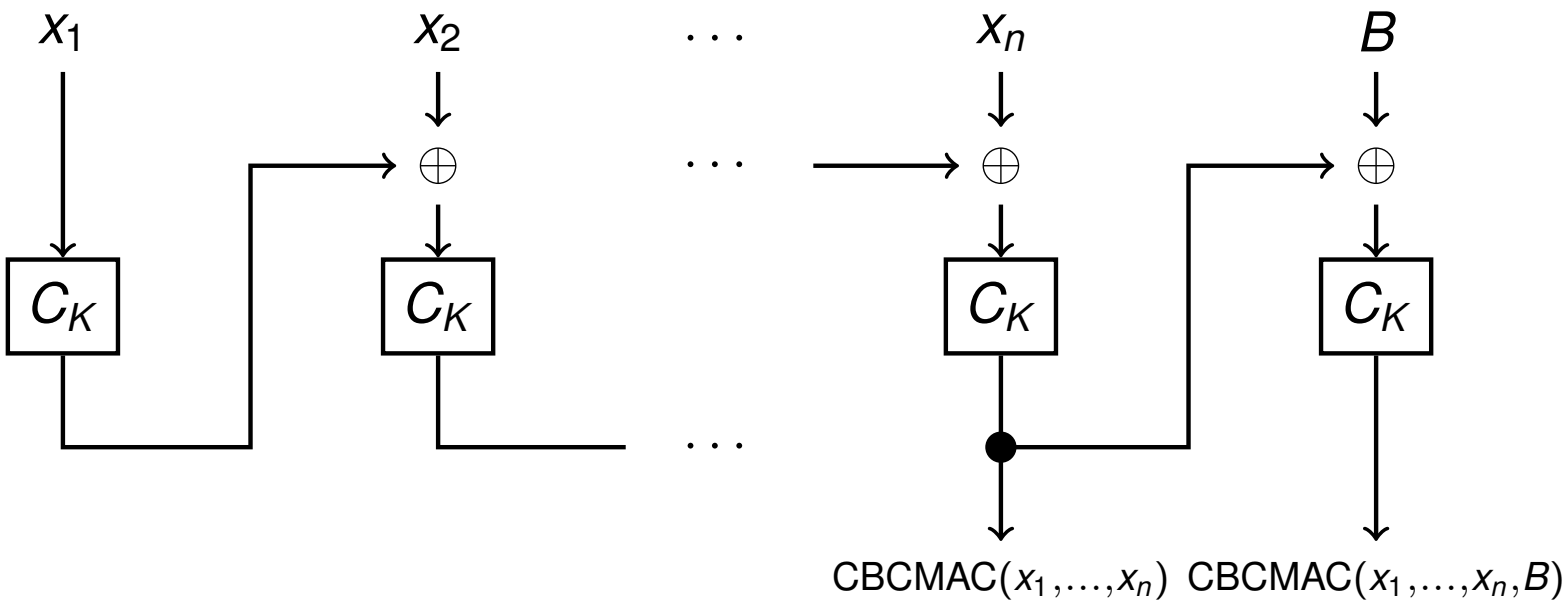
$$H^{(L)}\left(\text{cst}\|\overbrace{01\ A8\|\underbrace{\lceil|\ell|/8\rceil}_{1B}\|\underbrace{\ell}_{\text{bytestring}}\|K\|0\cdots0}^{168\text{ bits}}\|X\|\underbrace{L}_{\text{bytestring}}\|\underbrace{\lceil|L|/8\rceil}_{1B}\|00\right)$$

CBCMAC - (A Bad MAC)



= last ciphertext block of CBC encryption (IV $= 0$)

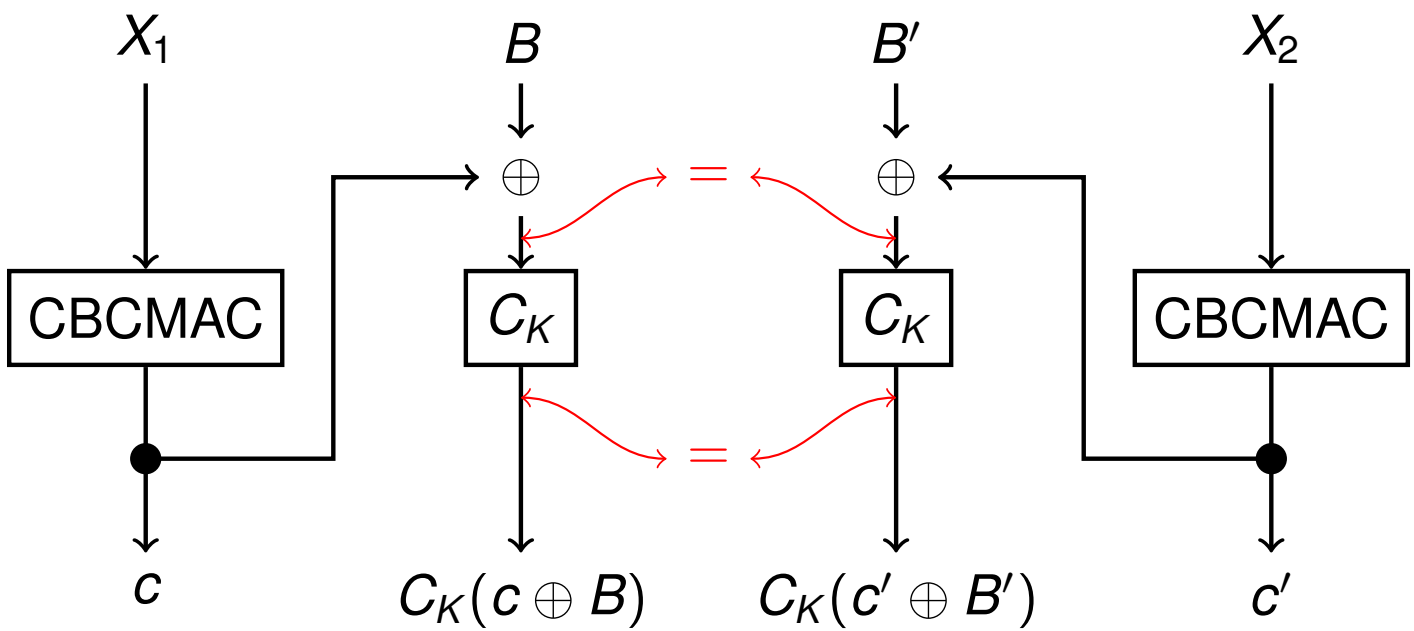
Property of CBCMAC



$$\text{CBCMAC}(X\|B) = C_K\left(\text{CBCMAC}(X) \oplus B\right)$$

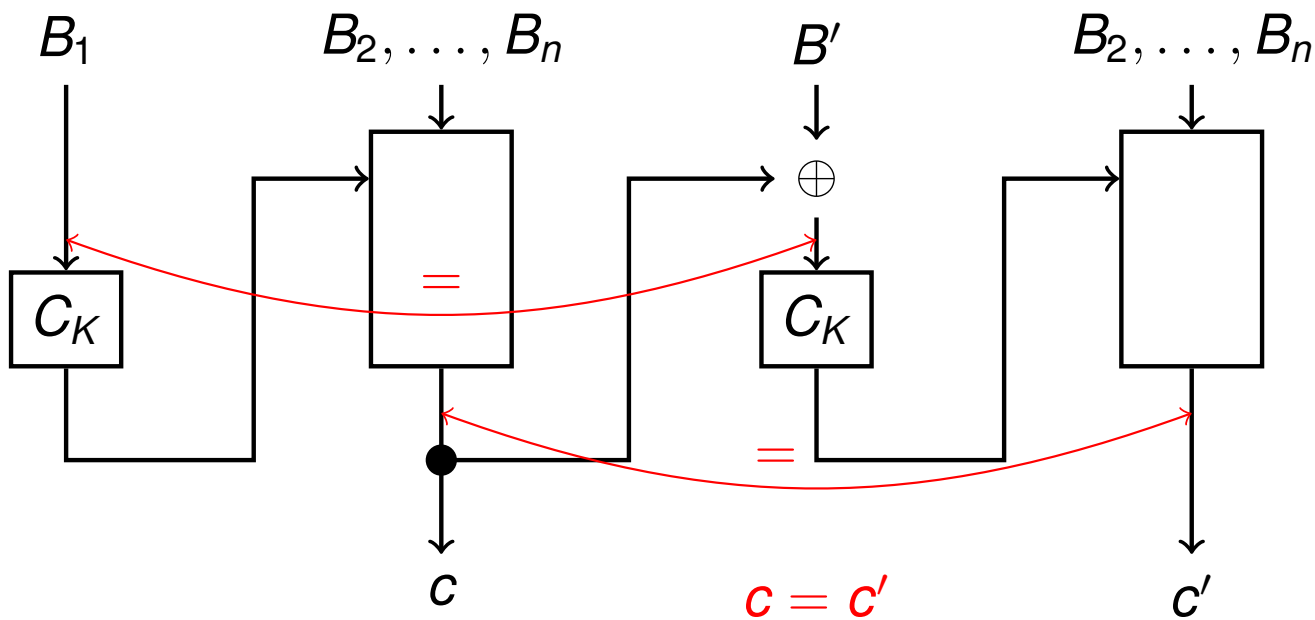
A MAC Forgery

$$
\begin{array}{lcl}
X_1 & = & \text{random} \\
X_2 & = & \text{random} \\
X_3 & = & X_1 \| B \\
\\
X_4 & = & X_2 \| B' \\
B' & = & B \oplus c \oplus c'
\end{array}
\qquad
\begin{array}{lcl}
\text{MAC}(X_1) & = & c \\
\text{MAC}(X_2) & = & c' \\
\text{MAC}(X_3) & = & C_K(c \oplus B) \\
\\
\text{MAC}(X_4) & = & C_K(c' \oplus B') \\
\text{MAC}(X_4) & = & \text{MAC}(X_3)
\end{array}
$$



Other Attack with 1 Known Message

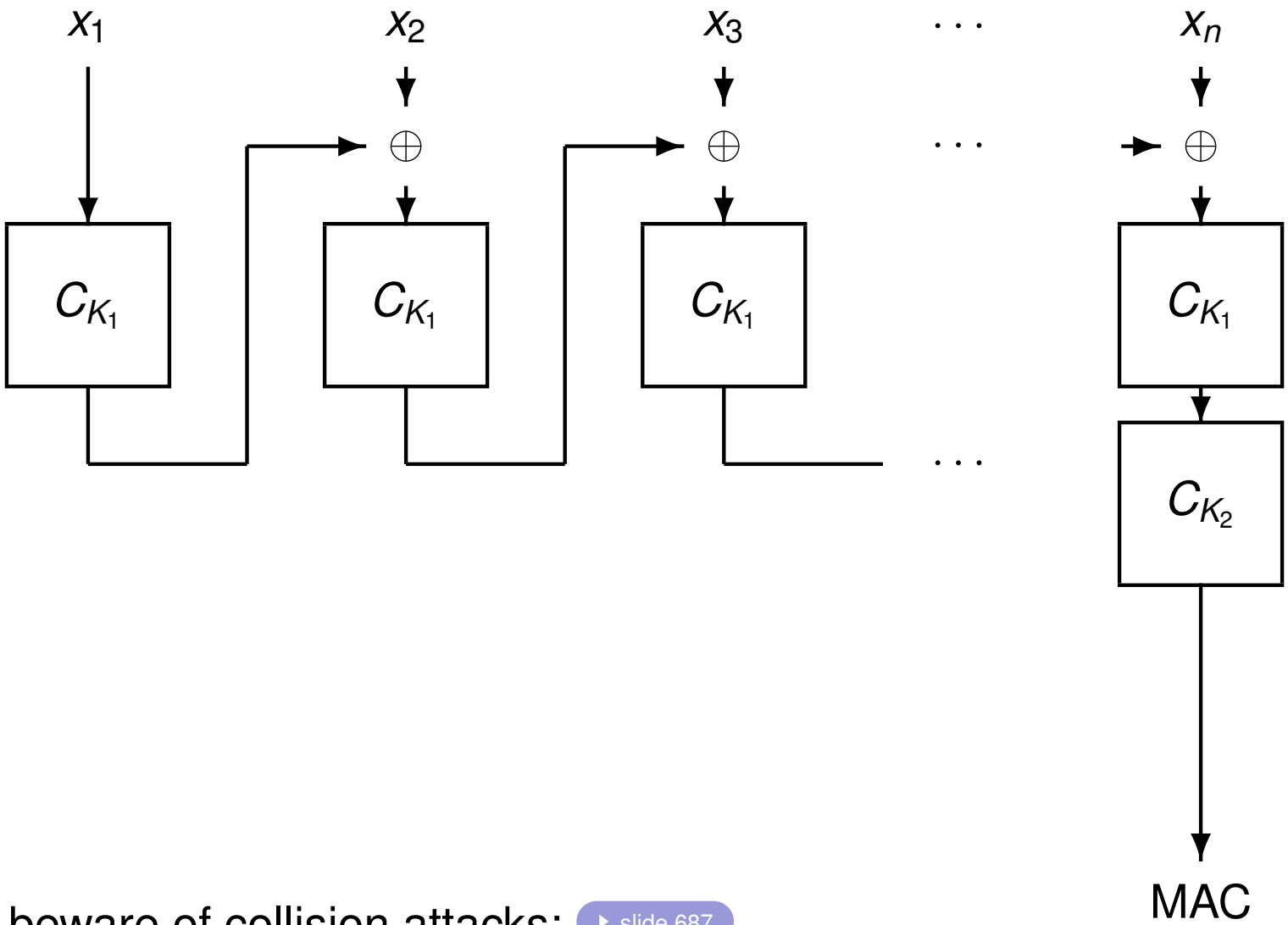
- $X_1 = B_1 \| \cdots \| B_n$ arbitrary
- $c = \mathrm{MAC}(X_1)$
- $X_2 = X_1 \| B' \| B_2 \| \cdots \| B_n$ with $B' = c \oplus B_1$
- forgery: $c = \mathrm{MAC}(X_2)$



Result on CBCMAC

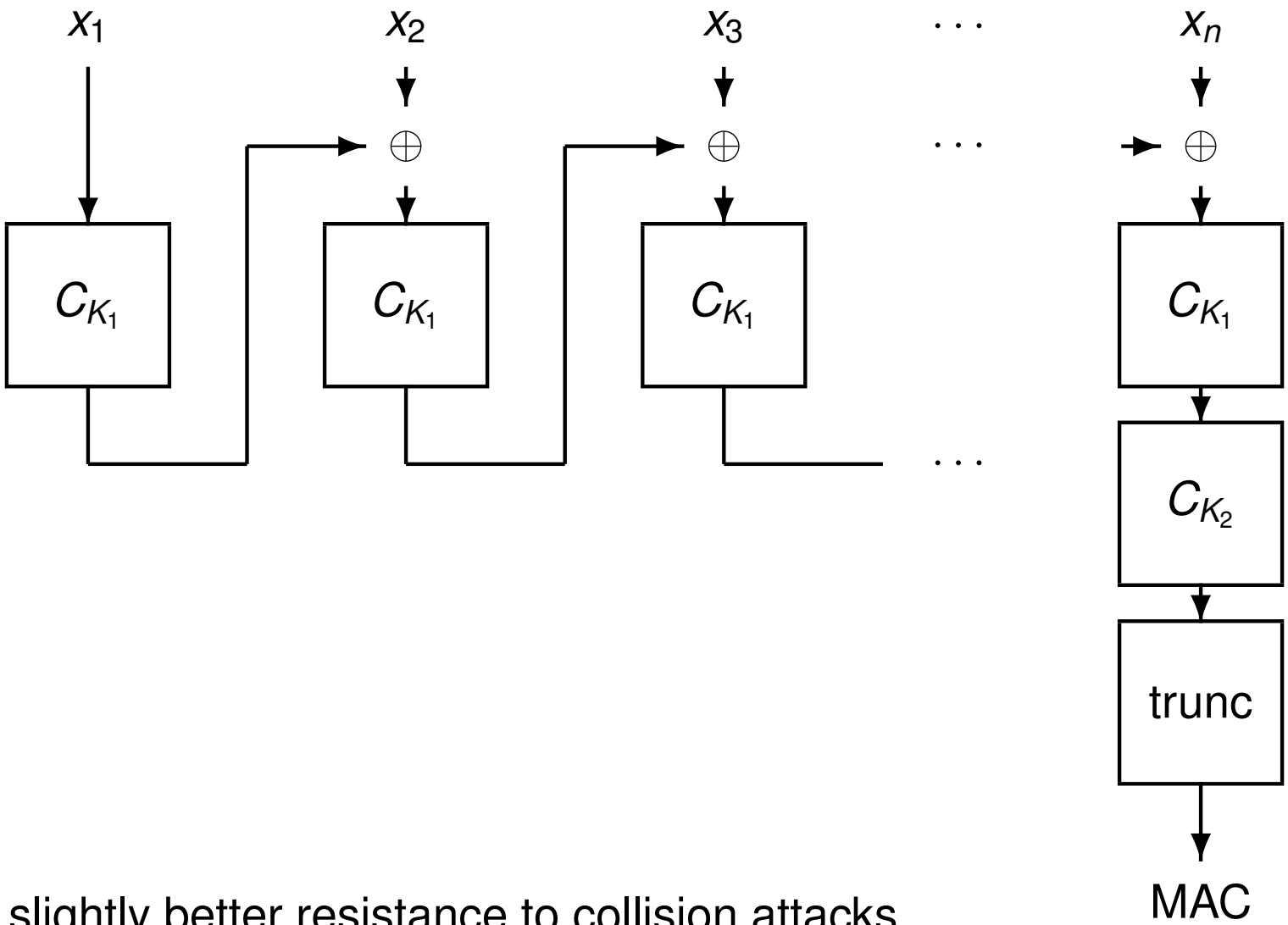
- insecure when used alone as a MAC
- secure when restricted to messages of same fixed length
- might be secure if encrypted (next constructions)

EMAC (Encrypted MAC) - (CBCMAC Variant)



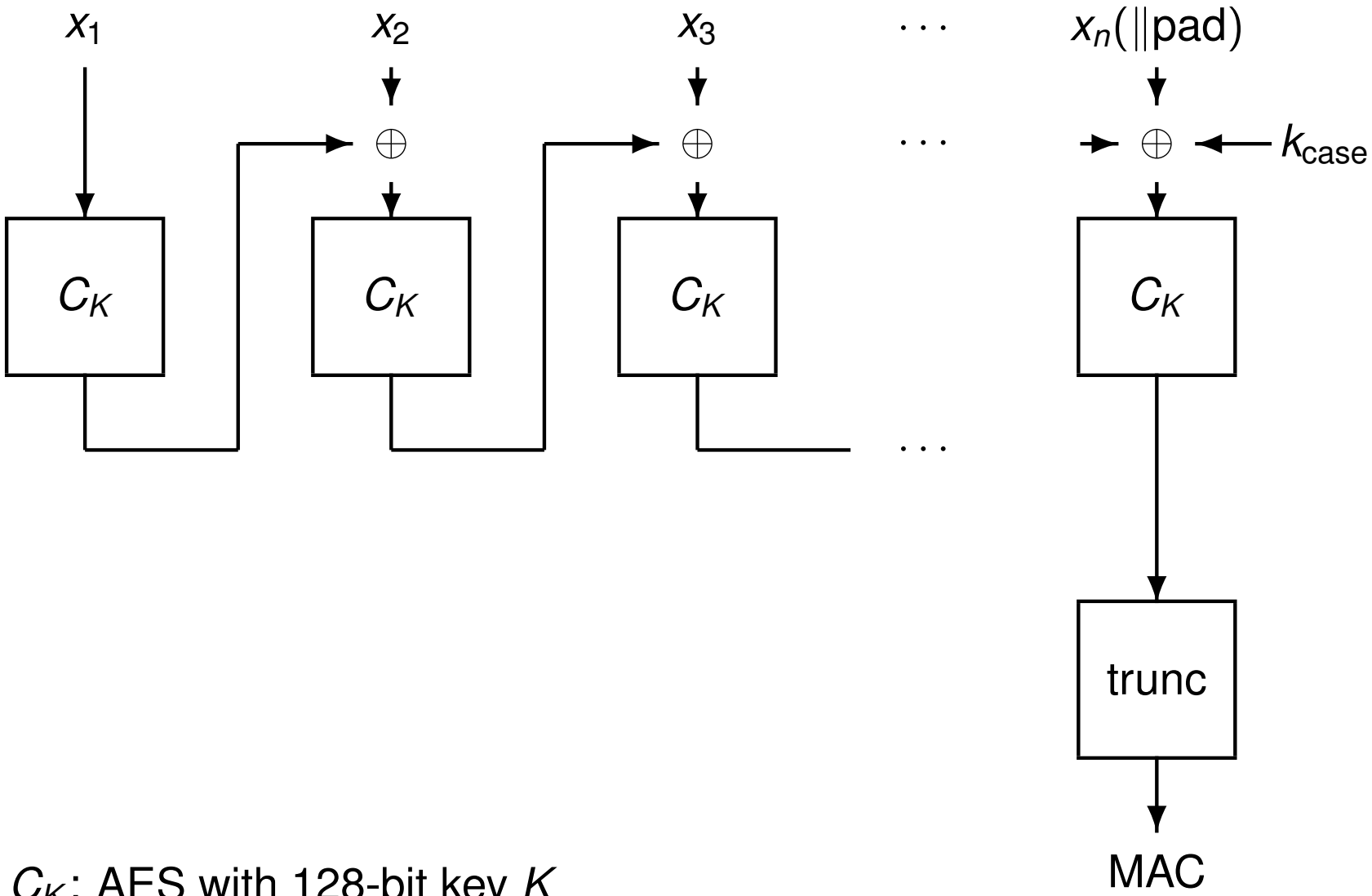
beware of collision attacks: ▸ slide 687

ISO/IEC 9797 - (Another CBCMAC Variant)



slightly better resistance to collision attacks

CMAC [RFC4493] - (Best CBCMAC Variant)



C_K: AES with 128-bit key K

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 a constant if any carry
  k_2 is k_1 shifted to the left by one bit XOR a constant if any carry
  constant:
  `0x000000000000001b` for 64-bit blocks and
  `0x00000000000000000000000000000087` for 128-bit blocks
- actually, this is the GF multiplication by the variable x
- (Iwata-Kurosawa 2003)
  if C is a pseudorandom permutation then CMAC is unforgeable
  (existential forgeries under chosen message attack)

PMAC

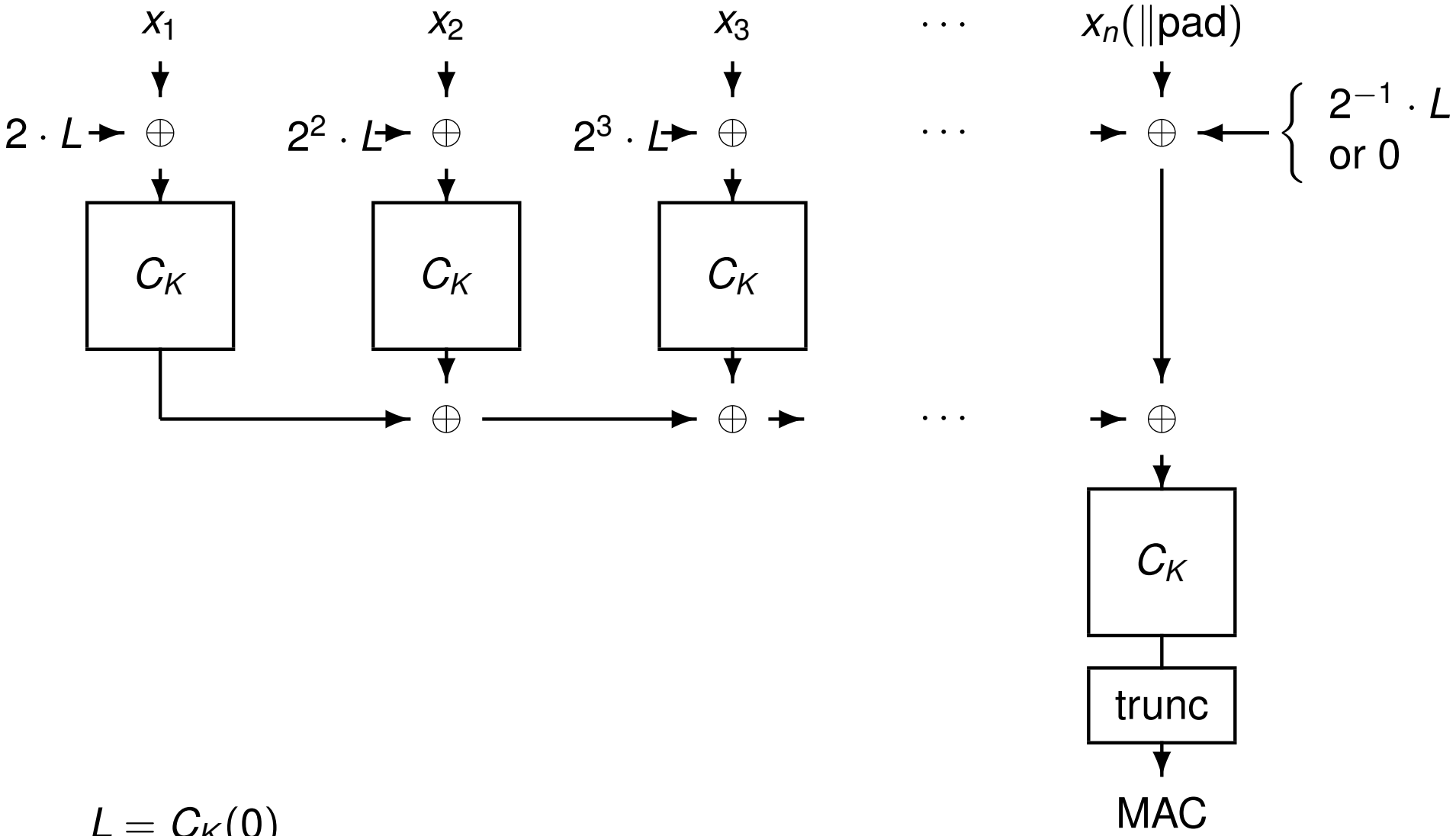
parameter: t, the MAC length
input: K and the message split into blocks $x_1, \ldots, x_n$
(last block can be incomplete)

1:
$$\Sigma = \bigoplus_{i=1}^{n-1} C_K\left(x_i \oplus 2^i \cdot C_K(0)\right) \oplus x_n \oplus 2^n \cdot C_K(0)$$

2: if last block complete: $\Sigma \leftarrow \Sigma \oplus 2^{-1} \cdot C_K(0)$ **return** the first t bits of $C_K(\Sigma)$

- $2^i \cdot x$: multiplication of x by 2^i in GF (like for OMAC)
(i.e. i times a shift with XOR if carry)
- (Black-Rogaway 2002)
if C is a pseudorandom permutation then PMAC is unforgeable
(existential forgeries under chosen message attack)

PMAC


x_1
x_2
x_3
$\cdots$
$x_n(\|\text{pad})$
$2 \cdot L$
$2^2 \cdot L$
$2^3 \cdot L$
$2^{-1} \cdot L$
or 0
C_K
C_K
C_K
C_K
trunc
MAC

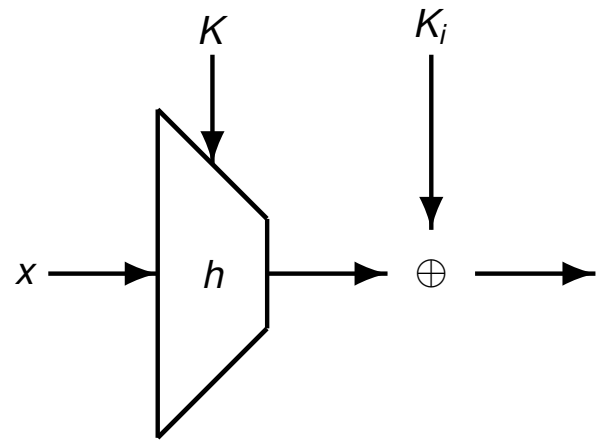
$L = C_K(0)$

WC-MAC

High-level idea: $\mathrm{MAC}(x) = \mathrm{Enc}(h(x))$ for $x \in \mathcal{D}$ with Vernam cipher

- K: uniformly at random in $\mathcal{K}$
- $K_1, K_2, \ldots$: independent and uniformly distributed over $\{0,1\}^m$
- $(h_K)_{K \in_U \mathcal{K}}$: ε-**XOR-universal** family of hash functions from $\mathcal{D}$ to $\{0,1\}^m$
- i: index (used only once: kind of nonce)

$$\mathrm{MAC}_{K,K_1,K_2,\ldots}(i,x) = h_K(x) \oplus K_i$$



Theorem (Wegman-Carter 1981)

No chosen message attack can forge a new authenticated message with a probability of success greater than ε.

Universal Hash Function

High-level idea:
for x and y fixed, the distribution of $h_K(x) \oplus h_K(y)$ is almost flat

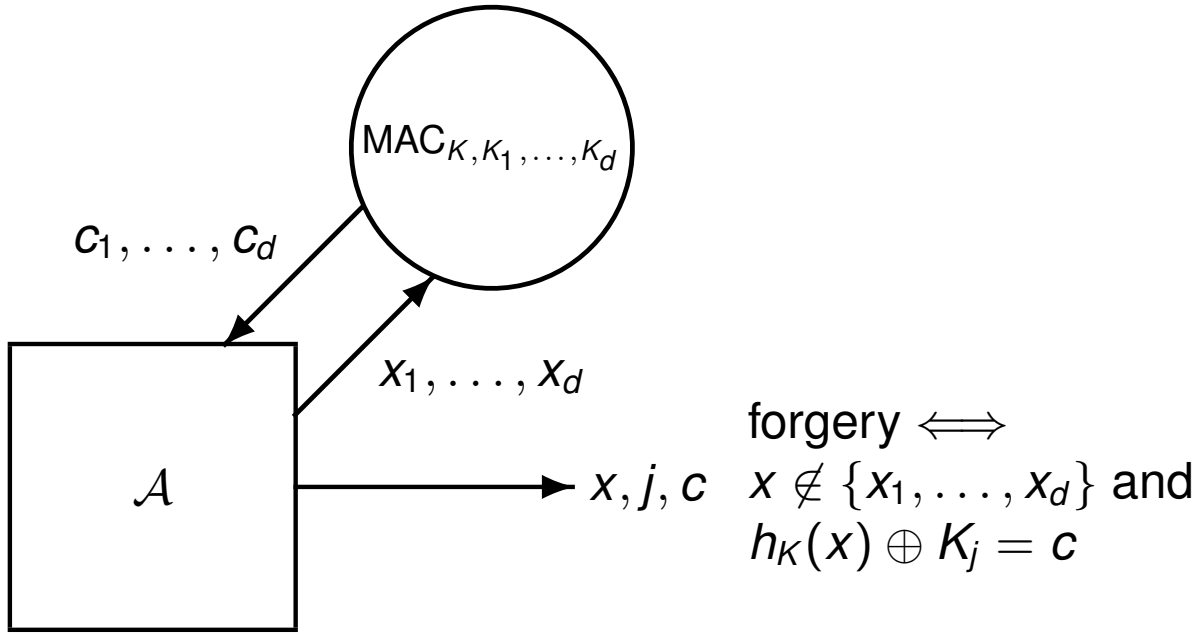
Definition (Krawczyk 1994)

Let $(h_K)_{K \in_U \mathcal{K}}$ be a family of hash functions from $\mathcal{D}$ to $\{0,1\}^m$ defined by a random key K which is chosen uniformly at random in a key space $\mathcal{K}$.
This family is ε**-XOR-universal** if for any a and $x \neq y$ in $\mathcal{D}$, we have

$$\Pr[h_K(x) \oplus h_K(y) = a] \leq \varepsilon.$$

Note: $1 = \sum_a \Pr[h_K(x) \oplus h_K(y) = a] \leq 2^m \varepsilon$ so $\varepsilon \geq 2^{-m}$

WC-MAC - Proof — i



Proof.
At the end, the attacker collects d triplets (x_i, i, c_i) for $i = 1, \ldots, d$ and forges (x, j, c). Let $p_j = \Pr[\text{success}|j]$. If $\forall j \quad p_j \leq \varepsilon$ then $\Pr[\text{success}] \leq \varepsilon$.

For any $j \notin [1, d]$, K_j is uniformly distributed and independent from $c_1, \ldots, c_d$, so the probability that c is a valid MAC of (x, j) is $p_j = 2^{-m}$. (Note that $2^{-m} \leq \varepsilon$.)

WC-MAC - Proof — ii

We have View $= \bigwedge_{i=1}^{d} h_K(x_i) \oplus K_i = c_i$.
For any $j \in [1, d]$, let View$' = \bigwedge_{i \in [1,d] \setminus \{j\}} h_K(x_i) \oplus K_i = c_i$
The success probability is

$$p_j = \Pr[h_K(x) \oplus K_j = c | h_K(x_j) \oplus K_j = c_j, \text{View}', j]$$

Due to the distribution of $K_1, \ldots, K_{j-1}, K_{j+1}, \ldots, K_d$, we can see that View$'$ is useless in the probability.

$$\begin{aligned}
p_j &= \Pr[h_K(x) \oplus K_j = c | h_K(x_j) \oplus K_j = c_j, j] \\
&= \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j | h_K(x_j) \oplus K_j = c_j, j] \\
\text{(Bayes)} \quad &= \frac{\Pr[K_j = h_K(x_j) \oplus c_j | h_K(x) \oplus h_K(x_j) = c \oplus c_j, j]}{\Pr[K_j = h_K(x_j) \oplus c_j | j]} \times \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j] \\
&= \Pr[h_K(x) \oplus h_K(x_j) = c \oplus c_j] \quad \leq \quad \varepsilon
\end{aligned}$$

since K_j is independent from K. □

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 $\mathcal{D} = \{0,1\}^n$ 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 GF(2) and an $s = (s_0, \ldots, s_{m-1})$ is an m-bit string.
- K defines an LFSR with connection polynomial $p(x)$ and initial state s

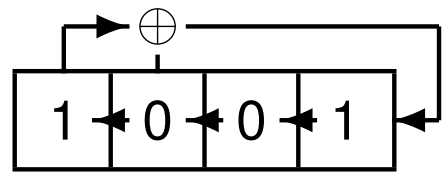
$$s_{t+m} = \bigoplus_{j=0}^{m-1} p_j s_{t+j} \qquad h_K(x_0, \ldots, x_{n-1}) = \bigoplus_{\substack{0 \le t < n \\ x_t = 1}} (s_t, \ldots, s_{t+m-1})$$

$$= \bigoplus_{0 \le t < n} x_t \times (s_t, \ldots, s_{t+m-1})$$

- For any m and n, the family of all h_K defined from $\{0,1\}^{\le n}$ to $\{0,1\}^m$ is $\underbrace{n2^{1-m}}_{\varepsilon}$-XOR-universal

Example

$p(x) = 1 + x + x^4$, $s = (1, 0, 0, 0)$
compute $h_K(1, 1, 0, 1, 0)$:



$$
\begin{array}{cccccc}
 & 1 & 0 & 0 & 0 & 1 \\
\oplus & 0 & 0 & 0 & 1 & 1 \\
\oplus & 0 & 0 & 0 & 0 & 0 \\
\oplus & 0 & 1 & 0 & 0 & 1 \\
\oplus & 0 & 0 & 0 & 0 & 0 \\
\hline
= & 1 & 1 & 0 & 1 &
\end{array}
$$

$$h_K(1, 1, 0, 1, 0) = (1, 1, 0, 1)$$

WC-MAC using a Stream Cipher

$$
\begin{aligned}
N &\leftarrow \text{nonce} \\
\mathrm{MAC}_{K,K'}(N,x) &= h_K(x) \oplus \mathrm{Keystream}_{K',N}
\end{aligned}
$$

idea: “encrypt $h_K(x)$ using a stream cipher”

CAUTION: using the same N twice could be a disaster!
(e.g. reveal information about K then allow easy forgery attacks)

Example (Taken From GCM Mode)

- (mac) $\mathrm{GMAC}_K(\mathrm{IV}, A)$
  1: set $H = C_K(0^{128})$
  2: set $S = \mathrm{GHASH}_H(A \| 0^v \| \mathrm{length}(A) \| 0^{128})$
  3: set $T = \mathrm{trunc}(\mathrm{GCTR}_K((\mathrm{IV} \| 0^{31}1), S))$ (encrypt S)
  4: return T
- (hash) $\mathrm{GHASH}_H(X_1, \ldots, X_m) = X_1 H^m + \cdots + X_m H$ in $\mathrm{GF}(2^{128})$
- (CTR encryption) $\mathrm{GCTR}_K(\mathrm{ct}, X) =$
  $\mathrm{trunc}_{\mathrm{length}(X)}\left(C_K(\mathrm{ct}) \| C_K(\mathrm{ct}+1) \| C_K(\mathrm{ct}+2) \cdots\right) \oplus X$

$\mathrm{GHASH}_H(X) \oplus \mathrm{GHASH}_H(Y) = a \iff P(H) = a$ for a polynomial P defined by $X \oplus Y$
$P(H) = a$ has up to m roots so GHASH is $m2^{-128}$-XOR-universal over $\mathcal{D}$: messages of up to m blocks

Variant: Poly1305 [RFC7539]

one-time authenticator

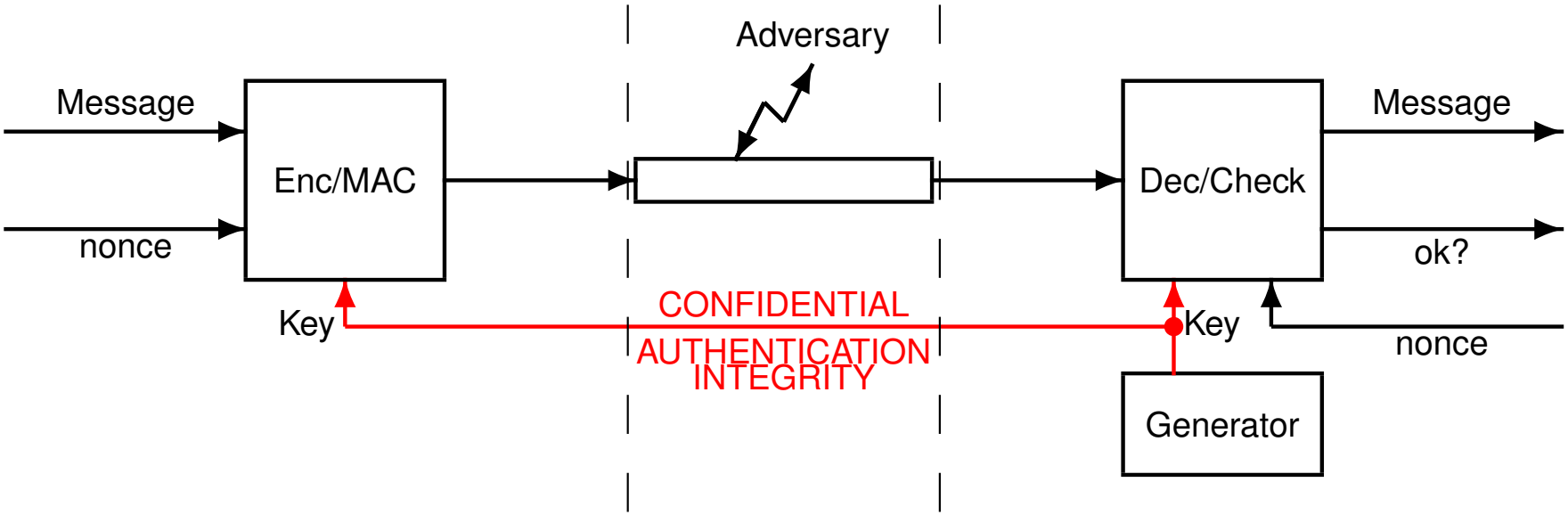
one-time encryption

$$\text{Poly1305}_{r,s}(m_1,\ldots,m_\ell) = (m_1+2^{128})r^\ell+\cdots+(m_\ell+2^{128})r+s \bmod (\underbrace{2^{130}-5}_{\text{prime}})$$

where (r,s) is a key to be used only once and $m_i \in \{0,\ldots,2^{128}-1\}$

example: $(r,s) = \text{Enc}_K(\text{nonce})$

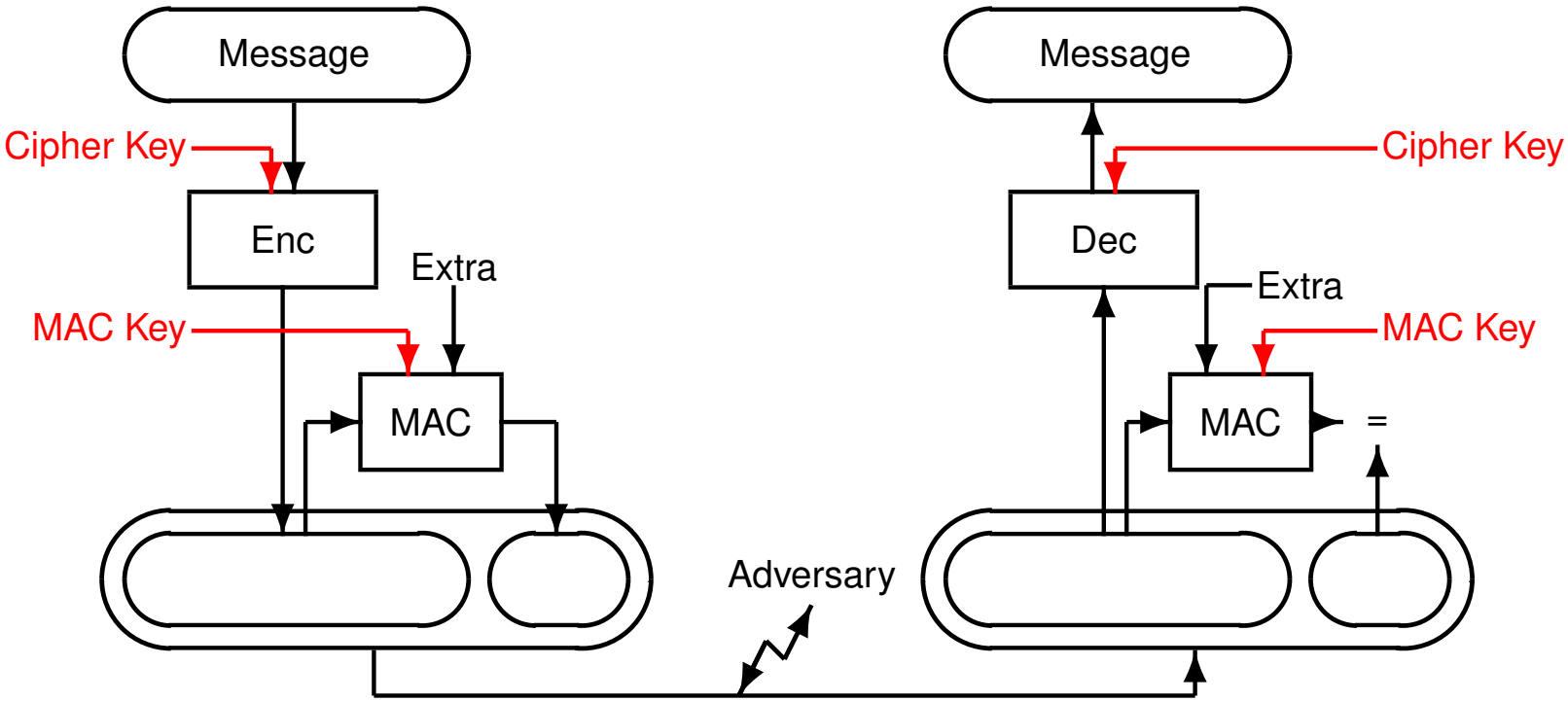
Authenticated Modes of Operation



Roadmap

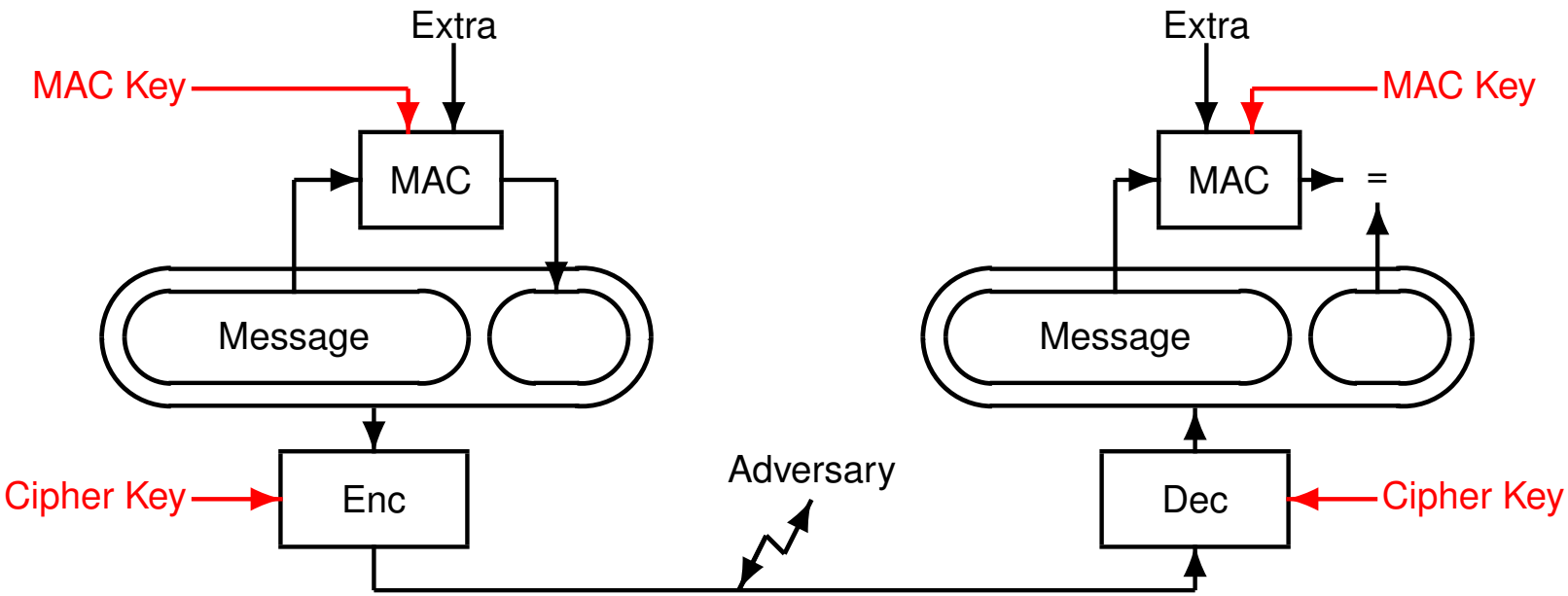
- home-made constructions
  - encrypt-then-MAC
  - MAC-then-encrypt
  - encrypt-and-MAC
- authenticated modes of operation
  - CCM
  - GCM
  - AES-GCM-SIV
  - CHACHA20-POLY1305

Encrypt-then-MAC



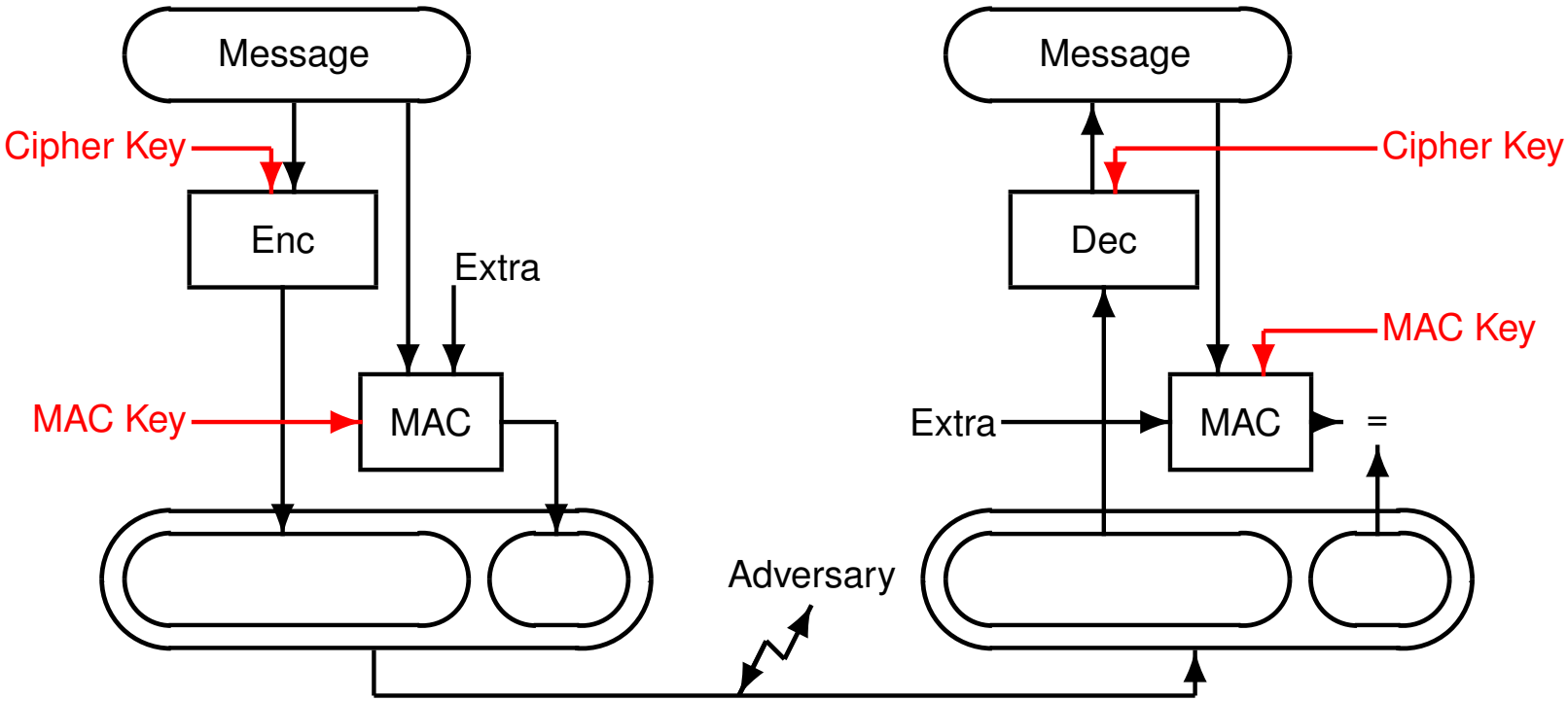
example: IPSEC

MAC-then-Encrypt



example: TLS ($<$ 1.3)

Encrypt-and-MAC

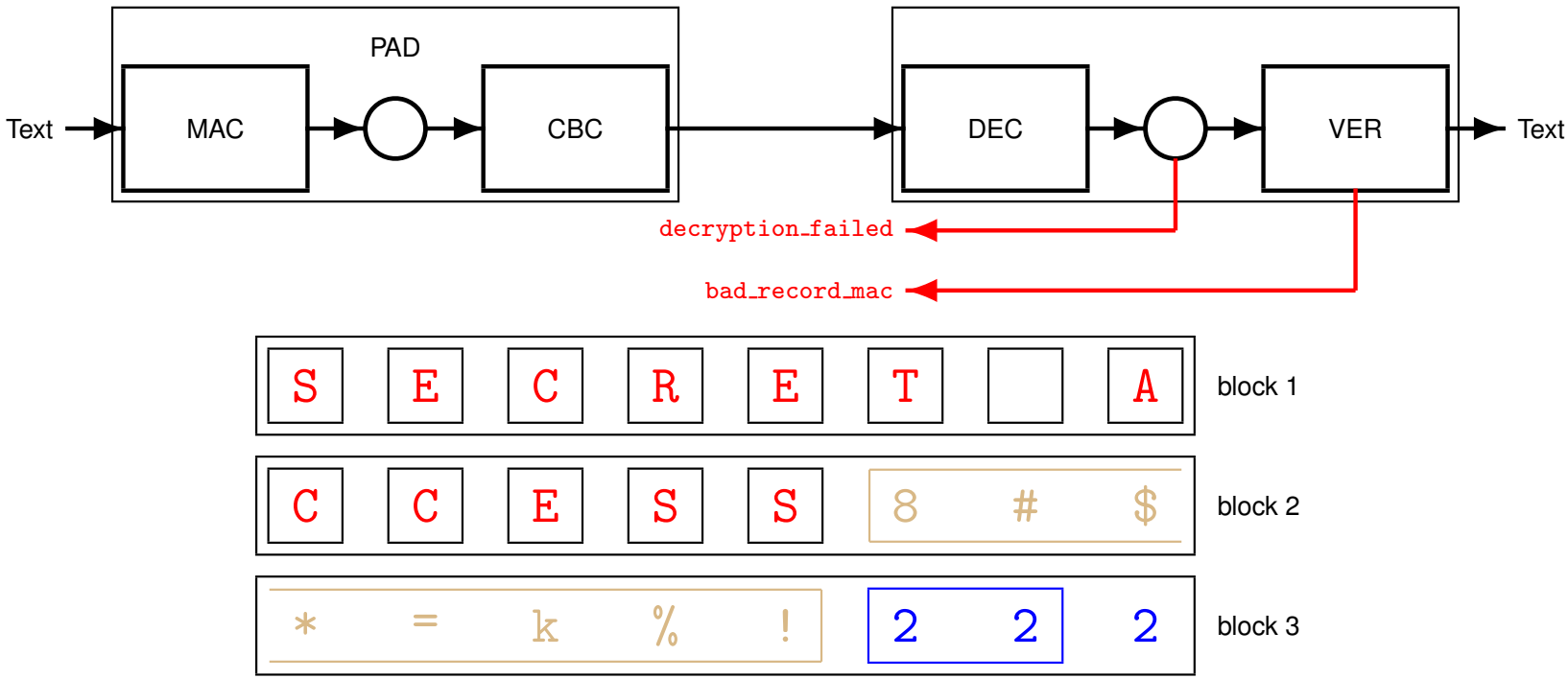


example: SSH

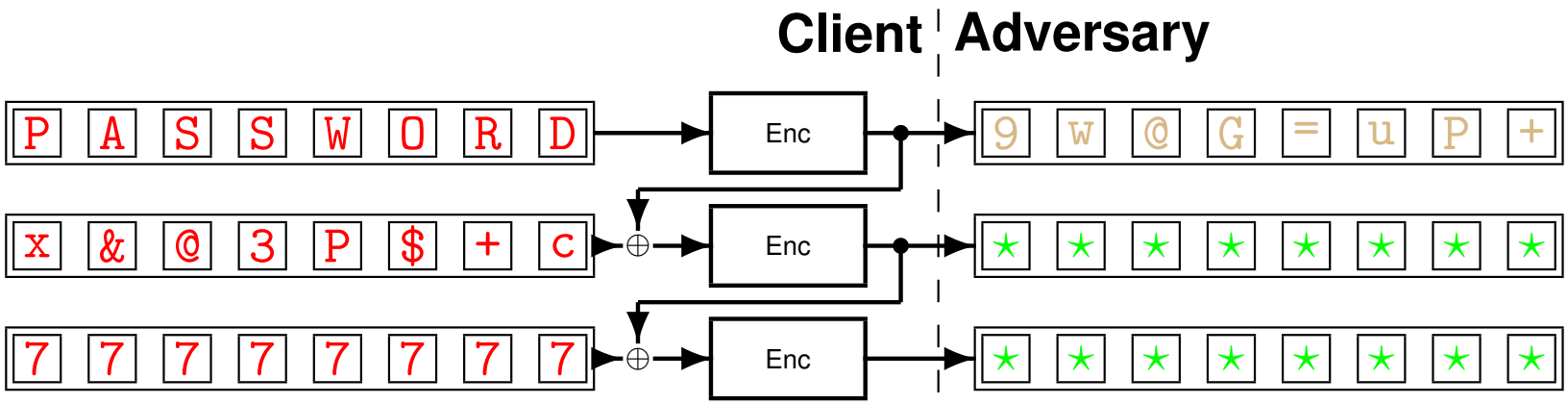
Some Tricky Additional Things

- 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

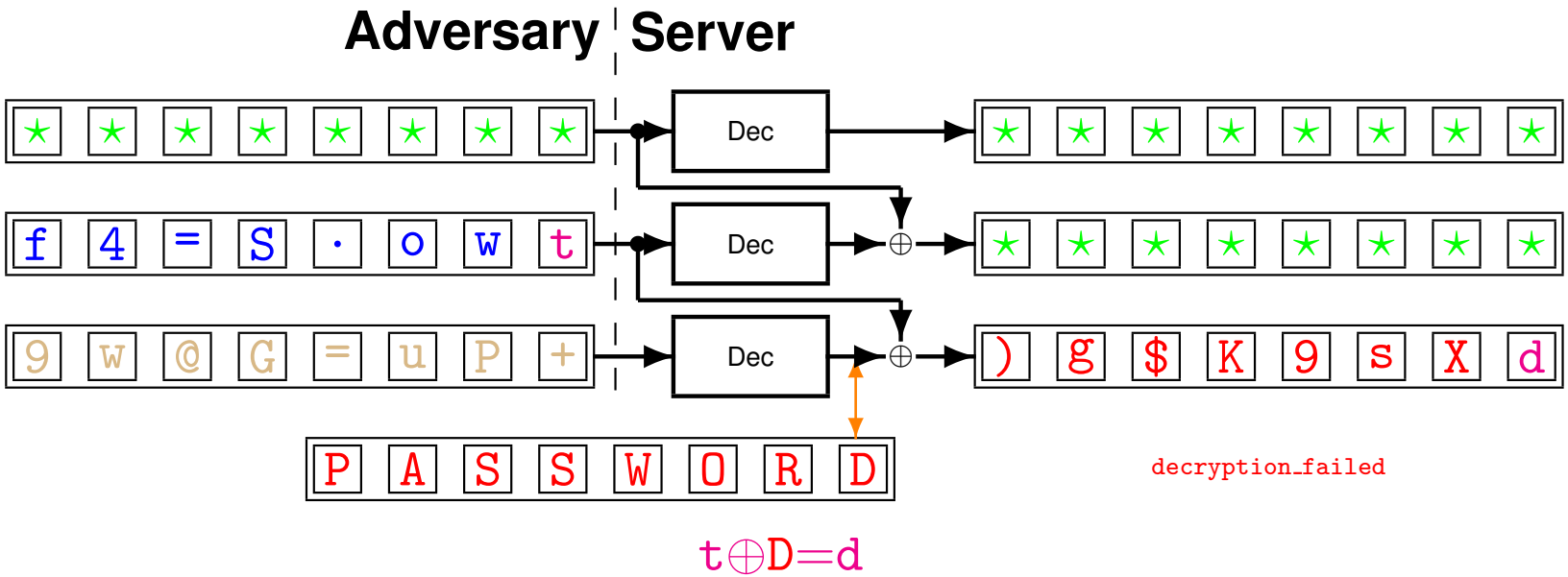


Padding Oracle Attack: Encryption



We would like to decrypt `9w@G=uP+`

Padding Oracle Attack: Decryption



CCM (Counter with CBC-MAC)

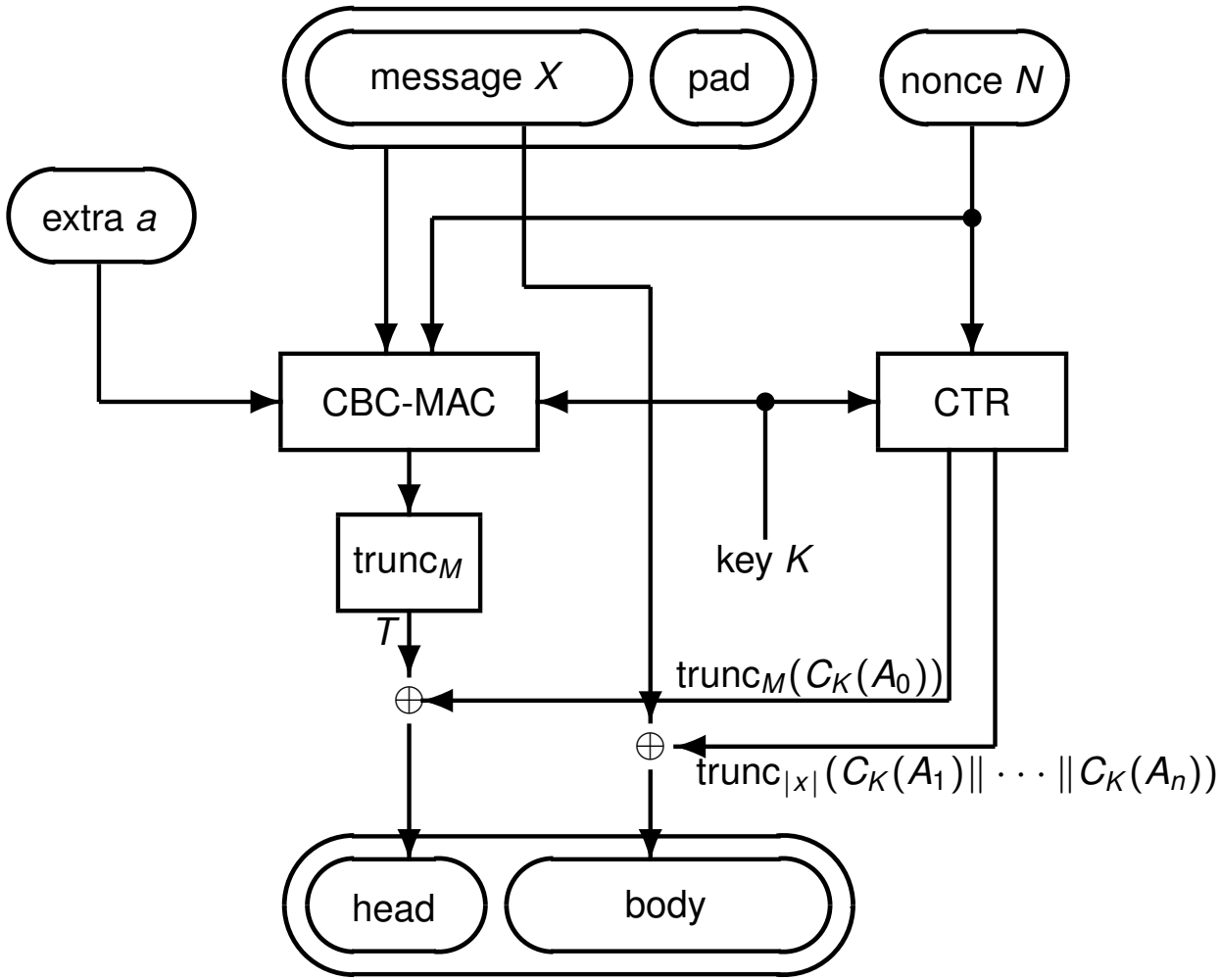
Roughly speaking:

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


message X
pad
nonce N
extra a
CBC-MAC
CTR
trunc_M
T
key K
$\text{trunc}_M(C_K(A_0))$
$\text{trunc}_{|x|}(C_K(A_1)\| \cdots \|C_K(A_n))$
head
body

CCM Processing

- pad X with enough zero bytes to reach the block boundary
- split $X \| \text{pad}$ as $B_1 \| \cdots \| B_n$
- make $B_0 = \text{byte}_1 \| N \| \text{length}(X)$ where byte_1 encodes M and L
- compute the CBCMAC of $B_0 \| B_1 \| \cdots \| B_n$, truncate it to M bytes, and get T
- make $A_i = \text{byte}_2 \| N \| i$ where byte_2 encodes L
- encrypt $T \| X$ by

$$Y = (T \oplus \text{trunc}_M(C_K(A_0))) \| \left(X \oplus \text{trunc}_{|X|}(C_K(A_1) \| \cdots \| C_K(A_n))\right)$$

Processing with an Extra Data

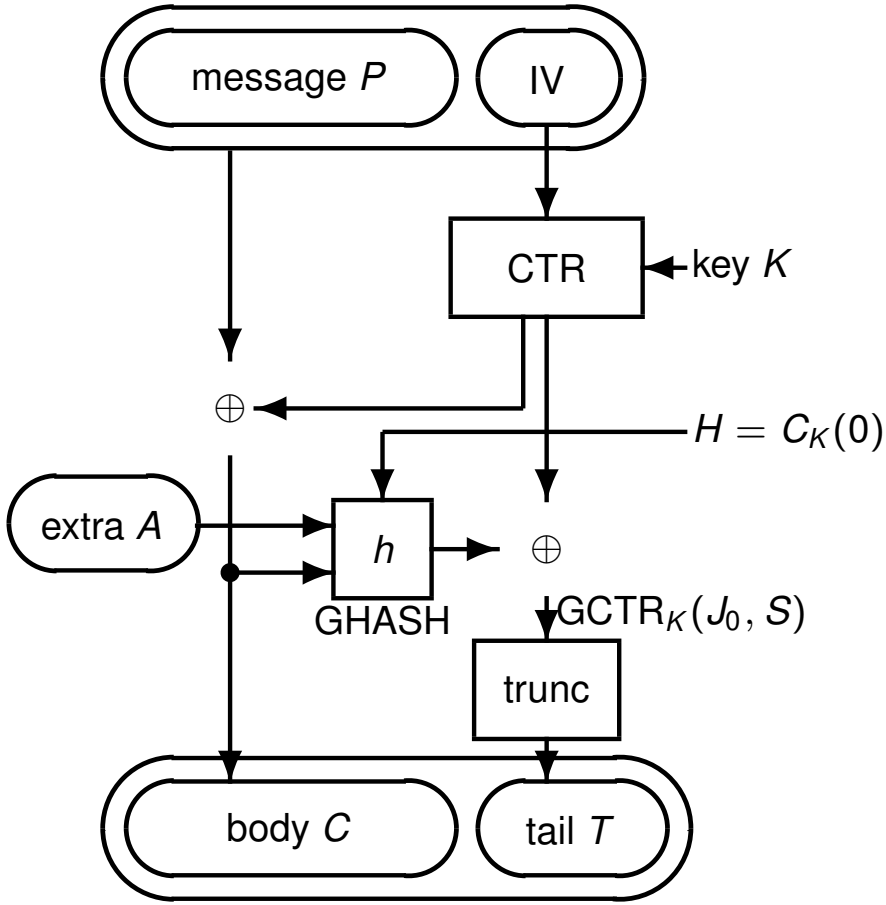
If we wish to send X together with a protocol data a which also needs to be authenticated (e.g. a sequence number, and IP address...)

- add a special bit in 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 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) $\mathrm{GCMAE}_K(\mathrm{IV}, P, A)$ with **plaintext** P and **extra data** A
  1: set $H = C_K(0^{128})$
  2: set $J_0 = \mathrm{IV} \| 0^{31}1$ (IV concatenated with a 32-bit counter)
  3: set $C = \mathrm{GCTR}_K(J_0 + 1, P)$
  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 $S = \mathrm{GHASH}_H(A\|0^v\|C\|0^u\|\mathrm{length}(A)\|\mathrm{length}(C))$
  6: set $T = \mathrm{trunc}(\mathrm{GCTR}_K(J_0, S))$
  7: return (C, T)
- (MAC) $\mathrm{GMAC}_K(\mathrm{IV}, A) = \mathrm{GCMAE}_K(\mathrm{IV}, \emptyset, A)$
- (hash) $\mathrm{GHASH}_H(X_1, \ldots, X_m) = X_1 H^m + \cdots + X_m H$ in $\mathrm{GF}(2^{128})$
- (CTR encryption) $\mathrm{GCTR}_K(\mathrm{ct}, X) = \mathrm{trunc}_{\mathrm{length}(X)}\left(C_K(\mathrm{ct})\|C_K(\mathrm{ct}+1)\|C_K(\mathrm{ct}+2)\cdots\right) \oplus X$

GCM



+ encryption on-the-fly (in stream mode)

Misuse Attack on GCM

misuse: get the encryption of 2 messages with same N

- get $\mathrm{GCMAE}_K(\mathrm{IV}', P_i, \emptyset) = (C_i, T_i)$ for $i = 1, 2$ with P_i of one block
- deduce $\mathrm{GHASH}_H(C_1 \| x) \oplus \mathrm{GHASH}_H(C_2 \| x) = T_1 \oplus T_2$ with $x = (\text{length } 0) \| (\text{length } 128)$
- so $H^2(C_1 \oplus C_2) + H(x \oplus x) = T_1 \oplus T_2$
- deduce that H is the root of $H^2(C_1 \oplus C_2) = T_1 \oplus T_2$ (it is the only root as squaring is bijective)

forgery from a known message $\mathrm{IV}, P', A', C', T'$:

- $\mathrm{GCMAE}_K(\mathrm{IV}, P', A') = (C', T')$
- take arbitrary A and P of same length as A' and P'
- deduce $\mathrm{GCMAE}_K(\mathrm{IV}, P, A) =$ $(C' \oplus P \oplus P', T' \oplus \mathrm{GHASH}_H([A, C]) \oplus \mathrm{GHASH}_H([A', C']))$

GCM

Advantage:

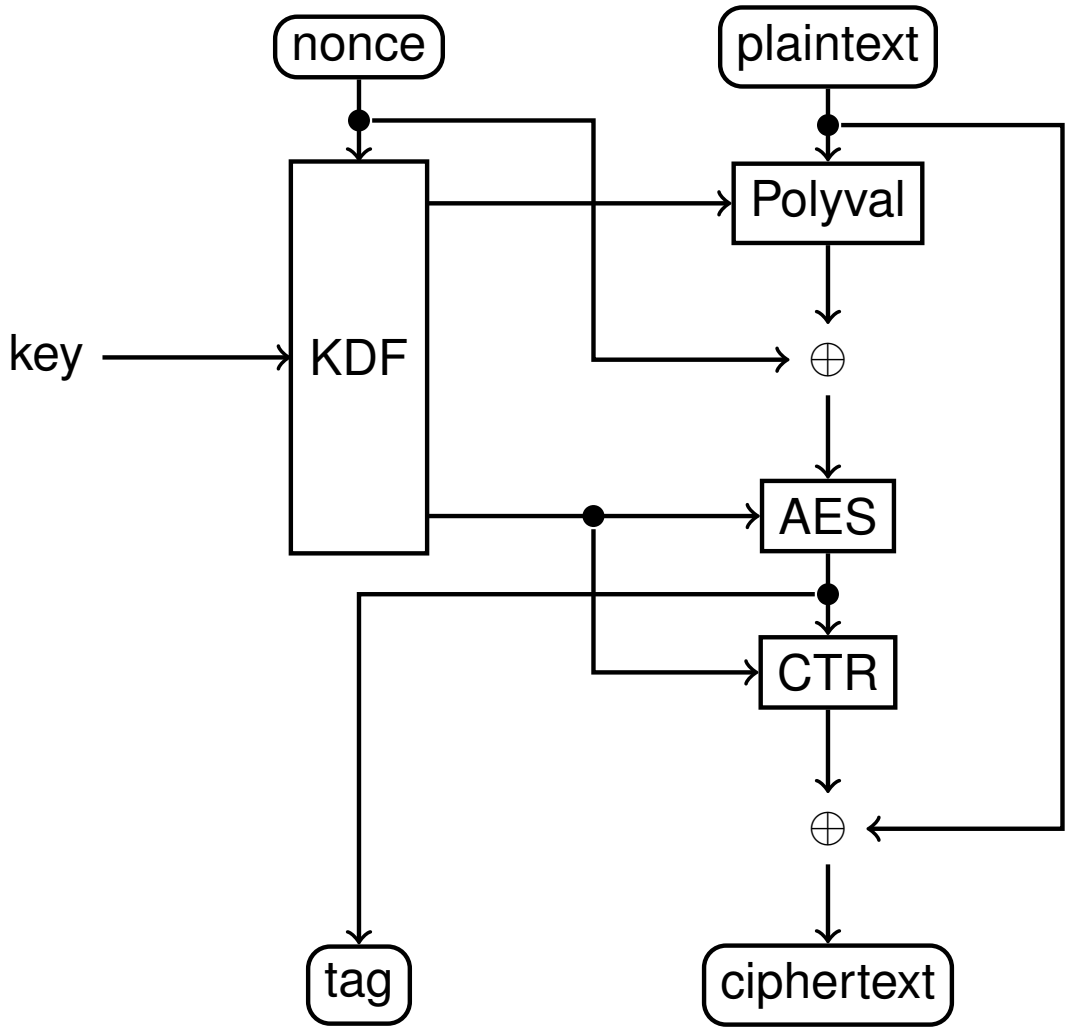
- secure (unless misused)
- fast
- ubiquitous (part of TLS 1.3)
- NIST standard

Disadvantage

- badly broken is misused
- nonce IV is short (96 bits)
- after 2^{32} messages, $\Pr[\text{collision}] \sim 2^{-32}$
- limited to 2^{36} bytes in CTR mode (64GB)

$$J_0 = \text{IV} \| 0^{31} 1$$

Variant: AES-GCM-SIV (RFC 8452)



- Polyval ≈ GHASH
- misuse-resistant
- not on-the-fly encryption (needs 2 passes)

The CHACHA20-POLY1305 AEAD

input: key K (256-bit), nonce N (96-bit), a plaintext P, associated data AD

1: generate otkey using K and N (ad-hoc Chaha20-based PRNG)
2: run ChaCha20 with K and N, and counter set to 1
3: ciphertext $\leftarrow P \oplus$ keystream
4: run Poly1305 with otkey and the input

AD||padding1||ciphertext||padding2||length(AD)||length(ciphertext)

output: ciphertext, tag

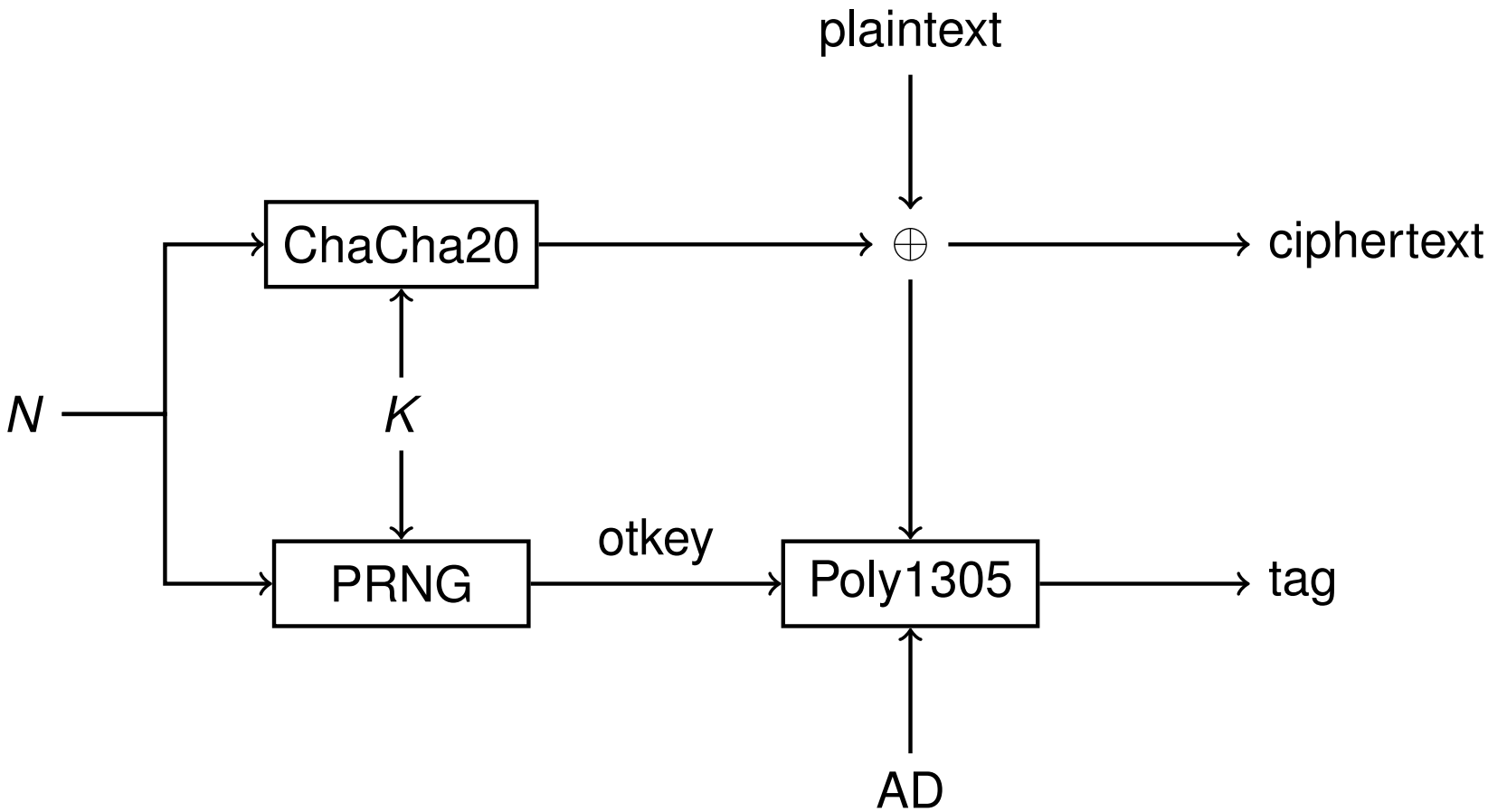
- **ChaCha20**: stream cipher

$$\text{key}, \text{nonce}, \text{counter} \mapsto \text{keystream}$$

- **Poly1305**: authenticator

$$\text{one-time key}, \text{input} \mapsto \text{tag}$$

The CHACHA20-POLY1305 AEAD



Authenticated Modes to Remember

mode	comment
CCM	CTR + CBCMAC
GCM	CTR + WC-MAC, misuse attack
GCM-SIV	CTR + WC-MAC
ChaCha20-Poly1305	stream cipher + WC-MAC

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

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

$$\begin{array}{rccc} h: & \mathcal{D} & \longrightarrow & \{0,1\}^\tau \\ & X & \longmapsto & h(X) \end{array}$$

One-Wayness

Definition

A hash function $(\mathcal{D}, \{0,1\}^\tau, h)$ is (t, ε)-**one-way** if for any probabilistic algorithm $\mathcal{A}$ limited to a time complexity* t, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

1: $x \xleftarrow{\$} \mathcal{D}$ ▷ assume $\mathcal{D}$ is finite or specify a distribution
2: $y \leftarrow h(x)$
3: $\mathcal{A}(y) \to x'$
4: **return** $1_{h(x')=y}$

* including the size of the code (as for all security definitions, actually)

Large-Code Inversion Attack

```
1: if y_0 = 0 then
2:     if y_1 = 0 then
3:         ...
4:         return 2948                ▷ preimage of 00...
5:     else
6:         ...
7:         return 8374                ▷ preimage of 01...
8:     end if
9: else
10:    if y_1 = 0 then
11:        ...
12:        return 8635                ▷ preimage of 10...
13:    else
14:        ...
15:        return 2533                ▷ preimage of 11...
16:    end if
17: end if
```

Security Against Collision Attack (Bad Definition)

Definition

A hash function $(\mathcal{D}, \{0,1\}^\tau, h)$ is (t, ε)-**secure against collision attacks** if for any probabilistic algorithm $\mathcal{A}$ limited to a time complexity t, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

1: $\mathcal{A} \to x, x'$
2: **return** $1_{h(x)=h(x'),x\neq x'}$

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 $(\{0,1\}^k, \mathcal{D}, \{0,1\}^\tau, \mathrm{MAC})$ 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 MAC implementing a function

$$\begin{array}{rccc} \mathrm{MAC}: & \{0,1\}^k \times \mathcal{D} & \longrightarrow & \{0,1\}^\tau \\ & (K, X) & \longmapsto & \mathrm{MAC}_K(X) \end{array}$$

(we could define a variant with nonces)

Security against Key Recovery

Definition

A message authentication code $(\{0,1\}^k, \mathcal{D}, \{0,1\}^\tau, \text{MAC})$ is (q, t, ε)-**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, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns } 1]$$

Game

1: $K \xleftarrow{\$} \{0,1\}^k$
2: $\mathcal{A}^{\text{OMac}} \to K'$
3: **return** $1_{K=K'}$

Oracle OMac(X):

4: **return** MAC(K, X)

(+ 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, ε)-**secure against forgery under chosen message attacks** if for any probabilistic algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

1: $K \xleftarrow{\$} \{0,1\}^k$
2: Queried $\leftarrow \emptyset$
3: $\mathcal{A}^{\text{OMac}} \rightarrow (X, t)$
4: **if** $X \in$ Queried **then return** 0
5: **return** $1_{\text{MAC}(K,X)=t}$

Oracle OMac(X):

6: Queried $\leftarrow$ Queried $\cup \{X\}$
7: **return** MAC(K, X)

(+ similar notion with **known message attacks**)

Forgery Security is Stronger than Key Recovery Security

$(q, t + t_0, \varepsilon)$-secure against forgeries $\Longrightarrow$ (q, t, ε)-secure against key recoveries, where t_0 is constant

Proof: let $\mathcal{A}$ be a (q, t) key recovery adversary
We want to prove $\Pr[\mathcal{A} \text{ succeeds}] \leq \varepsilon$

- we define $\mathcal{B}$:
  1: run $\mathcal{A}^{\mathcal{O}} \rightarrow K'$
  2: pick a fesh X arbitrarily
  3: compute $t = \mathrm{MAC}(K', X)$
  4: return (X, t)
- if Steps 2–3 take time t_0, $\mathcal{B}$ is a $(q, t + t_0, \varepsilon)$ forgery attack
- $\Pr[\mathcal{A} \text{ succeeds}] \leq \Pr[\mathcal{B} \text{ succeeds}]$
- due to unforgeability, $\Pr[\mathcal{B} \text{ succeeds}] \leq \varepsilon$
- so, $\Pr[\mathcal{A} \text{ succeeds}] \leq \varepsilon$

□

$$\text{key recovery-breaking} \Longrightarrow \text{forge}$$
$$\text{forgery-secure} \Longrightarrow \text{key recovery-secure}$$

Security against Distinguisher (PRF)

Definition

A message authentication code $(\{0,1\}^k, \mathcal{D}, \{0,1\}^\tau, \text{MAC})$ is a (q, t, ε)-**pseudorandom function (PRF)** if for any probabilistic algorithm $\mathcal{A}$ limited to a time complexity t and to q queries, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game Γ_b

1: $K \xleftarrow{\$} \{0,1\}^k$
2: pick $F : \mathcal{D} \to \{0,1\}^\tau$
3: $\mathcal{A}^{\mathsf{O}} \to z$
4: **return** z

Oracle $\mathsf{O}(X)$:

5: **if** $b = 0$ **then return** $F(X)$
6: **return** $\text{MAC}(K, X)$

(see slide 586)

PRF-Security is Stronger than Unforgeability

$(q+1, t+t_0, \varepsilon)$-PRF $\Longrightarrow$ $(q, t, \varepsilon + 2^{-\tau})$-secure against forgeries, where t_0 is constant

Proof: let $\mathcal{A}$ be a (q, t)-forger. We want to prove
$\Pr[\mathcal{A} \text{ succeeds}] \leq \varepsilon + 2^{-\tau}$

- we construct a distinguisher $\mathcal{D}$:
  1: run $\mathcal{A}^{\mathcal{O}} \to (X, t)$
  2: if X was queried by $\mathcal{A}$, output 0 and stop
  3: query X to $\mathcal{O}$ and get t'
  4: output $1_{t=t'}$
- with $\mathcal{O} = \mathsf{MAC}(K, .)$, we have $\Pr[\mathcal{D}^{\mathsf{MAC}(K,.)} \to 1] = \Pr[\mathcal{A} \text{ wins}]$
- with $\mathcal{O} = F(\cdot)$, we have $\Pr[\mathcal{D}^{F(\cdot)} \to 1] \leq 2^{-\tau}$
- so,

$$\Pr[\mathcal{A} \text{ wins}] \leq \Pr[\mathcal{D}^{\mathsf{MAC}(K,.)} \to 1] - \Pr[\mathcal{D}^{F(\cdot)} \to 1] + 2^{-\tau} \leq \varepsilon + 2^{-\tau}$$

therefore, PRF-security implies unforgeability □

PRF vs MAC

- PRF aims at being indistinguishable
- MAC aims at being unforgeable (unguessable)
- secure PRF $\Longrightarrow$ secure MAC

Security Notions

	key recovery	forgery	PRF
CMA			stronger security

Case Study: Mobile Telephony

▸ case study

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

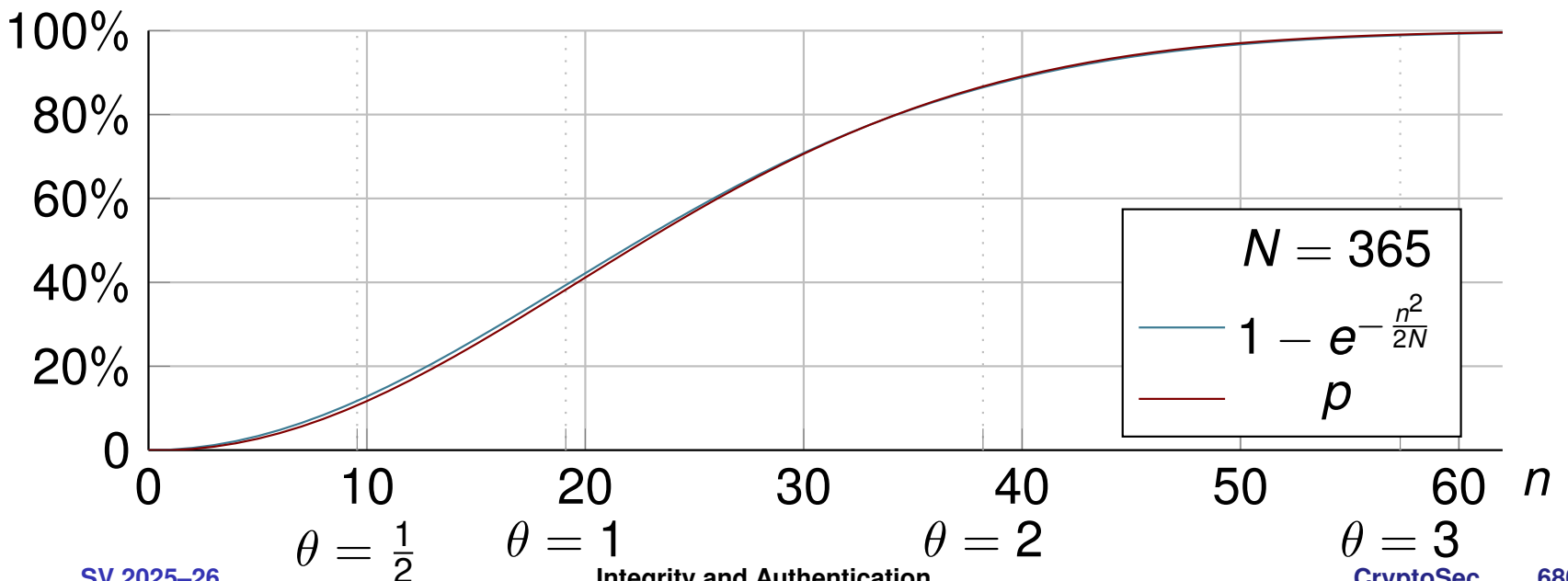
Birthday Paradox

Theorem

If we pick independent random numbers in $\{1, 2, \ldots, N\}$ *with uniform distribution, n times, we get at least one number twice with probability* $p = 1 - \frac{N!}{N^n (N-n)!}$.

If $N \to +\infty$ *and* $n = o(N)$, *we have* $p = 1 - e^{-\frac{n^2}{2N} + o(1)}$.
If $n \sim \theta\sqrt{N}$, *then*

$$p \underset{N \to +\infty}{\longrightarrow} 1 - e^{-\frac{\theta^2}{2}}.$$



Birthday Paradox - Informal Proof

$$
\begin{aligned}
p &\approx 1 - \left(1 - \frac{1}{N}\right)^{\binom{n}{2}} \\
&\approx 1 - \left(1 - \frac{1}{N}\right)^{\frac{n^2}{2}} \\
&= 1 - e^{\frac{n^2}{2}\ln\left(1-\frac{1}{N}\right)} \\
&\approx 1 - e^{-\frac{n^2}{2N}} \\
&= 1 - e^{-\frac{\theta^2}{2}}
\end{aligned}
$$

Birthday Paradox - Proof — i

Proof. We use the Stirling Approximation

$$n! \underset{n\to+\infty}{\sim} \sqrt{2\pi n} e^{-n} n^n$$

We have

$$\begin{aligned}
1-p &= \frac{\#\text{injections from } N \text{ to } n}{\#\text{functions from } N \text{ to } n} \\
&= \frac{N!}{N^n (N-n)!} \\
&\sim \left(1-\frac{n}{N}\right)^{-N+n} e^{-n} \\
&= \exp\left[-n + (-N+n)\log\left(1-\frac{n}{N}\right)\right]
\end{aligned}$$

Birthday Paradox - Proof — ii

We now use $\log(1-\varepsilon) = -\varepsilon - \frac{\varepsilon^2}{2} + o(\varepsilon^2)$ as $\varepsilon \to 0$

$$\begin{aligned} 1-p &\sim \exp\left[-n + (-N+n)\log\left(1-\frac{n}{N}\right)\right] \\ &\sim \exp\left[-\frac{n^2}{2N} + o(1)\right] \end{aligned}$$

Finally, $1-p \to e^{-\frac{\theta^2}{2}}$ when $n \sim \theta\sqrt{N}$ □

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** $T\{y\}$ defined **then**
4: yield $(x, T\{y\})$ and stop
5: **end if**
6: set $T\{y\} = x$
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: loop
2:     pick a (new) random x
3:     compute y = h(x)
4:     if T{y} defined then
5:         yield (x, T{y}) and stop
6:     end if
7:     set T{y} = x
8: end loop
```

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 SHA-2: $N = 2^{256}$, complexity $\sim 2^{128}$

Note: for collision search I, this is a worst-case complexity

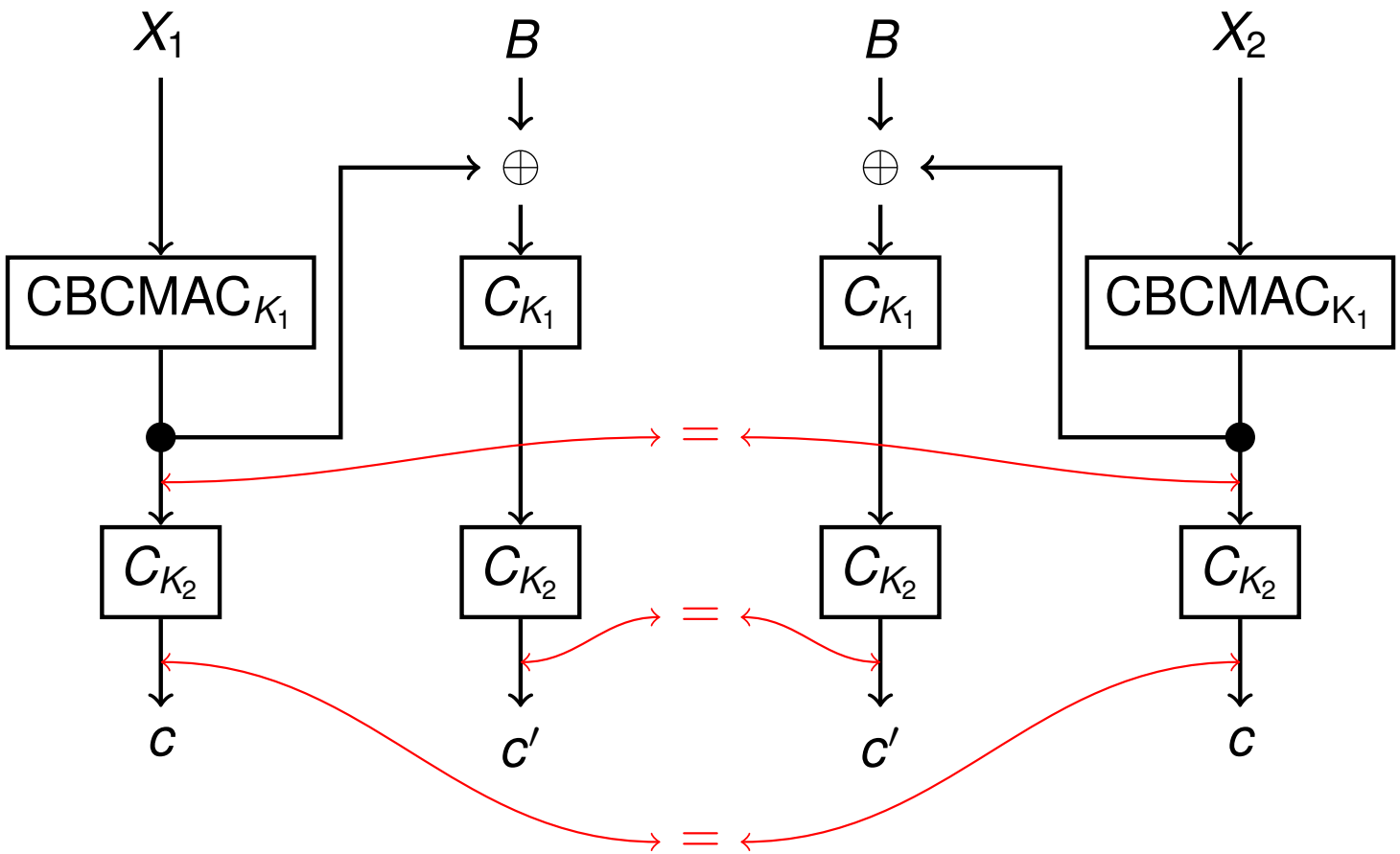
Example: Birthday Attack on EMAC

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 $\text{MAC}(X_2 \| B) = c'$



Variant: Collision between Two Lists

Theorem

If $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ are independent and uniformly distrbuted in $\{1, \ldots, N\}$, the probability there exist i and j such that $x_i = y_j$ is

$$p = 1 - e^{-\frac{mn}{N} + o\left(\frac{mn}{N}\right)}$$

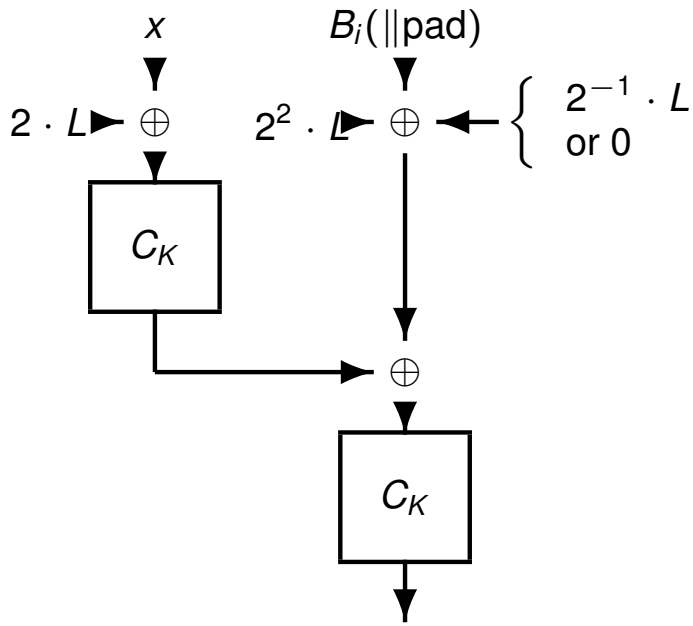
Proof.

$$\begin{aligned} 1 - p &\approx \left(1 - \frac{1}{N}\right)^{mn} \\ &= e^{mn \log\left(1 - \frac{1}{N}\right)} \\ &= e^{-\frac{mn}{N} + o\left(\frac{mn}{N}\right)} \end{aligned}$$

□

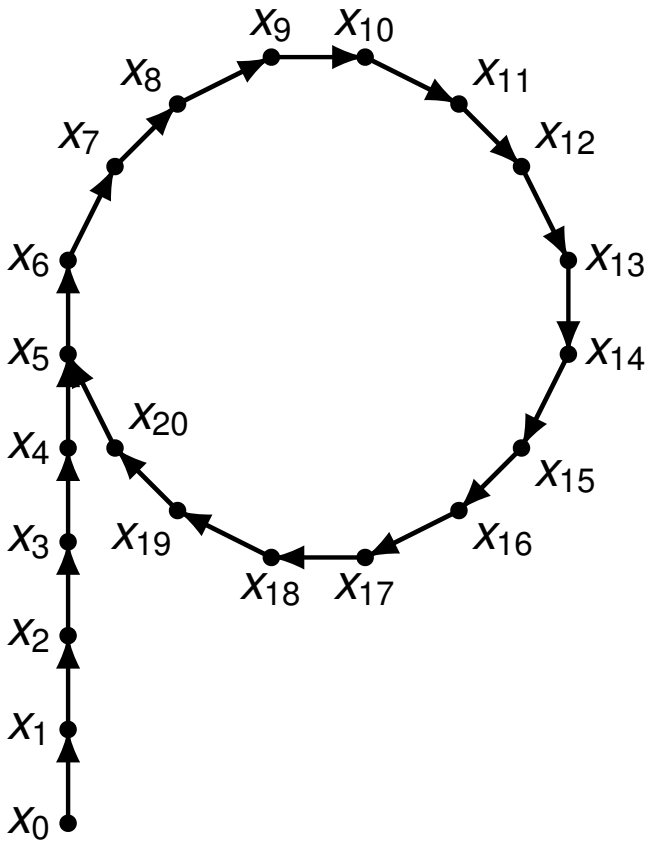
Example: Birthday Attack on PMAC

1: select a block x arbitrarily
2: for $2^{\frac{\text{blocklength}}{2}}$ pairwise different incomplete blocks B_i, get $t_i = \text{MAC}(x \| B_i)$
3: for $2^{\frac{\text{blocklength}}{2}}$ pairwise different complete blocks B'_i, get $t'_j = \text{MAC}(x \| B'_j)$
4: look for a collision $t_i = t'_j$
5: deduce $(B_i \| \text{pad}) \oplus B'_j = 2^{-1} \cdot L$
6: (when truncation is used) check if the obtained L is correct
7: make forgeries using L ▷ exercise: how?



(Almost) Memoryless Collision Search

The Rho (ρ) Effect



- $x_{i+1} = F(x_i)$
- ρ 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: $\quad a \leftarrow F(a)$
6: $\quad b \leftarrow F(F(b))$
7: **until** $a = b$
8: $a \leftarrow x_0$
9: **if** $a = b$ **then** fail
10: **while** $a \neq b$ **do**
11: $\quad a_{\text{old}} \leftarrow a$
12: $\quad b_{\text{old}} \leftarrow b$
13: $\quad a \leftarrow F(a)$
14: $\quad b \leftarrow F(b)$
15: **end while**
16: output a_{old}, b_{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 $3i + 2\lambda$ computations F
- which is on average

$$\begin{aligned} & 5E(\lambda) + \frac{3}{2}E(\tau) \\ = \quad & 6.5\sqrt{\pi/8} \times \sqrt{N} \end{aligned}$$

as $E(i) = E(\lambda) + \frac{1}{2}E(\tau)$

(fail if $\lambda = 0$)

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 $i \geq \lambda$

Example

A Random Function

$x \mapsto uv \mapsto$ first byte of SHA256("3.1415927-*uv*") mod 128

where *uv* is *x* in hexadecimal

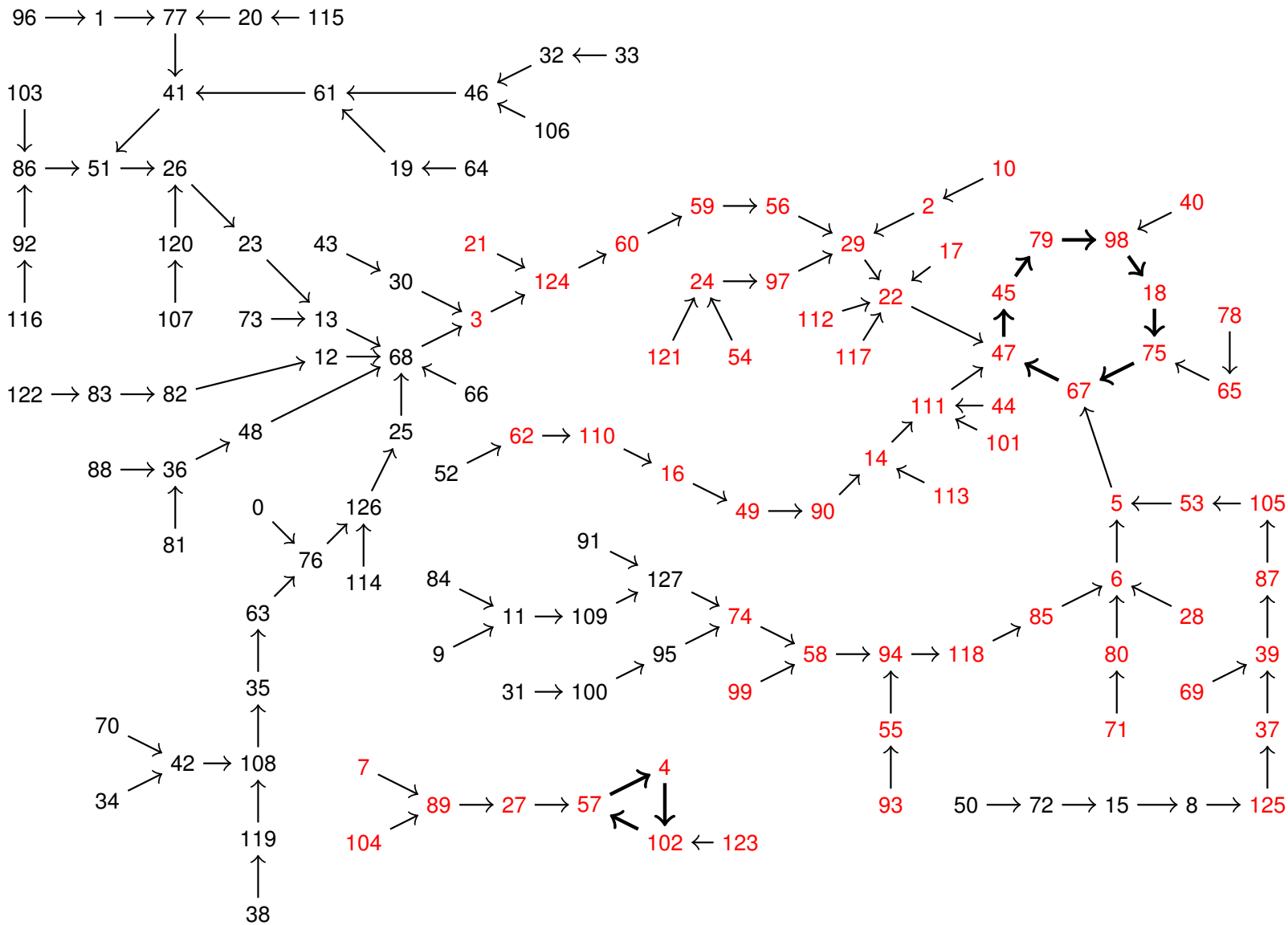
```
#! /bin/bash

string="3.1415927"

for i in {0..127}
do
  j=‘printf "$string-%02x" $i | sha256sum‘
  j=‘echo $j | tr "abcdef" "ABCDEF"‘
  j=‘echo $j | sed "s/^\(..\).*$/ibase=16;obase=2;\1/g" | bc‘
  j=‘echo $j | sed "s/.$//g"‘
  j=‘echo $j | sed "s/^/ibase=2;obase=A;/g" | bc‘
  echo "$i -> $j"
done
```

Note: $\sqrt{\frac{128\pi}{8}} \approx 7$

Example



red: points at distance ≤ 7 to a loop

Cycle Detection Algorithms

- Floyd (1967)
- Gosper (1972)
- Brent (1980)
- Sedgewick-Szymanski-Yao (1982)
- Quisquater-Delescaille (1989)
- van Oorschot-Wiener (1999)
- Nivasch (2004)

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

Breaking Symmetric Cryptography

- we know generic attacks are unavoidable
- we do not know how to prove security
- empirical security: assume (hope) there is no better attack then known ones
- security $\Longrightarrow$ 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

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 MAC

if we have a n-bit key ($N = 2^n$) and a τ-bit tag,

strategy	preprocessing	memory	time	success proba.
exhaustive search	0	1	2^n	1
random guess	0	1	1	$2^{-\tau}$
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 $n \geq s$
- select $\tau \geq s$

Summary of Generic Attacks against Hash Functions

if we hash onto n bits, ($N = 2^n$)

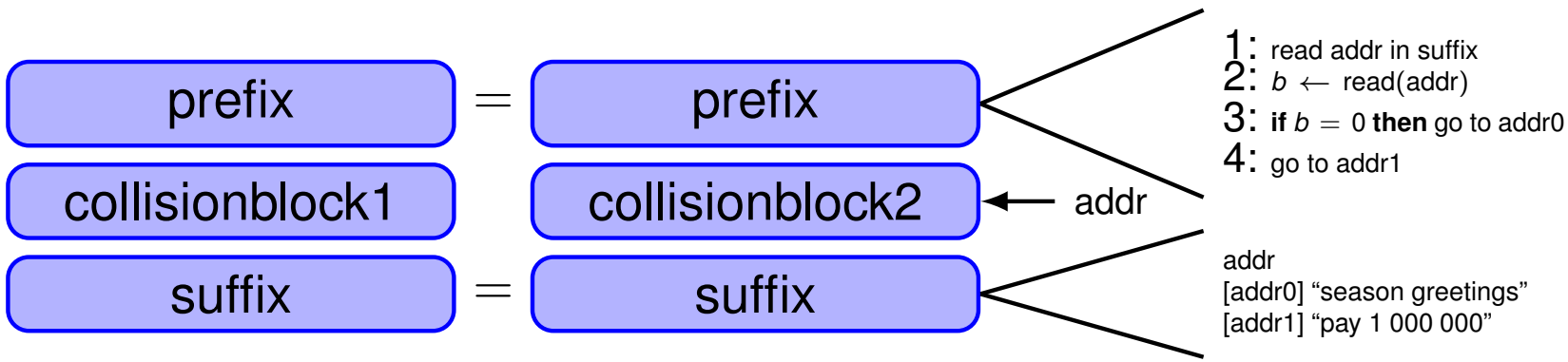
attack	complexity
preimage attack	2^n
collision attack	$2^{\frac{n}{2}}$

Want a security of 2^s?

- want security against inversion only: select $n \geq s$
- want security against collisions: select $n \geq 2s$

Risks When Underestimating Collision Attacks

- some people think that “academic” collisions are of no threat as they are no real documents
- they are random-looking but we can cast them in real-life format
- postscript, jpg, pdf, ...: media format looks like programming
- we can cast a collision



- demonstrated to forge a rogue certificate authority

  [Stevens-Lenstra-Benne de Weger: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities; EUROCRYPT 2007]

- demonstrated with two arbitrary images with SHA-1

  [Albertini: Exploiting Hash Collisions;BlackAlps 2017]

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

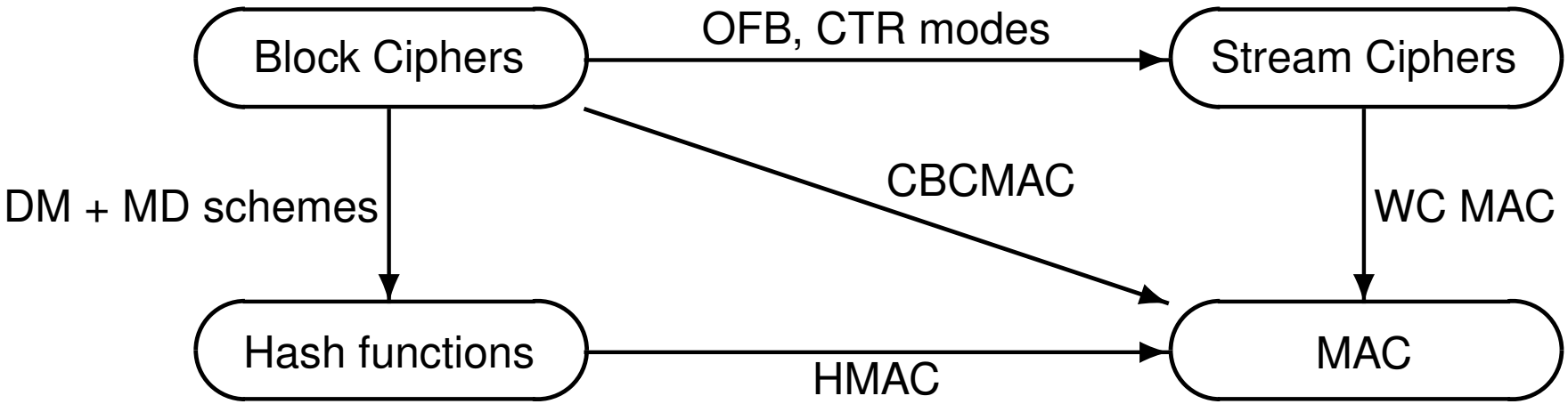
Cryptanalytic Advances

- security is often empirical
  $\rightarrow$ dedicated attacks
- heuristic security against attack methods
  $\rightarrow$ arguments may be wrong, other attack methods can be discovered
- all eggs in the same basket (lack of crypto-diversity)
  $\rightarrow$ more exposure, attacks more devastating
- the quantum threat
  $\rightarrow$ 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, KMAC, CBCMAC, WC-MAC, CCM mode, GCM mode
- **hash functions**: SHA-2, SHA-3
- **commitment**
- **bruteforce collision** within complexity $\mathcal{O}\left(\sqrt{\#\text{range}}\right)$

Dedicated Primitives and Reductions



References

- **Schneier**. *Applied Cryptography.* Wiley & Sons. 1996.
  Crypto for dummies!
- **Ferguson–Schneier**. *Practical Cryptography.* Wiley & Sons. 2003.
  Crypto for dummies!
- **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: SHA-2, SHA-3
- 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
  final exam 2019–20 ex1
- collisions:
  final exam 2013–14 ex2
  final exam 2012–13 ex2
  final exam 2010–11 ex1
  final exam 2016–17 ex4
  final exam 2017–18 ex1
  final exam 2018–19 ex2 (CBC mode)
- GCM issue: final exam 2016–17 ex3
- meaningful collisions: final exam 2024–25 ex1

Roadmap

- Diffie-Hellman: new directions in cryptography
- RSA standards for encryption and signature
- the ElGamal signature dynasty
- post-quantum crypto

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

Diffie-Hellman

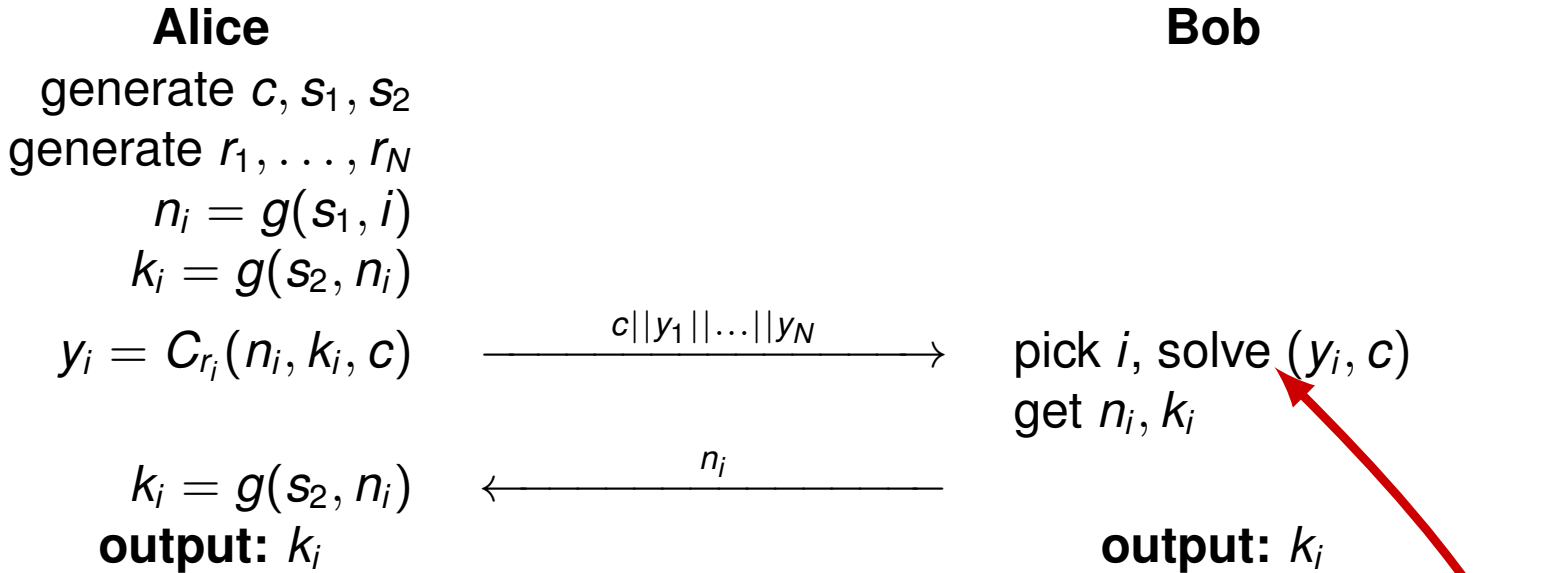
“New Directions in Cryptography” (1976)

[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

Merkle

"Secure Communications over Insecure Channels" (1978)



- Complexity for Alice: N
- Complexity for Bob: S (solving a puzzle)
- Complexity for an attacker: $N \times S$

exhaustive search on r_i s.t. $C_{r_i}^{-1}(y_i)$ has c inside

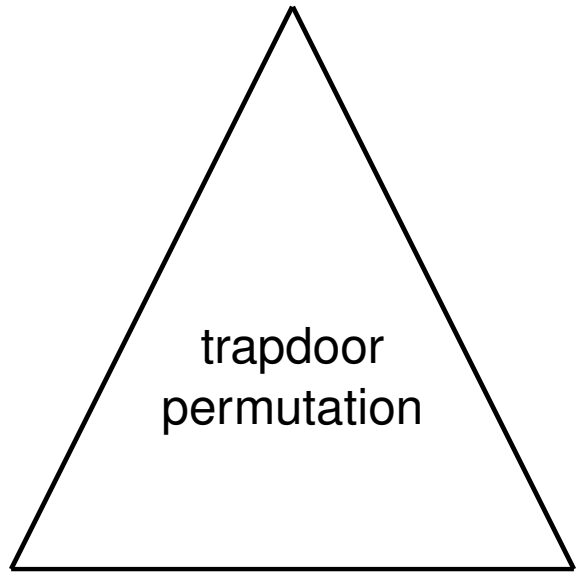
Trapdoor Permutation

- 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

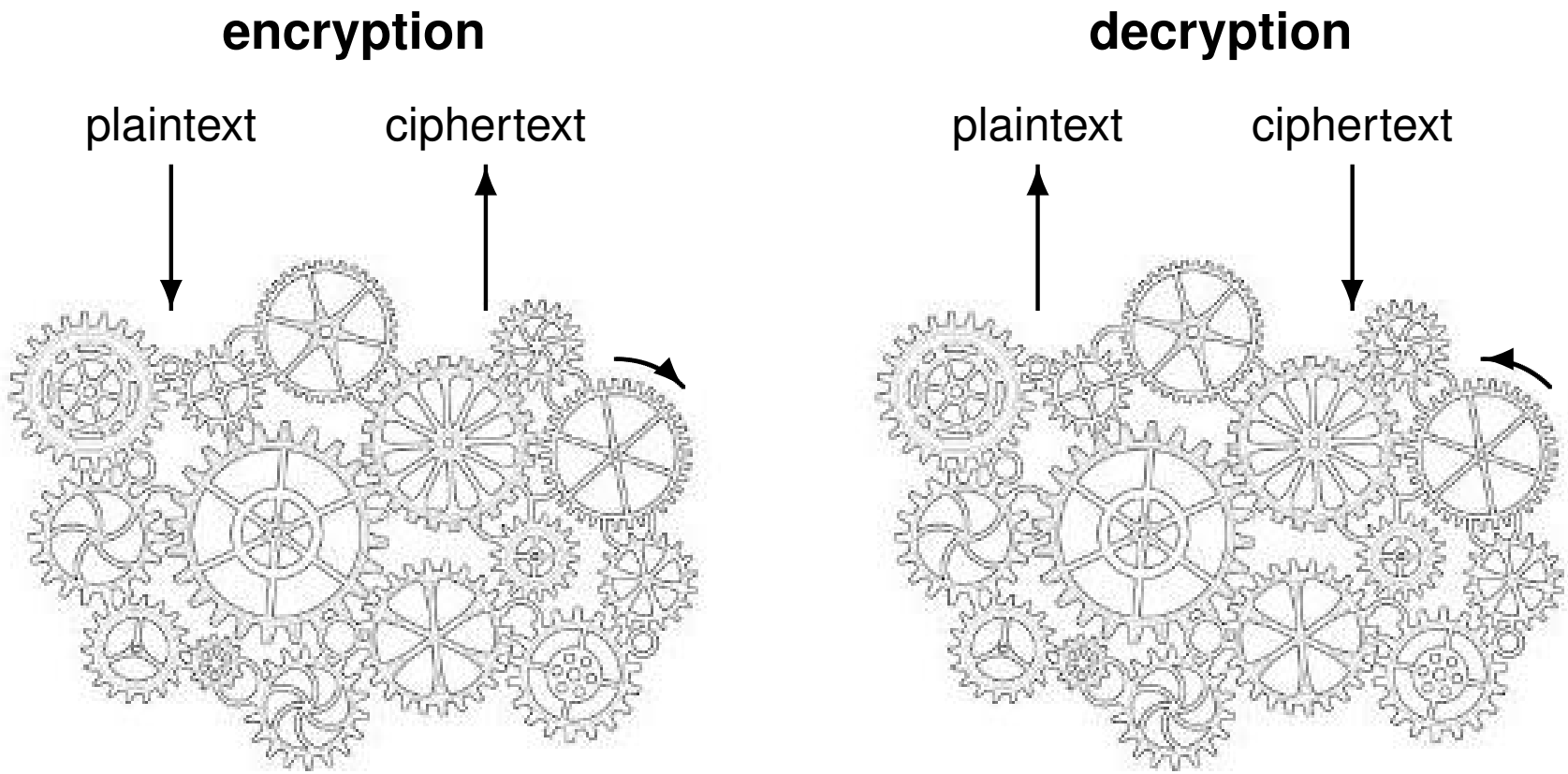
$\text{Gen} \rightarrow (\text{param}, K)$

$\text{InvPerm}_K(\text{Perm}_{\text{param}}(X)) = X$

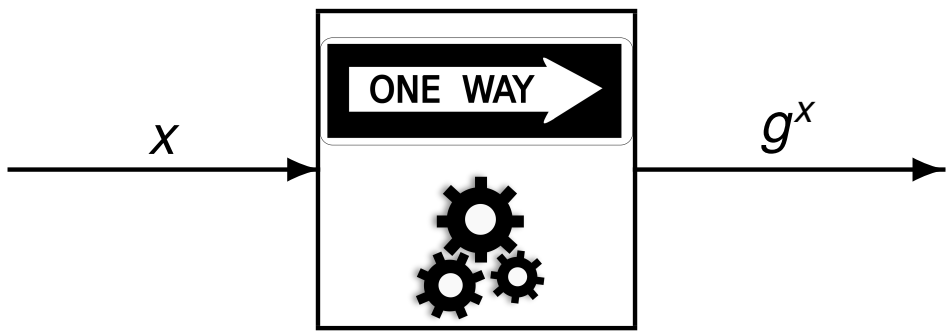
security

confidentiality is preserved

Reversibility in Symmetric Encryption

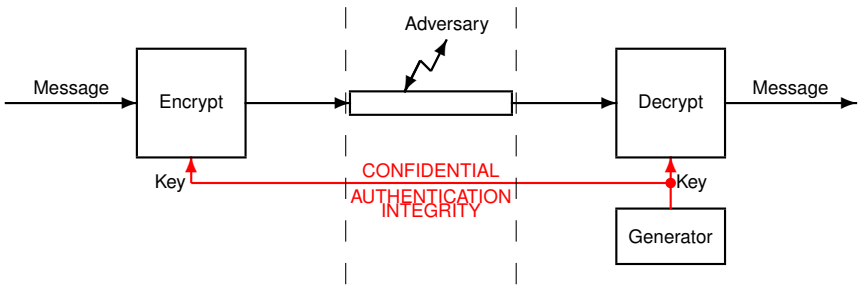


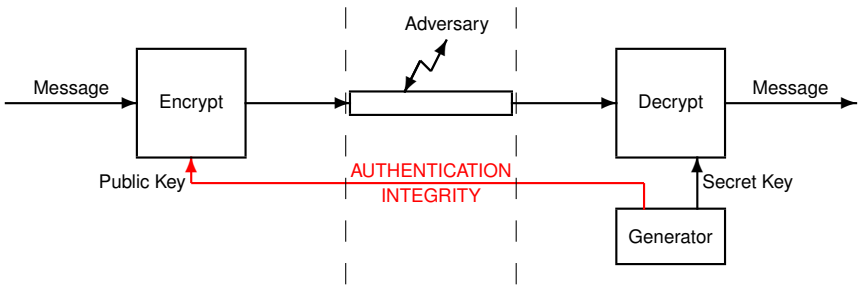
Hard-To-Invert Computation



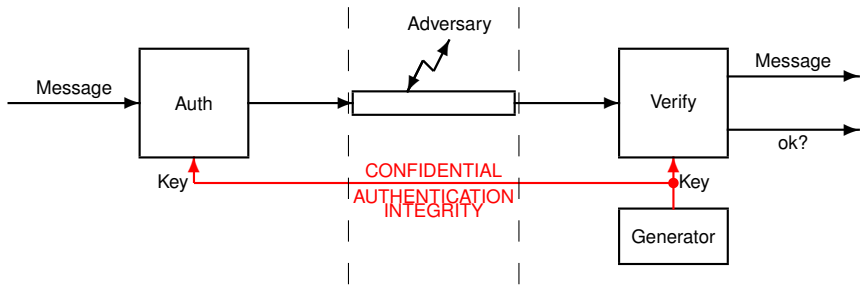
Big Picture

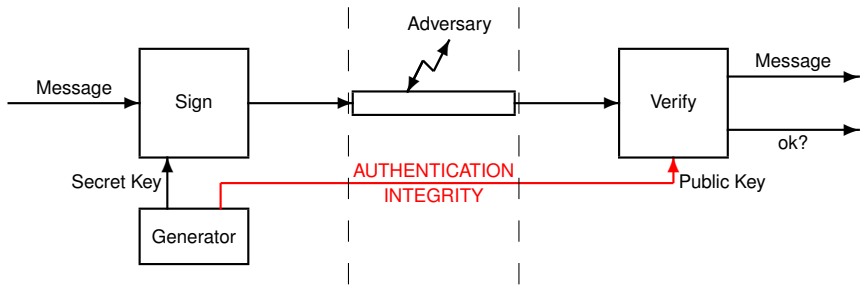
confidential transmission





authenticated transmission

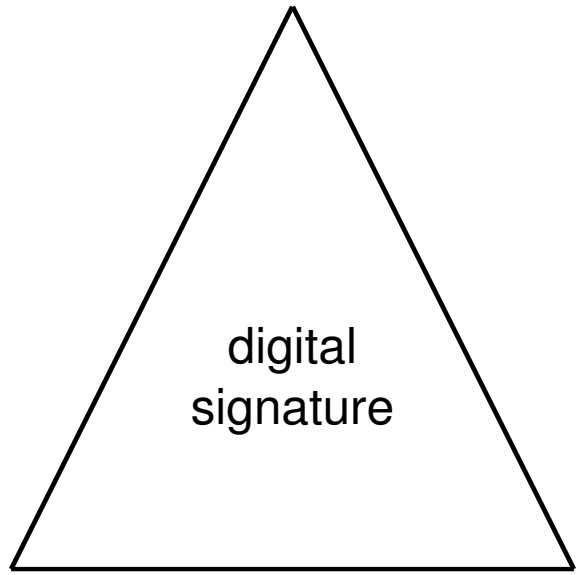




Digital Signature Primitive

Alice and Bob, Gen, Sig, Ver

components



functionality

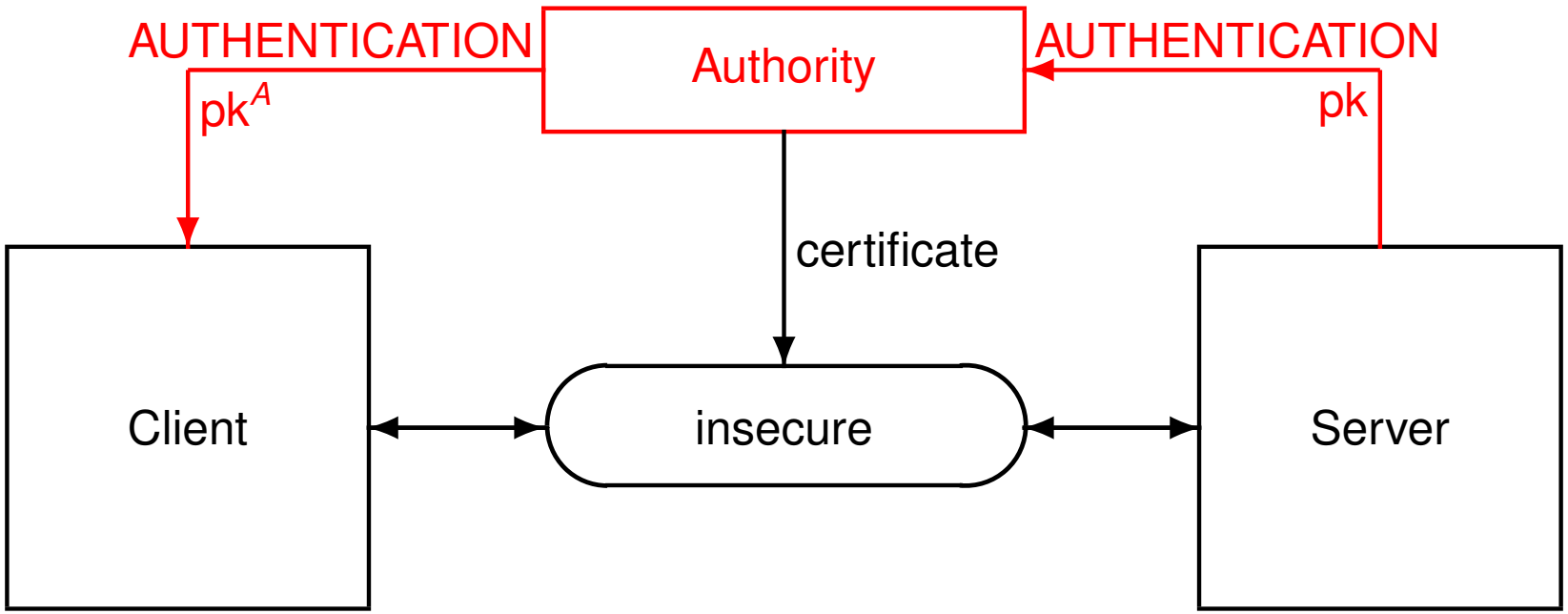
$\text{Gen} \rightarrow (\text{pk}, \text{sk})$

$\text{Ver}_{\text{pk}}(\text{Sig}_{\text{sk}}(X; r)) = X$

security

signature is non-repudiable

Application: Certificates



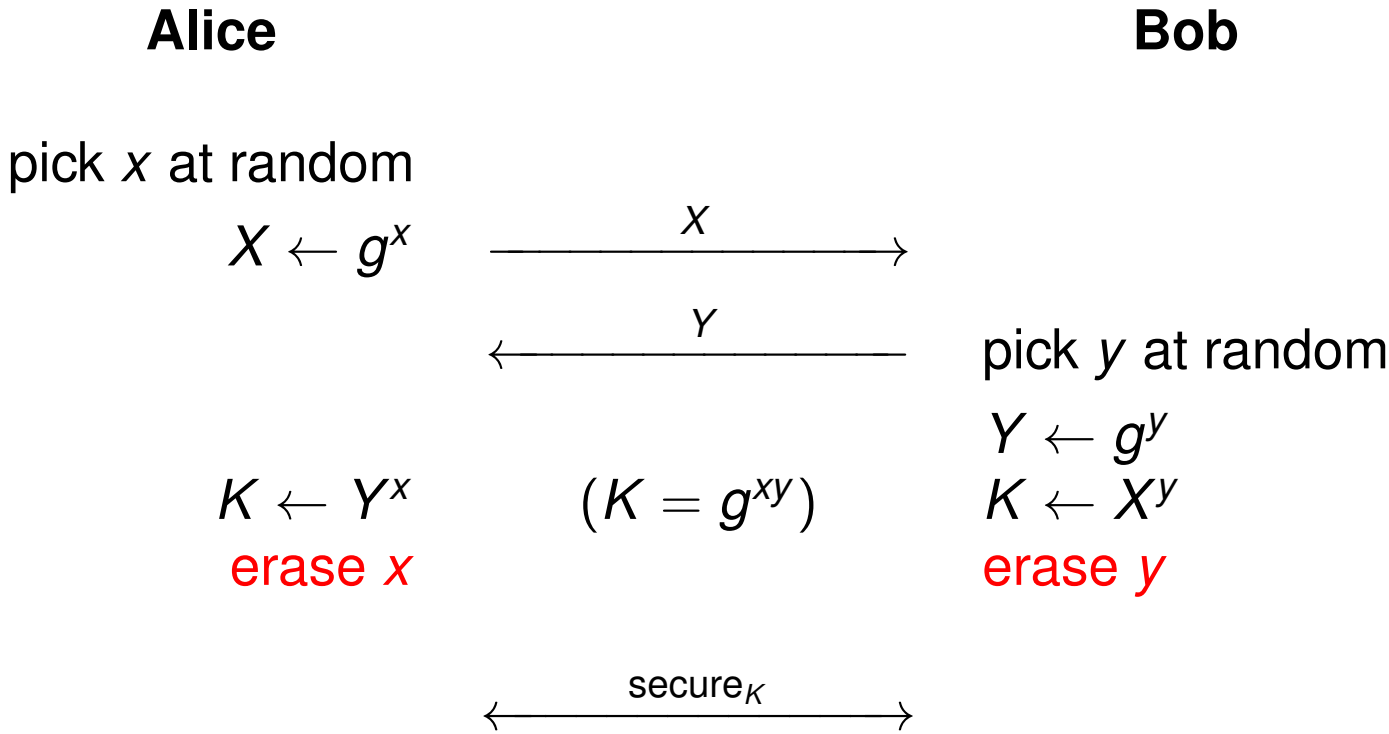
certificate = signature("I certify that public key pk belongs to *S*")

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

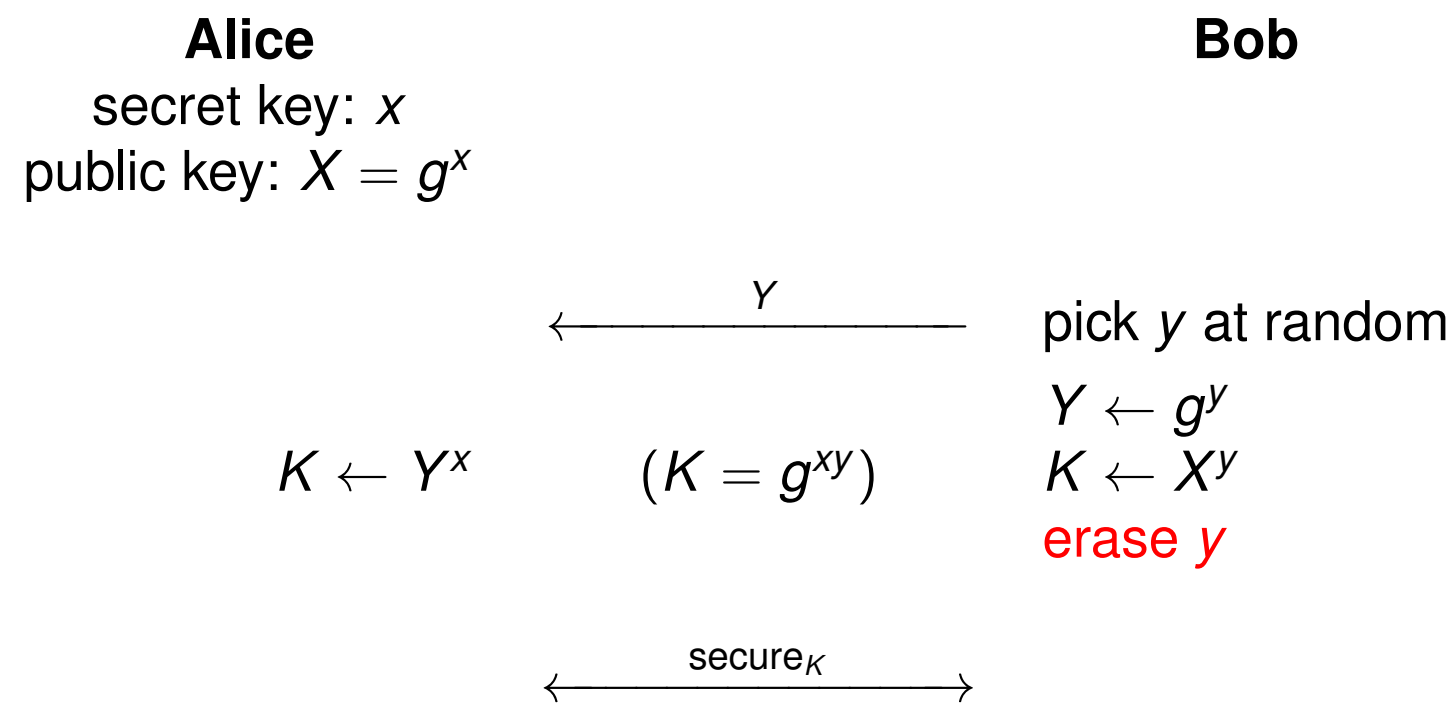
Ephemeral Diffie-Hellman Key Agreement

Assume a group generated by some g



Semi-Static Diffie-Hellman Key Agreement

Assume a group generated by some g



Static Diffie-Hellman Key Agreement

Assume a group generated by some g

Alice		**Bob**
secret key: x		secret key: y
public key: $X = g^x$		public key: $Y = g^y$
$K \leftarrow Y^x$	$(K = g^{xy})$	$K \leftarrow X^y$

$$\longleftrightarrow^{\text{secure}_K}$$

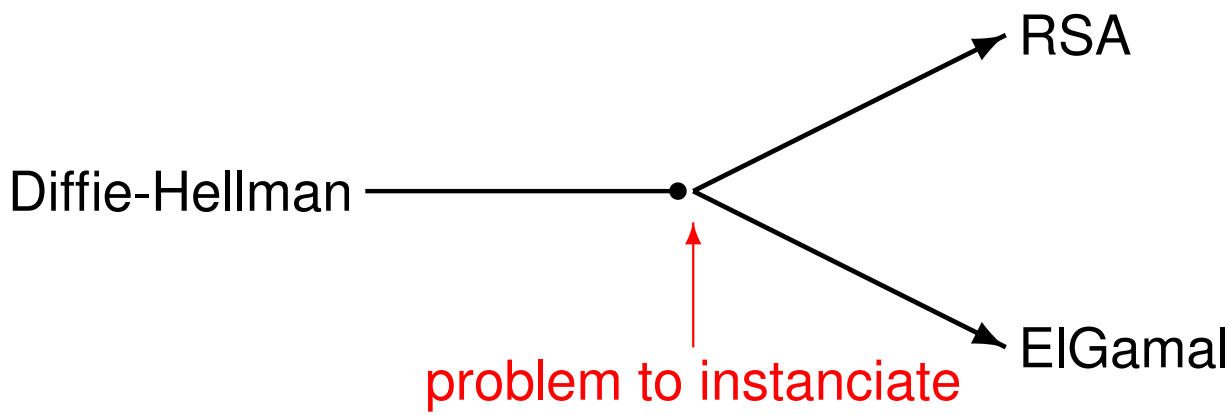
Forward Secrecy

- **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

Case Study: Signal

case study

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 $Y^x = \text{DH}(g, g^x, Y)$

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

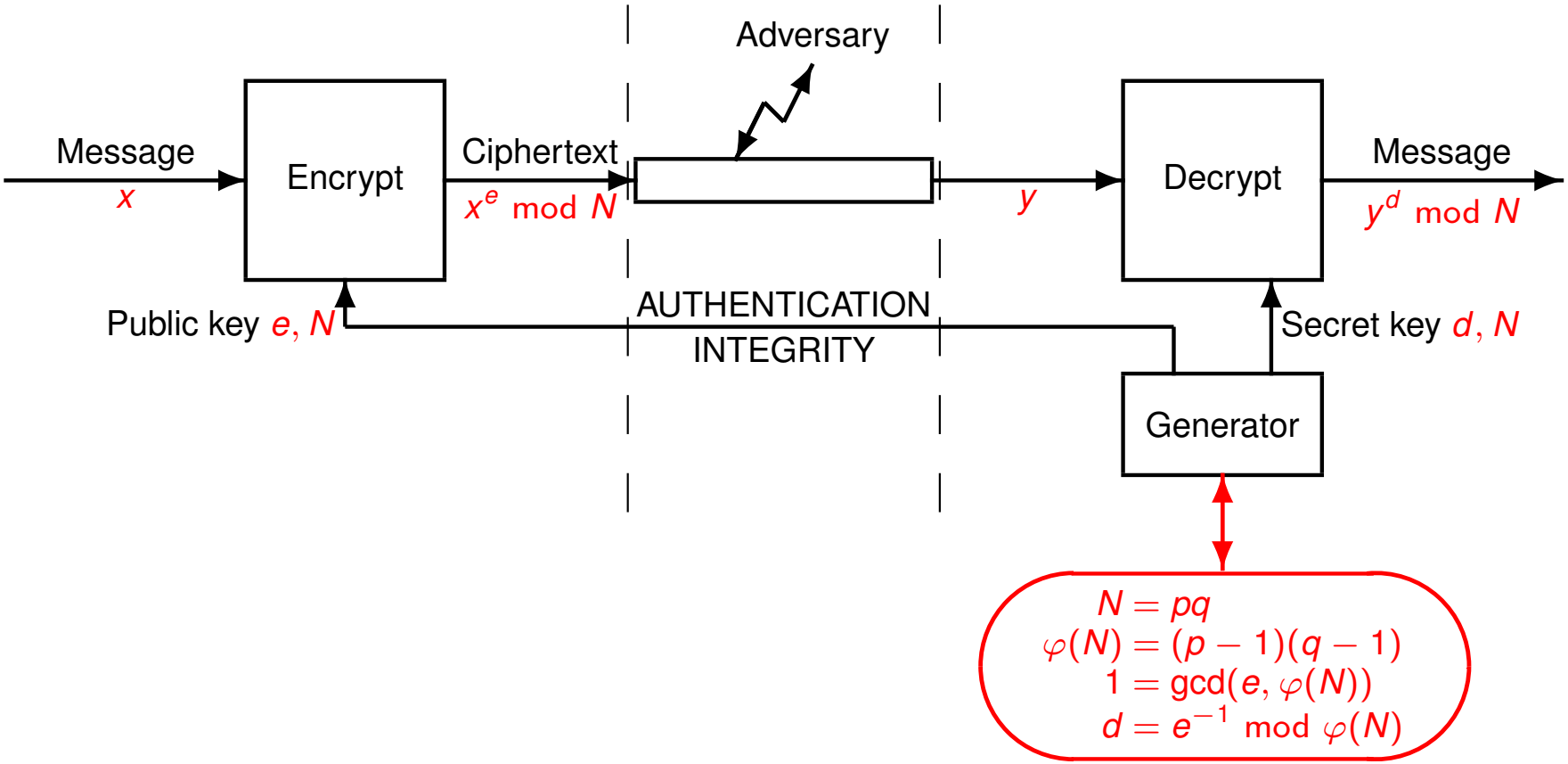
Rivest-Shamir-Adleman (RSA)

(1978)

[Shamir, Rivest, Adleman]

- concrete trapdoor permutation
- $\longrightarrow$ public-key cryptosystem
- $\longrightarrow$ signature scheme

Plain RSA


Adversary
Message
x
Encrypt
Ciphertext
$x^e \bmod N$
y
Decrypt
Message
$y^d \bmod N$
Public key e, N
AUTHENTICATION
INTEGRITY
Secret key d, N
Generator
$N = pq$
$\varphi(N) = (p-1)(q-1)$
$1 = \gcd(e, \varphi(N))$
$d = e^{-1} \bmod \varphi(N)$

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.)

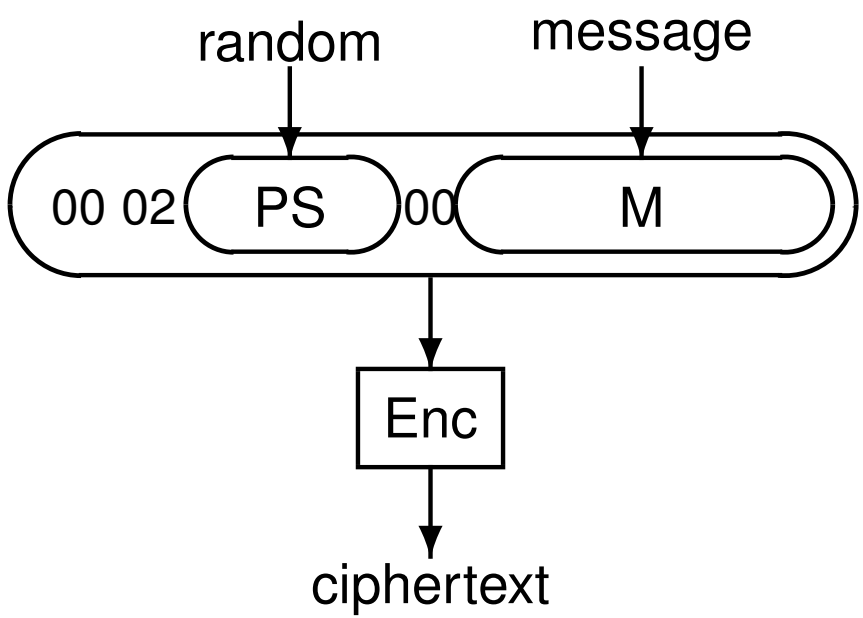
Encryption:

1. generate a pseudorandom string PS of non-zero bytes so that $M \| \text{PS}$ is of $k - 3$ bytes
2. construct string $00 \| 02 \| \text{PS} \| 00 \| M$ 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

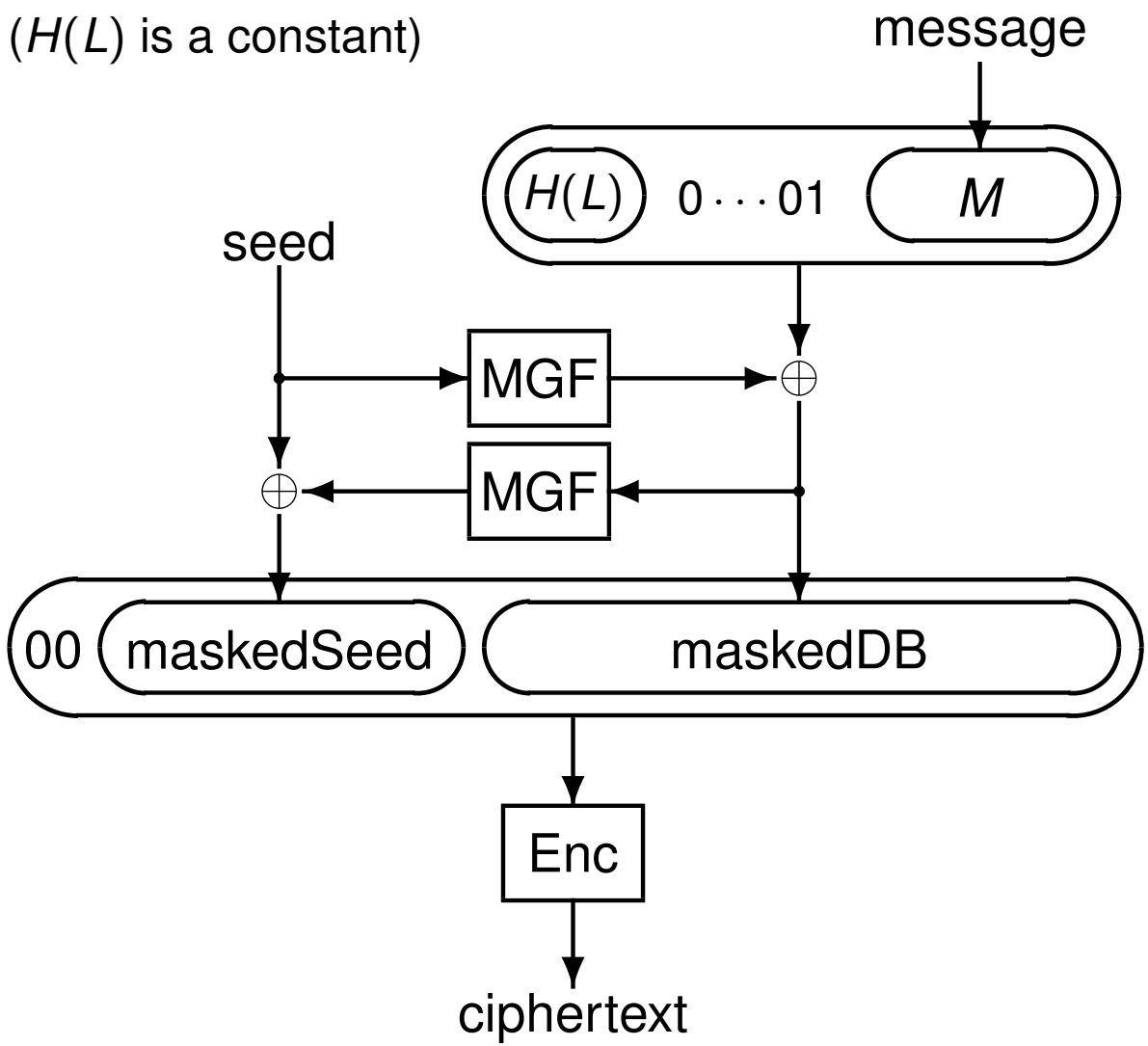
Decryption:

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 $00 \| 02 \| \text{PS} \| 00 \| M$ format for some byte strings PS and M where PS has no zero bytes
5. output M

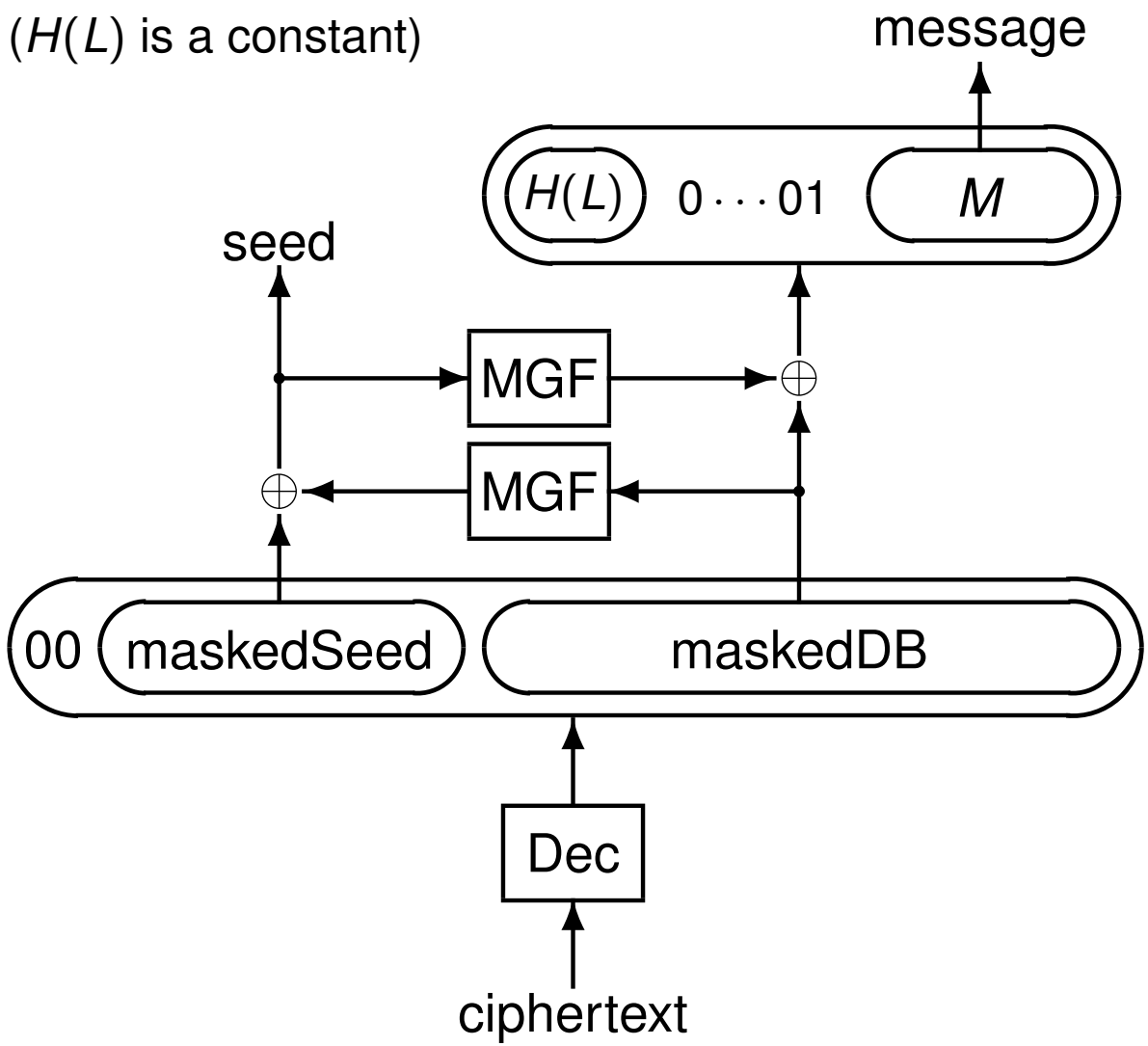
PKCS#1v1.5 Encryption



RSA-OAEP Encryption



RSA-OAEP Decryption



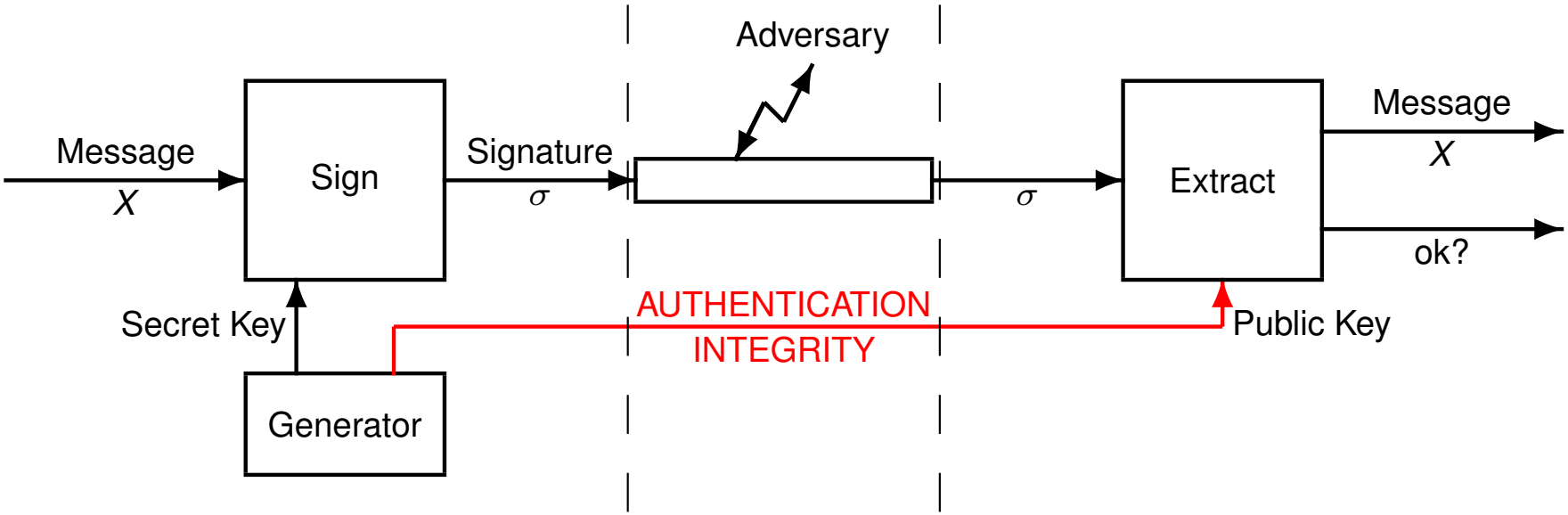
Mask Generation Function in RSA-OAEP

The PKCS specifications further suggests an mask generation function MGF1 which is based on a hash function. The $\mathrm{MGF1}_\ell(x)$ string simply consists of the ℓ leading bytes of

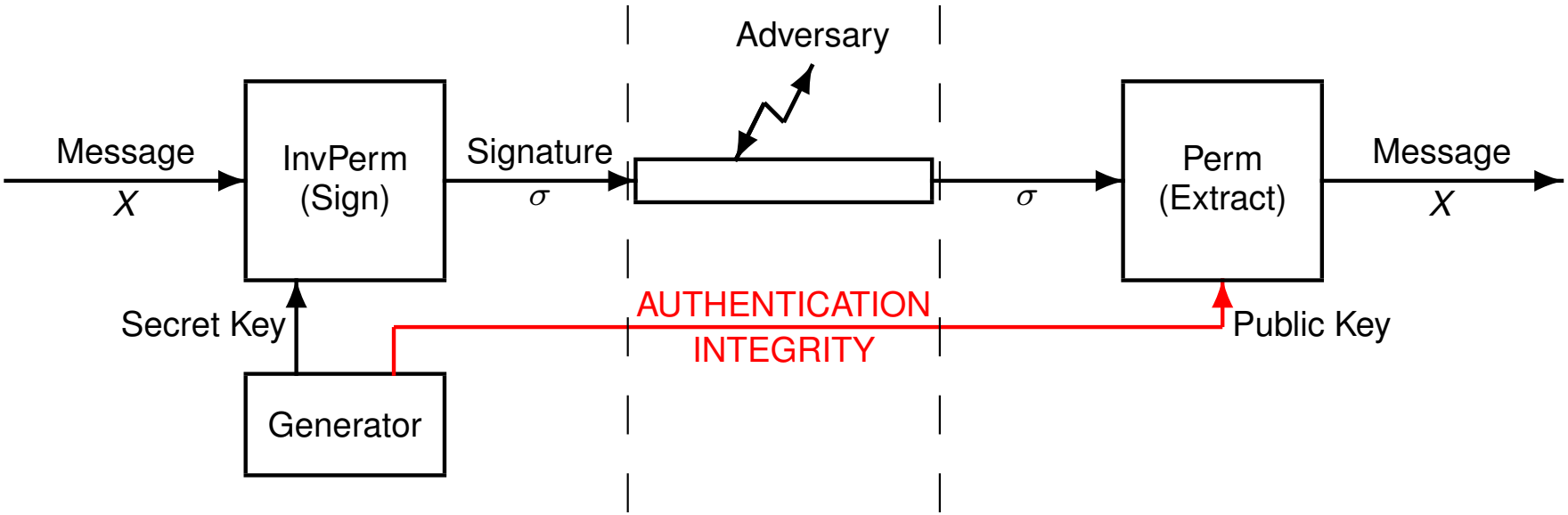
$$H(x\|00000000)\|H(x\|00000001)\|H(x\|00000002)\|\cdots$$

in which x is concatenated to a four-byte counter.

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: $\mathsf{sk} = (d, N)$.

Public key: $\mathsf{pk} = (e, N)$.

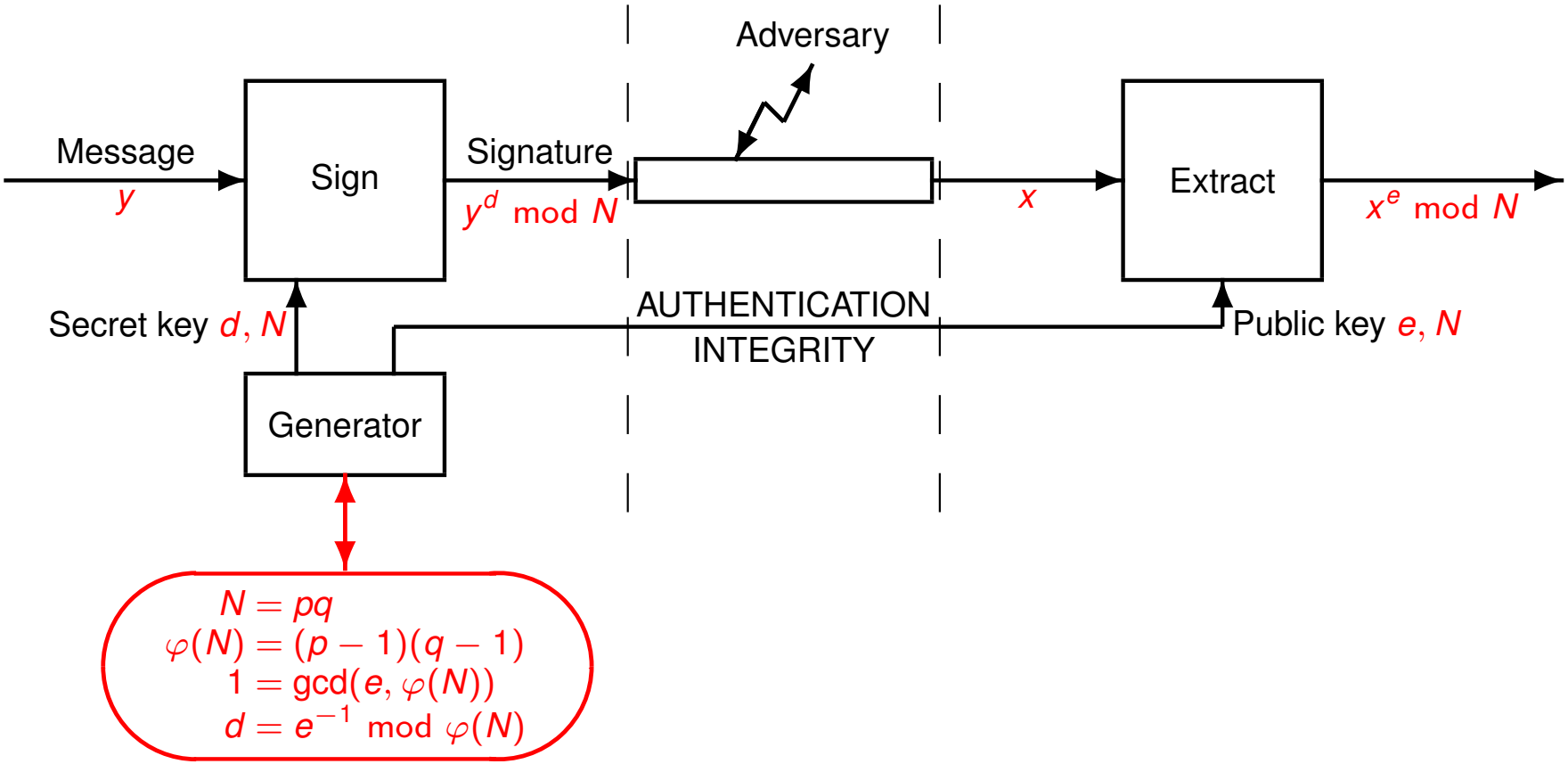
Message: an element $y \in \mathbf{Z}_N$.

Signature generation: $x = y^d \bmod N$.

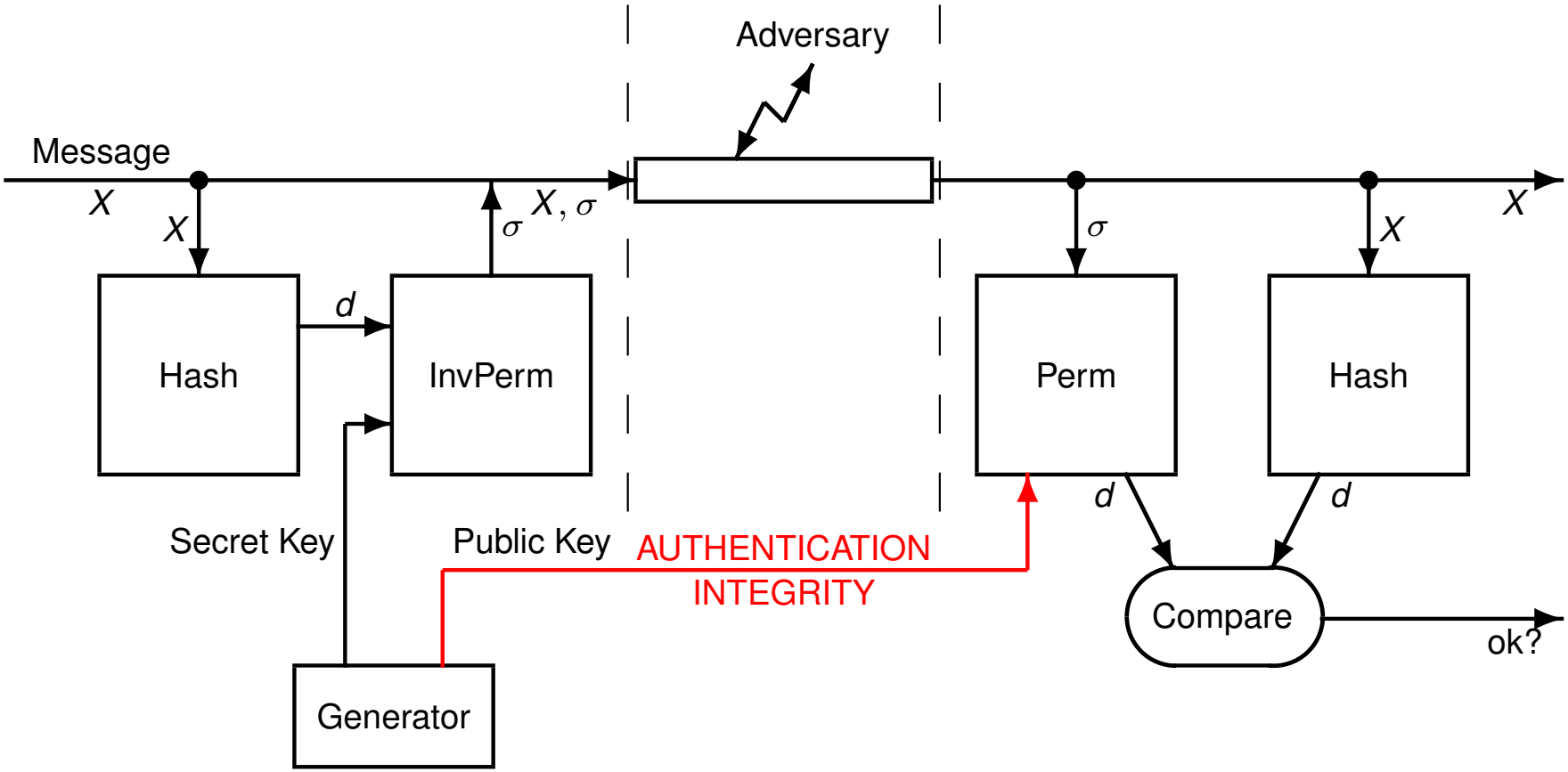
Extraction: $y = x^e \bmod N$.

(Signature with message recovery)

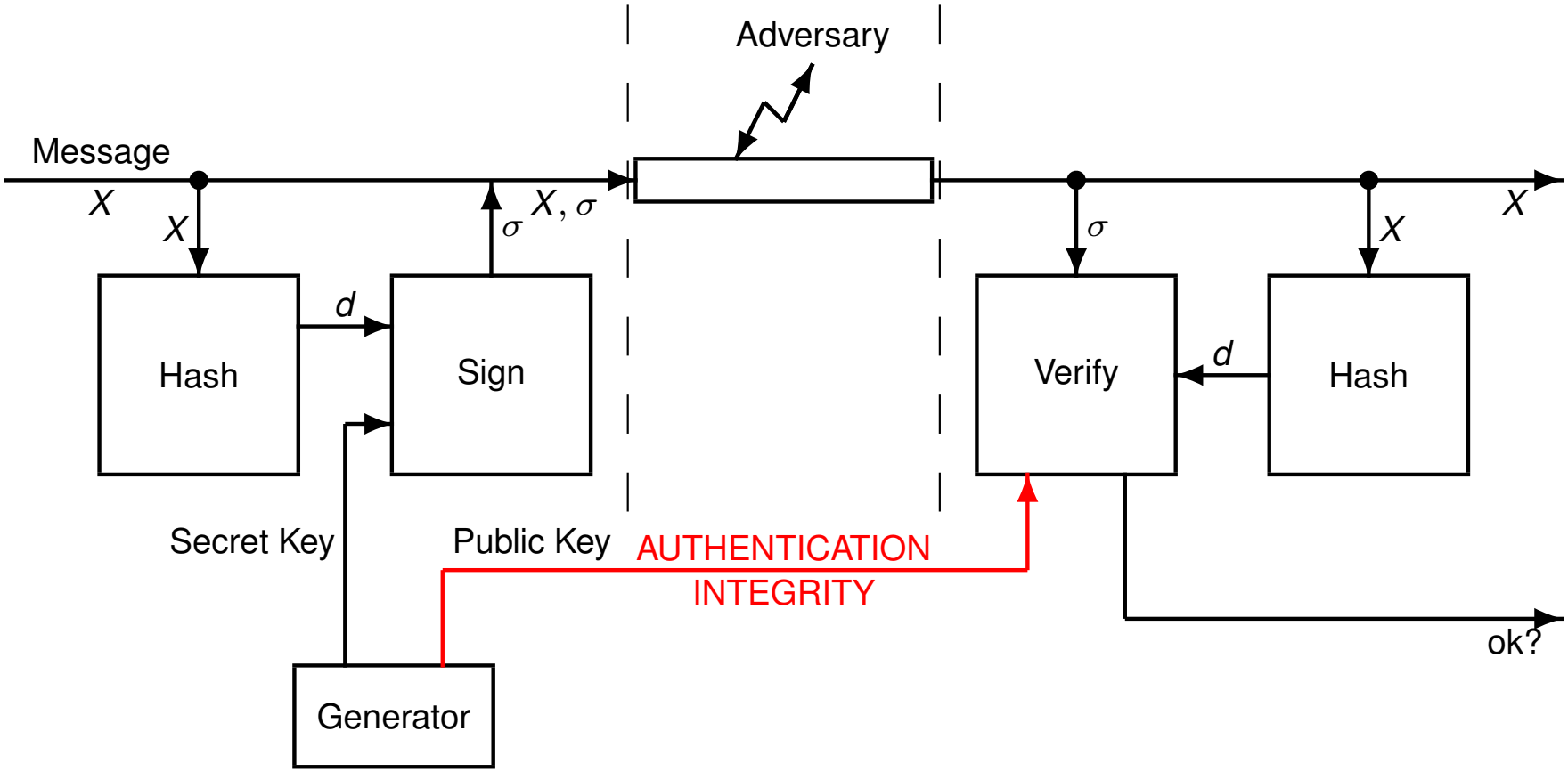
Plain RSA Signature



Trapdoor Permutation to Signature



More Generally: Hash-and-Sign Paradigm



PKCS#1v1.5

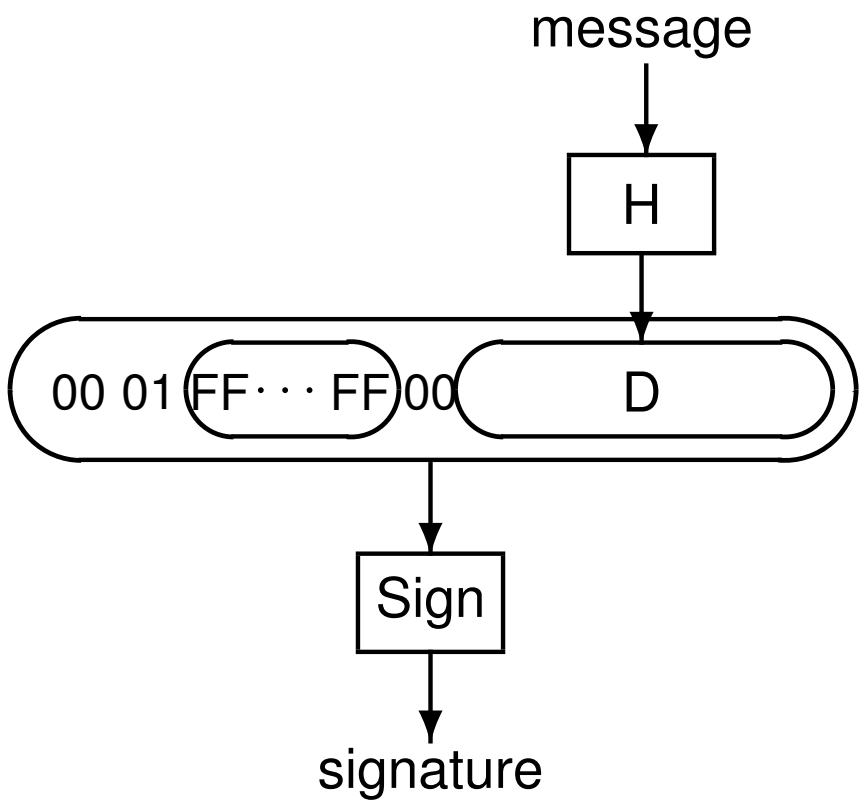
We are given a modulus N of k bytes.

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 $00 \| 01 \| \mathrm{FF} \cdots \mathrm{FF} \| 00 \| D$ is converted into an integer.
5. compute the plain RSA signature.
6. convert the result into a string of k bytes.

Signature Verification

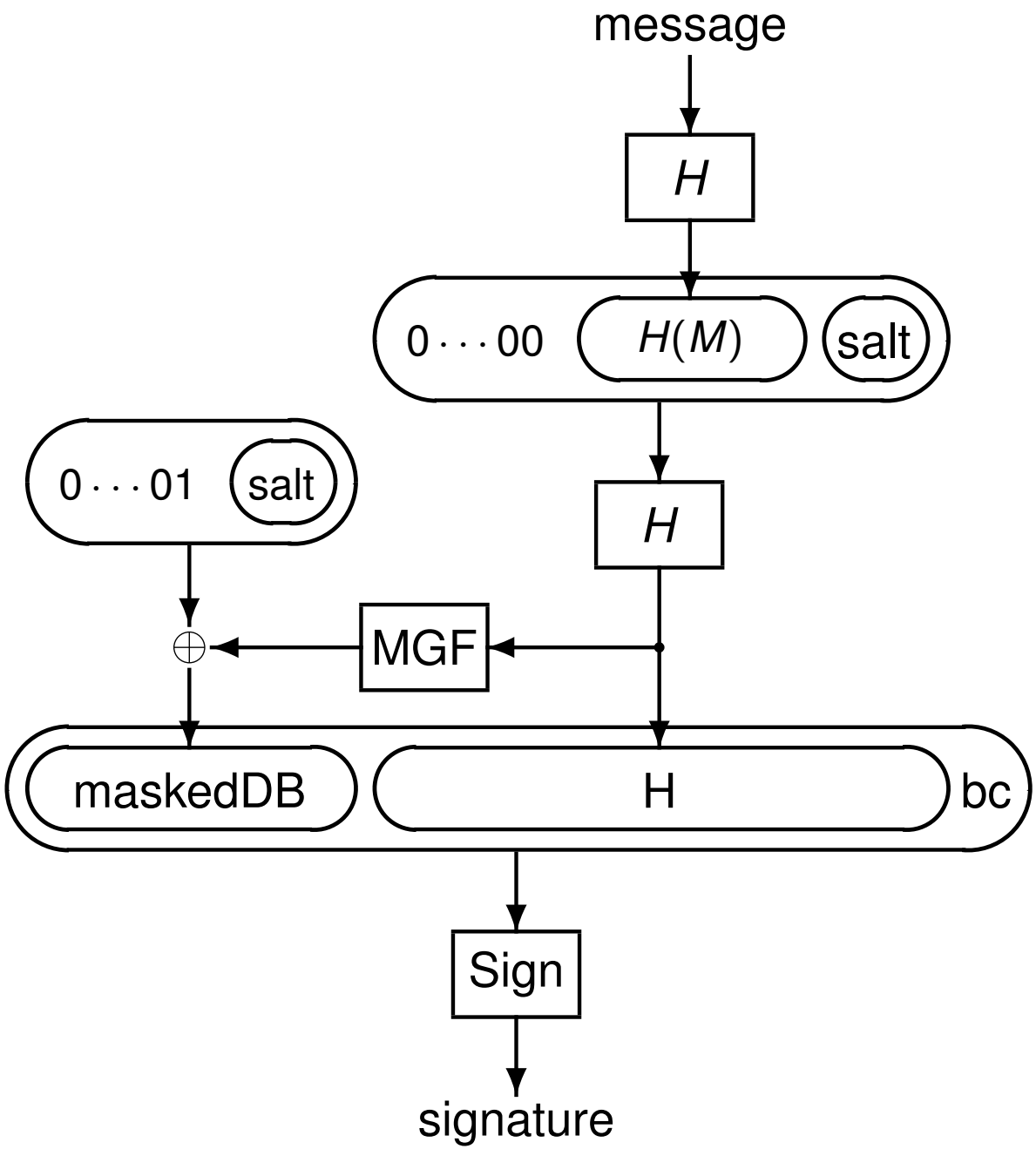
1. convert the signature into an integer. Reject it if it is greater than the modulus.
2. perform the plain RSA extraction and obtain another integer.
3. convert back the integer into a byte string.
4. check that the string has the $00\|01\|FF\ldots FF\|00\|D$ format for a byte string D.
5. decode the 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

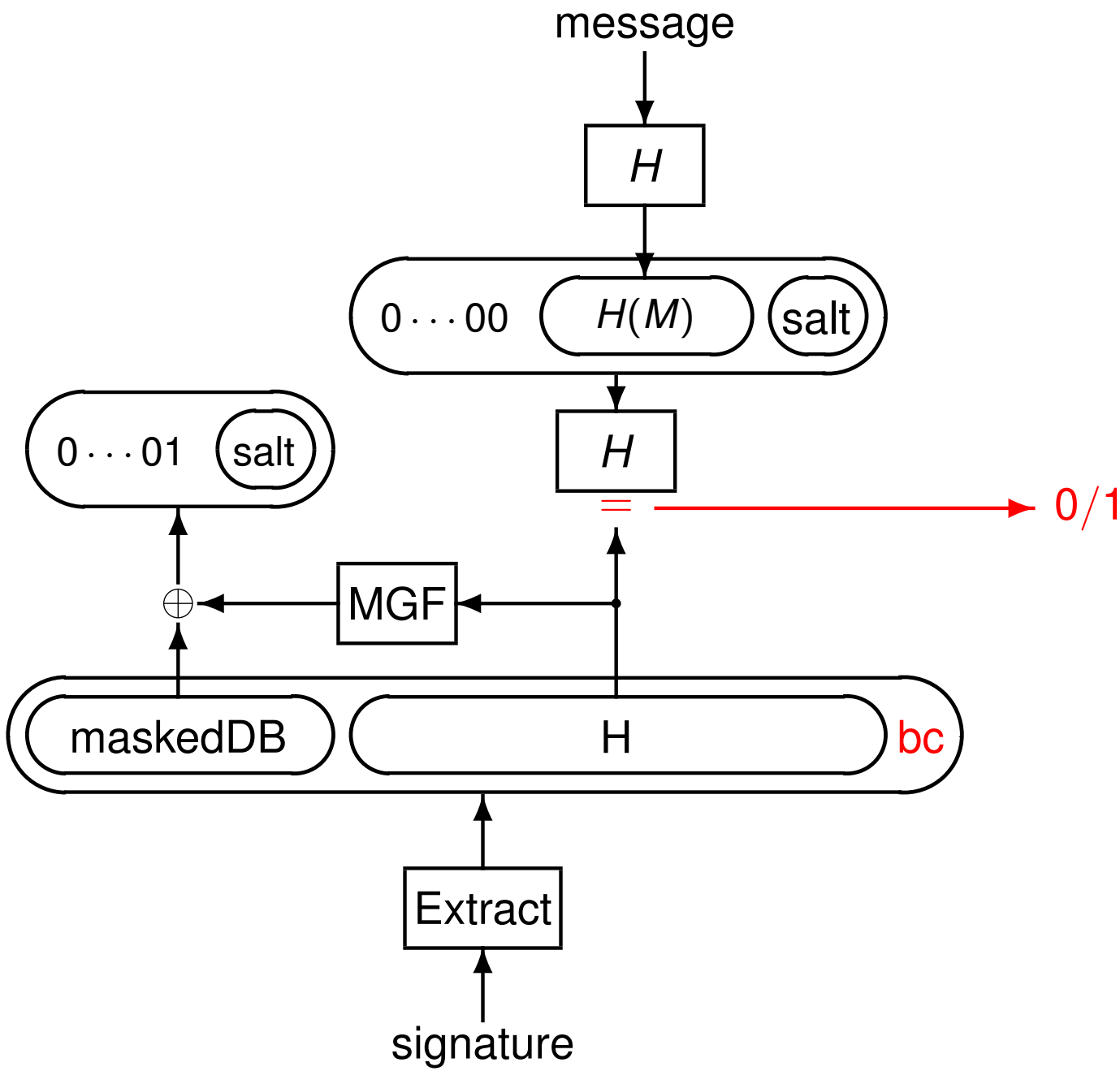


RSA signature without message recovery

RSA-PSS


message
H
0 · · · 00
H(M)
salt
0 · · · 01
salt
H
MGF
maskedDB
H
bc
Sign
signature

RSA-PSS Verification



Case Study: TLS

case study

7 Public-Key Cryptography

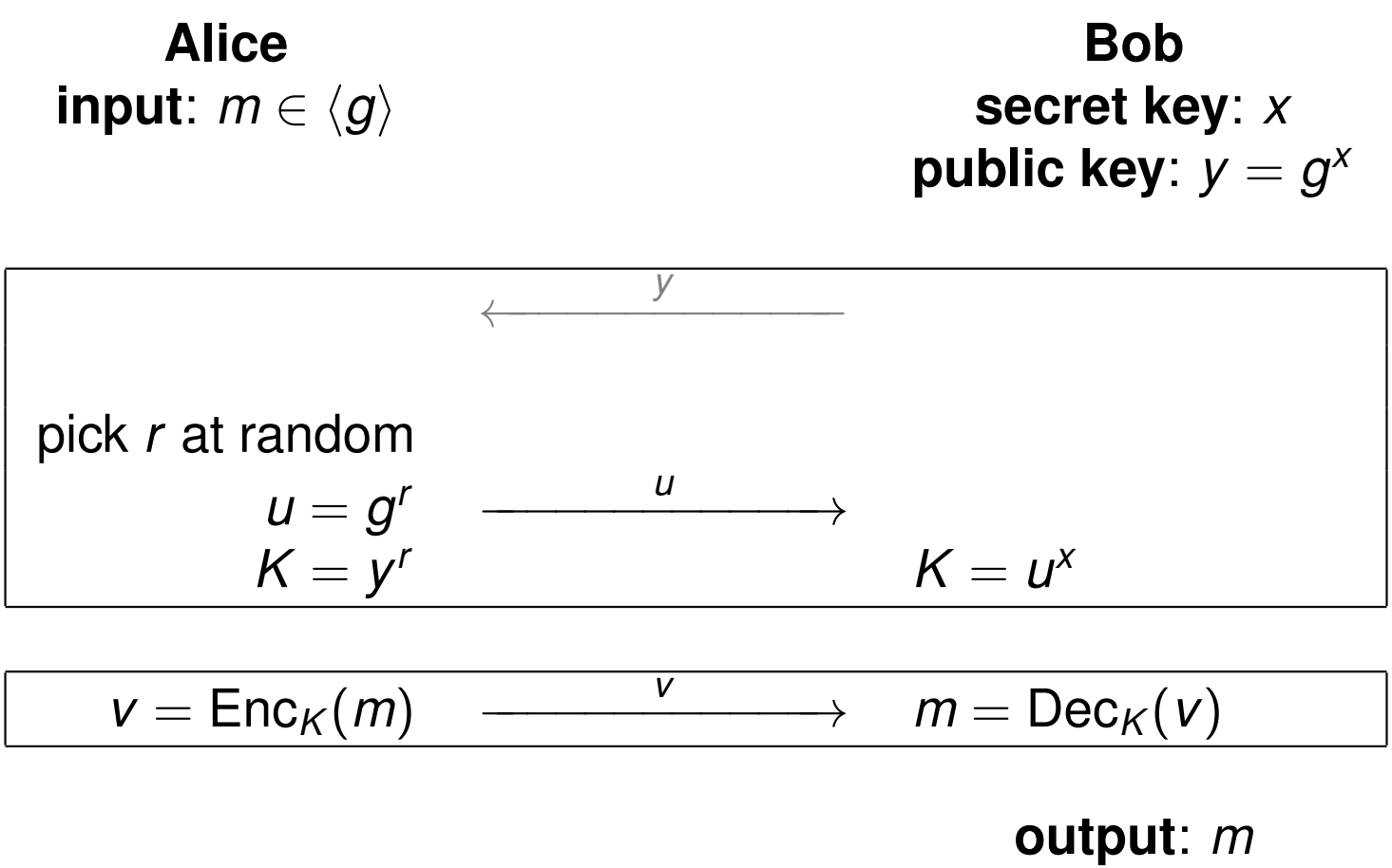
- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

ElGamal Cryptosystem Generalized (Reminder)

Alice		**Bob**
input: $m \in \langle g \rangle$		**secret key**: x **public key**: $y = g^x$
	$\xleftarrow{y}$	
pick r at random $u = g^r$	$\xrightarrow{u}$	
$K = y^r$		$K = u^x$
$v = mK$	$\xrightarrow{v}$	$m = vK^{-1}$

output: m

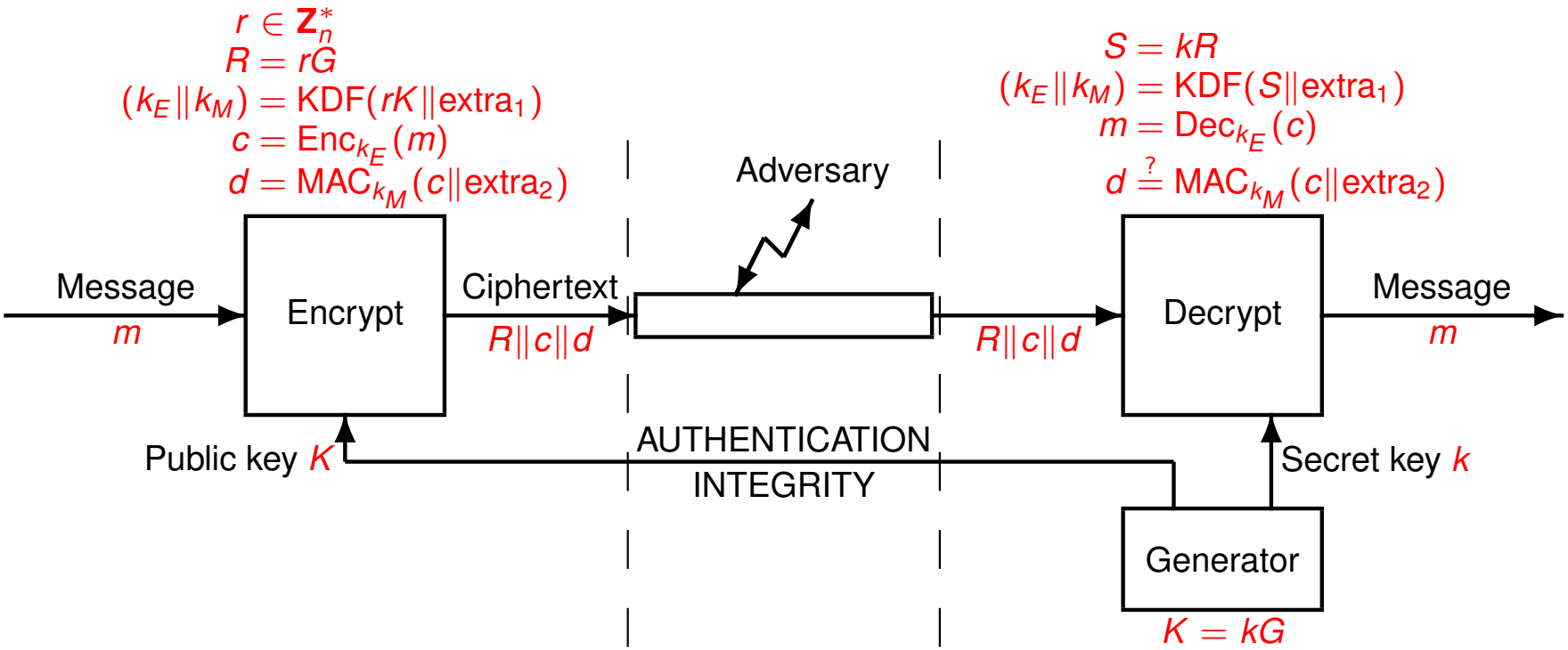
ElGamal Cryptosystem More Generalized



From ElGamal to ECIES

Alice **input**: $m \in \langle g \rangle$		**Bob** **secret key**: k **public key**: $K = kG$
	$\xleftarrow{K}$	
pick r at random		
$R = rG$	$\xrightarrow{R}$	
$(k_E \Vert k_M) = \mathrm{KDF}(rK)$		$(k_E \Vert k_M) = \mathrm{KDF}(kR)$
$c = \mathrm{Enc}_{k_E}(m)$	$\xrightarrow{c}$	$m = \mathrm{Dec}_{k_E}(c)$
$d = \mathrm{MAC}_{k_M}(c)$	$\xrightarrow{d}$	$d \stackrel{?}{=} \mathrm{MAC}_{k_M}(c)$
		output: m

ECIES (EC Integrated Encryption Scheme)



select field, elliptic curve
G point of order n
n prime

extra is context-based information (public)

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: $\mathrm{sk} = x$.

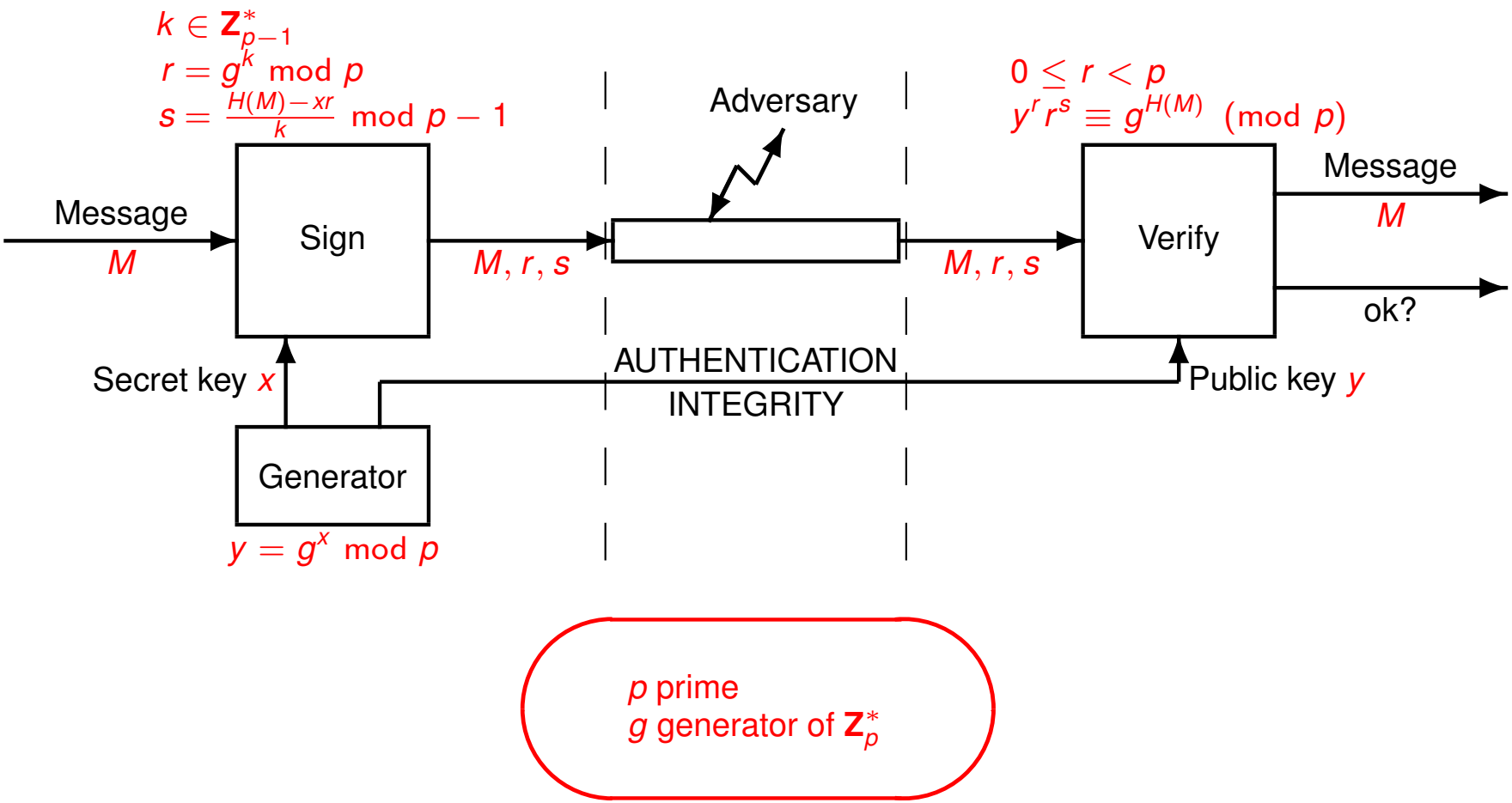
Public key: $\mathrm{pk} = y$.

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



Drawbacks of ElGamal Signatures

- signatures are pretty long
- security issues related to subgroups
- lack of security proof for *arbitrary* public parameter

Major Drawbacks of ElGamal-Like Signatures

leaking or reusing the ephemeral k reveals the long-term secret x

- leaking k: as p, M, r, and s are known, get x from

$$s = \frac{H(M) - xr}{k} \bmod (p-1)$$

- reusing k: as p, (M_1, r_1, s_1), (M_2, r_2, s_2) are known, get x from

$$\frac{s_1}{s_2} = \frac{H(M_1) - xr_1}{H(M_2) - xr_2}$$

this happened in the software install protection of Sony PS2
this does not happen with RSA

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 $\mathbf{Z}_p^*$)

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 $y = g^x \bmod p$.

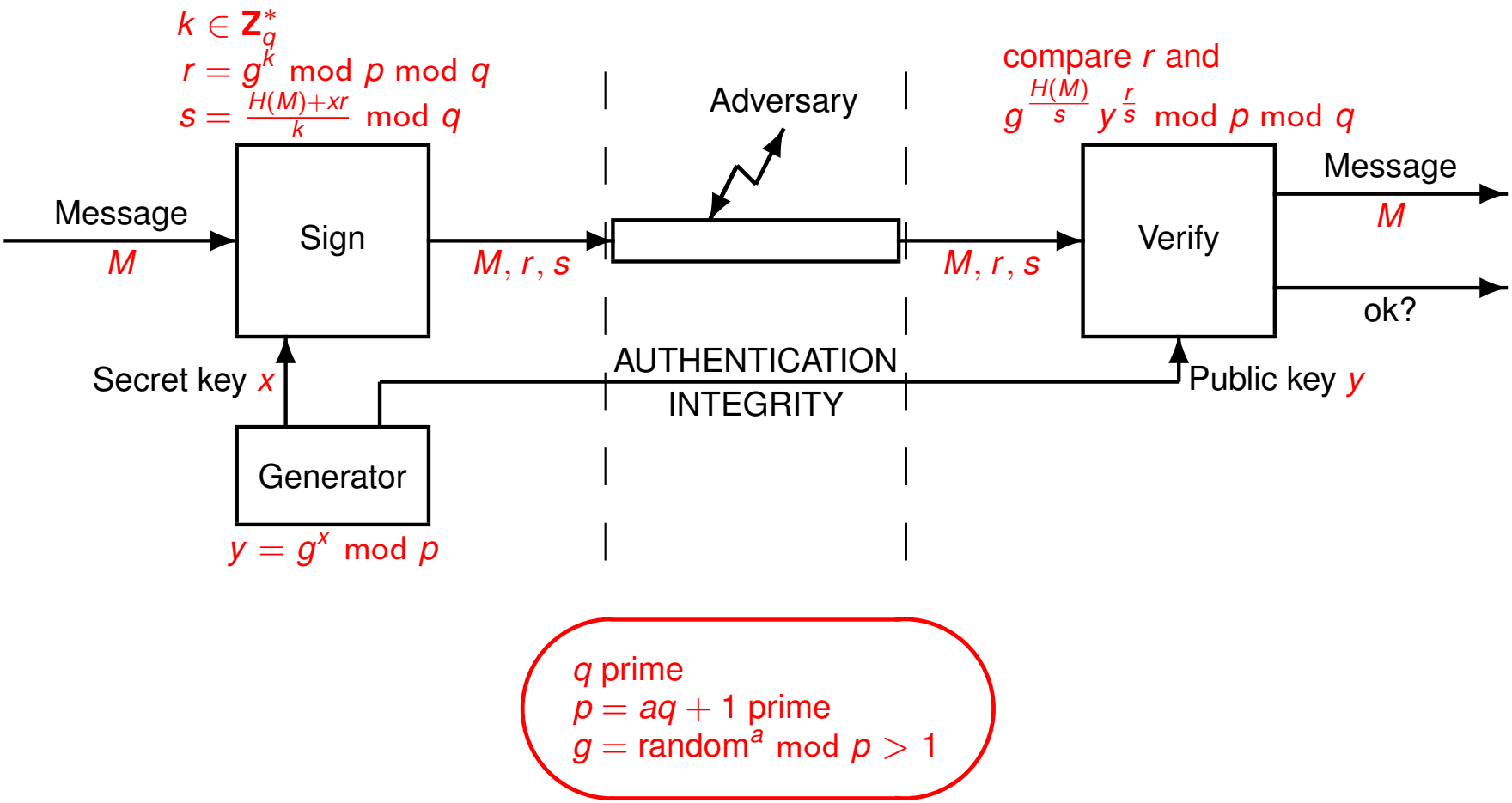
Secret key: $\mathrm{sk} = x$.

Public key: $\mathrm{pk} = y$.

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



Benefits

- signatures are shorter
- no proper subgroup (only $\{1\}$ and the group itself)
- some form of provable security

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 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 $(\mathsf{pk}, \mathsf{sk}) = (Q, d)$.

ECDSA

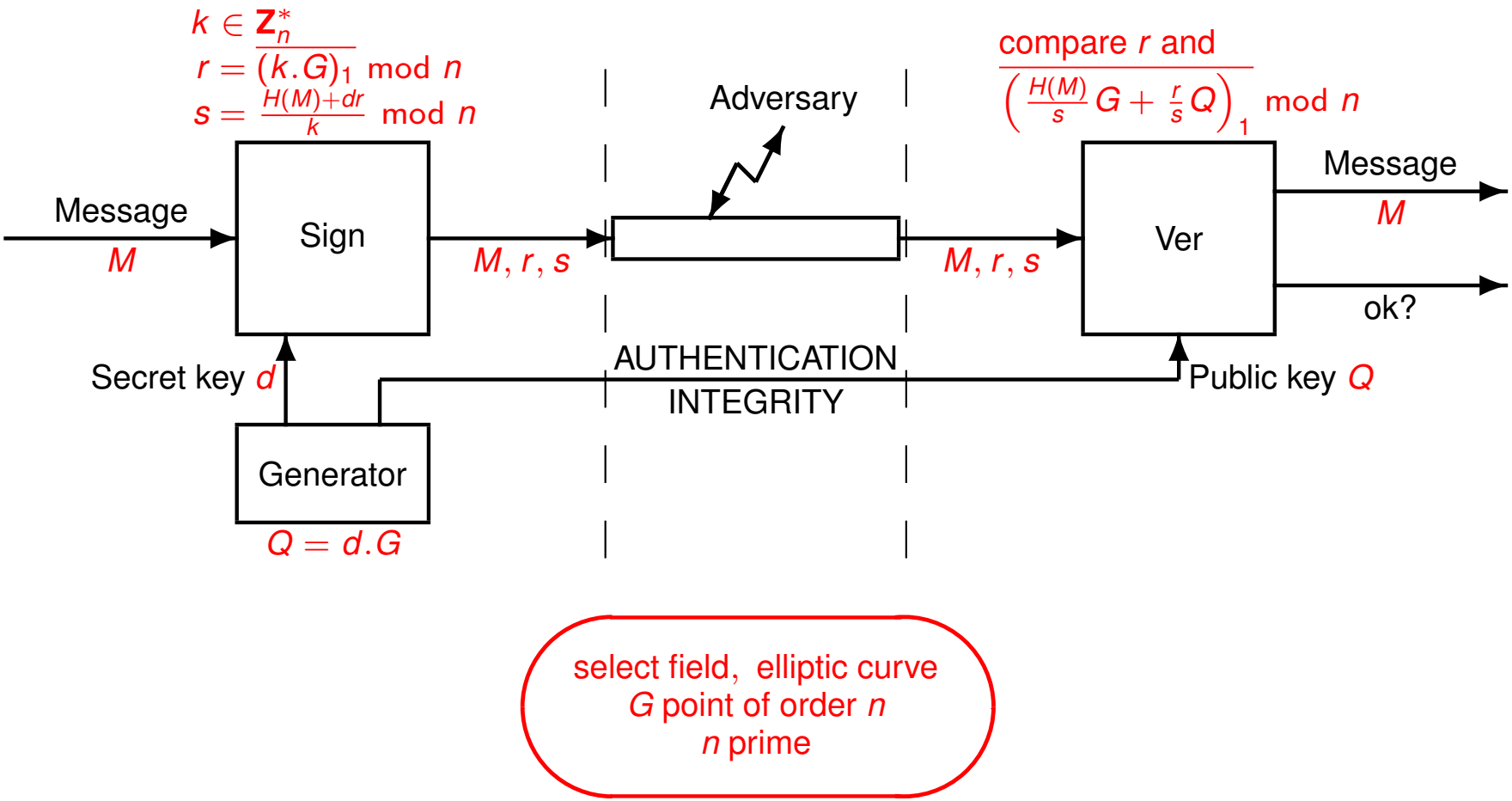
Signature generation: pick k in $[1, n-1]$ at random and compute

$$\begin{aligned}(x_1, y_1) &= kG \\ r &= \overline{x_1} \bmod n \\ s &= \frac{H(M) + dr}{k} \bmod n\end{aligned}$$

($\overline{x_1}$ 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 $Q \neq \mathcal{O}$, $Q \in C$, and $nQ = \mathcal{O}$. Check that r and s are in $[1, n-1]$ and that $r = \overline{x_1} \bmod n$ for $(x_1, y_1) = u_1 G + u_2 Q$, $u_1 = \frac{H(M)}{s} \bmod n$, and $u_2 = \frac{r}{s} \bmod n$.

ECDSA Signature



Example of Parameters and Key

secp192r1:

$$
\begin{aligned}
q &= 6277101735386680763835789423207666416083908700390324961279 \\
a &= \texttt{fffffffe ffffffff ffffffff fffffffe ffffffff fffffffc} \\
b &= \texttt{64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1} \\
n &= 6277101735386680763835789423176059013767194773182842284081 \\
G &= \texttt{03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012} \\
\text{seed} &= \texttt{3045ae6f c8422f64 ed579528 d38120ea e12196d5}
\end{aligned}
$$

(the leading "03" is for point compression)

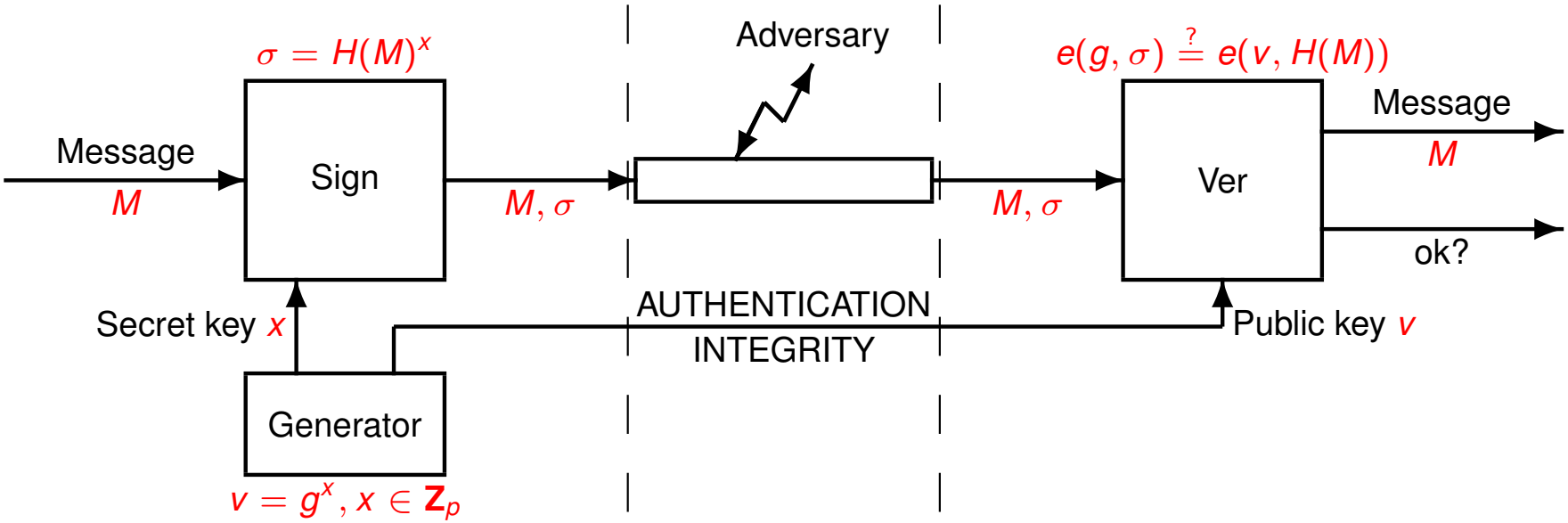
$$
\begin{aligned}
d &= 651056770906015076056810763456358567190100156695615665659 \\
Q &= \texttt{02 62b12d60 690cdcf3 30babab6 e69763b4 71f994dd 702d16a5}
\end{aligned}
$$

(the leading "02" is for point compression)

Benefits of ECDSA (Compared to DSA)

- public key is shorter
- computation is lighter

Boneh-Lynn-Shacham (BLS) Signature

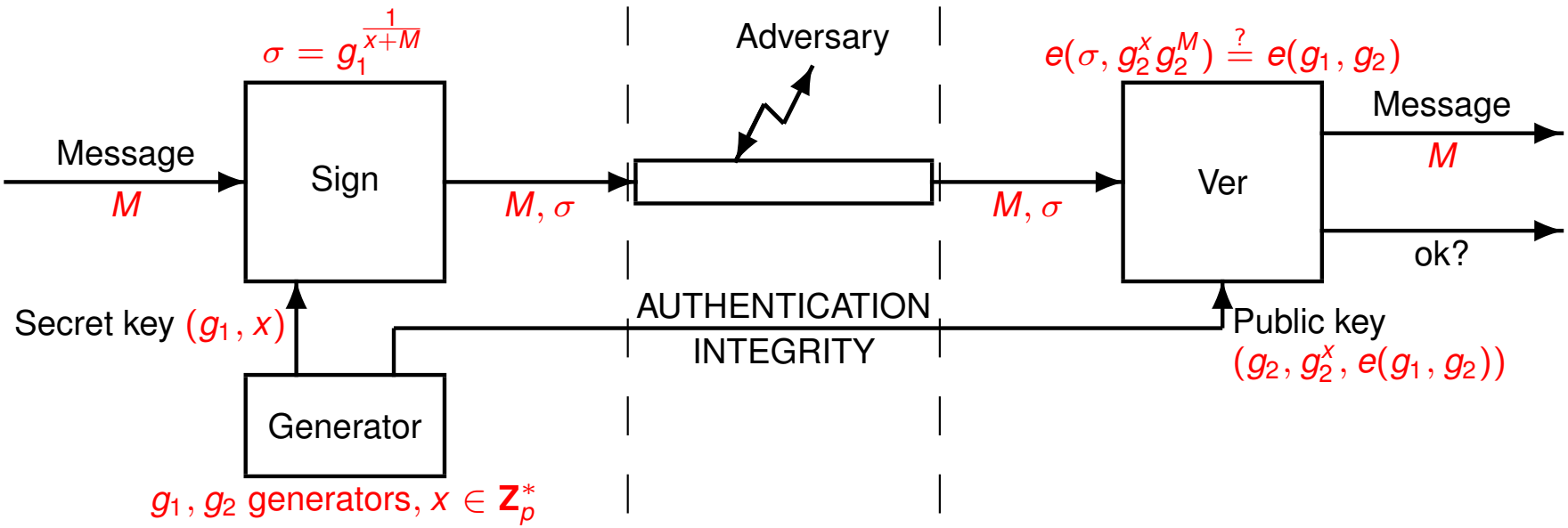


▶ slide 388

$g \in G$ generator
G, G_T groups of prime order p
$e : G \times G \to G_T$ pairing
H hashes into G

shortest signature!

(Simple) Boneh-Boyen Signature



▶ slide 388

p prime
G_1, G_2, G_T groups of order p
$e : G_1 \times G_2 \to G_T$ pairing

H-free signature!

Case Study: NFC Creditcard Payment

case study

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

Popular Algorithms

- symmetric encryption: AES ($\rightarrow$ key length)
- hash function: SHA2, SHA3 ($\rightarrow$ hash length)
- MAC/PRF: HMAC-SHA2 ($\rightarrow$ key length, tag length)
- authentication encryption: AES-CCM, AES-GCM

  be careful with the nonce
- key agreement: DH, ECDH ($\rightarrow$ public parameters)
- cryptosystem: RSA ($\rightarrow$ modulus length), EC crypto ($\rightarrow$ pp)
- signature: RSA ($\rightarrow$ modulus length), ECDSA ($\rightarrow$ pp)

  be careful with the randomness

  be careful with the key length

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
- $\rightarrow$ NFS

Breaking DH Cryptography by Discrete Logarithm

Best attack (ideally): discrete logarithm computation

Fact

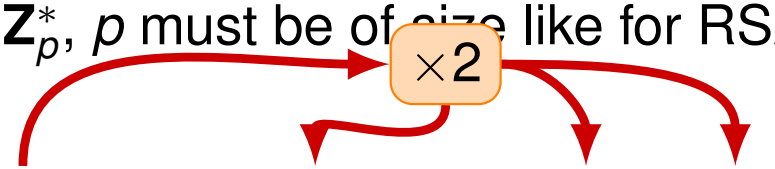
If we can compute the discrete logarithm x of g^x then from g, g^x, g^y we can compute g^{xy}.

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** $\rightarrow$ GNFS
  - **elliptic curve**
    - random over **prime field** $\rightarrow$ generic algorithms

Meta-comparison of Cryptographic Strengths

- symmetric encryption/MAC: bit-security
- RSA: check tables
- hash with collision resistance: digest of **twice** bit-security
- hash without collision resistance: digest of bit-security
- discrete logarithm/DH in a group: **twice** bit-security
  caveat: if subgroup of $\mathbf{Z}_p^*$, p must be of size like for RSA



method	sym.	RSA	DL		EC	hash
Lenstra-Verheul	82	1613	145	1613	154	163
Lenstra updated	78	1245	156	1245	156	156
ECRYPT II	80	1248	160	1248	160	160
NIST	112	2048	224	2048	224	224
FNISA	100	2048	200	2048	200	200
BSI	–	1976	224	2048	224	224

(`http://www.keylength.com` by Quisquater)

(Typical) Example of Coherent Choice

- AES 128 bits in AES-CCM
- SHA256
- EC crypto on P256

Impact of Quantum Computers

cryptosystem	key size	security	algorithm	#qbits	time
AES-GCM	128	128	Grover	2953	2.6×10^{12}Y
AES-GCM	192	192	Grover	4449	2.0×10^{22}Y
AES-GCM	256	256	Grover	6681	2.3×10^{32}Y
RSA	1024	80	Shor	2050	3.6H
RSA	2048	112	Shor	4098	28.6H
RSA	4096	128	Shor	8194	229H
ECC	256	128	Shor	2330	10.5H
ECC	384	192	Shor	3484	37.7H
ECC	521	256	Shor	4719	55H

https://nas.nationalacademis.org/read/25196/chapter/6#98

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

Definition

A **public-key cryptosystem** is a tuple $(\mathsf{Gen}, \mathcal{M}, \mathsf{Enc}, \mathsf{Dec})$ with a plaintext domain $\mathcal{M} \subseteq \{0,1\}^*$ and three efficient algorithms Gen, Enc, and Dec. The algorithm Dec is deterministic and outputs either something in $\mathcal{M}$ or an error $\perp$. It is such that

$$\forall \mathsf{pt} \in \mathcal{M} \quad \Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})) = \mathsf{pt}] = 1$$

where $(\mathsf{pk}, \mathsf{sk})$ is generated from running Gen. The probability is over the randomness used in Gen and 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 predictible plaintext, if encryption is deterministic, it is easy to recognize from 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

IND-CPA Security

Definition

A PKC $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$ is (t, ε)-**secure under chosen plaintext attacks** (IND-CPA-secure) if for any interactive process $(\mathcal{A}_1, \mathcal{A}_2)$ limited to a time complexity t, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game Γ_b

1: $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
2: $\mathcal{A}_1(\text{pk}) \to (\text{pt}_0, \text{pt}_1, \text{st})$
3: **if** $|\text{pt}_0| \neq |\text{pt}_1|$ **then return** 0
4: $\text{Enc}(\text{pt}_b) \xrightarrow{\$} \text{ct}^*$
5: $\mathcal{A}_2(\text{st}, \text{ct}^*) \to z$
6: **return** z

Problem with Deterministic Cryptosystems

deterministic encryption is *not* IND-CPA secure

$\mathcal{A}_1(\mathsf{pk})$

1: pick pt_0 and pt_1 arbitrarily (same length)
2: $\mathsf{st} \leftarrow \mathsf{pt}_1$
3: **return** $(\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$

$\mathcal{A}_2(\mathsf{st}, \mathsf{ct}^*)$

4: $\mathsf{st} \rightarrow \mathsf{pt}_1$
5: $\mathsf{ct} \rightarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_1)$
6: **return** $1_{\mathsf{ct}=\mathsf{ct}^*}$

$\rightarrow$ plain RSA is *not* IND-CPA secure

$\rightarrow$ example: plain ElGamal cryptosystem is IND-CPA secure

IND-CCA Security

Definition

A PKC $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$ is (t, ε)-**secure under chosen ciphertext attacks** (IND-CCA-secure) if for any interactive process $(\mathcal{A}_1, \mathcal{A}_2)$ limited to a time complexity t, the advantage Adv is bounded by ε, where

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game Γ_b

1: $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
2: $\mathcal{A}_1^{\text{ODec}_1}(\text{pk}) \to (\text{pt}_0, \text{pt}_1, \text{st})$
3: **if** $|\text{pt}_0| \neq |\text{pt}_1|$ **then return** 0
4: $\text{Enc}(\text{pt}_b) \xrightarrow{\$} \text{ct}^*$
5: $\mathcal{A}_2^{\text{ODec}_2}(\text{st}, \text{ct}^*) \to z$
6: **return** z

Oracle $\text{ODec}_1(\text{ct})$

7: **return** $\text{Dec}(\text{sk}, \text{ct})$

Oracle $\text{ODec}_2(\text{ct})$

8: **if** $\text{ct} = \text{ct}^*$ **then return** $\perp$
9: **return** $\text{Dec}(\text{sk}, \text{ct})$

Basic Constructions

- plain ElGamal is *not* IND-CCA secure

  $\mathcal{A}_1(\mathsf{pk})$
  1: pick pt_0 and pt_1 arbitrarily (same length)
  2: $\mathsf{st} \leftarrow \mathsf{pt}_1$
  3: **return** $(\mathsf{pt}_0, \mathsf{pt}_1, \mathsf{st})$

  $\mathcal{A}_2(\mathsf{st}, \mathsf{ct}^*)$
  4: $\mathsf{st} \rightarrow \mathsf{pt}_1$
  5: $\mathsf{ct}^* \rightarrow (u, v)$
  6: $\mathsf{ct} \leftarrow (u, 2v)$
  7: $\mathsf{ODec}_2(\mathsf{ct}) \rightarrow \mathsf{pt}$
  8: **return** $1_{\mathsf{pt}=2\mathsf{pt}_1}$
- RSA-OAEP and ECIES are IND-CCA secure (under some conditions)

Fujisaki-Okamoto Transform

- γ**-spread** and **OWCPA-secure** PKC $(\mathsf{Gen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$
- **one-time secure** cipher (e.g. one-time pad)
- **random oracles** G and H

$\rightarrow$ construct a PKC which is INDCCA-secure
(many variants possible)

$$\mathsf{Gen} \rightarrow (\mathsf{pk} = \mathsf{pk}_0, \mathsf{sk} = (\mathsf{sk}_0, \mathsf{pk}_0))$$

$$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{pt}; \mathsf{coins}) \rightarrow \left(\mathsf{Enc}_{0,\mathsf{pk}_0}(\mathsf{coins}; \underbrace{H(\mathsf{coins}, \mathsf{ct}_2)}_{\text{new coins}}), \overbrace{\mathsf{pt} \oplus G(\mathsf{coins})}^{\mathsf{ct}_2} \right)$$

$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_1, \mathsf{ct}_2)$:

1: $\mathsf{Dec}_{0,\mathsf{sk}_0}(\mathsf{ct}_1) \rightarrow \mathsf{coins}$
2: **if** $\mathsf{ct}_1 \neq \mathsf{Enc}_{0,\mathsf{pk}_0}(\mathsf{coins}; H(\mathsf{coins}, \mathsf{ct}_2))$ **then return** $\perp$
3: **return** $\mathsf{ct}_2 \oplus G(\mathsf{coins})$

Fujisaki-Okamoto KEM

- **random oracles** G and H

$\rightarrow$ construct a KEM which is INDCCA-secure

$$\text{Gen} \rightarrow (\text{pk} = \text{pk}_0, \text{sk} = (\text{sk}_0, \text{pk}_0))$$

$\text{KemEnc}_{\text{pk}}(;\text{coins})$:

1: $(K, r) \leftarrow H(\text{coins})$
2: $\text{ct} \leftarrow \text{Enc}_{0,\text{pk}_0}(\text{coins}; r)$
3: **return** (K, ct)

$\text{KemDec}_{\text{sk}}(\text{ct})$:

4: $\text{Dec}_{0,\text{sk}_0}(\text{ct}) \rightarrow \text{coins}$
5: $\text{KemEnc}_{\text{pk}_0}(;\text{coins}) \rightarrow (K, \text{ct}')$
6: **if** $\text{ct} \neq \text{ct}'$ **then return** $\perp$
7: **return** K

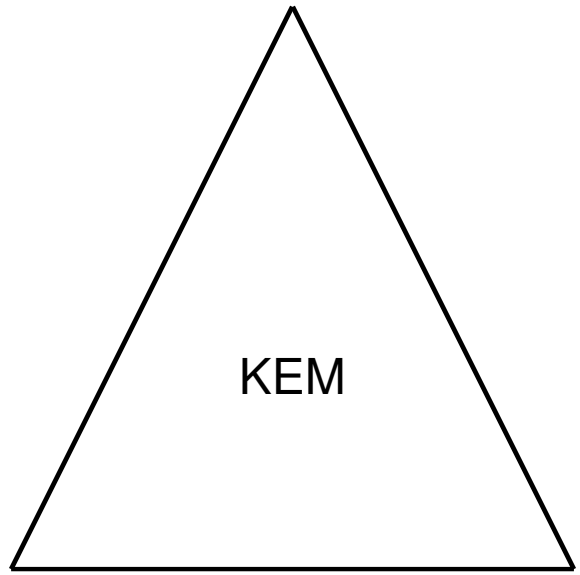
Key and Data Encapsulation Mechanisms

Hybrid Encryption

- DEM: same as symmetric encryption
- KEM: public-key algorithm producing an encrypted (encapsulated) key
  $\approx$ generate a random symmetric key and encrypt it using public-key encryption
- hybrid encryption: symmetric + asymmetric

KEM Primitive

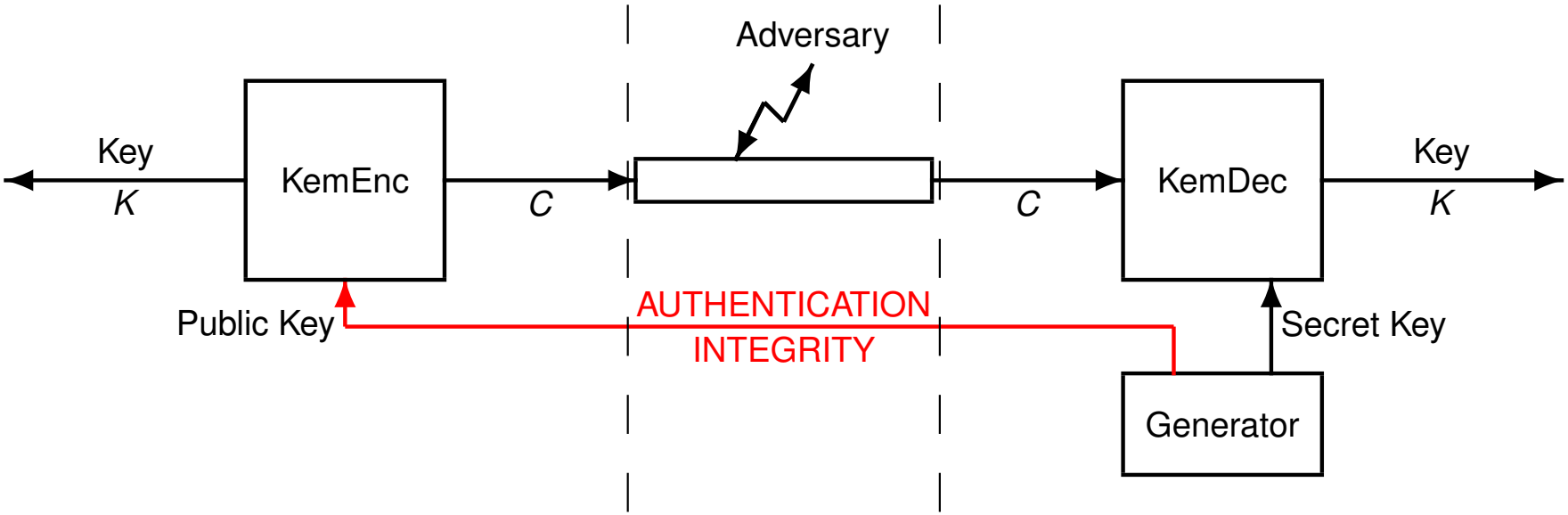
Generator, KemEnc, KemDec
components



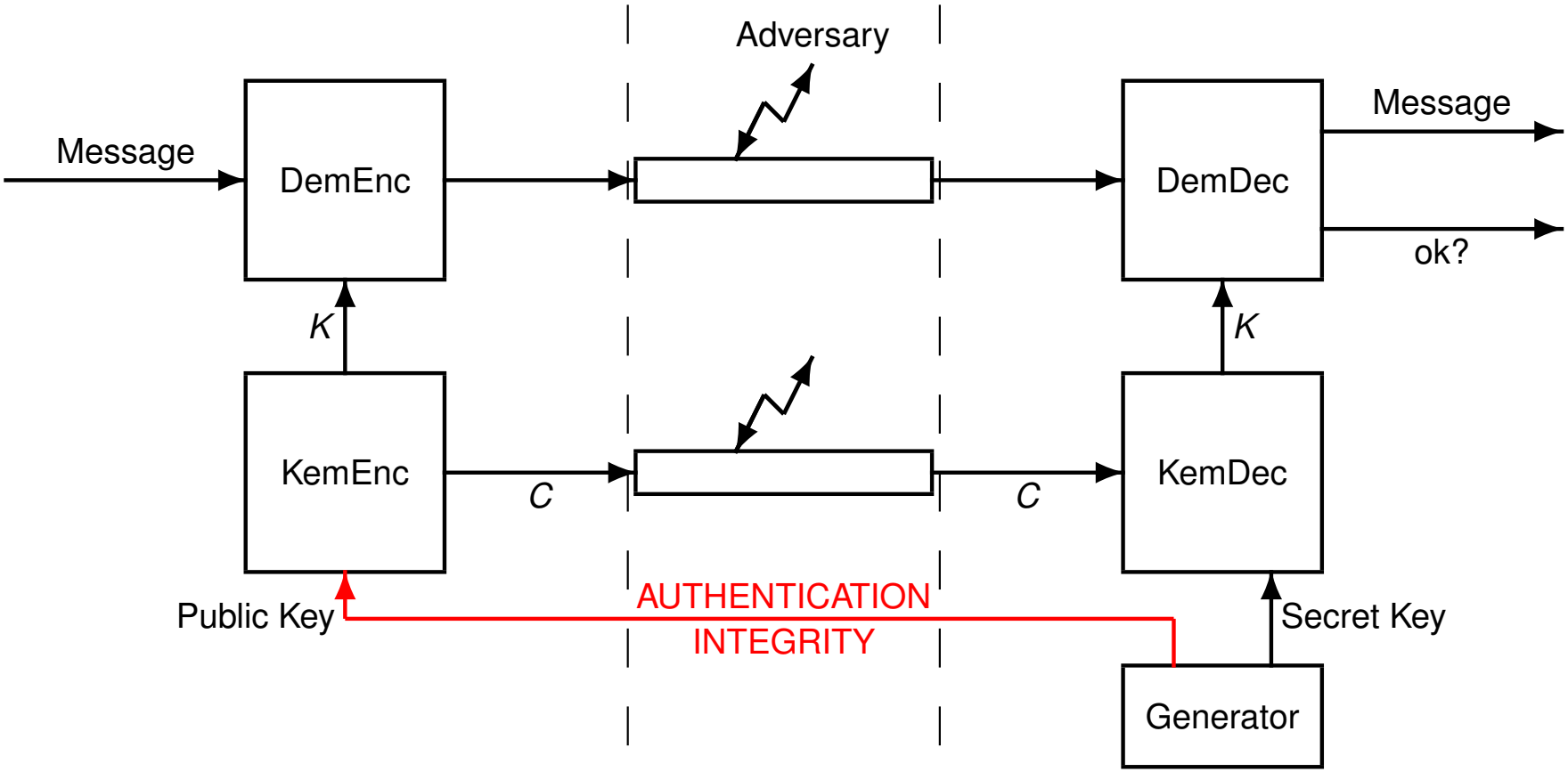
functionality
if $\mathsf{KemEnc}_{pk} \rightarrow (K, C)$
then $\mathsf{KemDec}_{sk}(C) = K$

security
key is confidential

KEM


Adversary
Key
K
KemEnc
C
C
KemDec
Key
K
Public Key
AUTHENTICATION
INTEGRITY
Secret Key
Generator

KEM + DEM Hybrid Encryption



DEM Example

Shoup: Using Hash Functions as a Hedge against Chosen Ciphertext Attack (2000)

$$\mathrm{DemEnc}_K(\mathrm{pt}) = (\mathrm{pt} \oplus G(K), \mathrm{MAC}_{G'(K)}(\mathrm{pt}))$$

- encrypt and MAC
- one-time encryption (Vernam-based) +
  one-time authenticator (WC-based)
- example: Poly1305
- could use an AE too (one-time: constant nonce ok)
  - CHACHA20-POLY1305 (encrypt-then-MAC)
  - AES-GCM
  - AES-CCM (MAC-then-encrypt)

HPKE (Hybrib Public Key Encryption)

RFC9180

common interface for many algorithms (KDF, AEAD, KEM)

- DHKEM (DH-Based KEM)

Gen:
1: pick sk
2: pk $\leftarrow$ sk $\cdot\, G$
3: **return** (pk, sk)

DH(a, B):
4: assert $B \in \text{group} - \{0\}$
5: **return** rep($a \cdot B$)

KemEnc(pk):
6: pick x
7: ct $\leftarrow x \cdot G$
8: $Z \leftarrow$ DH(x, pk)
9: $K \leftarrow H(Z, \text{ct}, \text{pk})$
10: **return** (K, ct)

KemDec(sk, ct):
11: pk $\leftarrow$ sk $\cdot\, G$
12: $Z \leftarrow$ DH(sk, ct)
13: $K \leftarrow H(Z, \text{ct}, \text{pk})$
14: **return** K

- an authenticated version of DHKEM (next slide)
- KDF: HKDF-SHA256, HKDF-SHA384 HKDF-SHA512
- KEM: DHKEM on P256, P384, P521, X25519, X448 with KDF
- AEAD: AES-128-GCM, AES-256-GCM, ChaCha20Poly1305
- HPKE is a KEM + AEAD

Authenticated DHKEM

include a static-DH key T for implicit authentication

Gen:

1: pick sk
2: $\mathsf{pk} \leftarrow \mathsf{sk} \cdot G$
3: **return** $(\mathsf{pk}, \mathsf{sk})$

KemEnc($\mathsf{pk}_R, \mathsf{sk}_S$):

4: pick x
5: $\mathsf{ct} \leftarrow x \cdot G$
6: $\mathsf{pk}_S \leftarrow \mathsf{sk}_S \cdot G$
7: $Z \leftarrow \mathrm{DH}(x, \mathsf{pk}_R)$
8: $T \leftarrow \mathrm{DH}(\mathsf{sk}_S, \mathsf{pk}_R)$
9: $K \leftarrow H(Z, T, \mathsf{ct}, \mathsf{pk}_R, \mathsf{pk}_S)$
10: **return** (K, ct)

DH(a, B):

11: **if** $B \notin$ group **then return** $\perp$
12: **if** $B = 0$ **then return** $\perp$
13: **return** $\mathrm{rep}(a \cdot B)$

KemDec($\mathsf{sk}_R, \mathsf{pk}_S, \mathsf{ct}$):

14: $\mathsf{pk}_R \leftarrow \mathsf{sk}_R \cdot G$
15: $Z \leftarrow \mathrm{DH}(\mathsf{sk}_R, \mathsf{ct})$
16: $T \leftarrow \mathrm{DH}(\mathsf{sk}_R, \mathsf{pk}_S)$
17: $K \leftarrow H(Z, T, \mathsf{ct}, \mathsf{pk}_R, \mathsf{pk}_S)$
18: **return** K

Signature Scheme

Definition

A **digital signature scheme** is a tuple $(\mathsf{Gen}, \mathcal{D}, \mathsf{Sig}, \mathsf{Ver})$ with a message domain $\mathcal{D} \subseteq \{0,1\}^*$ and three efficient algorithms Gen, Sig, and Ver. The algorithm Ver is deterministic and outputs 0 (reject) or 1 (accept). It is such that

$$\forall X \in \mathcal{D} \quad \Pr[\mathsf{Ver}(\mathsf{pk}, X, \mathsf{Sig}(\mathsf{sk}, X)) = 1] = 1$$

where $(\mathsf{pk}, \mathsf{sk})$ is generated from running Gen. The probability is over the randomness used in Gen and Sig.

EF-CMA Security

Definition

A digital signature scheme $(\mathsf{Gen}, \mathcal{D}, \mathsf{Sig}, \mathsf{Ver})$ is (q, t, ε)-**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, the advantage Adv is bounded by ε.

$$\mathsf{Adv} = \Pr[\text{game returns } 1]$$

Game

1: $\mathsf{Gen} \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$
2: $\mathsf{Queries} \leftarrow \emptyset$
3: $\mathcal{A}^{\mathsf{OSig}}(\mathsf{pk}) \rightarrow (X, \sigma)$
4: **if** $X \in \mathsf{Queries}$ **then return** 0
5: **return** $1_{\mathsf{Ver}(\mathsf{pk}, X, \sigma)}$

Oracle $\mathsf{OSig}(X)$:

6: $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, X)$
7: $\mathsf{Queries} \leftarrow \mathsf{Queries} \cup \{X\}$
8: **return** σ

Examples

- ElGamal signature is EF-CMA secure (under some conditions)
- RSA-PSS is EF-CMA secure (under some conditions)

Other Public-Key Cryptosystems

- **RSA**
- Rabin
- Paillier

based on factoring

- **ElGamal**
- ECC
- HECC

based on discrete logarithm

- NTRU
- lattice-based
- McEliece
- TCHo
- ...

"post-quantum"

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

On Real Quantum Computers

(unreliable numbers)

- 1998: 2 qbits
- 2000: 4, 5, 7 qbits
- 2006: 12 qbits
- 2011: 14 qbits
- 2017: 17, 49 qbits
- 2019: 54 qbits (quantum supremacy reached)
- 2020: 65 qbits
- ??
- (IBM wish) 2033: 2000 qbits

The Impact on Crypto

The Sky is Falling

- **symmetric crypto**: block ciphers, hash functions, MAC
  $\rightarrow$ may need to double sizes (we have time)
- **public-key crypto**: cryptosystems, signatures
  $\rightarrow$ discrete log and factoring become easy
  - encryption: “harvest now, decrypt later” attack
  - signature: forge a binding signature with a date in past

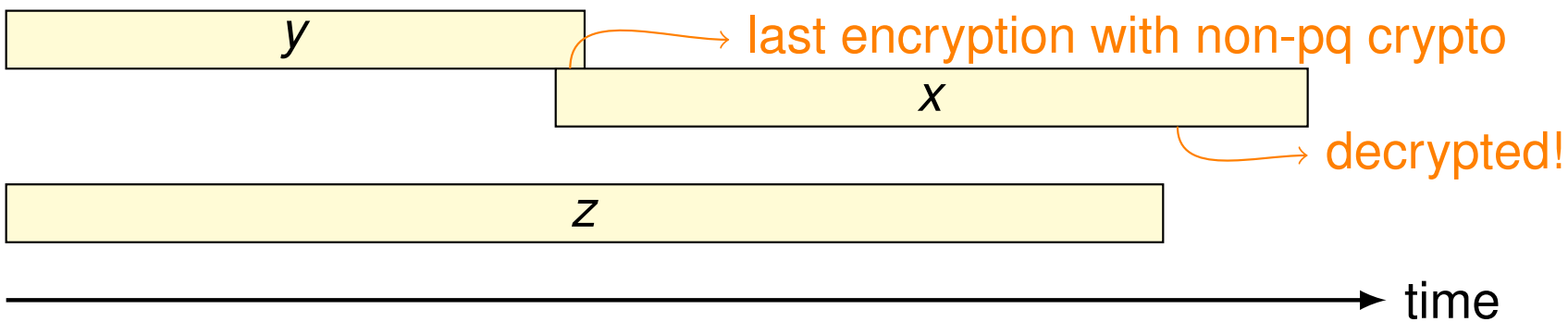
When to Worry?

Cybersecurity in an Era with Quantum Computers: Will we be Ready? [Mosca 2013]

- x: how long information must remain secure
- y: how long until pq-crypto is available
- z: how long until quantum computers really exist

Theorem

If $x + y > z$ then worry.

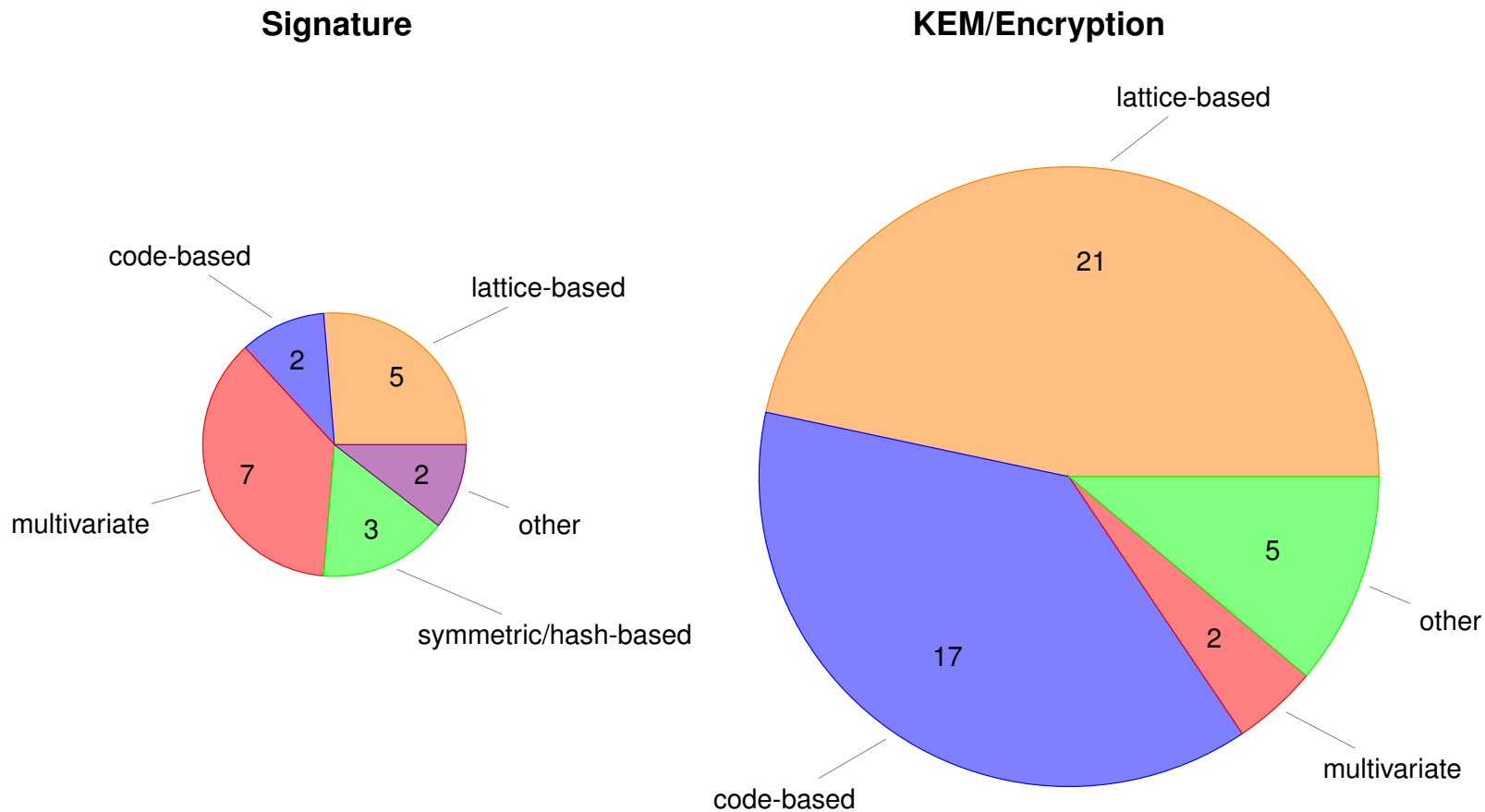


for signatures: replace x by the time a signature should remain binding (could be mitigated with a reliable timestamp)

NIST PQC Agenda

- $\sim$2012: NIST begins PQC project
- 2015: NIST workshop on cybersecurity in a pq world
- 2016: NIST plan: requirements and evaluation criteria
- 2017: submission deadline to NIST PQC Round #1
- 2018: 1st NIST PQC workshop
- 2019: NIST selects algorithms to go to Round #2
- 2019: 2nd NIST PQC workshop
- 2020: NIST selects algorithms to go to Round #3
- 2021: 3rd NIST PQC workshop
- 2022: NIST selects algorithms + Round #4
- 2024: standards

NIST PQC Round #1 Submission



source: `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/PQCrypto-April2018_Moody.pdf`

NIST PQC 2022 Selected Algorithms

	Signature		KEM/Encryption	
Selected	CRYSTALS-DILITHIUM FALCON SPHINCS+	lattice lattice hash	CRYSTALS-KYBER	lattice

Next:

- KYBER becomes ML-KEM in FIPS 203 (Module-Lattice)
- DILITHIUM becomes ML-DSA in FIPS 204 (Module-Lattice)
- SPHINCS+ becomes SLH-DSA in FIPS 205 (Stateless Hash)
- FALCON becomes FN-DSA (later)

Next Steps

- 4th round for KEM/Encryption: HQC (code-based)
- new candidates for signature (other than lattice-based): 14 left
- hybrids (postquantum+classical): SP 800-227
- DH, RSA, ECDSA, EdDSA, ECDH:
  - deprecated in 2030
  - disallowed in 2035

Hybrid KEM

$$\mathsf{KEM} = \mathsf{KEM}_1 \& \mathsf{KEM}_2$$

Gen:
1: $\mathsf{Gen}_1 \to (\mathsf{pk}_1, \mathsf{sk}_1)$
2: $\mathsf{Gen}_2 \to (\mathsf{pk}_2, \mathsf{sk}_2)$
3: $\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$
4: $\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
5: **return** $(\mathsf{pk}, \mathsf{sk})$

KemEnc(pk):
6: $\mathsf{pk} \to (\mathsf{pk}_1, \mathsf{pk}_2)$
7: $\mathsf{KemEnc}_1(\mathsf{pk}_1) \to (K_1, \mathsf{ct}_1)$
8: $\mathsf{KemEnc}_2(\mathsf{pk}_2) \to (K_2, \mathsf{ct}_2)$
9: $K \leftarrow f(\ldots, K_1, K_2)$
10: $\mathsf{ct} \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2)$
11: **return** (K, ct)

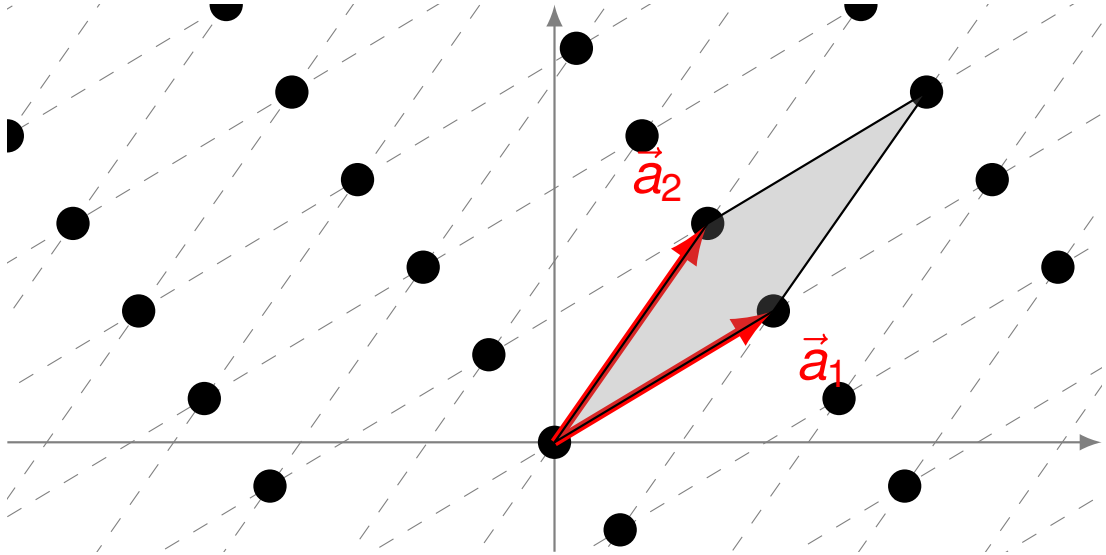
KemDec(sk, ct):
12: $\mathsf{sk} \to (\mathsf{sk}_1, \mathsf{sk}_2)$
13: $\mathsf{ct} \to (\mathsf{ct}_1, \mathsf{ct}_2)$
14: $\mathsf{KemDec}_1(\mathsf{sk}_1, \mathsf{ct}_1) \to K_1$
15: $\mathsf{KemDec}_2(\mathsf{sk}_2, \mathsf{ct}_2) \to K_2$
16: $K \leftarrow f(\ldots, K_1, K_2)$
17: **return** K

- IETF draft for TLS: $K = K_1 \| K_2$
  example: `SecP256r1Kyber768Draft00` (ML-KEM-768&P256)
- X-Wing with DH: $K = \text{SHA3-256}(K_1, K_2, \mathsf{ct}_{\mathsf{DH}}, \mathsf{pk}_{\mathsf{DH}})$
- SP 800-227: $K = H(K_1 \| K_2 \| \mathsf{ct} \| \mathsf{pk})$ (or other combiners)

7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

Lattices



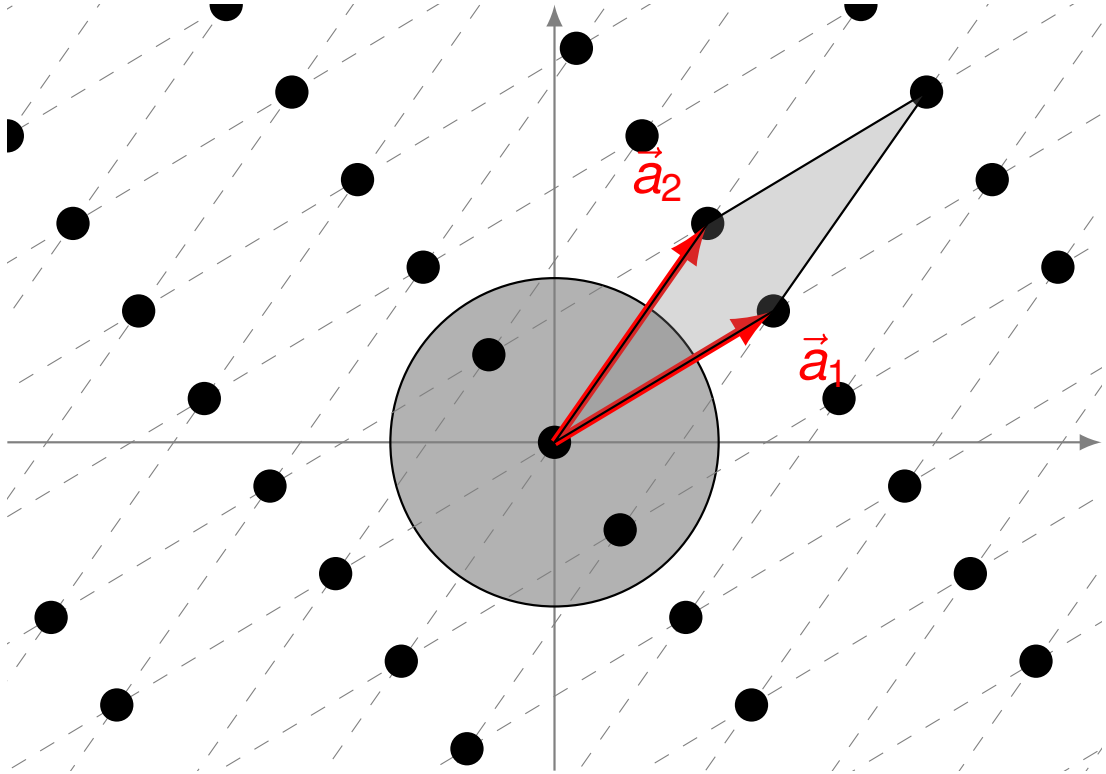
- **lattice**: discrete subgroup of $\mathbf{R}^m$
- specified by a basis:

$$\mathcal{L}(\vec{a}_1, \ldots, \vec{a}_n) = \left\{ \sum_{i=1}^{n} s_i \vec{a}_i ; s_1, \ldots, s_n \in \mathbf{Z} \right\} = \left\{ A\vec{s}; \vec{s} \in \mathbf{Z}^n \right\}$$

  rank n (if the $\vec{a}_i$ are independent), dimension m
- **fundamental domain** ("$\mathcal{L}$-tiles" of $\mathbf{R}^m$)

$$F(\vec{a}_1, \ldots, \vec{a}_n) = \left\{ \sum_{i=1}^{n} s_i \vec{a}_i \; ; \; \forall i \quad 0 \leq s_i < 1 \right\}$$

Lattices



- **determinent**/volume ($m = n$)

$$\det(\mathcal{L}) = \mathrm{vol}(F) = |\det(A)|$$

- "regular" number of lattice vectors of norm up to r is $\sim \frac{\mathrm{vol}(B_r)}{\det(\mathcal{L})}$

$$\lim_{r \to +\infty} \frac{\mathrm{vol}(B_r)}{\#\{\vec{x} \in \mathcal{L}; \|\vec{x}\| \leq r\}} = \det(\mathcal{L})$$

Lattice-Based Problems

- **CVP** (closest vector problem)
  given $\vec{b}$, find $\vec{x} \in \mathcal{L}(\vec{a}_1, \ldots, \vec{a}_n)$ making $\|\vec{b} - \vec{x}\|$ small
- **SVP** (smallest vector problem)
  find short (nonzero) vectors $\vec{x} \in \mathcal{L}(\vec{a}_1, \ldots, \vec{a}_n)$
- γ-SVP: (SVP approximation with gap γ)
  find a nonzero lattice vector $\vec{x}$ such that $\|\vec{x}\| \leq \gamma \min_{\vec{y} \in \mathcal{L} - \{0\}} \|\vec{y}\|$
  $\rightarrow$ easy for $\gamma > 2^{n \frac{\log \log n}{\log n}}$ (LLL algorithm)
  $\rightarrow$ used for $\gamma \sim n$
  $\rightarrow$ NP-hard for $\gamma < n^{\frac{c}{\log \log n}}$

Learning with Error

- **LWE** (learning with error):
  given access to a $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + \text{small} \bmod q)$ generator, it is hard to recover $\vec{s}$
- typical LWE variant:
  given a random A and $\vec{y} = A\vec{s} + \vec{e} \bmod q$ with $\vec{e}$ small, it is hard to recover $\vec{s}$
- typical distribution for $\vec{e}$: Gaussian of scale $\sigma \sim \sqrt{q}$

  density probability of $\vec{e}$: $\frac{1}{\sigma^n} e^{-\pi \frac{\|\vec{e}\|^2}{\sigma^2}}$

$$E(\|\vec{e}\|^2) = \sum_{i=1}^{n} \frac{1}{\sigma} \int_{e_i=-\infty}^{+\infty} e_i^2 . e^{-\pi \frac{e_i^2}{\sigma^2}} \, de_i = \frac{n\sigma^2}{2\pi} = \mathcal{O}(nq)$$

Primal Attack on LWE

reduction to SVP (when $\vec{s}$ is small):

$$\vec{y} = A\vec{s} + \vec{e} \bmod q \iff \begin{pmatrix} \vec{e} \\ \vec{s} \\ 1 \end{pmatrix} \in \operatorname{span}(B)$$

with

$$B = \begin{pmatrix} qI & -A & \vec{y} \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad I = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$$

since

$$\begin{pmatrix} qI & -A & \vec{y} \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} * \\ \vec{s} \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{e} \\ \vec{s} \\ 1 \end{pmatrix}$$

the lattice has "volume" $\det(B) = q^n$ and $\sqrt{\|\vec{e}\|^2 + \|\vec{s}\|^2 + 1} = \mathcal{O}(\sqrt{nq})$ is unusually short (for $n \ll q$) so it must be *the* shortest vector

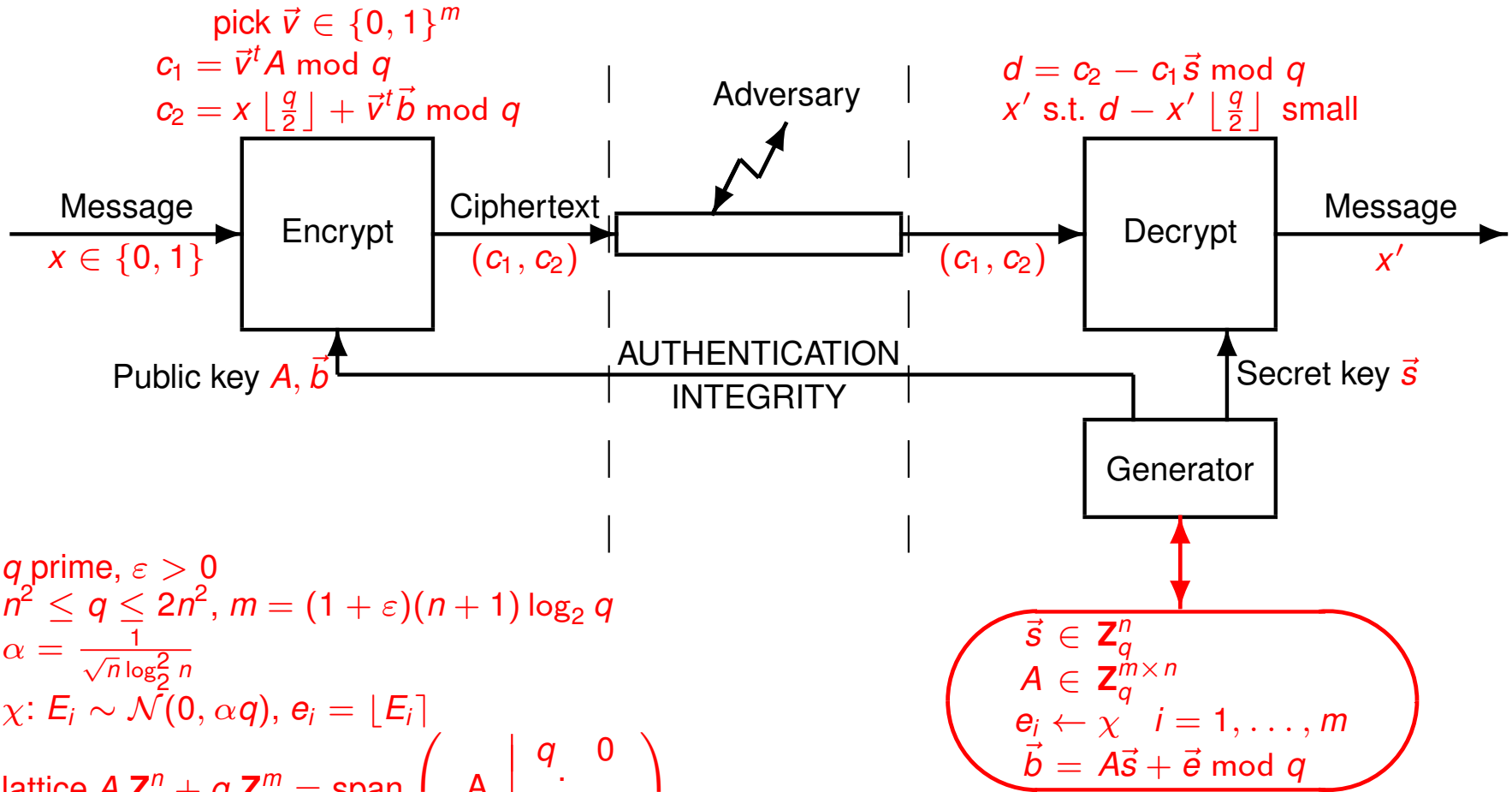
Module Lattices

"interpret vectors as polynomials and define multiplication"

example:

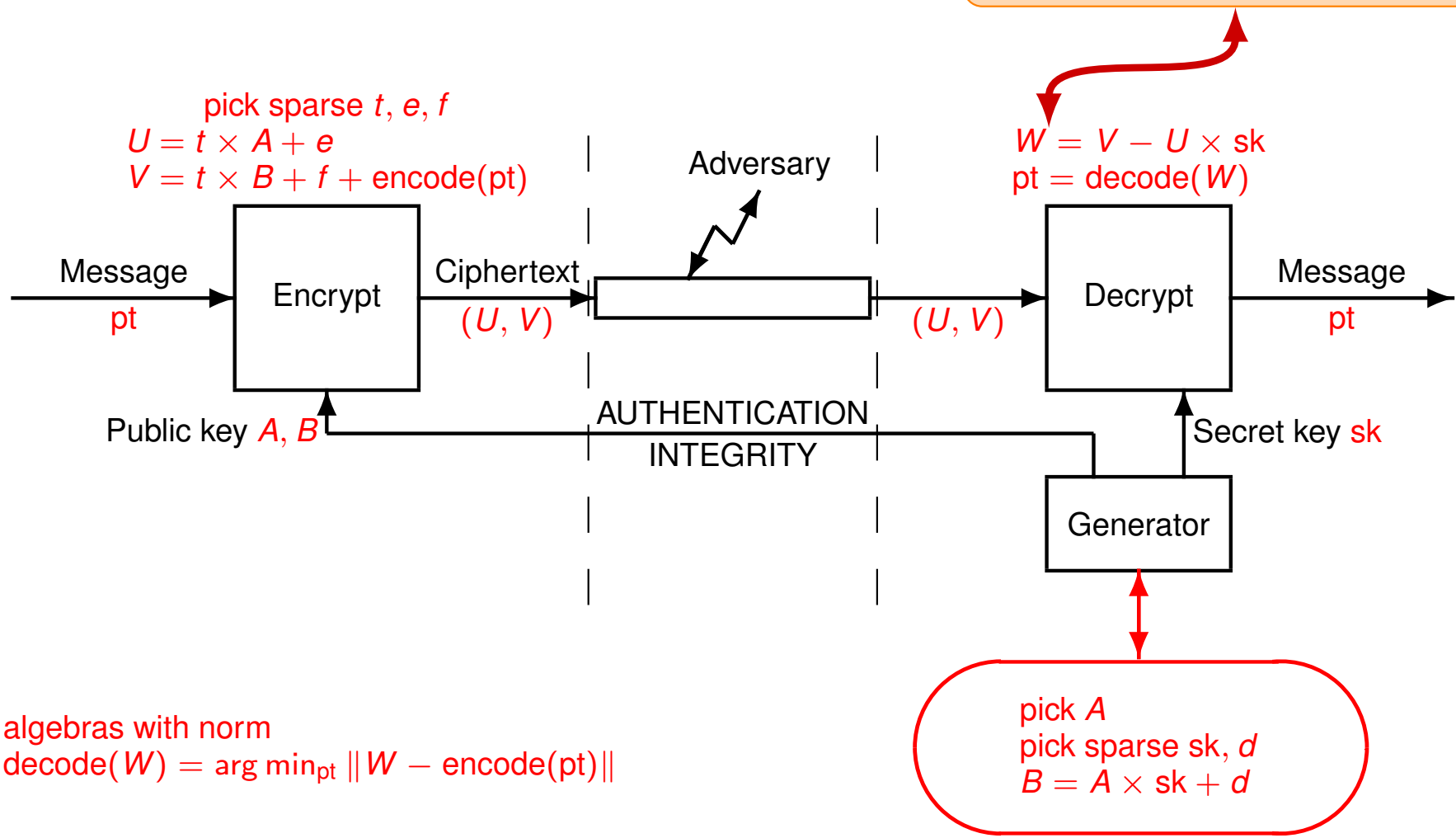
$$\mathcal{L} = \mathbf{Z}_q[z]/(z^n + 1)$$

The Regev Public-Key Cryptosystem



Meta (PQ) Cryptosystem

$$\delta = t \times d + f - e \times \mathrm{sk}$$
$$W = \delta + \mathrm{encode}(\mathrm{pt})$$



algebras with norm
$\mathrm{decode}(W) = \arg\min_{\mathrm{pt}} \|W - \mathrm{encode}(\mathrm{pt})\|$

Examples

- **FrodoPKE-640**: $q = 2^{15}$, $\bar{m} = \bar{n} = 8$, $n = 640$, $\ell = 2$

$$\mathsf{sk}, B, d \in \mathbf{Z}_q^{n \times \bar{n}} \quad A \in \mathbf{Z}_q^{n^2} \quad t, U, e \in \mathbf{Z}_q^{\bar{m} \times n} \quad V, f, W, \delta \in \mathbf{Z}_q^{\bar{m} \times \bar{n}}$$

$$\begin{aligned} \|X\| &= \max_{i,j} \left| \left(\left(X_{i,j} + \frac{q}{2} \right) \bmod q \right) - \frac{q}{2} \right| \\ (\mathsf{encode}(\mathsf{pt}))_{i,j} &= q 2^{-\ell} \sum_{k=1}^{\ell} 2^{k-1} \mathsf{pt}_{\ell((i-1)\bar{n}+(j-1))+k} \end{aligned}$$

- **NewHope512CPA-PKE**: $q = 12\,289$, $n = 512$

$$\mathsf{sk}, A, B, d, t, e, f, U, V, W, \delta \in \mathbf{Z}_q[z]/(z^n + 1)$$

$$\begin{aligned} \left\| \sum_{i=0}^{n-1} X_i z^i \right\| &= \max_i \left| \left(\left(X_i + \frac{q}{2} \right) \bmod q \right) - \frac{q}{2} \right| \\ \mathsf{encode}(\mathsf{pt}) &= \frac{q}{2} \sum_{i=1}^{n} (z^{i-1} + z^{i+255}) \mathsf{pt}_i \end{aligned}$$

Crystals-Kyber

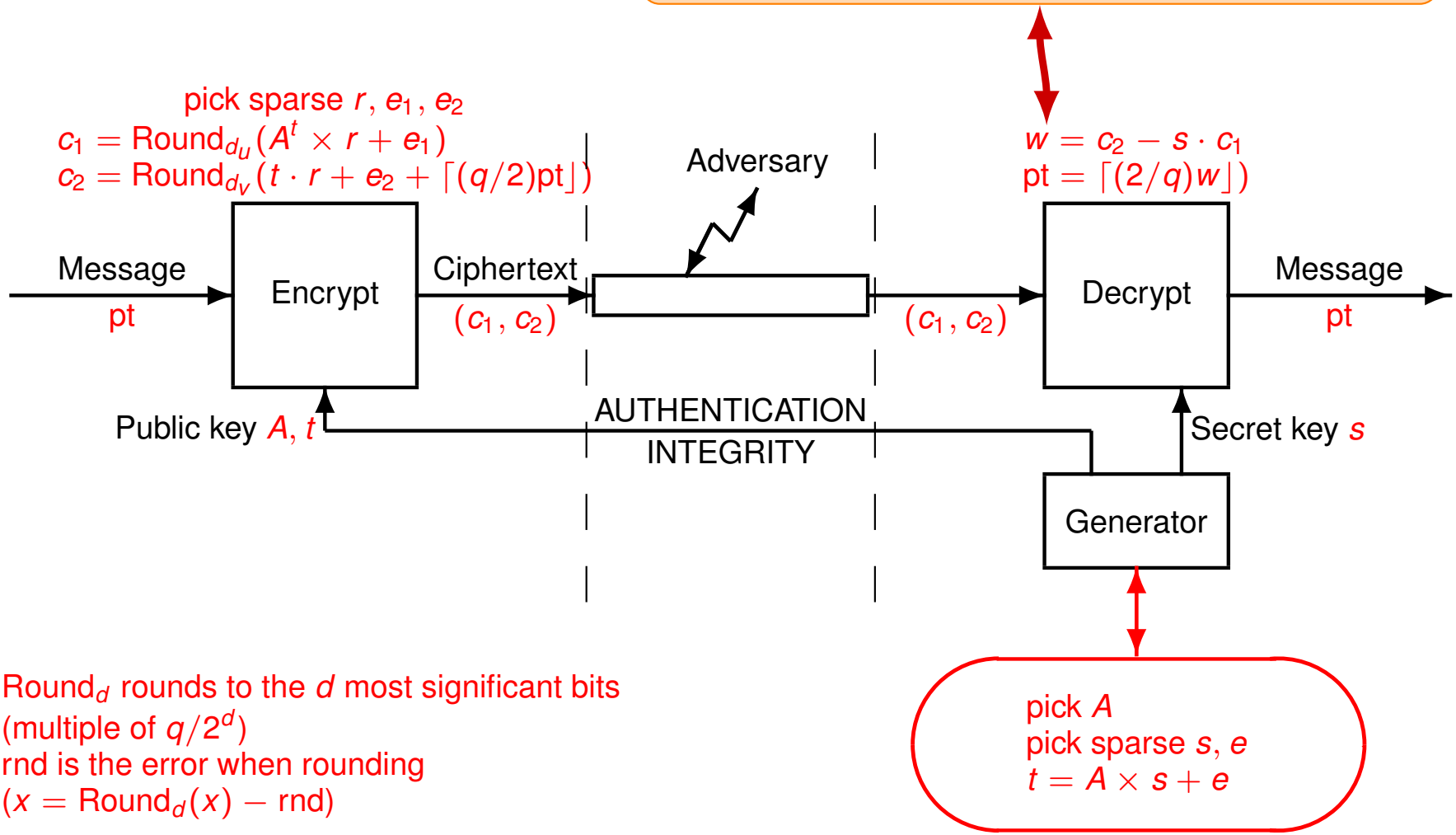
- $q = 3329$, $n = 256$, $L = \mathbf{Z}_q[z]/(z^n + 1)$, $k \in \{2, 3, 4\}$

$$\text{sk}, B, d, t, e, U \in L^k \qquad A \in L^{k \times k} \qquad f, V, W, \delta \in L$$

- K-PKE: weak version (IND-CPA secure)
- ML-KEM: strong version (IND-CCA secure by FO-like transform)

K-PKE (Unoptimized)

$$\delta = e \cdot r + e_2 - s \cdot e_1$$
$$w = \delta + \mathrm{rnd}_2 - s \cdot \mathrm{rnd}_1 + \lceil (q/2)\mathrm{pt} \rfloor)$$



Round_d rounds to the d most significant bits (multiple of $q/2^d$)
rnd is the error when rounding ($x = \mathrm{Round}_d(x) - \mathrm{rnd}$)

How to Multiple in Linear Time

there is a one-to-one correspondence

$$P(z) \in \mathbf{Z}_q[z]/(z^n + 1) \leftrightarrow (P(\zeta_0), P(\zeta_1), \ldots, P(\zeta_{n-1})) \in \mathbf{Z}_q^n$$

when $z^n + 1 = (z - \zeta_0) \cdots (z - \zeta_{n-1})$

- good news: it is easier to multiply!

$$((PQ)(\zeta_0), \ldots, (PQ)(\zeta_{n-1})) = (P(\zeta_0)Q(\zeta_0), \ldots, P(\zeta_{n-1})Q(\zeta_{n-1}))$$

- bad news: $z^n + 1$ has only half of its roots in $\mathbf{Z}_q$
- hint: $P(\zeta_i) = P(z) \bmod (z - \zeta_i)$

Facts about the Kyber Field

The Number Theoretic Transform (NTT)

$n = 256$, $q = 3329 = 13n + 1$, $\zeta = 17$

- ζ is a root of $z^{128} + 1$
- $z^n + 1 = \prod_{k=0}^{127}(z^2 - \zeta^{2k+1})$
- $\mathbf{Z}_q[z]/(z^n + 1)$ is isomorphic to $\prod_{k=0}^{127} \mathbf{Z}_q[z]/(z^2 - \zeta^{2k+1})$
- 1-to-1 correspondence between $P(z) \in \mathbf{Z}_q[z]/(z^n + 1)$ and $\hat{P} = \left(P(z) \bmod (z^2 - \zeta^{2k+1})\right)_{0 \le k < 128}$
- in the NTT domain, additions and multiplications are coordinate-wise

Kyber K-PKE

name	security	n	q	k	η_1	η_2	d_u	d_v
ML-KEM-512	128	256	3329	2	3	2	10	4
ML-KEM-768	192	256	3329	3	2	2	10	4
ML-KEM-1024	256	256	3329	4	2	2	11	5

Gen:

1: pick $\hat{A}$ in NTT domain ($k \times k$)
2: $s, e \leftarrow \text{Sample}_k(\eta_1)$
3: $\hat{s} \leftarrow \text{NTT}(s)$
4: $\hat{e} \leftarrow \text{NTT}(e)$
5: $\hat{t} \leftarrow \hat{A} \times \hat{s} + \hat{e}$
6: $\text{pk} \leftarrow (\hat{t}, \hat{A})$
7: $\text{sk} \leftarrow \hat{s}$
8: **return** (pk, sk)

$\text{Sample}_\ell(\eta)$:

9: pick $\vec{x}_1, \ldots, \vec{x}_\eta \xleftarrow{\$} B^\ell$
10: pick $\vec{y}_1, \ldots, \vec{y}_\eta \xleftarrow{\$} B^\ell$
11: **return** $\sum \vec{x} - \sum \vec{y}$

$$B = \left\{ \sum_{i=0}^{n-1} b_i z^i ; b_i \in \{0, 1\} \right\}$$

return ℓ-vector of polynomials
coefficients with
expected value: 0
variance: $\eta/2$

Kyber K-PKE

Enc(pk, pt):

1: pk $\to (\hat{t}, \hat{A})$
2: $r \leftarrow \text{Sample}_k(\eta_1)$
3: $e_1 \leftarrow \text{Sample}_k(\eta_2)$
4: $e_2 \leftarrow \text{Sample}_1(\eta_2)$
5: $\hat{r} \leftarrow \text{NTT}(r)$
6: $u \leftarrow \text{NTT}^{-1}(\hat{A}^t \times \hat{r}) + e_1$
7: $\mu \leftarrow \text{Decompress}_1(\text{pt})$
8: $v \leftarrow \text{NTT}^{-1}(\hat{t}^t \times \hat{r}) + e_2 + \mu$
9: $c_1 \leftarrow \text{Compress}_{d_u}(u)$
10: $c_2 \leftarrow \text{Compress}_{d_v}(v)$
11: **return** (c_1, c_2)

Dec(sk, c_1, c_2):

12: $u \leftarrow \text{Decompress}_{d_u}(c_1)$
13: $v \leftarrow \text{Decompress}_{d_v}(c_2)$
14: sk $\to \hat{s}$
15: $w \leftarrow v - \text{NTT}^{-1}(\hat{s}^t \times \text{NTT}(u))$
16: pt $\leftarrow \text{Compress}_1(w)$
17: **return** pt

$\text{Compress}_d(x)$:

18: $y_i \leftarrow \lceil (2^d/q)x_i \rfloor$ for all i
19: **return** y

$\text{Decompress}_d(y)$:

20: $x_i \leftarrow \lceil (q/2^d)y_i \rfloor$ for all i
21: **return** x

$\text{Decompress}_d \circ \text{Compress}_d$ rounds to multiples of $q/2^d$

FO Variant for IND-CCA KEM (in ML-KEM)

KemGen:

1: $\text{Gen} \to (\text{pk}, \text{sk})$
2: pick z
3: $h \leftarrow H(\text{pk})$
4: $\text{dk} \leftarrow (\text{sk}, \text{pk}, h, z)$
5: **return** (pk, dk)

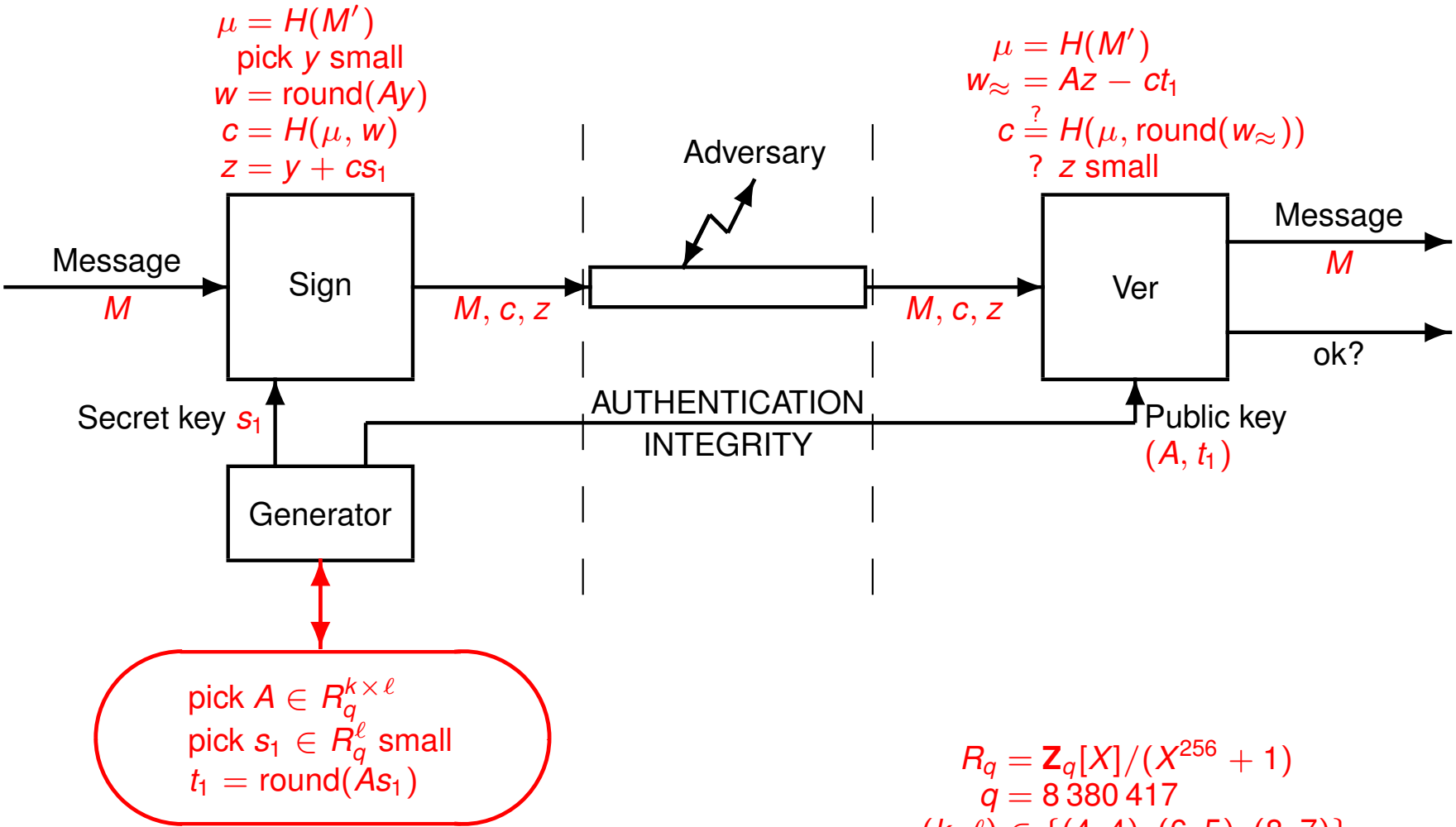
KemEnc(pk):

6: pick coins
7: $(\text{key}, r) \leftarrow G(\text{coins} \| H(\text{pk}))$
8: $\text{ct} \leftarrow \text{Enc}_{\text{pk}}(\text{coins}; r)$
9: **return** (key, ct)

KemDec(dk, ct):

10: $\text{dk} \to (\text{sk}, \text{pk}, h, z)$
11: $\text{coins} \leftarrow \text{Dec}_{\text{sk}}(\text{ct})$
12: $(\text{key}, r) \leftarrow G(\text{coins} \| h)$
13: $\text{key}' \leftarrow J(z \| \text{ct})$
14: $\text{ct}' \leftarrow \text{Enc}_{\text{pk}}(\text{coins}; r)$
15: **if** $\text{ct} \neq \text{ct}'$ **then** $\text{key} \leftarrow \text{key}'$
16: **return** key

ML-DSA [FIPS 204] (Simplified)



$$R_q = \mathbf{Z}_q[X]/(X^{256} + 1)$$
$$q = 8\,380\,417$$
$$(k, \ell) \in \{(4, 4), (6, 5), (8, 7)\}$$

ML-DSA: $M' = 0 \| M$
HashML-DSA: $M' = 1 \| H(M)$

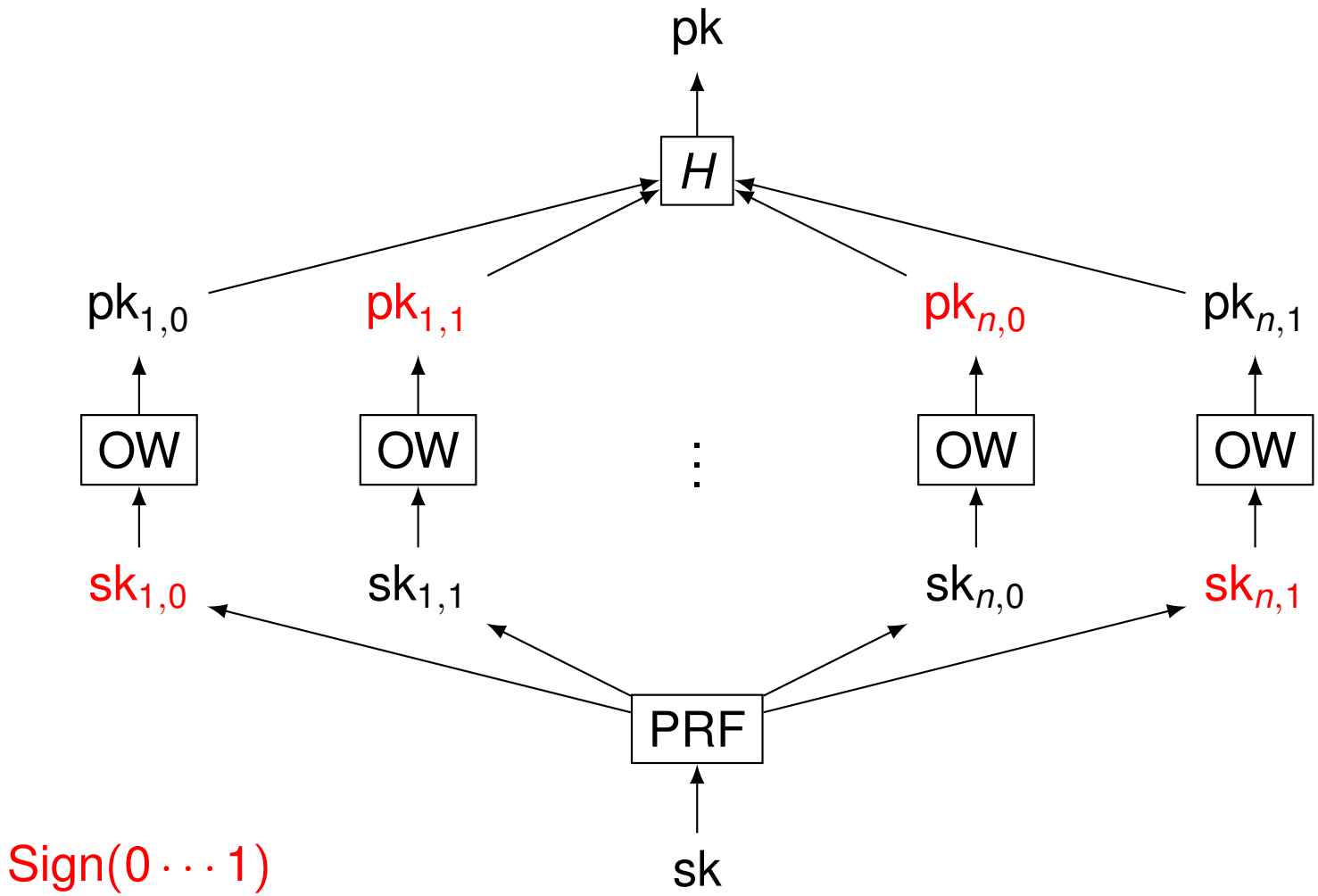
7 Public-Key Cryptography

- Public-Key Cryptography
- RSA Cryptography
- ElGamal Cryptography
- Selecting Key Lengths
- Formalism
- Towards Post-Quantum Cryptography
- Lattice-Based Cryptography
- Hash-Based Cryptography

Lamport (One-Time) Signature Scheme

- Parameter: n, the hash length
- **Secret key**: $(\mathrm{sk}_{i,b})_{i=1,\ldots,n,b=0,1}$
- **Public key**: $(\mathrm{OW}(\mathrm{sk}_{i,b}))_{i=1,\ldots,n,b=0,1}$
- **Signature** of m: $(\mathrm{sk}_{i,H(m)_i})_{i=1,\ldots,n}$
- **Verification**: $\mathrm{OW}(\sigma_i) = \mathrm{pk}_{i,H(m)_i}$ for $i = 1, \ldots, n$
- possible improvement for a shorter secret key: $\mathrm{sk}_{i,b} = \mathrm{PRF}_{\mathrm{seed}}(i, b)$ and keep seed
- possible improvement for a shorter public key: hash the public keys
- main drawback: large signature size, one-time pk use

Lamport (One-Time) Signature Scheme - Example



Lamport (One-Time) Signature Scheme - Example

- $n = 4$
- **Secret key**: sk
- **Public key**: $\mathrm{pk} = H(\text{all } \mathrm{pk}_{i,b})$ with

$$\mathrm{pk}_{i,b} = \mathrm{OW}(\mathrm{PRF}_{\mathrm{sk}}(i, b)) \qquad i = 1, \ldots, n, \quad b = 0, 1$$

- **Signature** of $m = 0101$:

$$\sigma_1 = \mathrm{PRF}_{\mathrm{sk}}(1, 0)\ ,\ \mathrm{pk}_{1,1} \qquad \sigma_2 = \mathrm{PRF}_{\mathrm{sk}}(2, 1)\ ,\ \mathrm{pk}_{2,0}$$
$$\sigma_3 = \mathrm{PRF}_{\mathrm{sk}}(3, 0)\ ,\ \mathrm{pk}_{3,1} \qquad \sigma_4 = \mathrm{PRF}_{\mathrm{sk}}(4, 1)\ ,\ \mathrm{pk}_{4,0}$$

- **Verification**: compute

$$\mathrm{pk}_{1,0} = \mathrm{OW}(\sigma_1) \qquad \mathrm{pk}_{2,1} = \mathrm{OW}(\sigma_2)$$
$$\mathrm{pk}_{3,0} = \mathrm{OW}(\sigma_3) \qquad \mathrm{pk}_{4,1} = \mathrm{OW}(\sigma_4)$$

then verify $\mathrm{pk} = H(\text{all } \mathrm{pk}_{i,b})$

Winternitz (One-Time) Signature Scheme

- Parameters: power-of-2 w, hash length $n \log_2 w$, $n' = n + \frac{\log n}{\log w} + 1$, message space $\{0, \ldots, w-1\}^n$
- **Secret key**: $(\mathsf{sk}_i)_{i=1,\ldots,n'}$
- **Public key**: $(\mathsf{OW}^{w-1}(\mathsf{sk}_i))_{i=1,\ldots,n'}$
- **Signature** of m:

  1: parse $m = j_1 | \cdots | j_n$ with $j_i \in \{0, \ldots, w-1\}$,

  2: parse $\sum_{i=1}^{n} (w - 1 - j_i) = [j_{n+1} | \cdots | j_{n'}]$,

  ▷ $j_1, \ldots, j_{n'}$ satisfies a checksum:

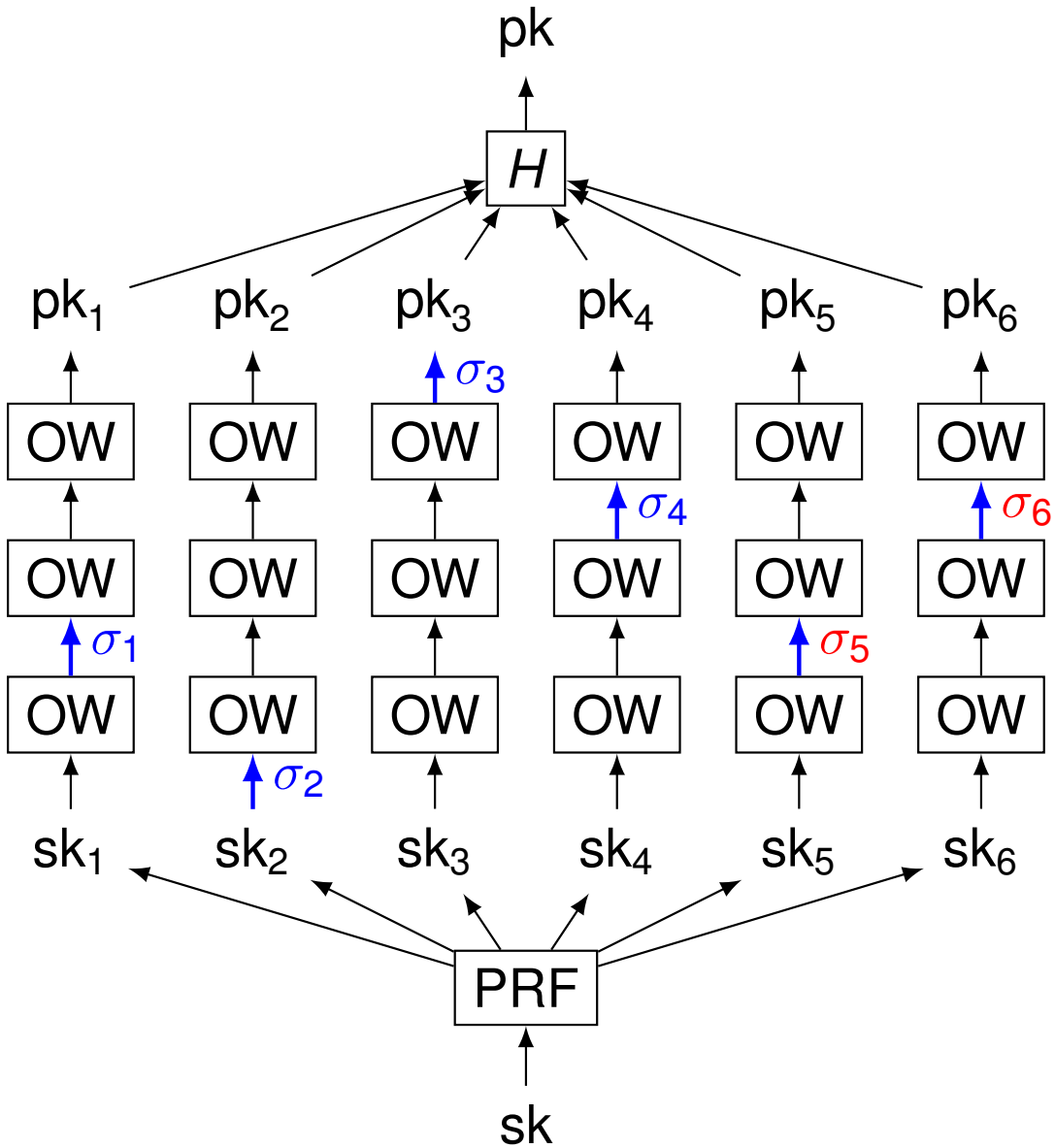
$$\left(\sum_{i=1}^{n} j_i \right) + [j_{n+1} | \cdots | j_{n'}] = n(w-1)$$

  3: $\sigma \leftarrow (\mathsf{OW}^{j_i}(\mathsf{sk}_i))_{i=1,\ldots,n'}$
- **Verification**:

  1: compute $j_1 | \cdots | j_{n'}$ as above

  2: check $\mathsf{OW}^{w-1-j_i}(\sigma_i) = \mathsf{pk}_i$ for $i = 1, \ldots, n'$
- possible improvement: all pk_i in a Merkle tree or an accumulator
- possible improvement: sk_i from seed sk

Winternitz Signature Scheme - Example



n=4, w=2^2, $n' = 6$

Sign(01 00 11 10)

10+11+00+01=01 10

Attack If We Had No Checksum

- **Secret key**: $(\mathsf{sk}_i)_{i=1,\ldots,n}$
- **Public key**: $(\mathsf{OW}^{w-1}(\mathsf{sk}_i))_{i=1,\ldots,n}$
- **Signature** of m:
  1: parse $m = j_1 | \cdots | j_n$ with $j_i \in \{0, \ldots, w-1\}$,
  2: $\sigma \leftarrow (\mathsf{OW}^{j_i}(\mathsf{sk}_i))_{i=1,\ldots,n}$
- **Verification**:
  1: compute $j_1 | \cdots | j_n$ as above
  2: check $\mathsf{OW}^{w-1-j_i}(\sigma_i) = \mathsf{pk}_i$ for $i = 1, \ldots, n$
- given (m, σ) we can forge the signature σ' of any $m' = j'_1 | \cdots | j'_n$ such that $j'_i \geq j_i$ for $i = 1, \ldots, n$:

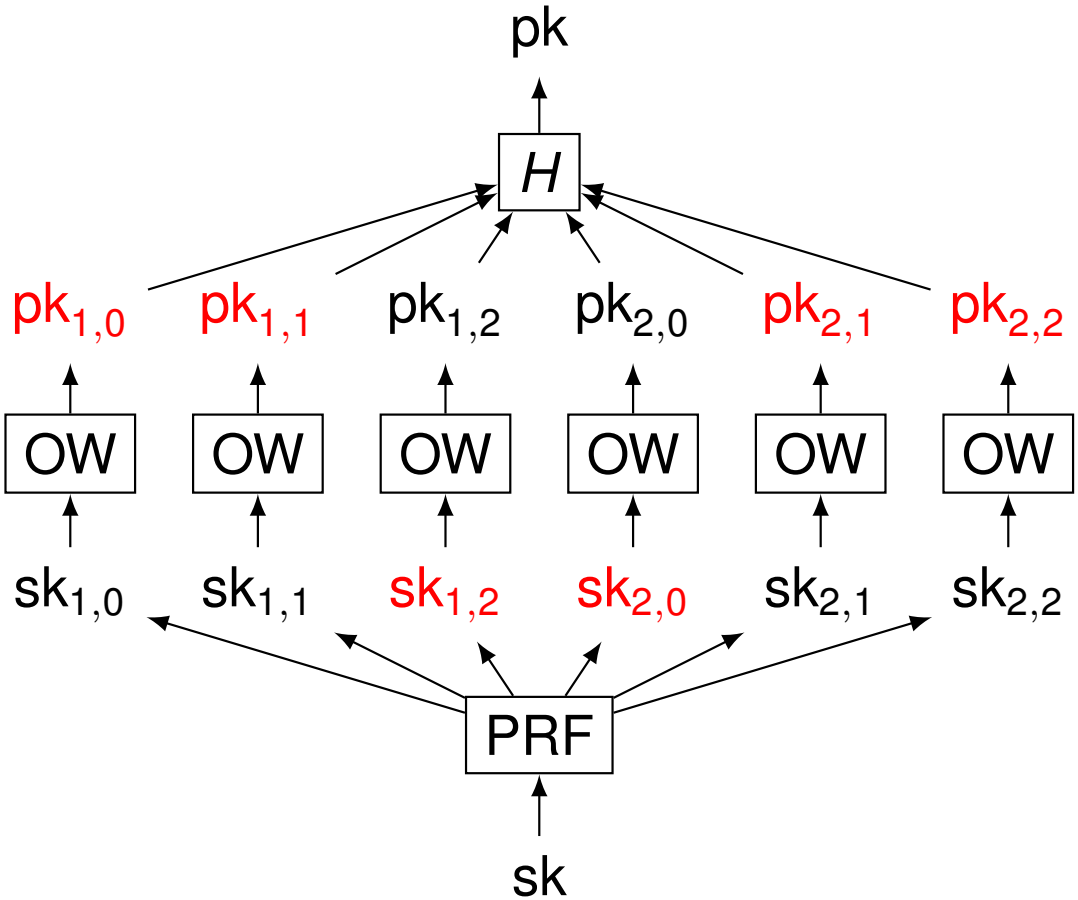
$$\sigma'_i = \mathsf{OW}^{j'_i - j_i}(\sigma_i)$$

- idea: hash the message so that $\bigwedge_i j'_i \geq j_i$ occurs with negligible probability

FORS (Few-Times) Signature Scheme

- Parameters: k, t
- **Secret key**: $(\mathsf{sk}_{i,j})_{i=1,\ldots,k,j=0,\ldots,t-1}$
- **Public key**: $(\mathsf{OW}(\mathsf{sk}_{i,j}))_{i,j}$
- **Signature** of m:
  1: parse $H(m) = j_1 | \cdots | j_k$ with $j_i \in \{0, \ldots, t-1\}$,
  2: $\sigma \leftarrow (\mathsf{sk}_{i,j_i})_{i=1,\ldots,k}$
- **Verification**:
  1: compute $j_1 | \cdots | j_k$ as above
  2: check $\mathsf{OW}(\sigma_i) = \mathsf{pk}_{i,j_i}$ for $i = 1, \ldots, k$
- tricky selection of parameters to make it secure
- same possible improvements

FORS Signature Scheme - Example

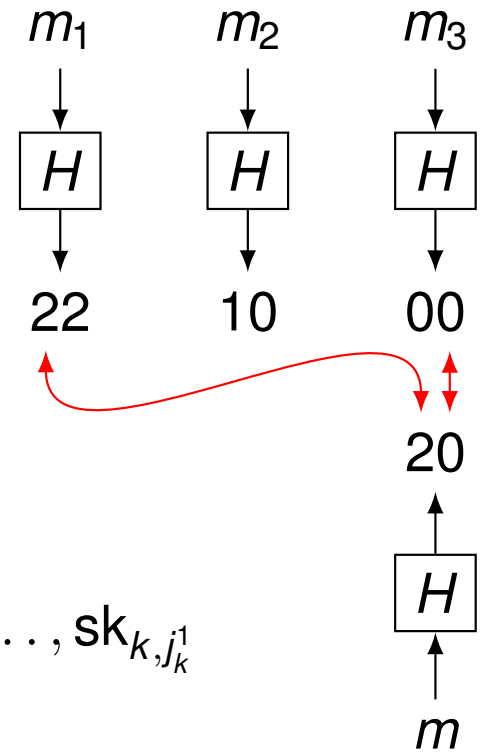


k=2, t=3
Sign(*m*)
H(m)=20

Trying to Attack FORS

- **Secret key**: $(\mathsf{sk}_{i,j})_{i=1,\ldots,k,j=1,\ldots,t}$
- **Public key**: $(\mathsf{OW}(\mathsf{sk}_{i,j}))_{i,j}$
- **Signature** of m:
  1: parse $H(m) = j_1 | \cdots | j_k$
  2: $\sigma \leftarrow (\mathsf{sk}_{i,j_i})_{i=1,\ldots,k}$
- collect q signatures

$$\begin{aligned} \mathsf{Sign}\left(m_1 \xrightarrow{H} j_1^1 \| \cdots \| j_k^1\right) &= \mathsf{sk}_{1,j_1^1}, \ldots, \mathsf{sk}_{k,j_k^1} \\ &\vdots \\ \mathsf{Sign}\left(m_q \xrightarrow{H} j_1^q \| \cdots \| j_k^q\right) &= \mathsf{sk}_{1,j_1^q}, \ldots, \mathsf{sk}_{k,j_k^q} \end{aligned}$$



- hash a new message until $\bigwedge_i j_i \in \{j_i^1, \ldots, j_i^q\}$
- complexity is $\left(\frac{t}{q}\right)^k$
- idea: make sure that q remains small ("few-times signature")

Sphincs+: a Hash-Based Signature Scheme

- **Secret key**: seed
- **Public key**: key + root of a Merkle tree, with leaves being roots of FORS trees (generated from secret seed)
- **Signature** of m:
  1: pick a random R
  2: parse $H(R, \text{key}, m)$ as a digest and the address of a FORS tree
  3: sign the digest using this FORS tree
  4: $\sigma \leftarrow (R, \text{FORS.signature}, \text{Merkle})$
- **Verification**:
  1: parse $H(R, \text{key}, m)$ as a digest and the address of a FORS tree
  2: verify FORS signature of digest with this FORS tree

Conclusion (on Chapters 2,3,4,7)

- 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
- new: digital signatures
- PQ-crypto

References

- **Merkle**. Secure Communications over Insecure Channels. *Communications of the ACM* vol. 21, 1978.
- **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

- PKCS#1
- ElGamal signature and variants: be careful about the nonce
- how to select key lengths
- hybrid encryption (symmetric + public-key)
- Fujisaki-Okamoto
- post-quantum crypto

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
- Learning Parity with Noise: final exam 2017–18 ex2
- Mersenne cryptosystem: final exam 2018–19 ex1
- PKC vs KEM vs KA: final exam 2018–19 ex3
- hash-based signature: final exam 2022–23 ex2
- PQ-cryptosystem: final exam 2019–20 ex3
- Windows 10 software protection: final exam 2019–20 ex4
- Merkle's puzzles: final exam 2020–21 ex1
- key length: final exam 2020–21 ex2
- ECDSA with bad randomness: final exam 2020–21 ex3
- blind Schnorr signature: final exam 2021–22 ex3
- privacy pass: final exam 2021–22 ex2
- security of KEM: final exam 2024–25 ex2

Roadmap

- access control: challenge-response, strong authentication
- password-based cryptography
- secure communication channels
- setup by narrowband channel
- setup by a trusted third party: Kerberos, PKI

8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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

Application: Access Control

many scenarios:

- access to a computer
- access to a door: “Sésame”
- access to a mailbox
- access 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 TLS)

Password Authentication Protocol — i

- server keeps a database of (ID, password) entries
- channel to server keeps confidentiality

Client		**Server**
	$\xrightarrow{\text{ID,password}}$	get entry (ID, w) check $w =$ password

Problem: if adversary has access to database he can get the password

Password Authentication Protocol — ii

- server keeps a database of (ID, OW(password)) entries
- channel to server keeps confidentiality

Client		**Server**
	ID,password $\longrightarrow$	get entry (ID, w) check $w =$ OW(password)

Problem: multi-target invertion 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

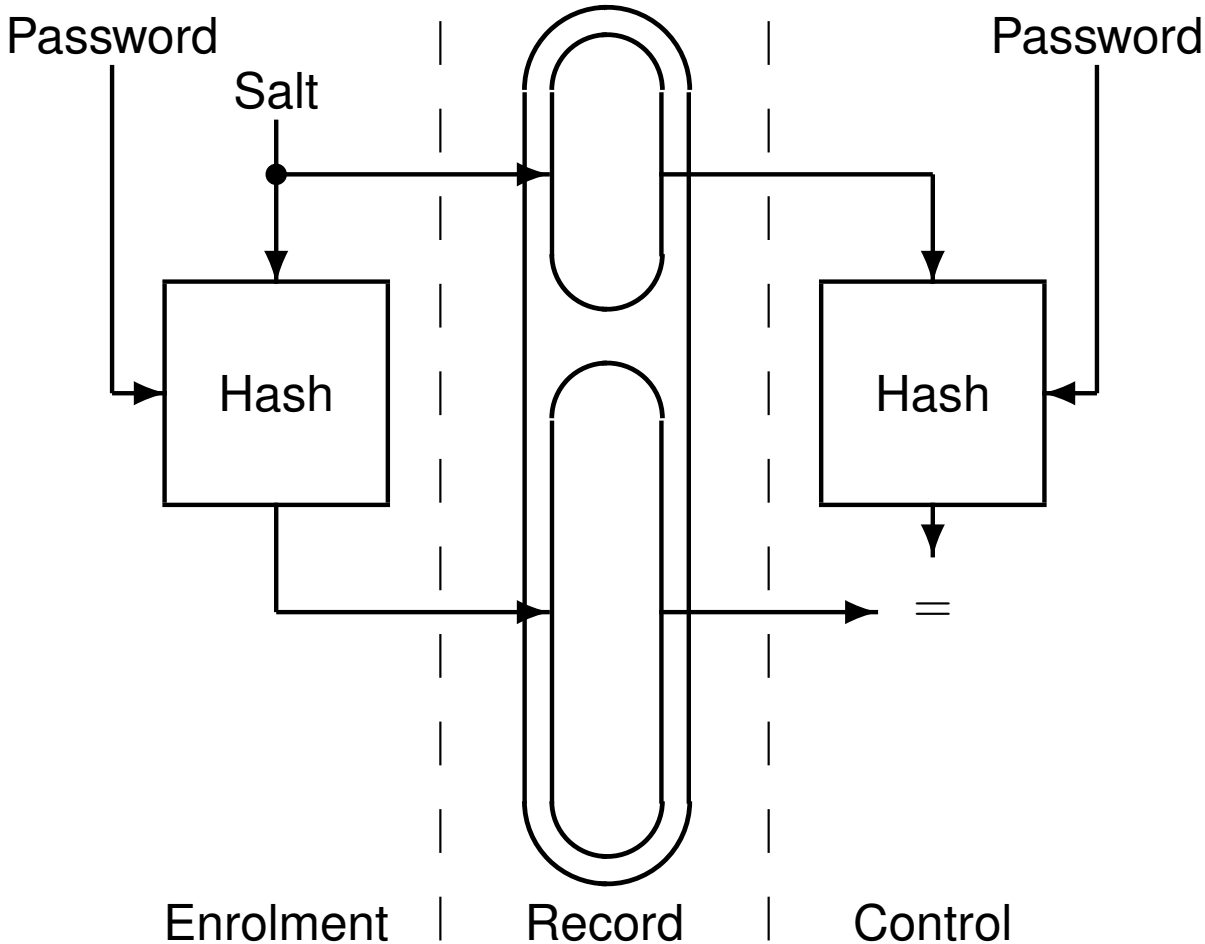
Client		**Server**
	ID,password $\longrightarrow$	get entry (ID, salt, w) check $w = \mathrm{OW}(\text{password}, \text{salt})$

advantages:

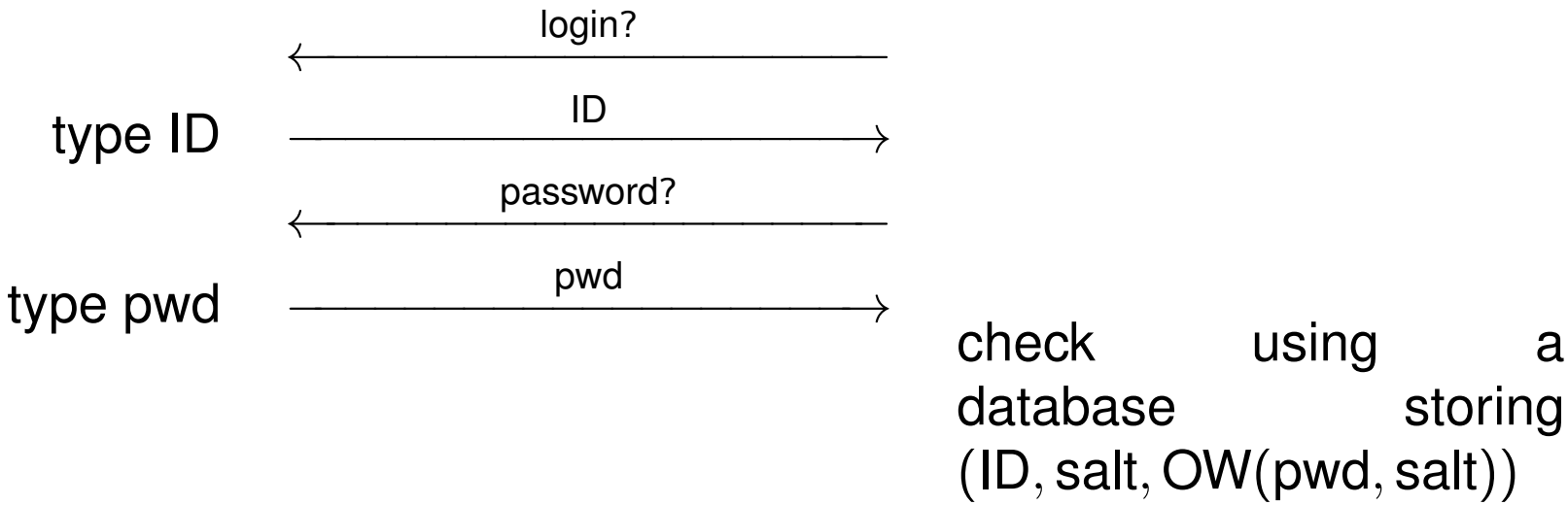
- avoid multi-target bruteforce attacks from database

(does not avoid single-target exhaustive search from database)

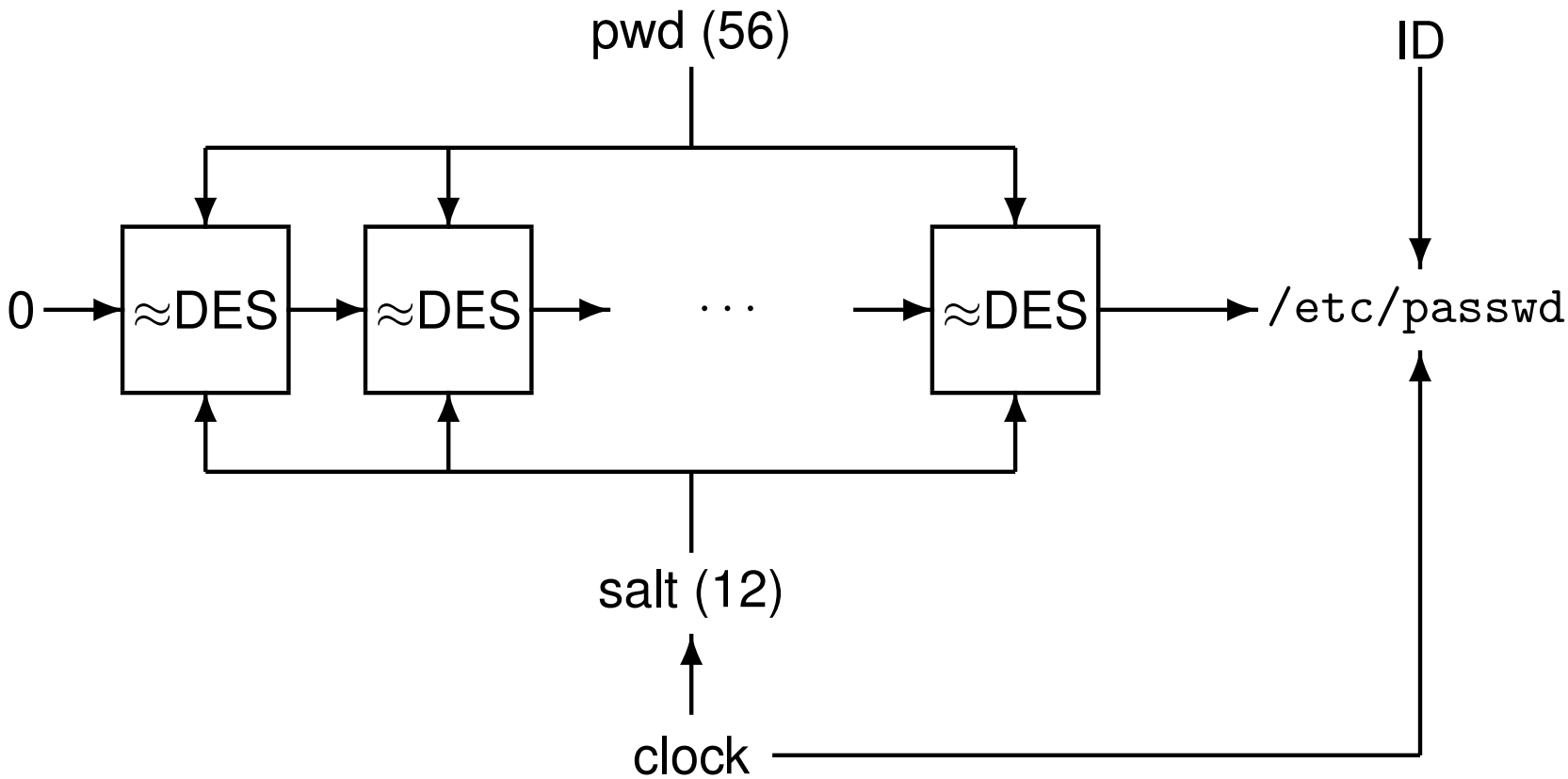
Password Access Control Using Salt



Example: UNIX Password Access Protocol



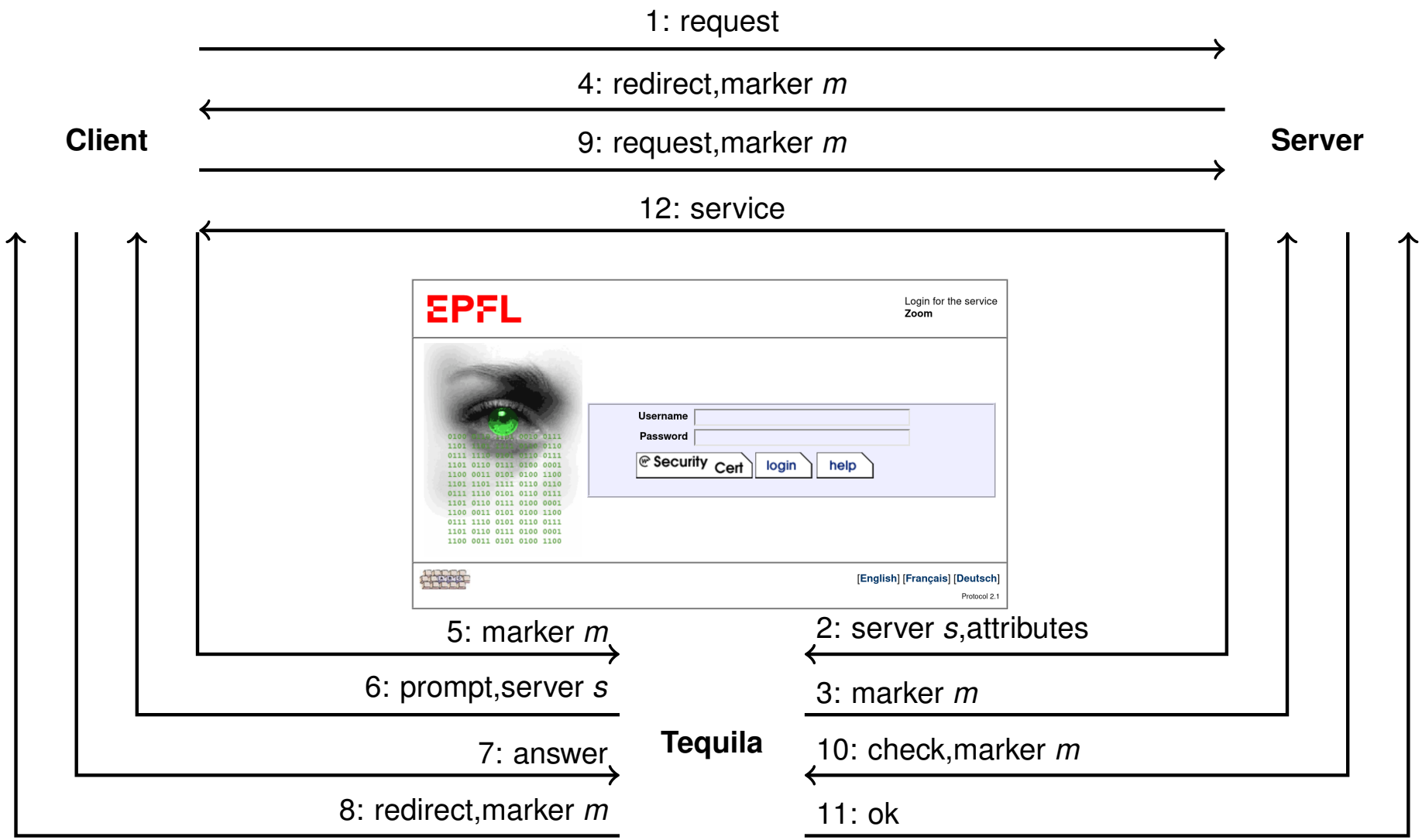
UNIX Passwords



Examples

- UNIX password authentication
- Basic Access Control in HTTP [RFC2617]
- IMAP4rev1 [RFC2060]
- tequila authentication at EPFL

tequila@epfl



problem: not sure the prompt comes from tequila
privacy: Tequila warns if a sensitive attribute is requested by Server

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)

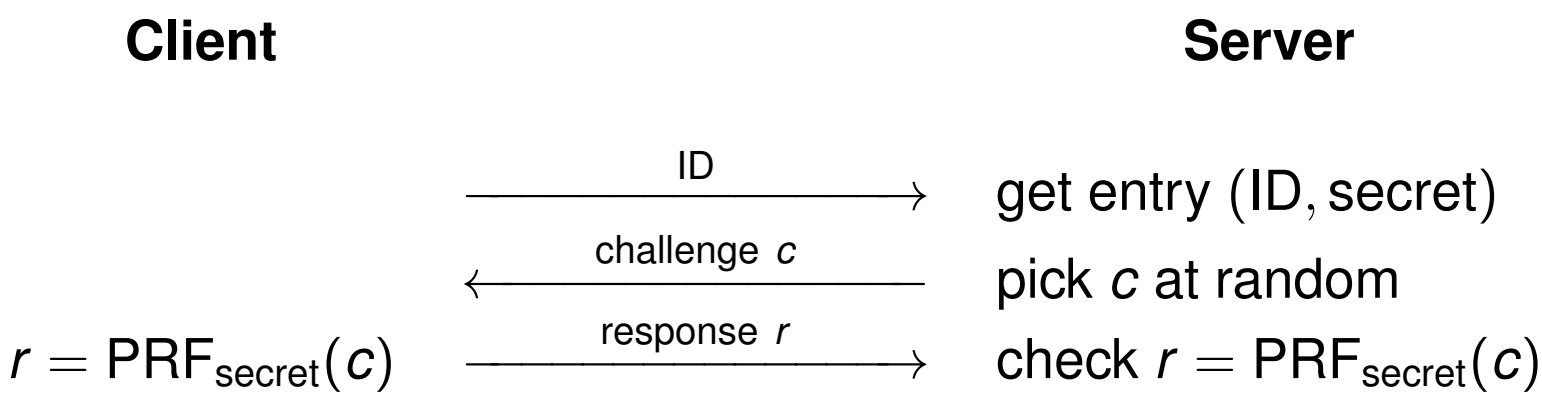
- PROBLEM: authenticate a client to a server
- HYPOTHESIS 2: adversary is passive
- example: unencrypted semi-authenticated channel (client authenticates the server e.g. using TLS but they are not allowed to use encryption)

Passive vs Active Adversary

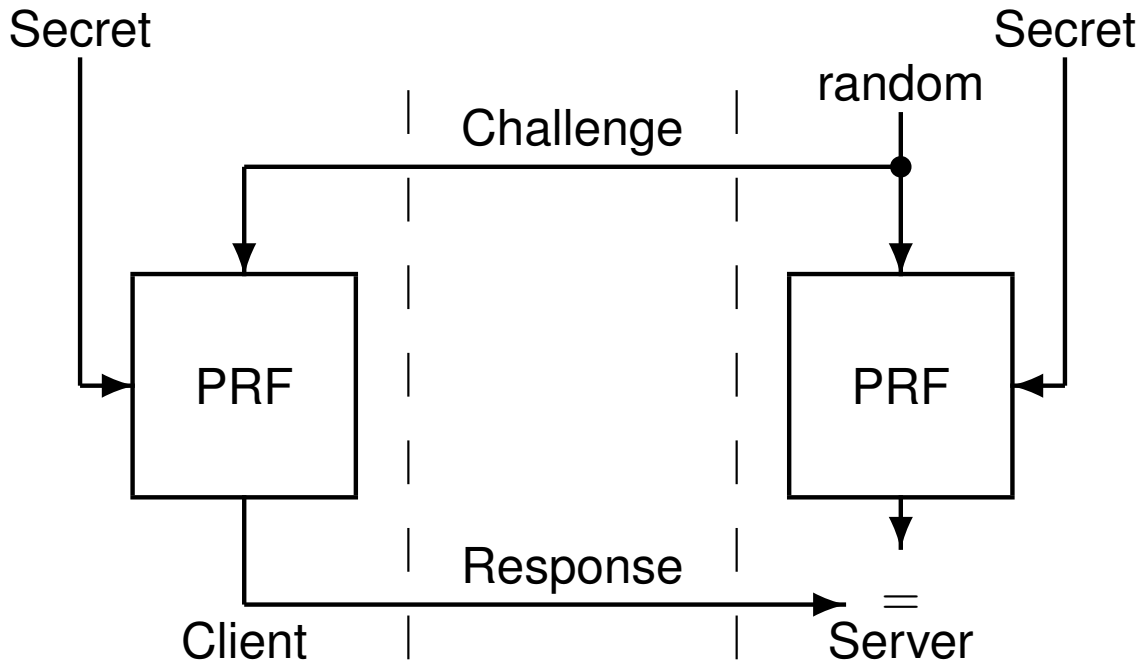
- **passive adversary**: only listen to communications and tries to get credential to later pass access control
- **active adversary**: can interfere with client or server communications e.g. man-in-the-middle

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)

Examples

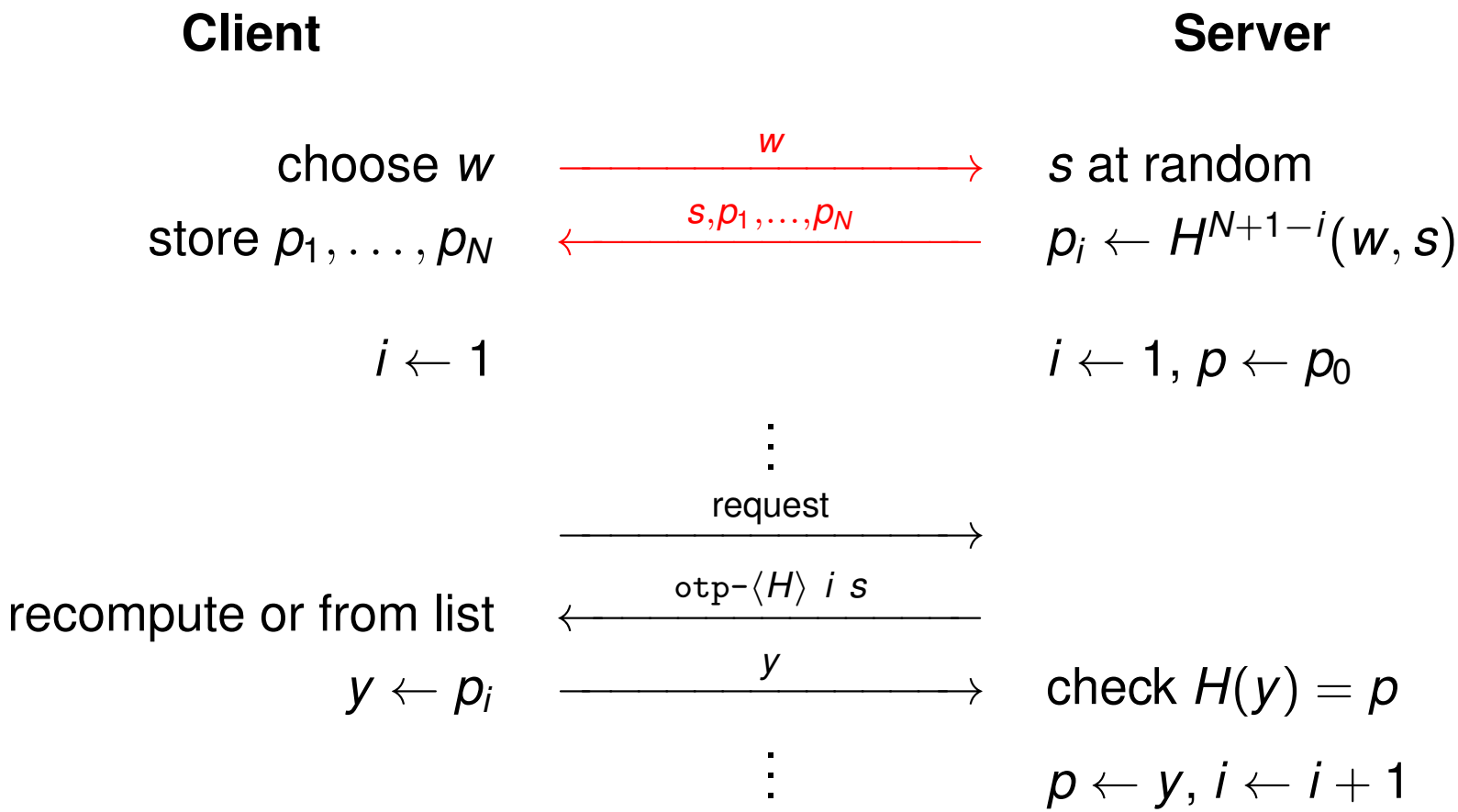
- GSM
- Digest Access Control in HTTP [RFC2617]
- Bluetooth peer authentication
- access control to UBS account (later in this chapter)

Case Study: Bluetooth

case study

S/Key - OTP [RFC2289]

possible hash function H: `md4`, `md5`, `sha1`



w must have a large entropy
challenges must be authenticated
responses shall be protected against delays in delivery

HMAC-Based One-Time Password (HOTP) [RFC4226]

HOTP generator (client) and HOTP validator (server)

$$\text{HOTP}(K, C) = \text{DT}(\text{HMAC}(K, C)) \bmod 10^d$$

- uses HMAC-SHA1 (output of 20 bytes)
- C (8 bytes) counter synchronized between client and server
- K shared secret
- DT: read 4 consecutive bytes of the input starting from the one of index equal to the last four bits of the input
- T number of unsuccessful attempts before the server blocks
- s: size of a look-ahead window (to resynchronize the counter)
- d: digit length of HOTP values in decimal

Time-Based One-Time Password (TOTP) [RFC6238]

- set C to a function of the cuttent time
- $C = \frac{\text{time} - t_0}{X}$ (typically: $X = 30\text{sec}$)
- can use SHA2
- application: Google Authenticator, Microsoft Authenticator
- used for 2-factor authentication

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, impossible to change

strong authentication = authentication using at least two methods (2-way authentication)
example: smart card + PIN code

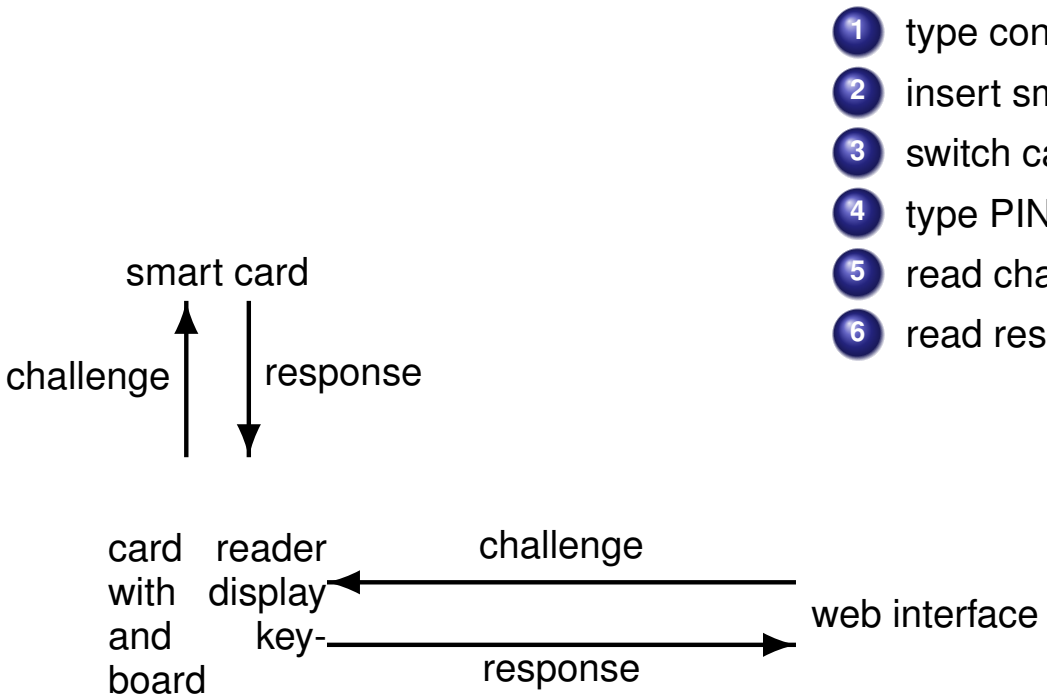
Example of Critical Application: UBS E-Banking

[E-banking from a browser]

Requirements for e-Banking

- strong bidirectional **authentication**
- **confidentiality** of communication
- **integrity** of communication
- **non-repudiation** of transaction
- **resilience** to clients in hostile environment

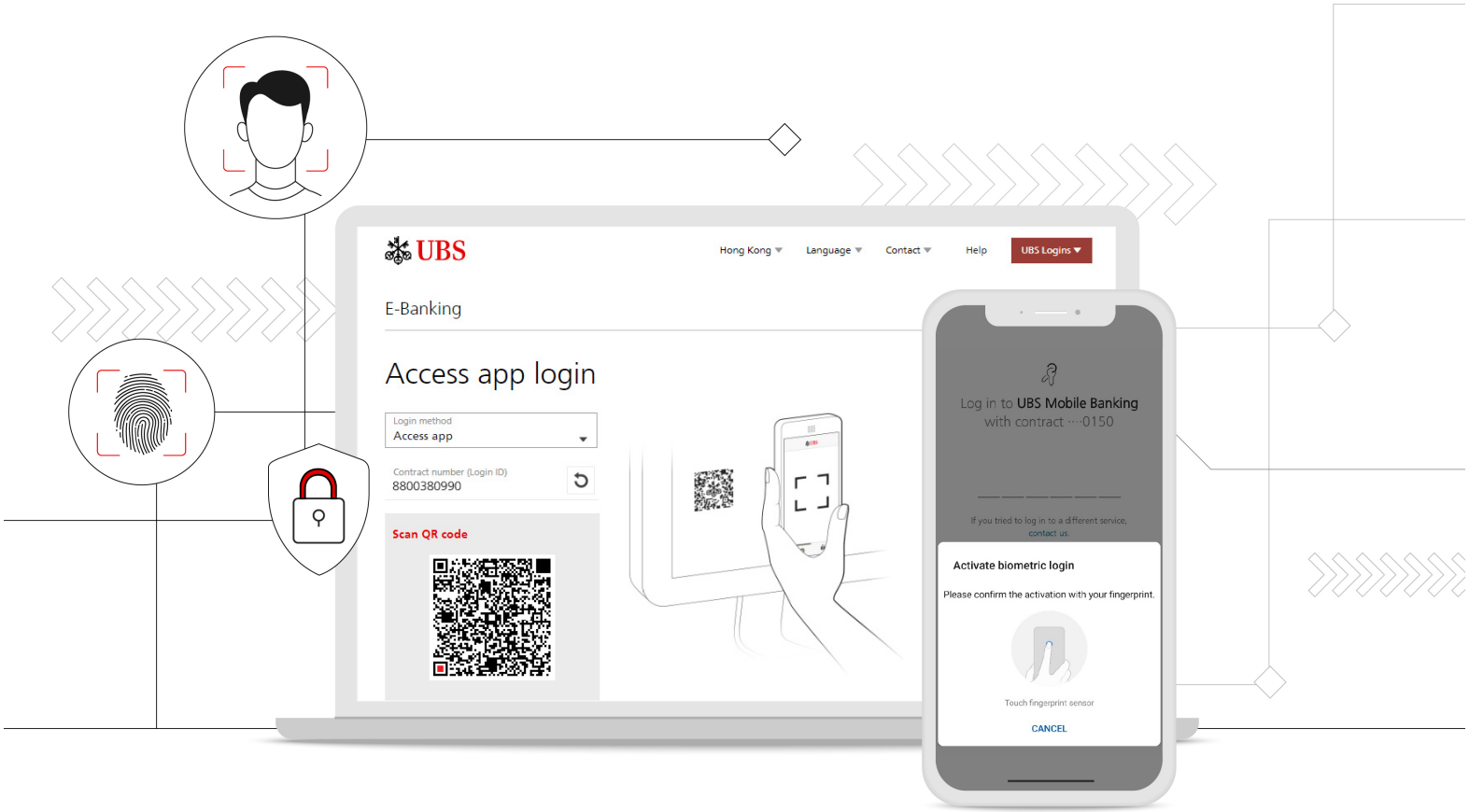
Example: (Old) UBS E-Banking



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

Example: (New) UBS E-Banking



- UBS app installed on smartphone (tedious setup)
- biometric access to app
- scan QR code and report to UBS

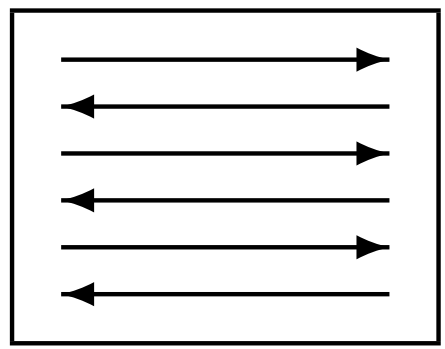
8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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

Password-based Access Control Protocol

Alice
password: w
random tape: r_A

Bob
password: w
random tape: r_B



output: ok (or abort)

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

Online Dictionary Attack: a Generic Attack

generic

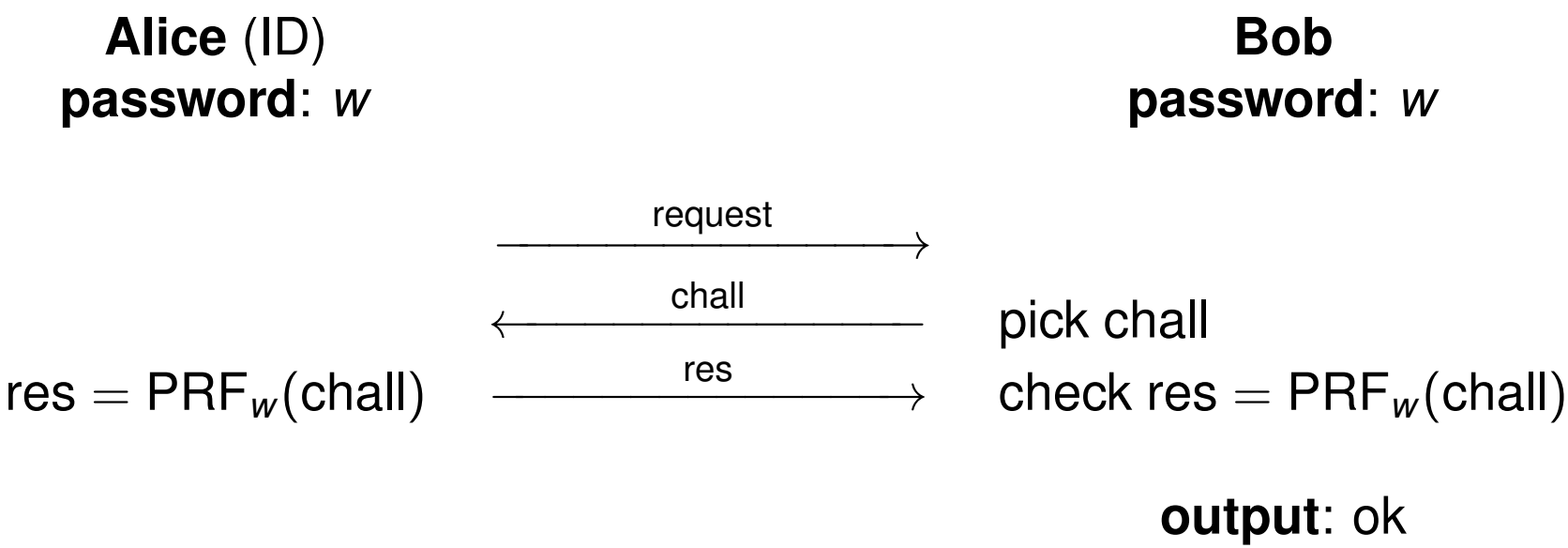
1: **repeat**
2: make a new guess $\hat{w}$ following a dictionary
3: simulate Alice with password input $\hat{w}$ to interact with Bob
4: **until** Bob accepts
5: print $\hat{w}$

a protocol is secure if this attack is the best one

Online and Offline Passwords Recovery

	online	**offline**
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	• increasing delay before new attempt • blocked after xx trials	• password with large entropy • use salt (mitigate multi-target) • leak no witness

(Bad) Example: Challenge/Response Protocol

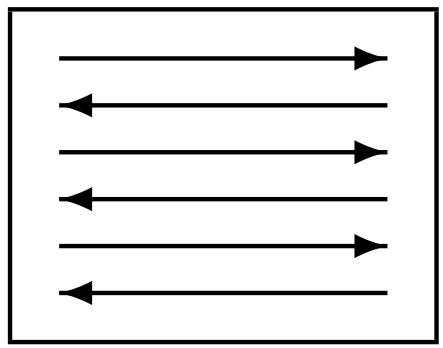


subject to offline exhaustive search

Password-Based Authenticated Key Agreement

Alice
password: w
random tape: r_A

Bob
password: w
random tape: r_B



output: K_A

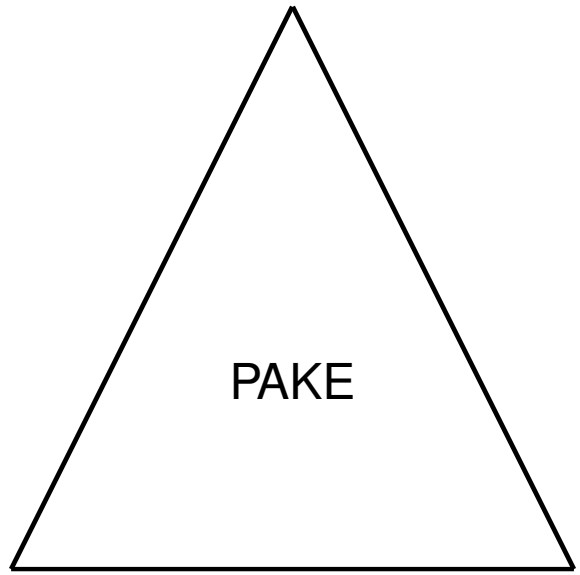
output: K_B

- 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

Alice and Bob, Proto_A, Proto_B

components



functionality

$\text{Proto}_A(w) \leftrightarrow \text{Proto}_B(w)$

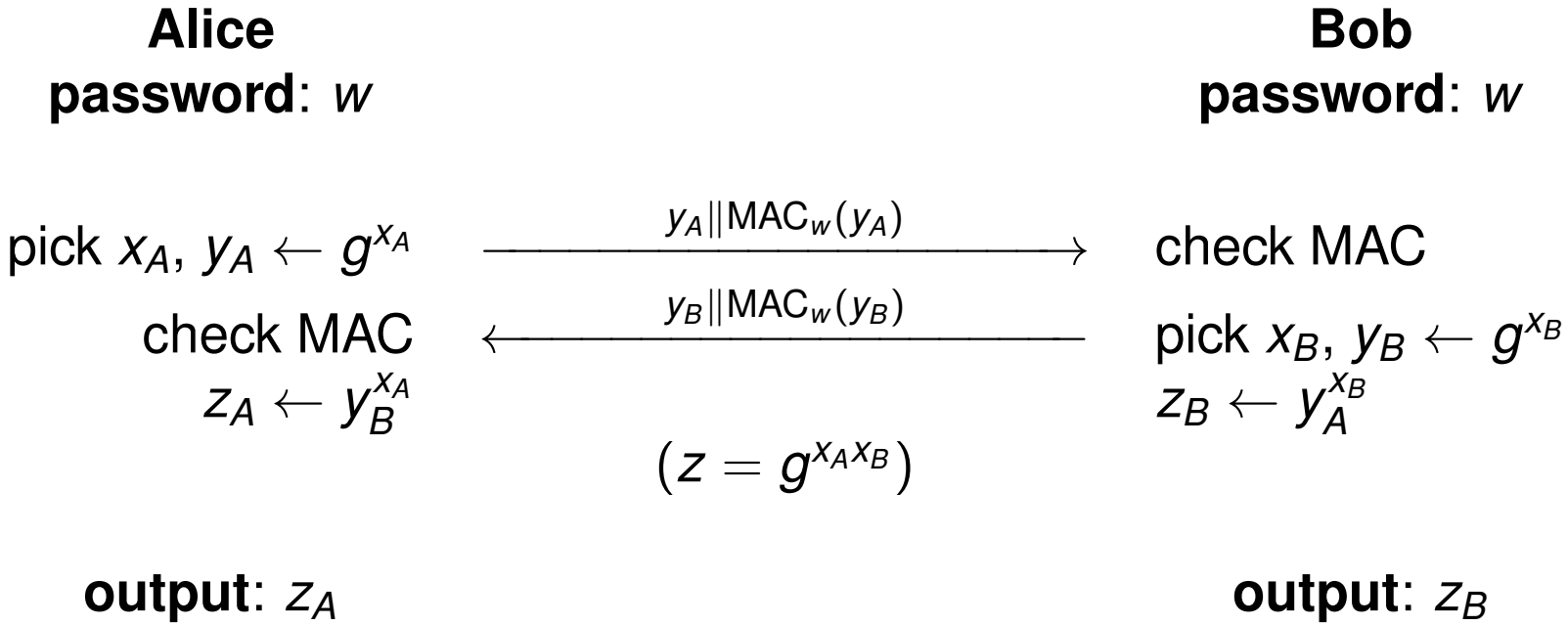
$\downarrow \quad \downarrow$

$K \quad = \quad K$

security

confidentiality of w, K

Key Agreement: a (Bad) Idea



subject to offline exhaustive search

Key Agreement: Another (Bad) Idea

Alice **password**: w		**Bob** **password**: w
$\text{RSA.Gen} \to (N, e, d)$	$\xrightarrow{N,e}$	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 is later revealed, offline exhaustive search possible
- partition attack: eliminate all $\hat{w}$ such that $\text{Dec}_{\hat{w}}(c) \geq N$

SPAKE2

Abdalla, Pointcheval [CT-RSA 2005]

Alice **password**: w		**Bob** **password**: w
pick $x \in \mathbf{Z}_q^*$, $X \leftarrow g^x$		pick $y \in \mathbf{Z}_q^*$, $Y \leftarrow g^y$
$X^* \leftarrow X g_2^w$	$\xrightarrow{X^*}$	$Y^* \leftarrow Y g_3^w$
$K_A \leftarrow (Y^* / g_3^w)^x$	$\xleftarrow{Y^*}$	$K_B \leftarrow (X^* / g_2^w)^y$
$K'_A \leftarrow \mathrm{KDF}(A, B, X^*, Y^*, w, K_A)$		$K'_B \leftarrow \mathrm{KDF}(A, B, X^*, Y^*, w, K_B)$
output: K'_A		**output**: K'_B

(public parameters: q prime, g, g_2, g_3 generators of the same order-q group

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.

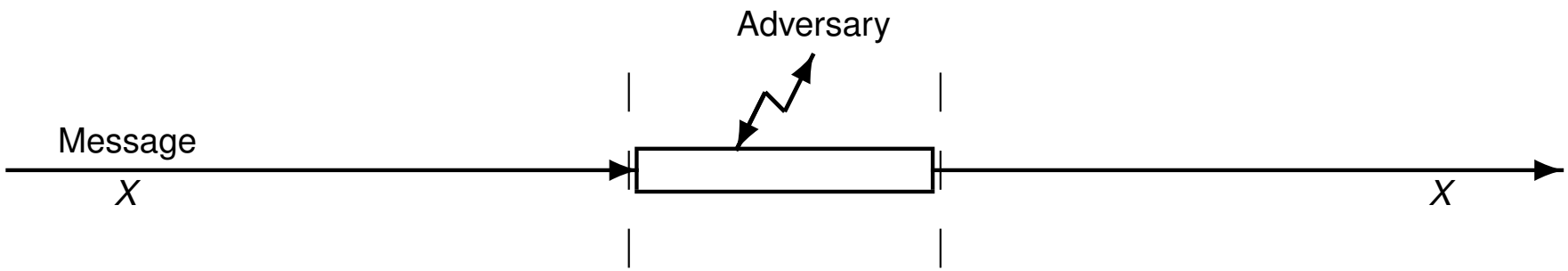
Case Study: The Biometric Passport

case study

8 Trust Establishment

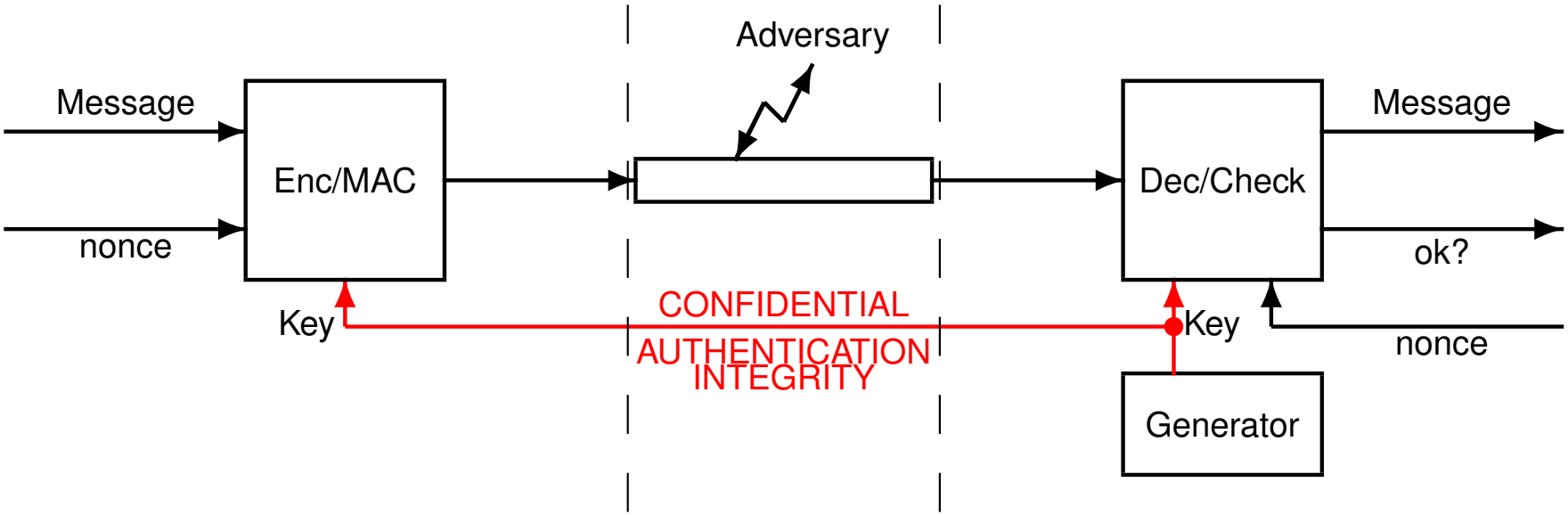
- Access Control
- Password-Based Cryptography
- From Secure Channel to Secure Communications
- Setup of Secure Channels
- Setup by Narrowband Secure Channel
- Setup by a Trusted Third Party
- Trust Management and Cryptography

The Cryptographic Trilogy

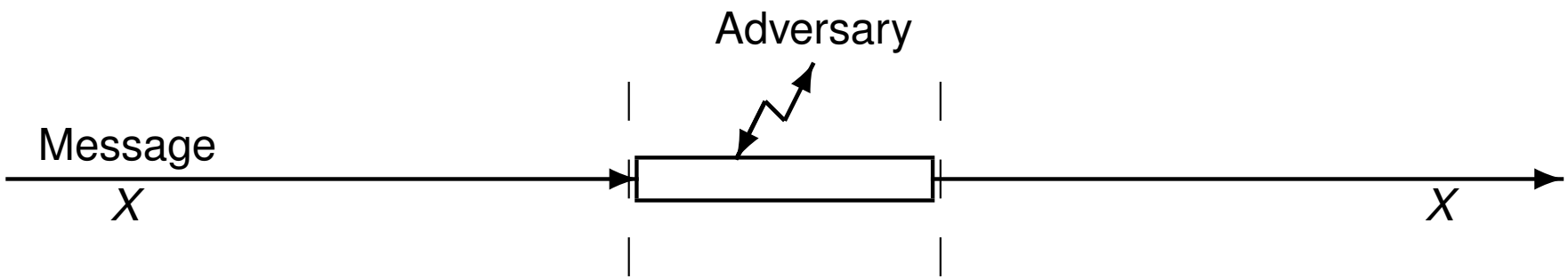


- **Confidentiality** (C): only the legitimate receiver can get X
- **Authentication** + **Integrity** (A+I): only the legitimate sender can insert X and the received message must be equal to X

A+I+C by Symmetric Cryptography

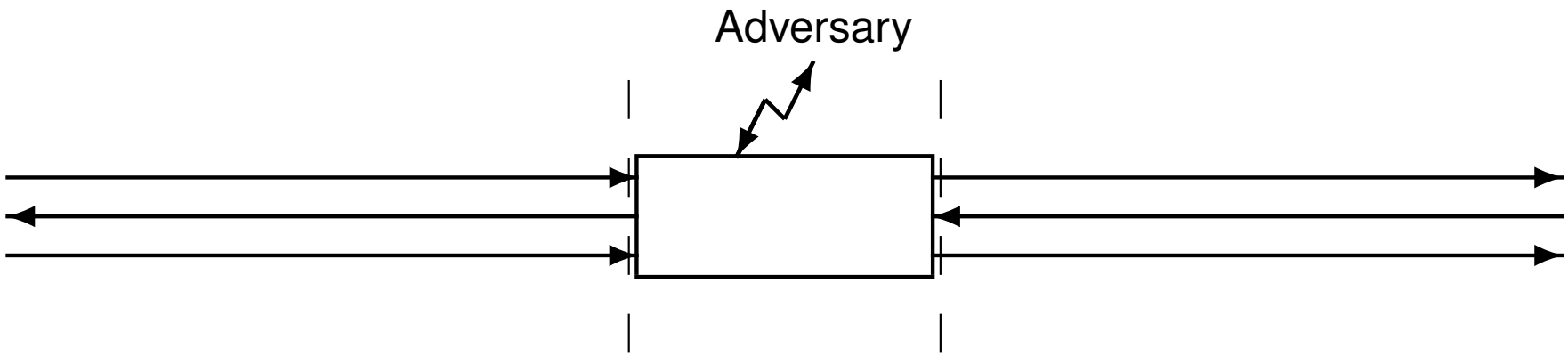


Security Property of Communication Channels



- **Confidentiality**, **Authentication**, **Integrity**
- **Freshness**: the received X was not received before (no replay)
- **Liveliness**: a sent message X is eventually delivered (no loss)
- **Timeliness**: ($>$ liveliness) time of delivery is upper bounded
- **Deniability**: no evidence of sending a message leaks
- **Non-repudiation**: sender cannot deny his sent messages
- **Forward secrecy**: secrecy even when states leak in the future
- **Postcompromise security**: healed secrecy even after leakage

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!
  (anonymity: cannot identify sender or receiver)
  (unlinkability: 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 $X_1, \ldots, X_t$, X_t coming in, then the other participant has seen a message sequence whose first t messages are $X_1, \ldots, X_t$
  ☺: 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 liveliness is not guaranteed

Sequentiality using A + I Message Security

common method:

- common method: authenticate a sequence number in packets and check that received packets have consecutive sequence numbers
- old TLS example: $Y = \mathrm{Enc}(X \| \mathrm{MAC}(\mathrm{seq} \| X))$
  where seq is implicit
- modern TLS example: $Y = \mathrm{AE.Enc}(\underbrace{\mathrm{seq}}_{\mathrm{ad}}, X)$

  where seq is implicit
  (authenticated encryption with associated data)

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

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 ?
all	privacy	?

8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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

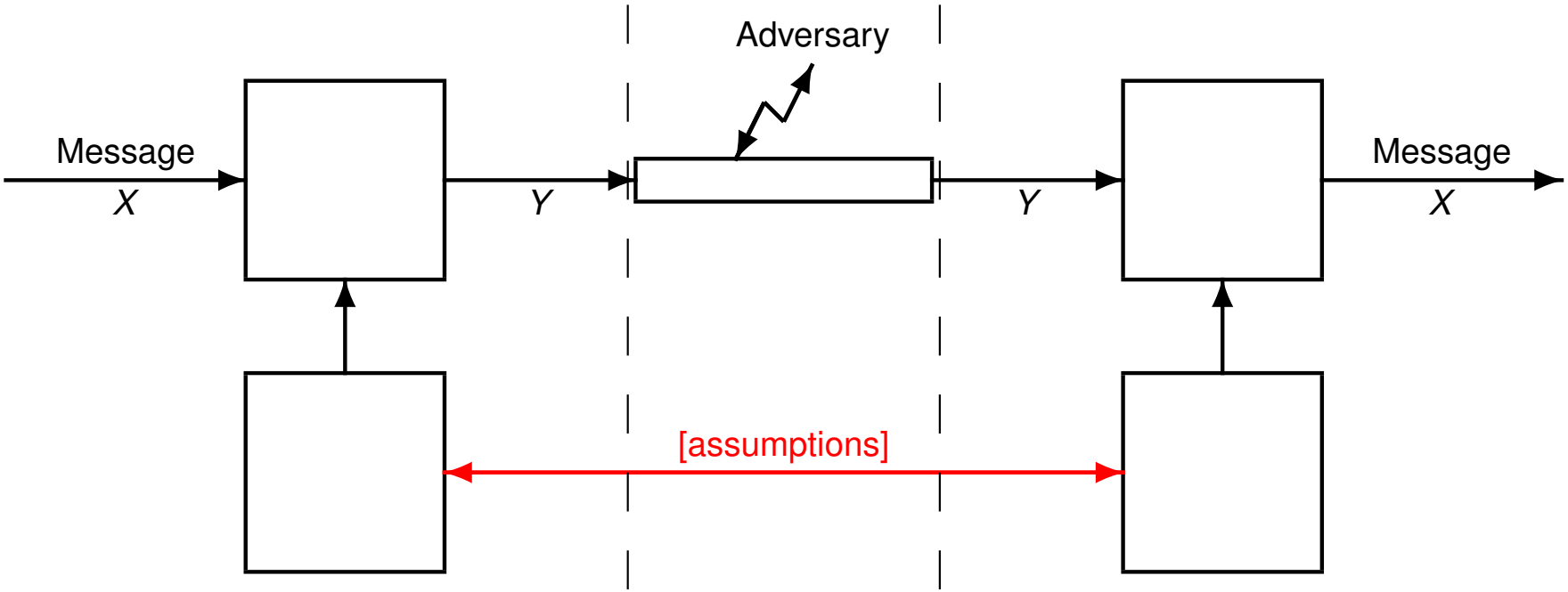
Problem

Q: How to setup a secure channel over an insecure channel?

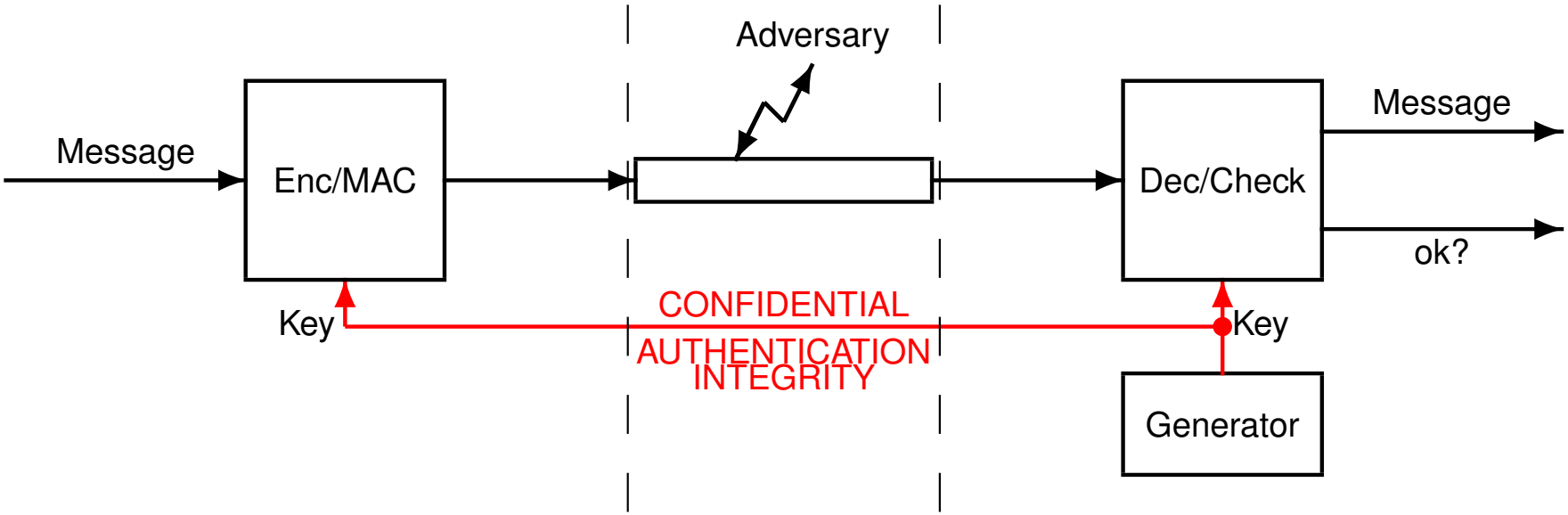
A: hfr n frpher punaary

▸ ROT13

Virtual Channels by Combination of Channels



Secure Channel from A+I+C Channel: PSK



PSK: PreShared Key

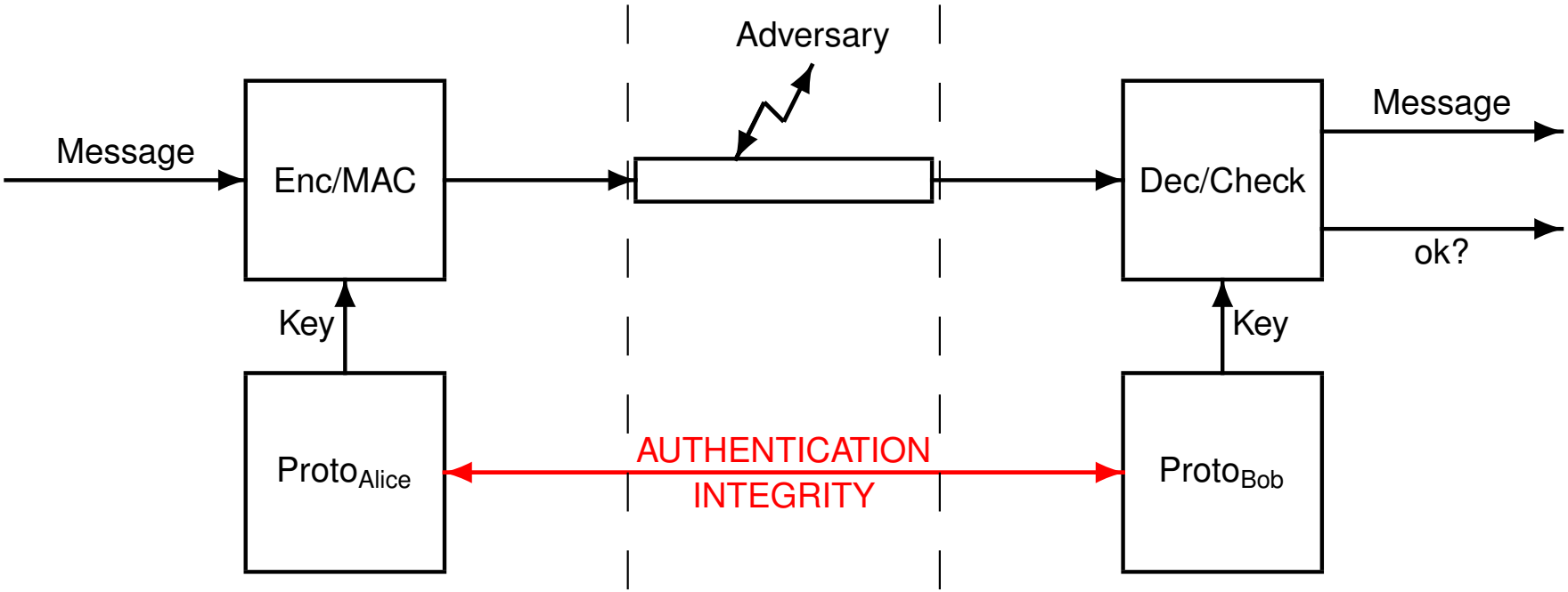
Next Step: Strongly Secure Channel From Weakly Secure Channel

Q: How to relax security properties at setup?

A: hfr choyvp-xrl pelcgbtencul

▶ ROT13

... with A+I Channel: Key Agreement Protocol



Passive vs Active Adversaries

- **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
- **active adversary:** can interfere with communication (modify messages, insert messages, replay messages)
  e.g. man-in-the-middle attack
  The Diffie-Hellman protocol requires A+I channel to protect against it

Approaches to Build an Initial Authenticated Channel

- **using really secure initial channel**
  setup cable, Near Field Comm. (see Bluetooth simple pairing)
- **by user monitoring**
  (see next)
- **using a trusted third party**
  examples: secure token, key server, certificate authority
  (see later)

Summary

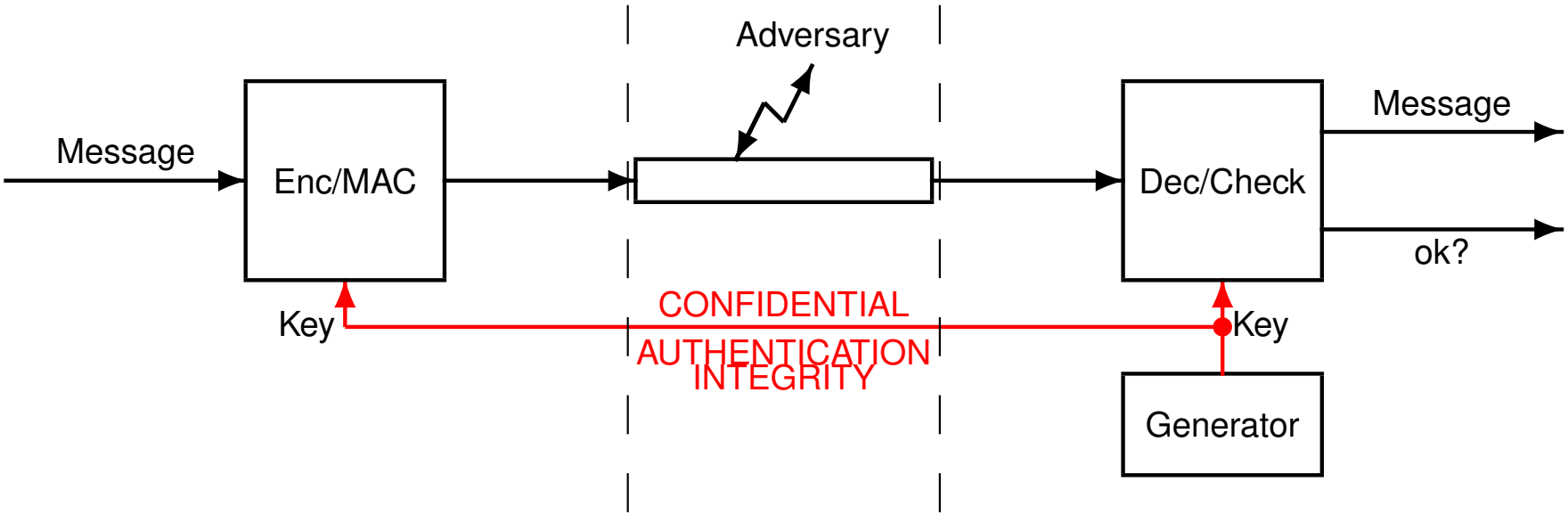
- we need specific means to A+I-securely transmit a public key
- we agree on a master key using public key cryptography
- we use conventional cryptography to set up secure channels
  - we derive several symmetric keys using key derivation functions
  - we use symmetric encryption and MAC
- we must live with the fear that termination may be unfair

8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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

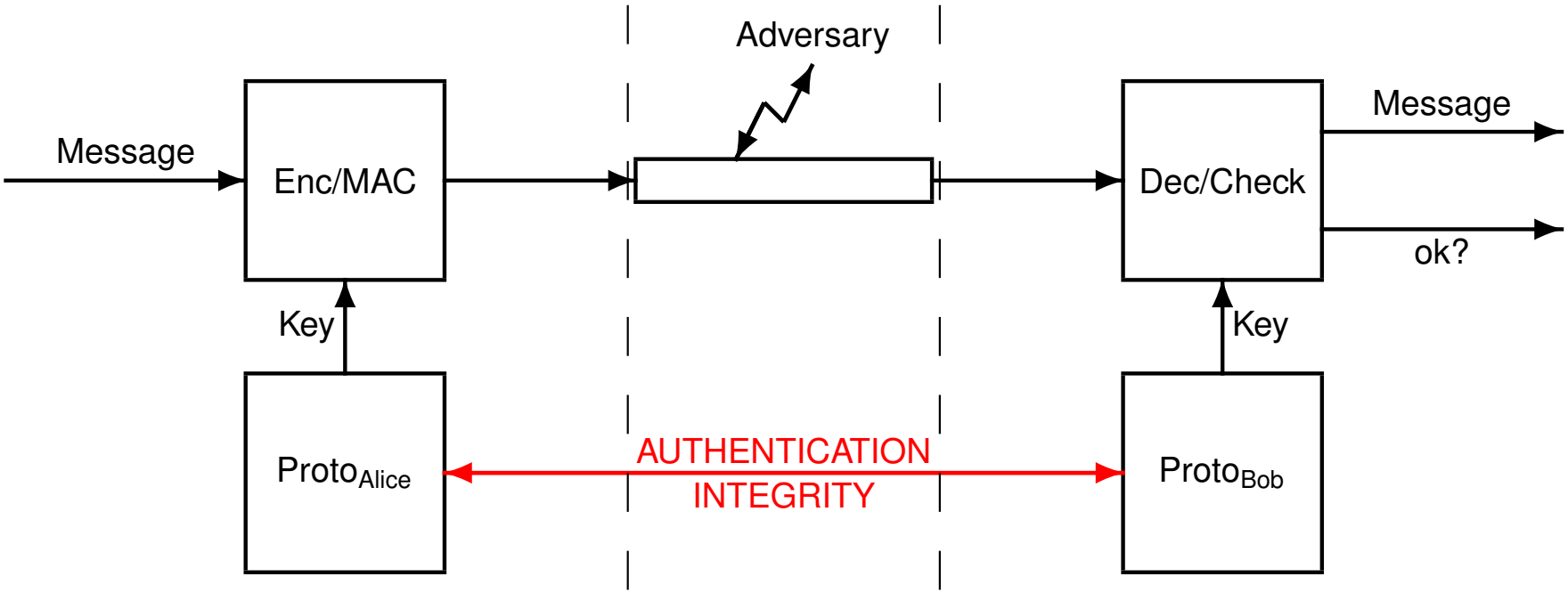
Secure Communication Step 1

Conventional Cryptography



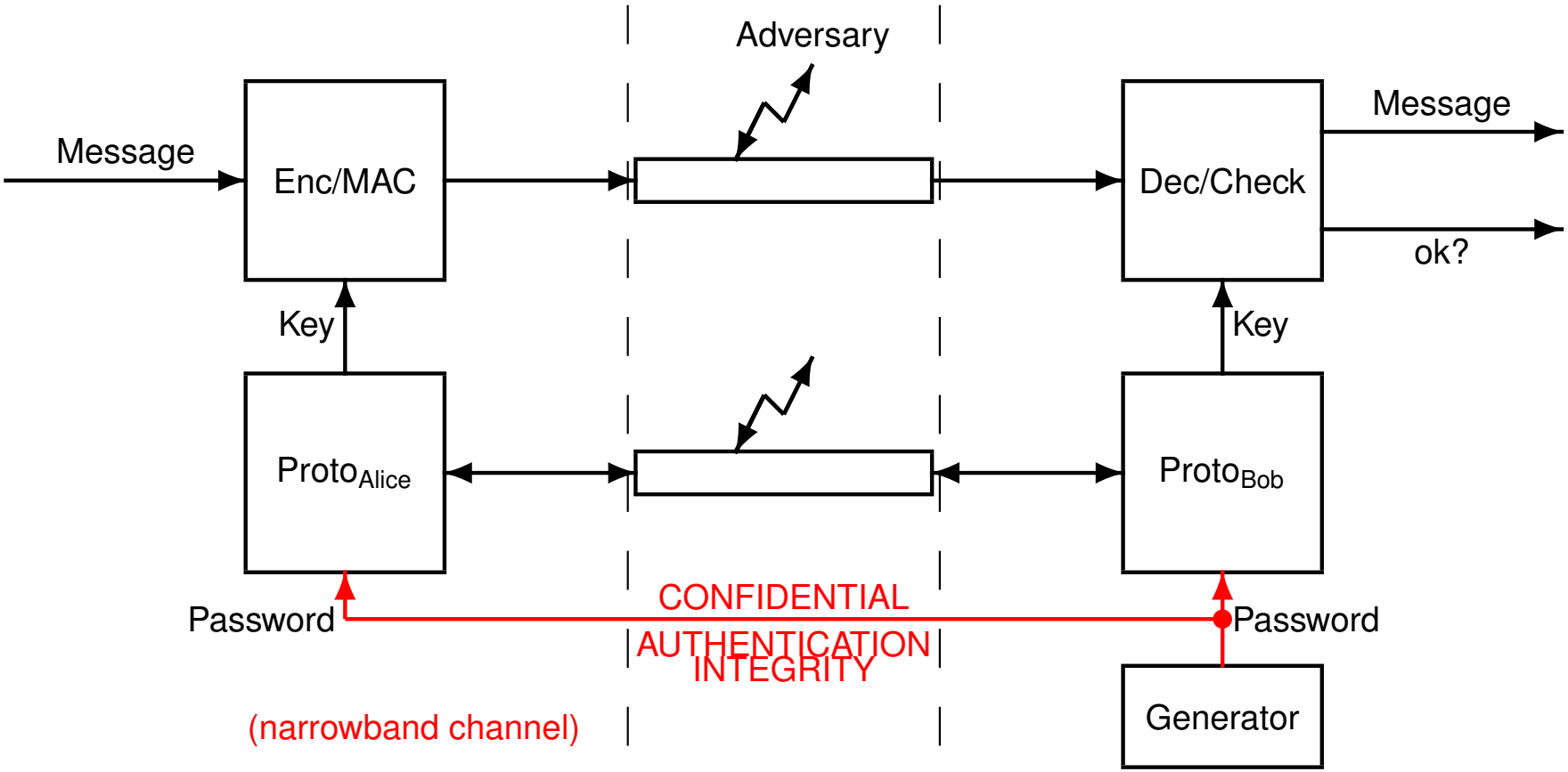
Secure Communication Step 2

Public-Key Cryptography



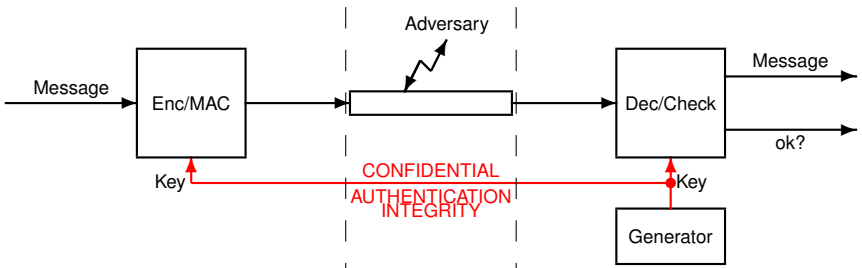
Secure Communication Step 3

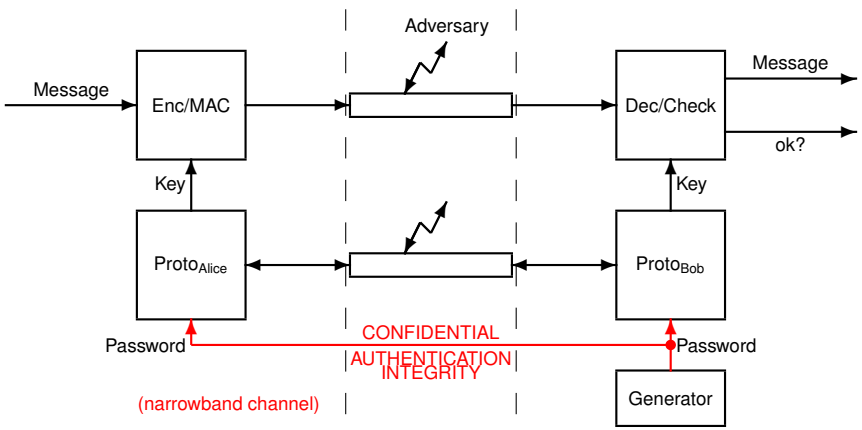
Password-Based Cryptography



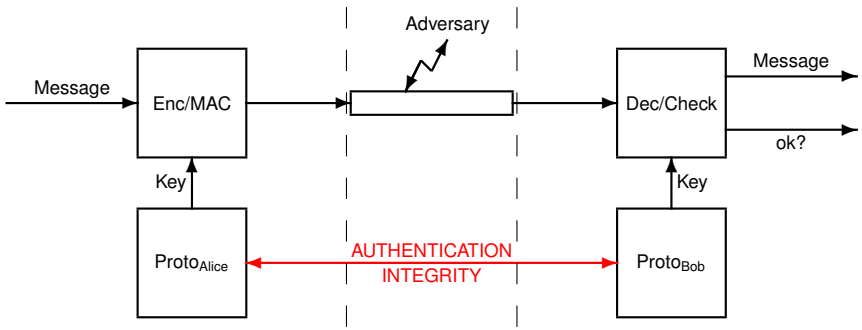
Secure Communication

with confidential channel



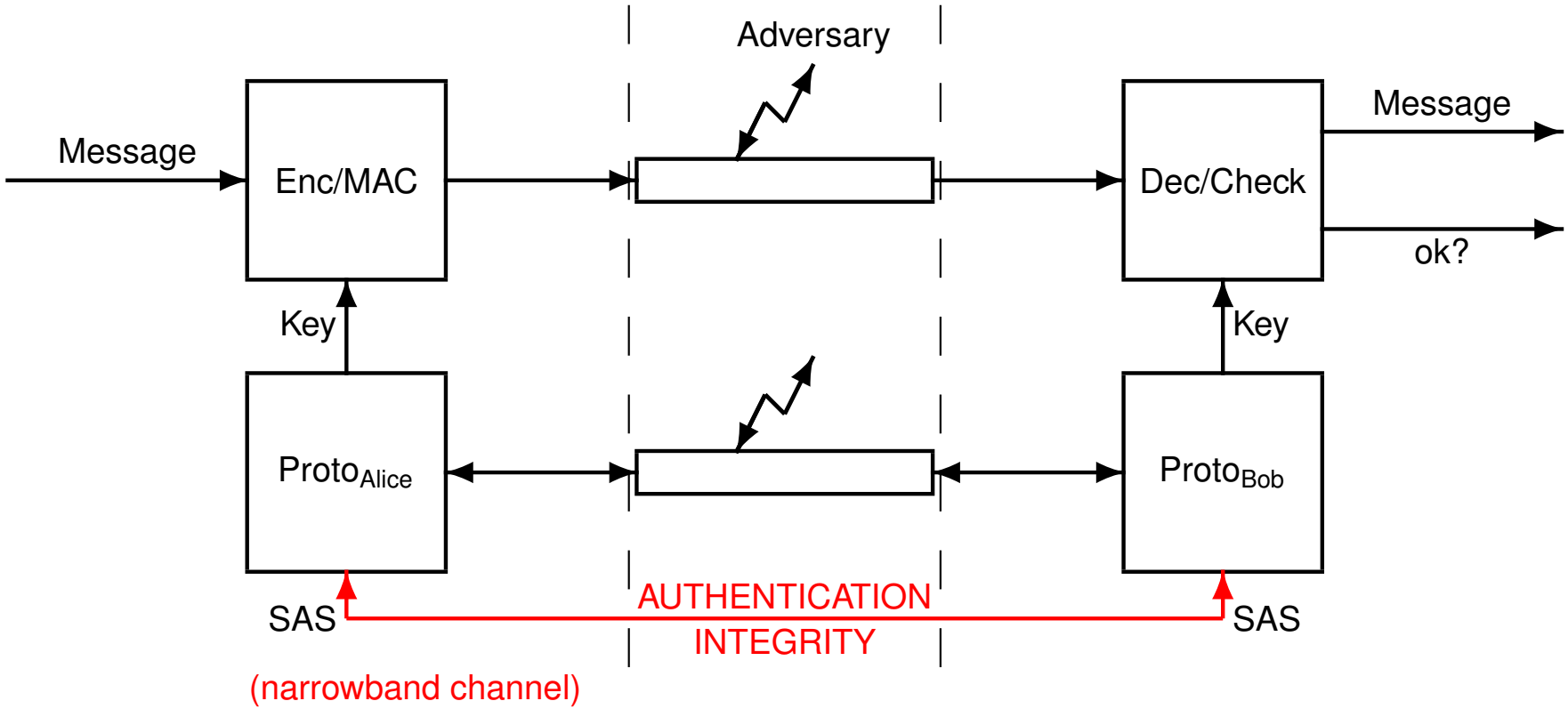


without confidential channel

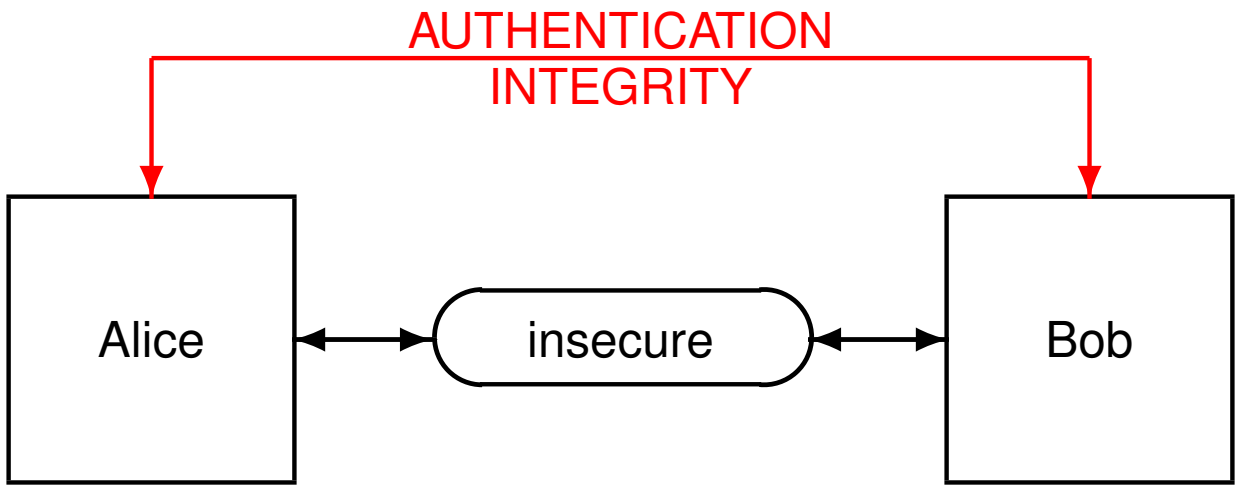


Secure Communication Step 4

Cryptography Based on Short Authenticated Strings



Security from Human-Monitored Short String Authentication

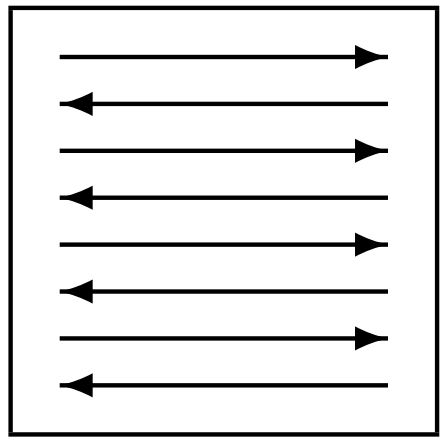


- communication over a cheap/efficient but insecure channel
- security set up with the help of a short authenticated string (SAS)
- authentication based on human monitoring

Message Authentication Protocols

Alice
input: m

Bob



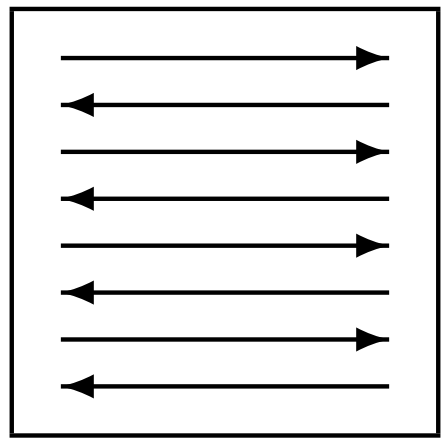
output: $\hat{m}$

- functionality: $\hat{m} = m$
- security: if $\hat{m} \neq \perp$, then Alice has run the protocol with $m = \hat{m}$
- 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
input: m_A

Bob
input: m_B

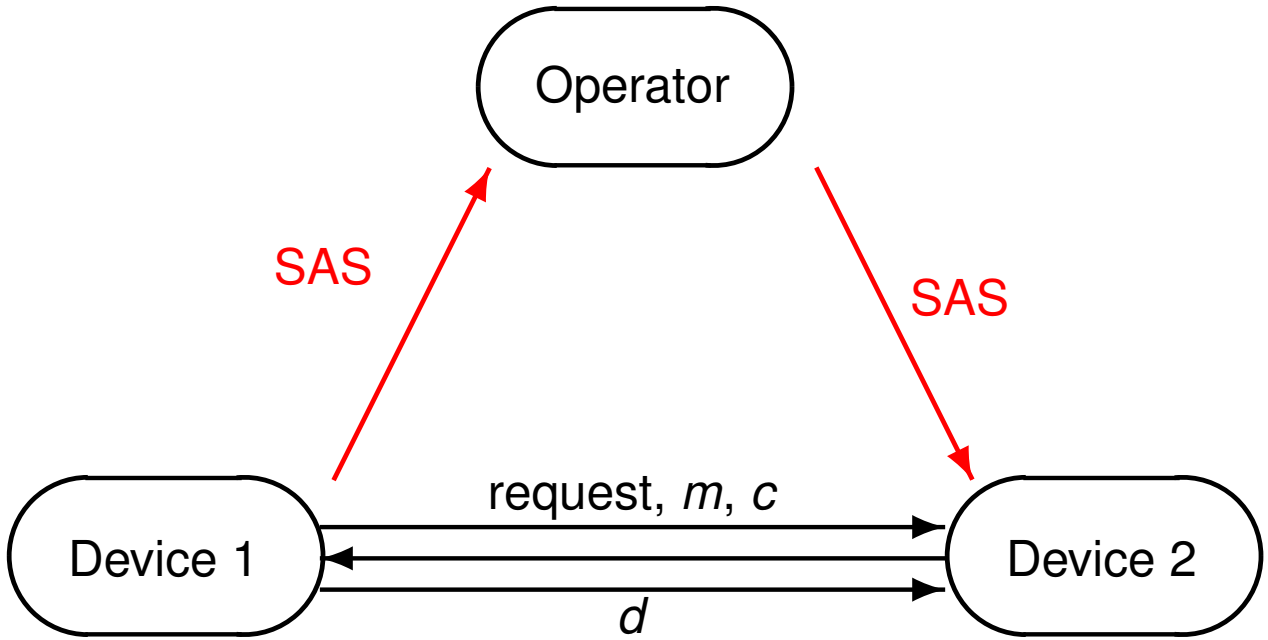


output: m_B

output: m_A

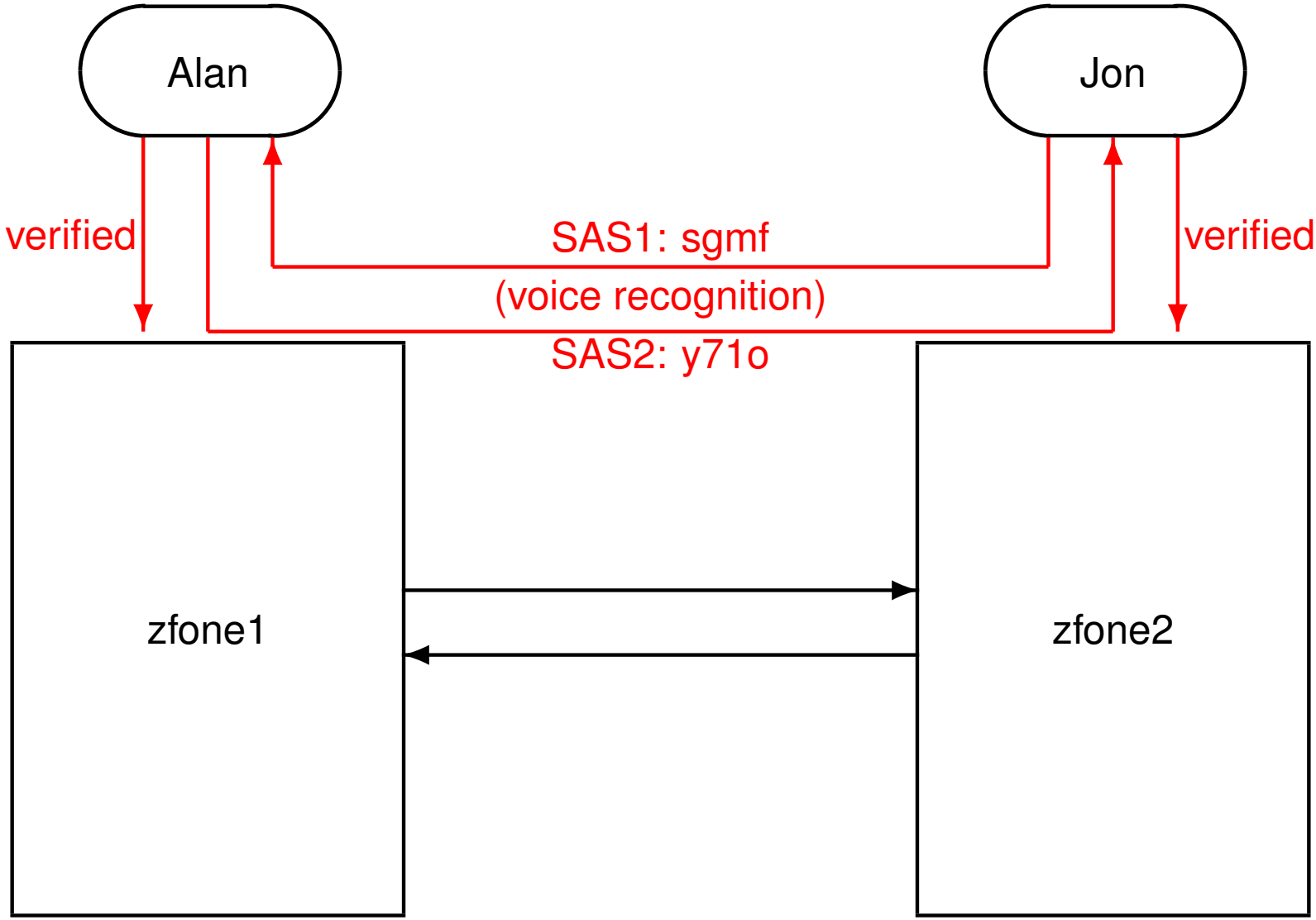
- two message authentication protocols at the same time
- application: authenticated key agreement
  (m_A and m_B are Diffie-Hellman public keys)

Application I: Personal Area Network Setup

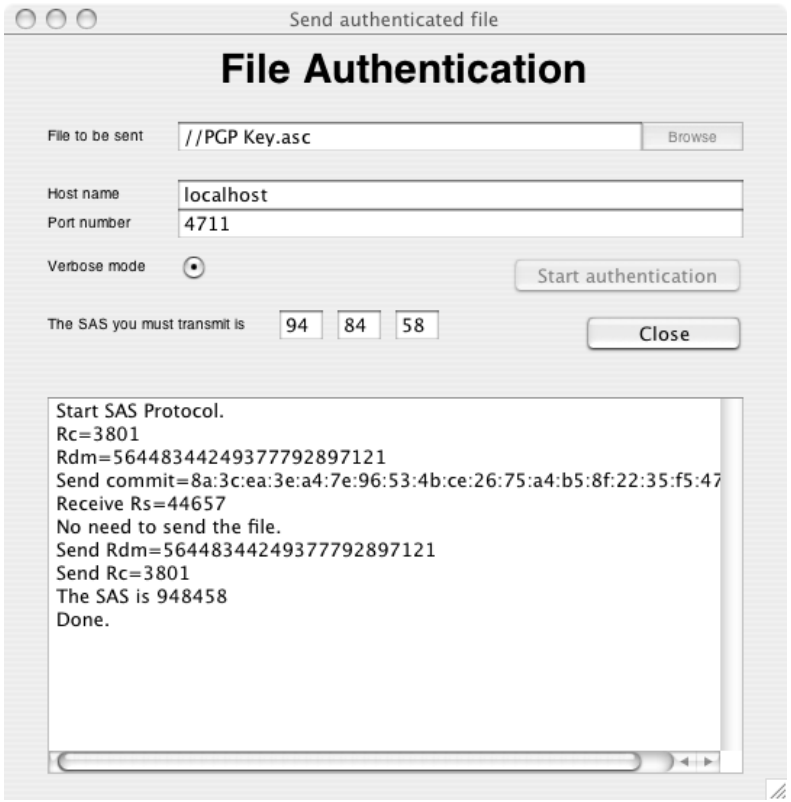


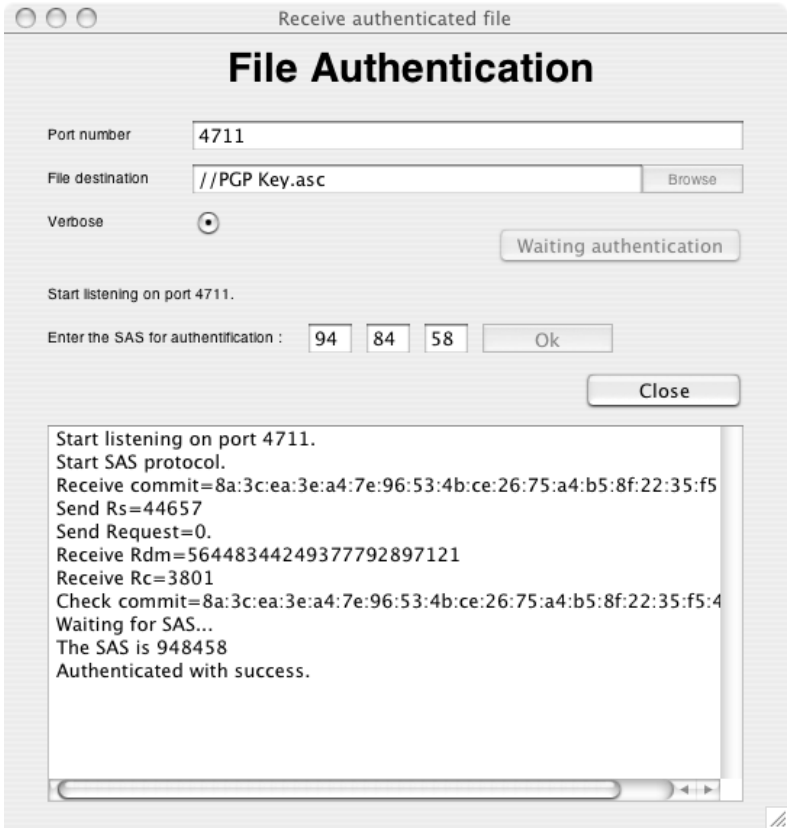
Application II: Voice over IP

Existing Standard: ZRTP



Application III: Peer-to-Peer PGP Channel Setup

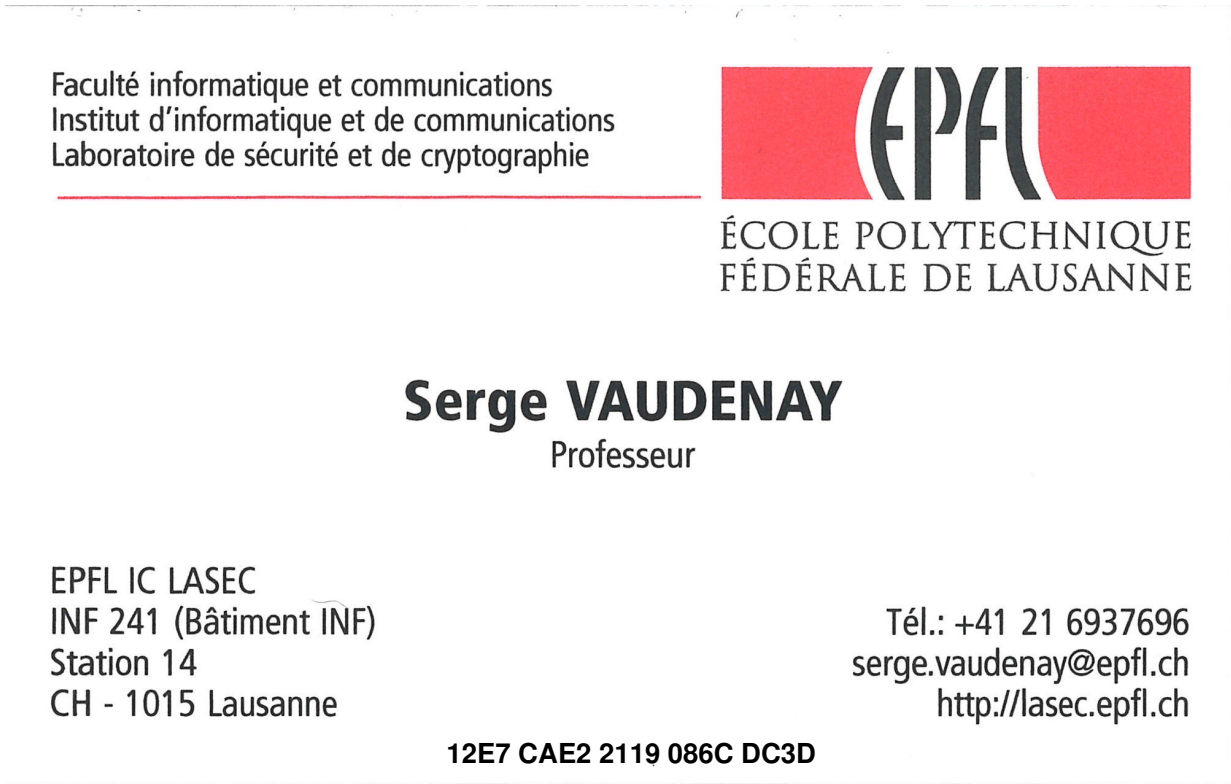




Application IV: Disaster Recovery

- on the road, after a key loss (computer crash, stolen laptop)
  $\longrightarrow$ set up of a security association
- PKI collapse (company bankrupt, main key sold, act of God)
  $\longrightarrow$ set up of a security association

Semi-Authenticated Non-Interactive: Application



Faculté informatique et communications
Institut d'informatique et de communications
Laboratoire de sécurité et de cryptographie

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Serge VAUDENAY
Professeur

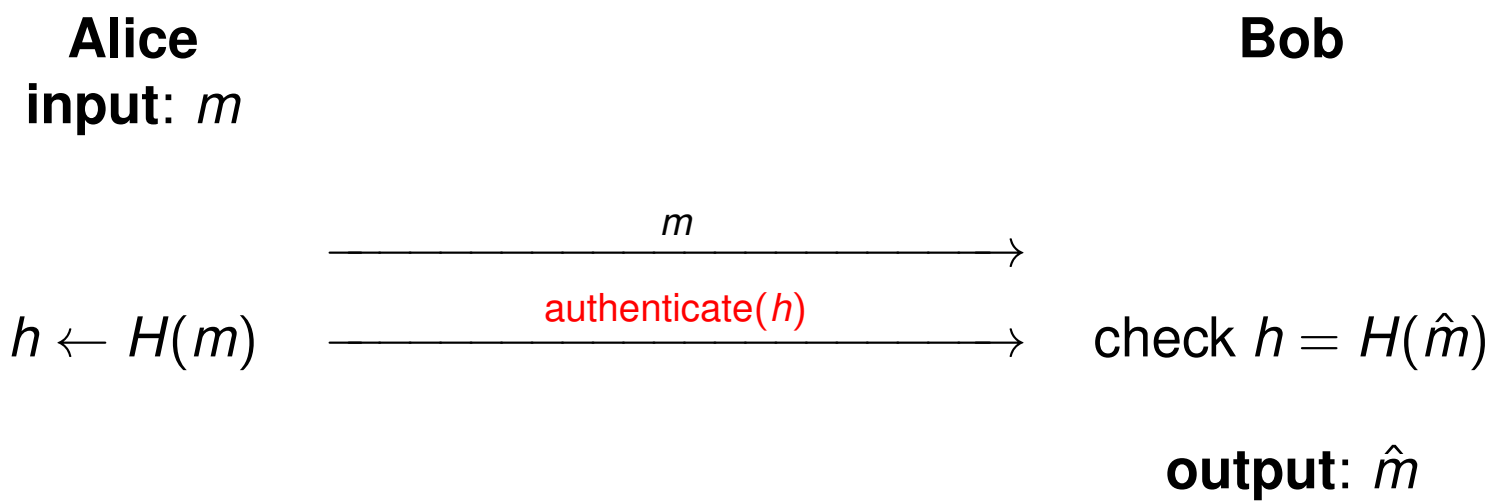
EPFL IC LASEC
INF 241 (Bâtiment INF)
Station 14
CH - 1015 Lausanne

Tél.: +41 21 6937696
serge.vaudenay@epfl.ch
http://lasec.epfl.ch

12E7 CAE2 2119 086C DC3D

Folklore

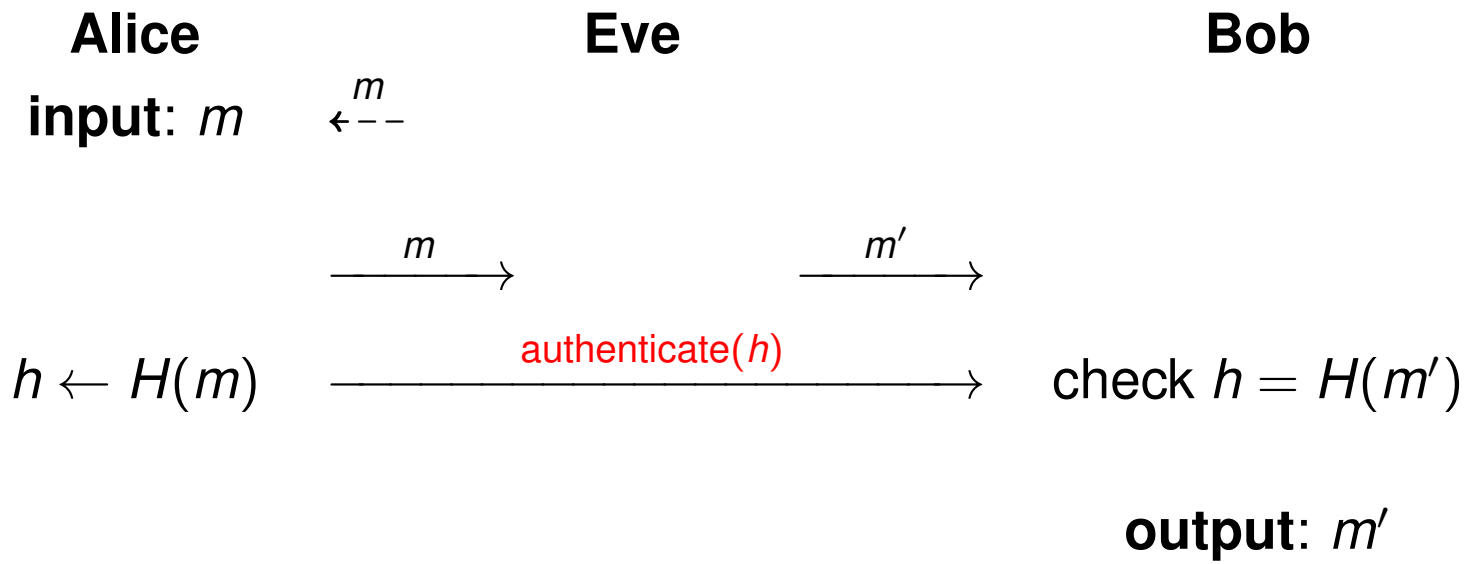
Protocol Balfanz-Smetters-Stewart-Chi Wong 2002



- ☺ efficient, provably security assuming collision resistance
- ☹ this requires SAS of at least 256 bits

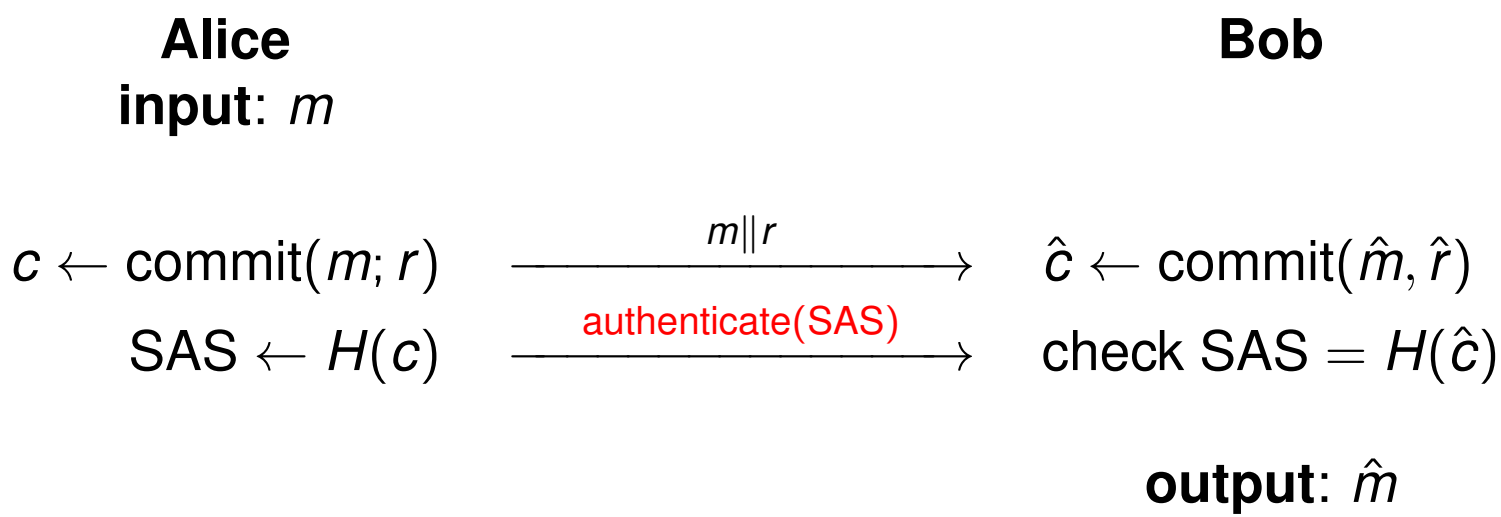
A Collision Attack

if SAS is so short that we can find collisions $H(m) = H(m')$, $m \neq m'$, make Alice run the protocol with m (chosen input attack) but change the message to Bob to m'



SAS-Based NIMAP

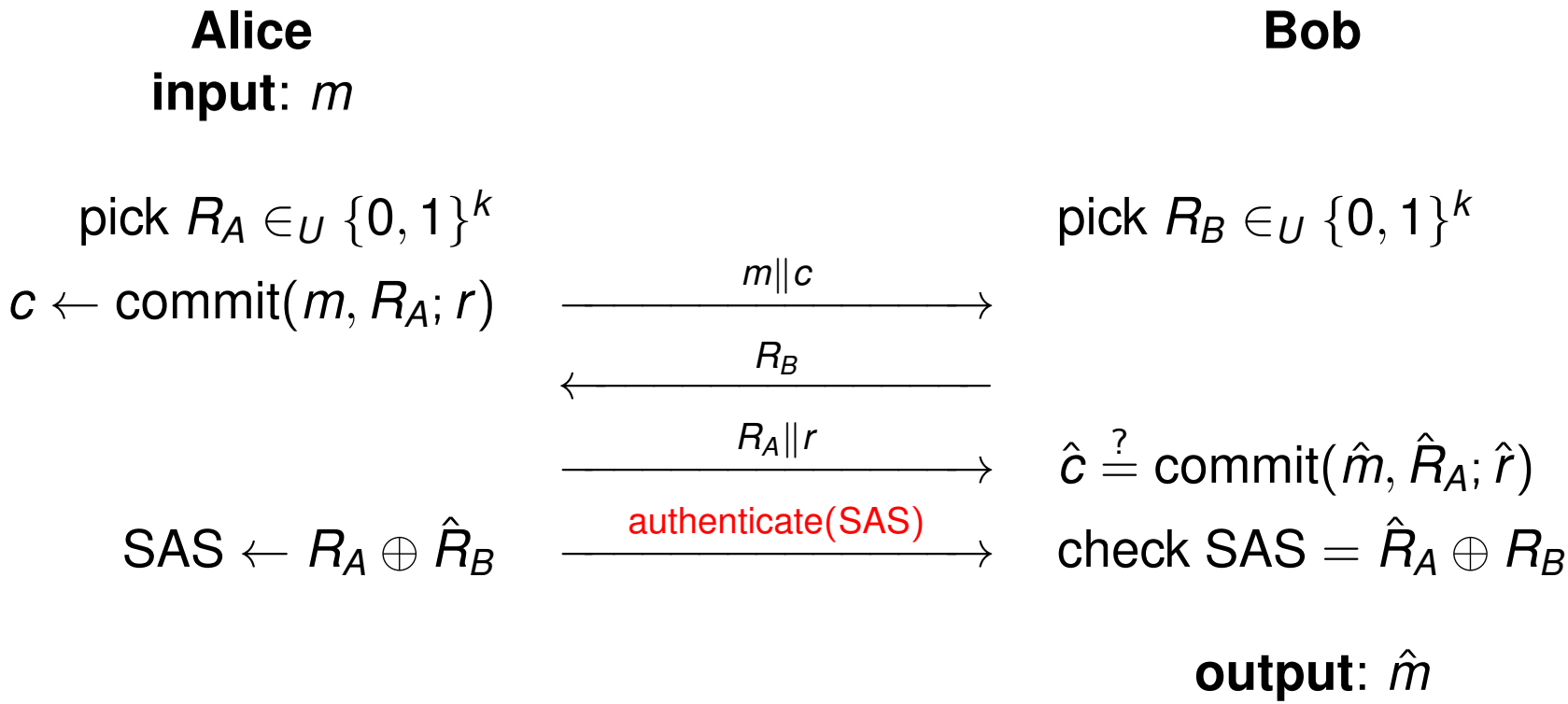
Pasini-Vaudenay 2006



- ☺ provable security, efficient
- ☺ can work with SAS of 128 bits (the least possible for NIMAP)

Semi-Authenticated Interactive

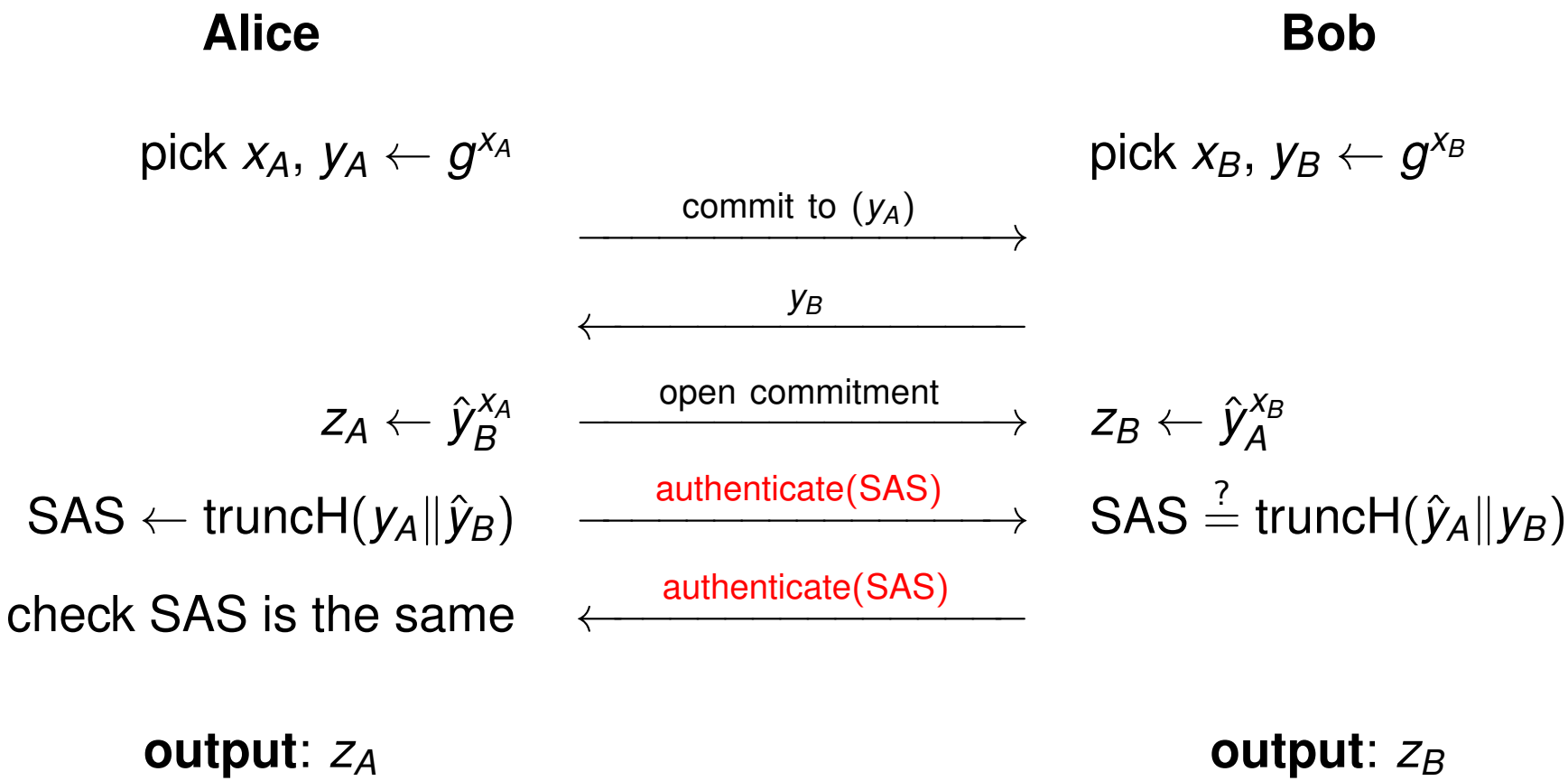
Vaudenay 2005



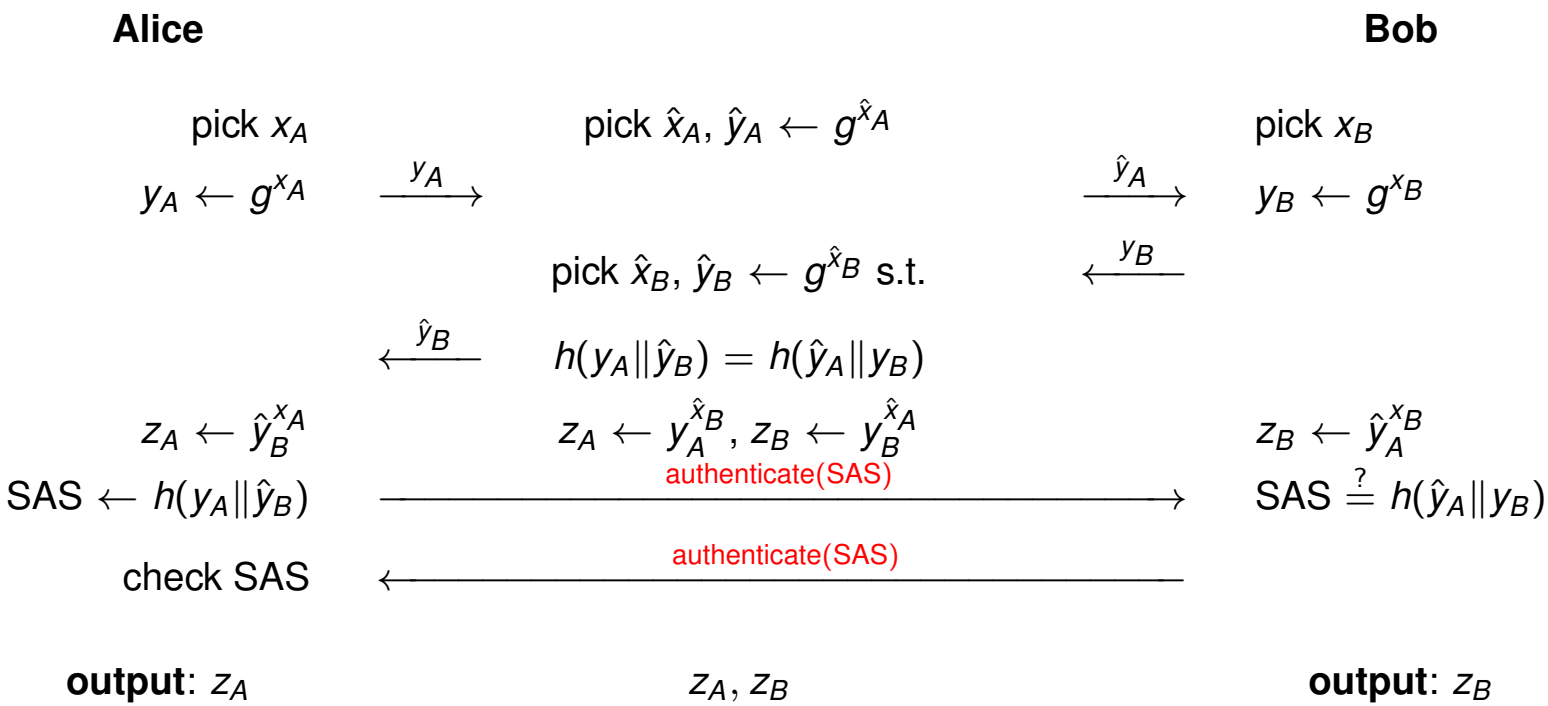
- ☺ provable security, efficient
- ☺ can work with SAS of 20 bits

Authenticated Interactive

Zimmermann 1995: PGPfone



Attack on a Variant Without Commitment



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. Gehrmann, 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.

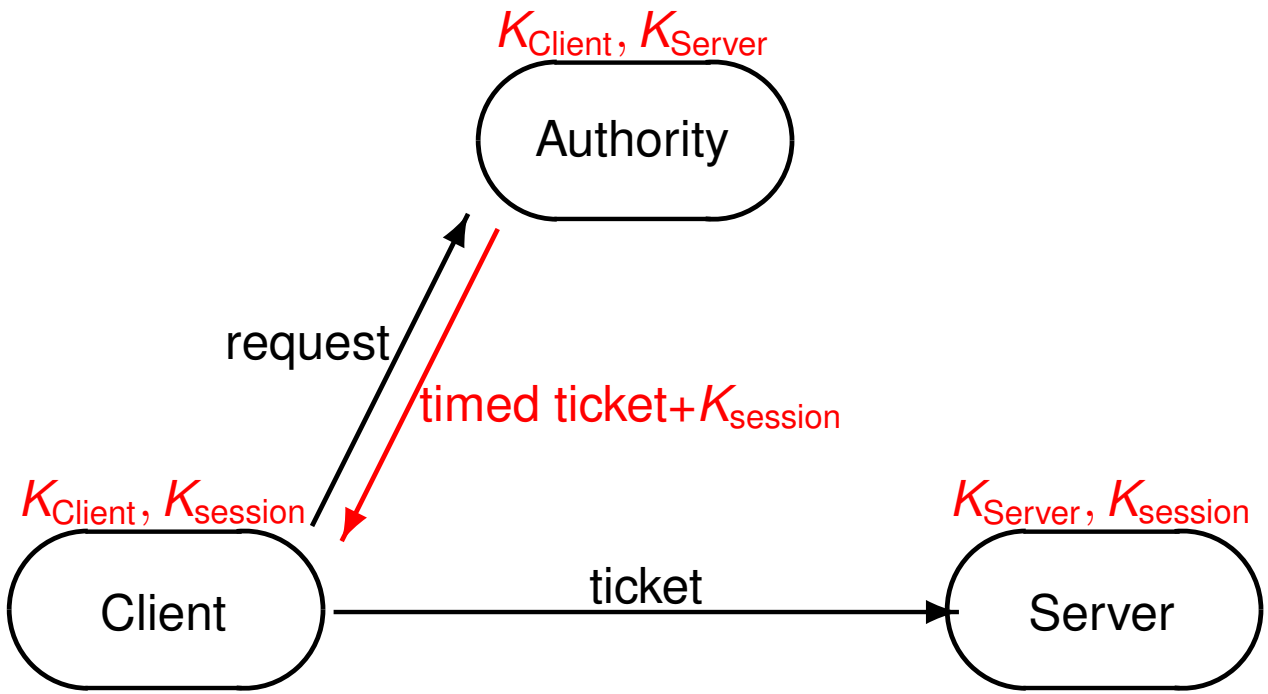
8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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+$K_{session}$ encrypted with K_{Client}
- ticket encrypted with K_{Server}

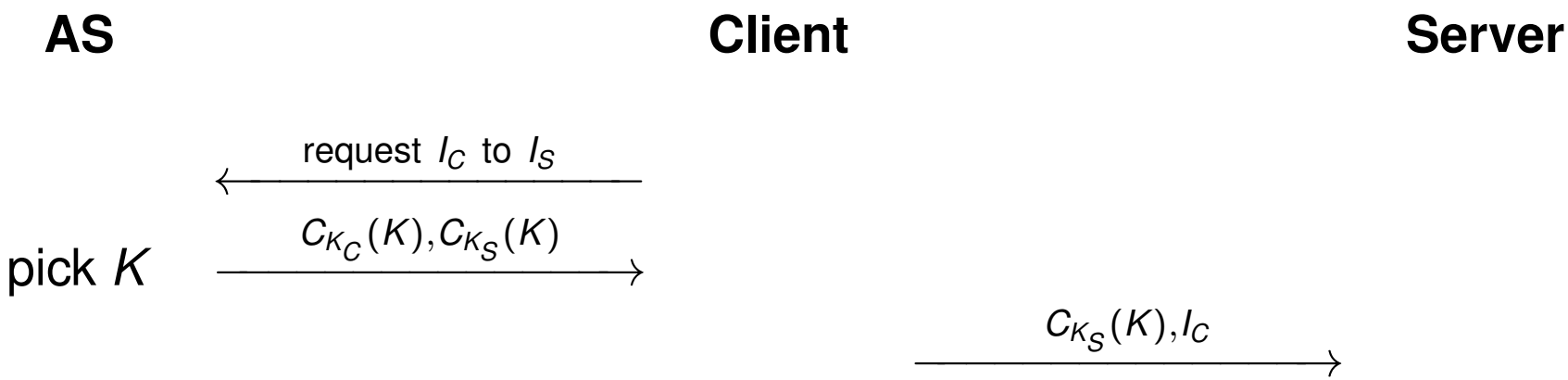
Kerberos

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

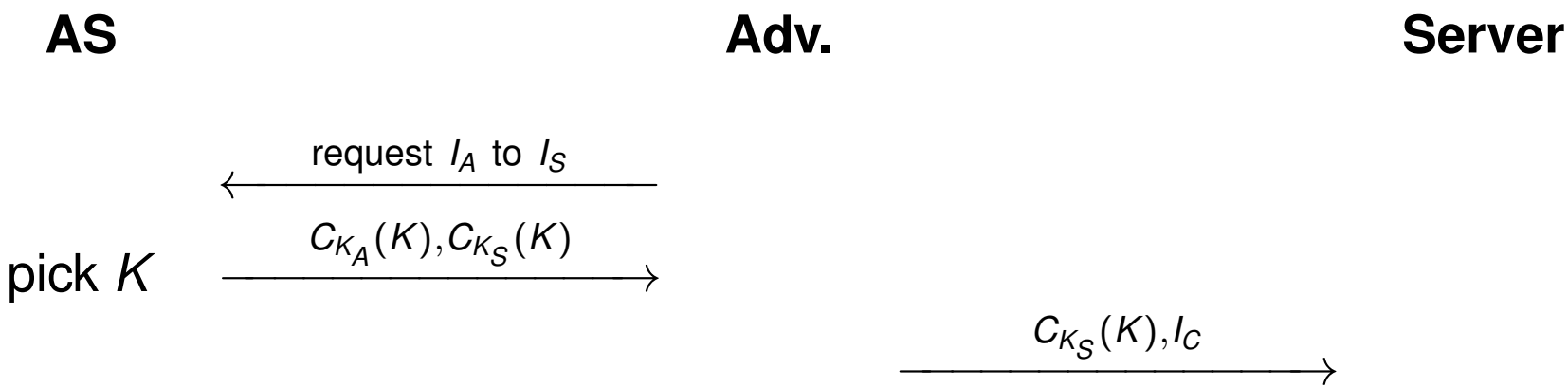
- Goal: to help I_C and I_S to share a session key K (and to help careless users to get privacy)

Server-Aided Authentication (Bad Protocol)



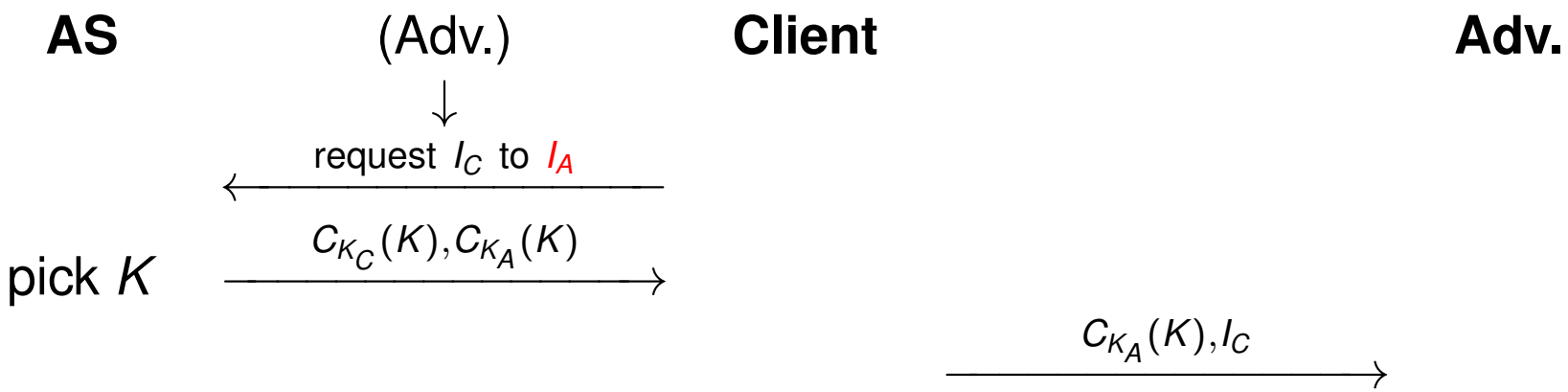
Problem: there is no authentication: an attacker can replace I_C or I_S

Attack



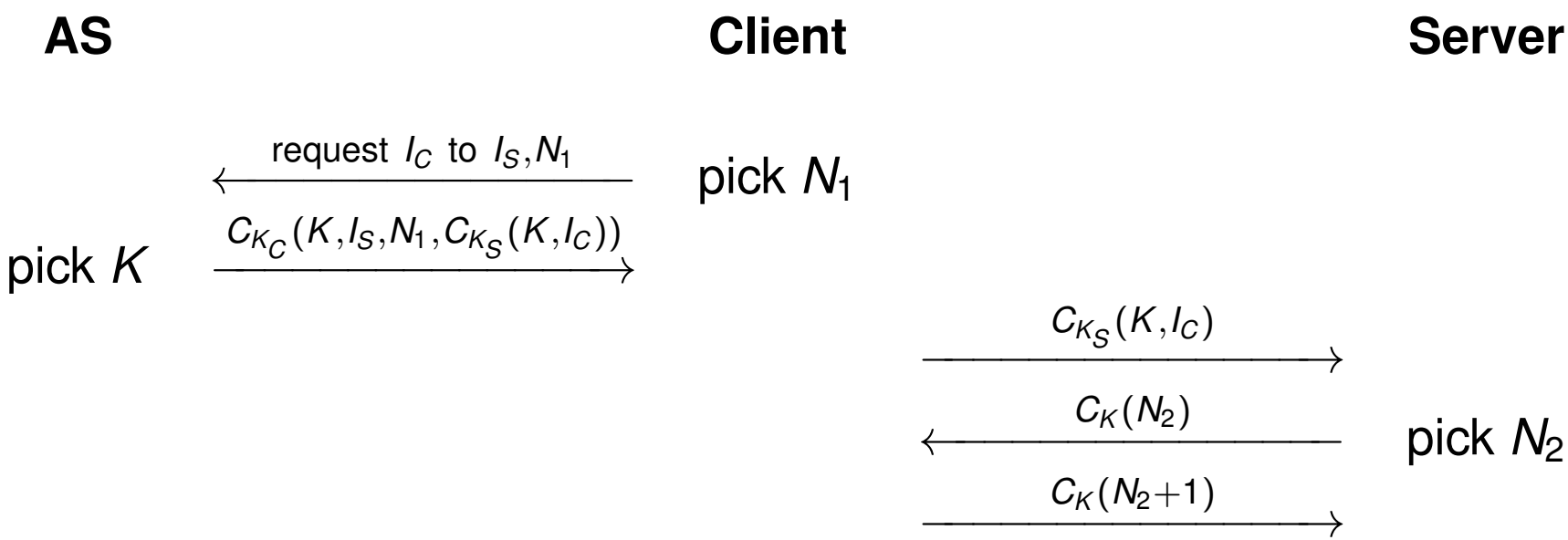
Server thinks he is talking to I_C!

Attack



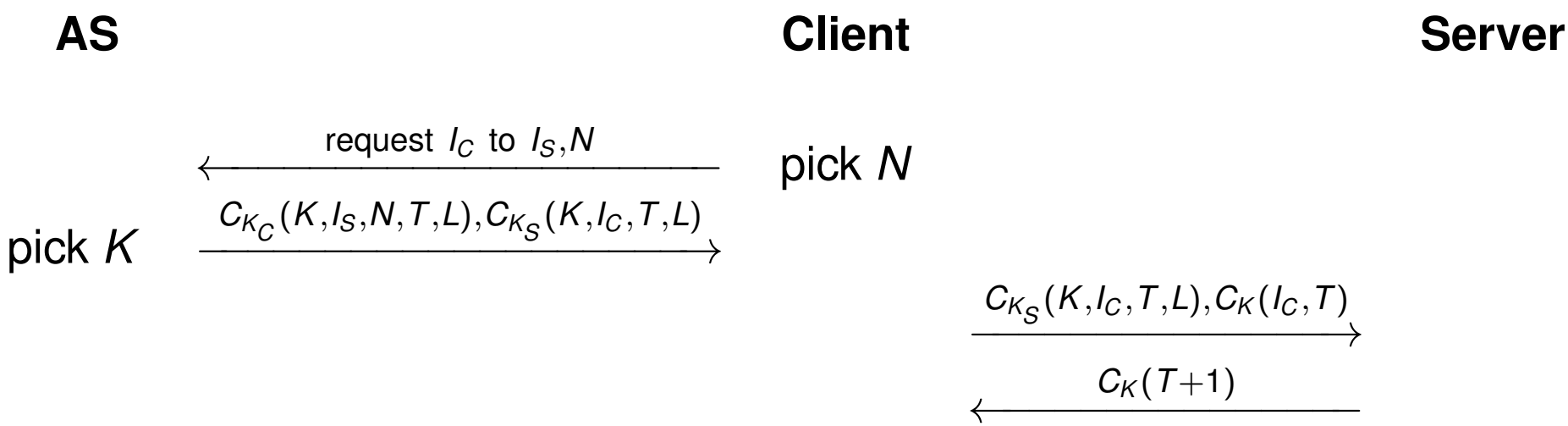
Client thinks he is talking to I_S!

Needham-Schroeder Authentication (Still Bad)



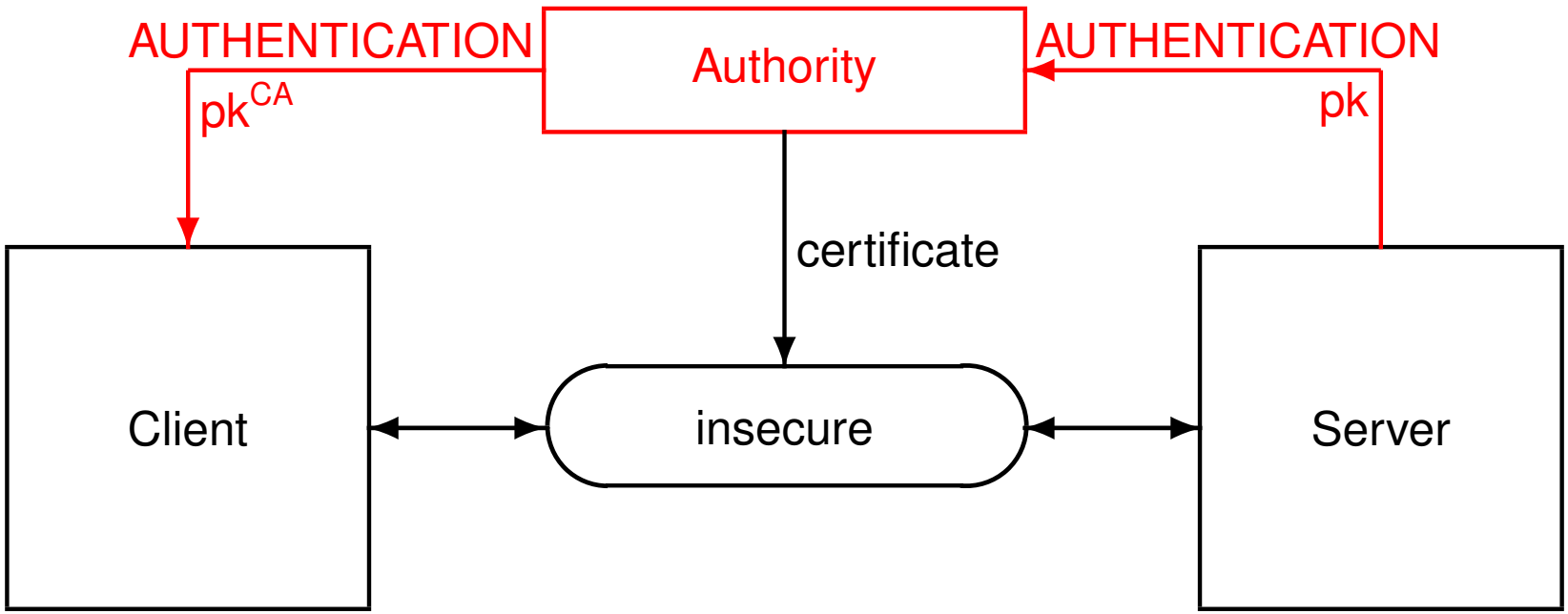
Problem: replay attack by impersonating C after K gets compromised

Basic Kerberos Protocol

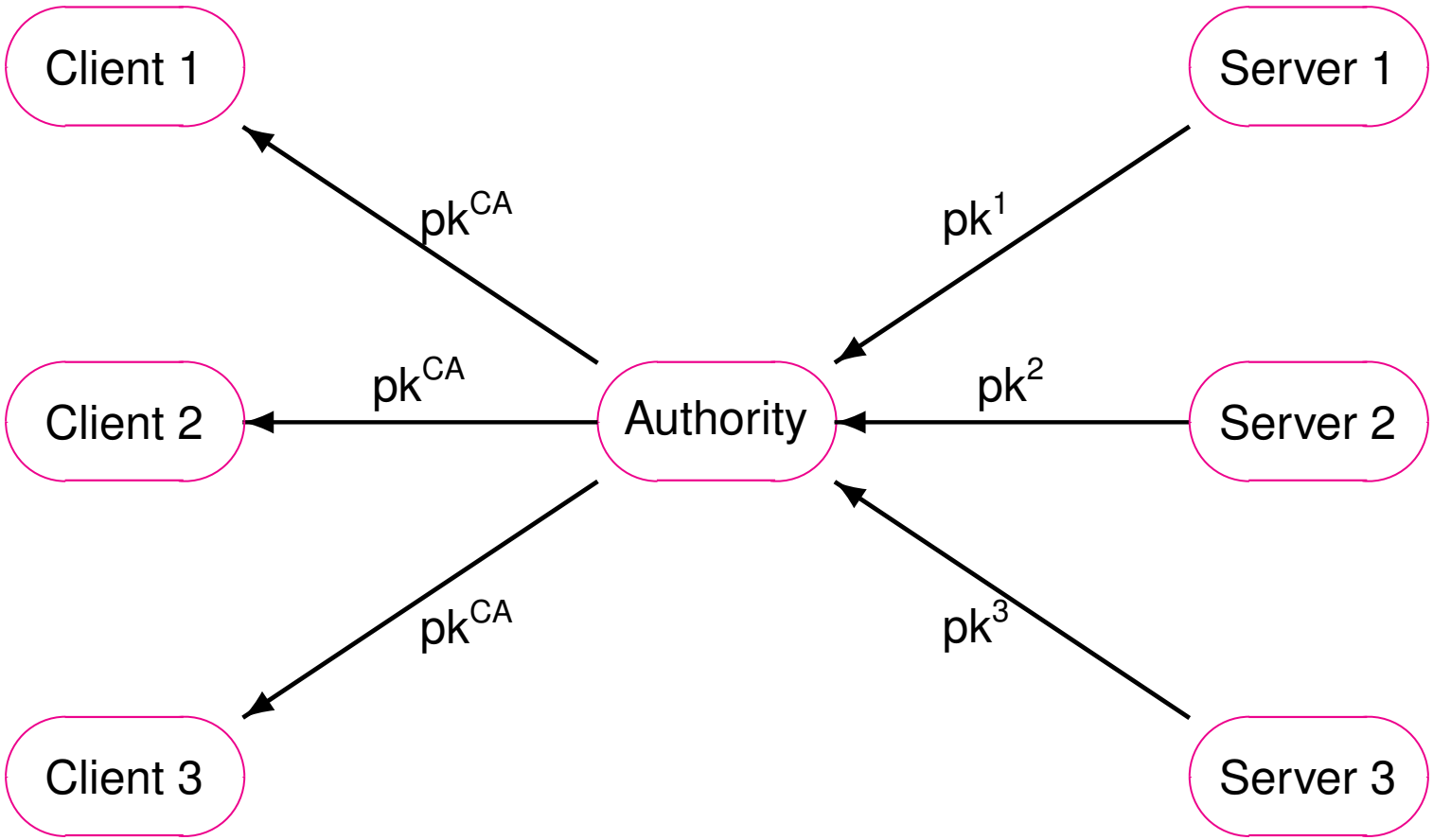


T: clock value; L: validity period

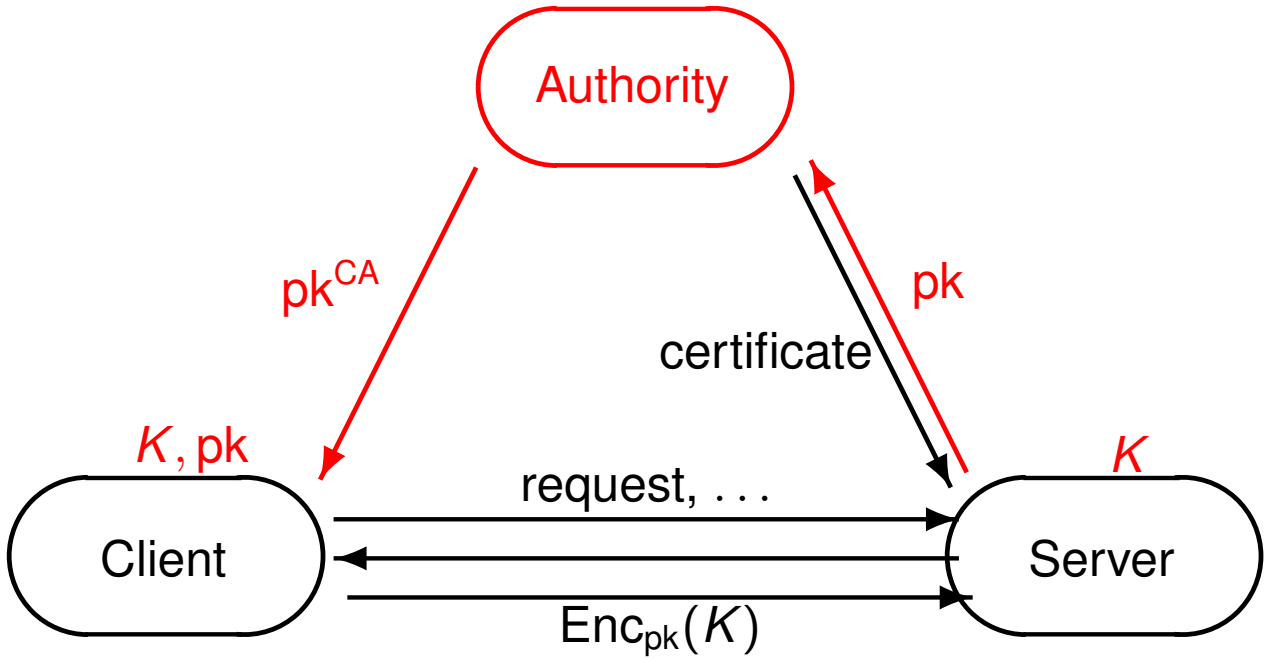
The Certificate Authority Model



Critical Secure Channels



Semi-A Key Exchange Using Certificates

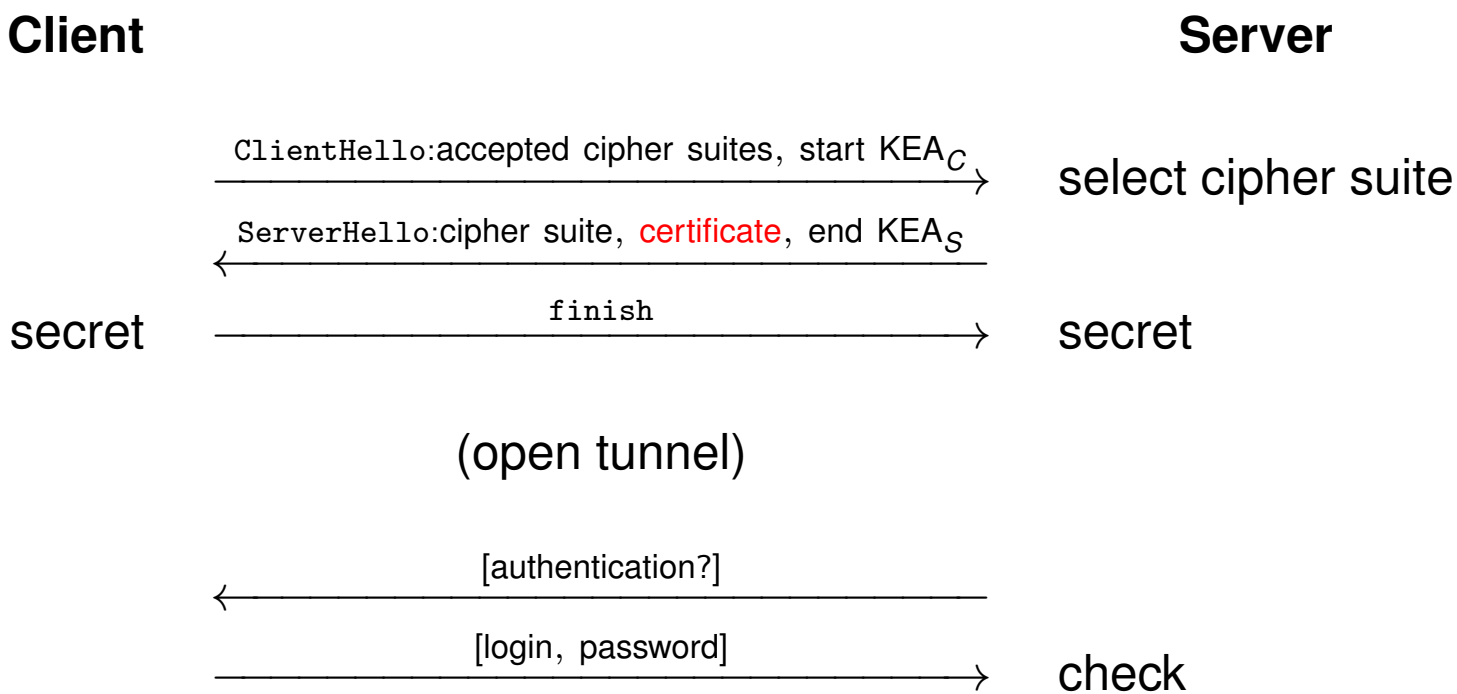


Semi-Authenticated Channel

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 1.3 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)
```

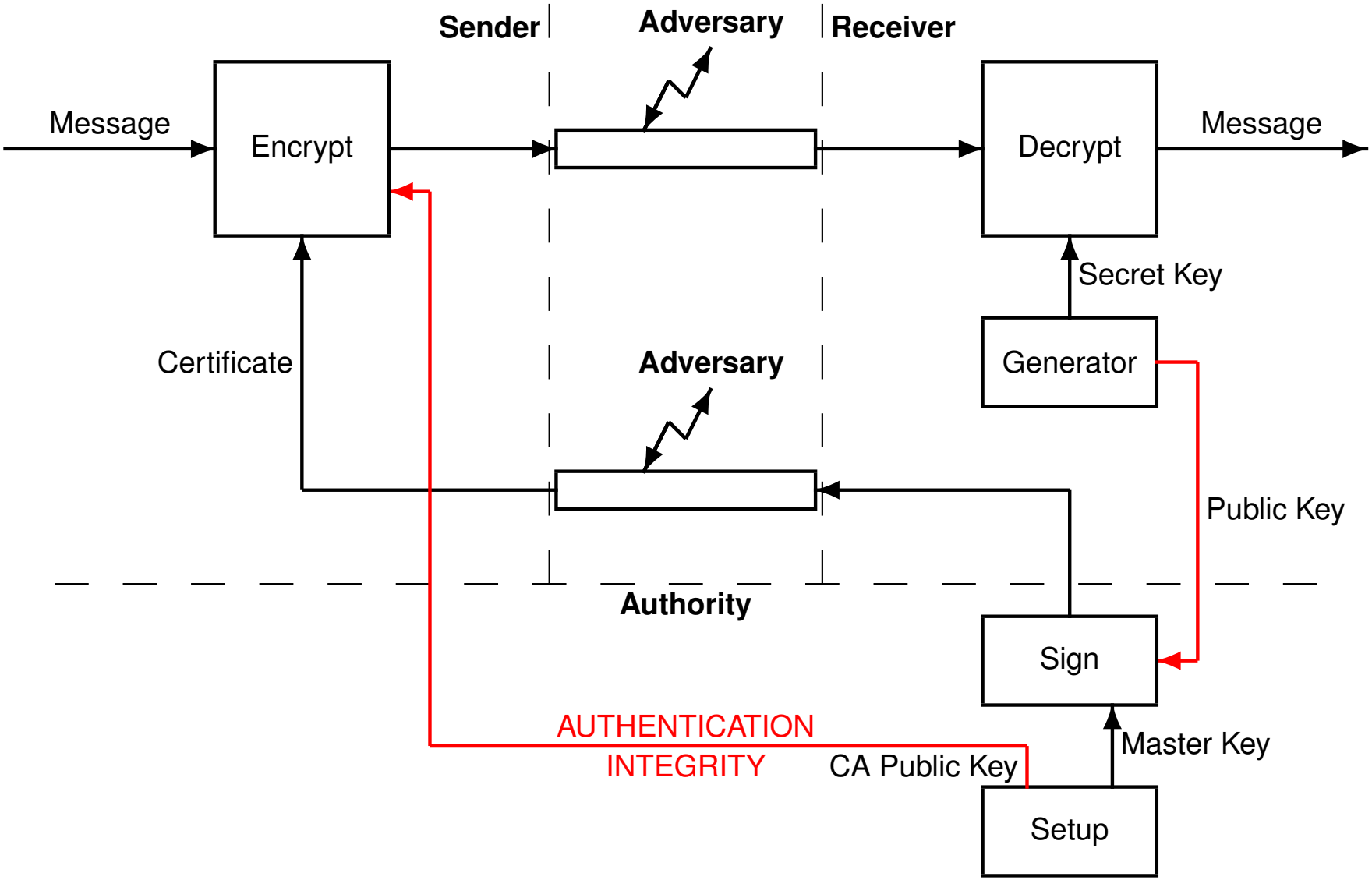
Two Approaches to Revocations

- **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

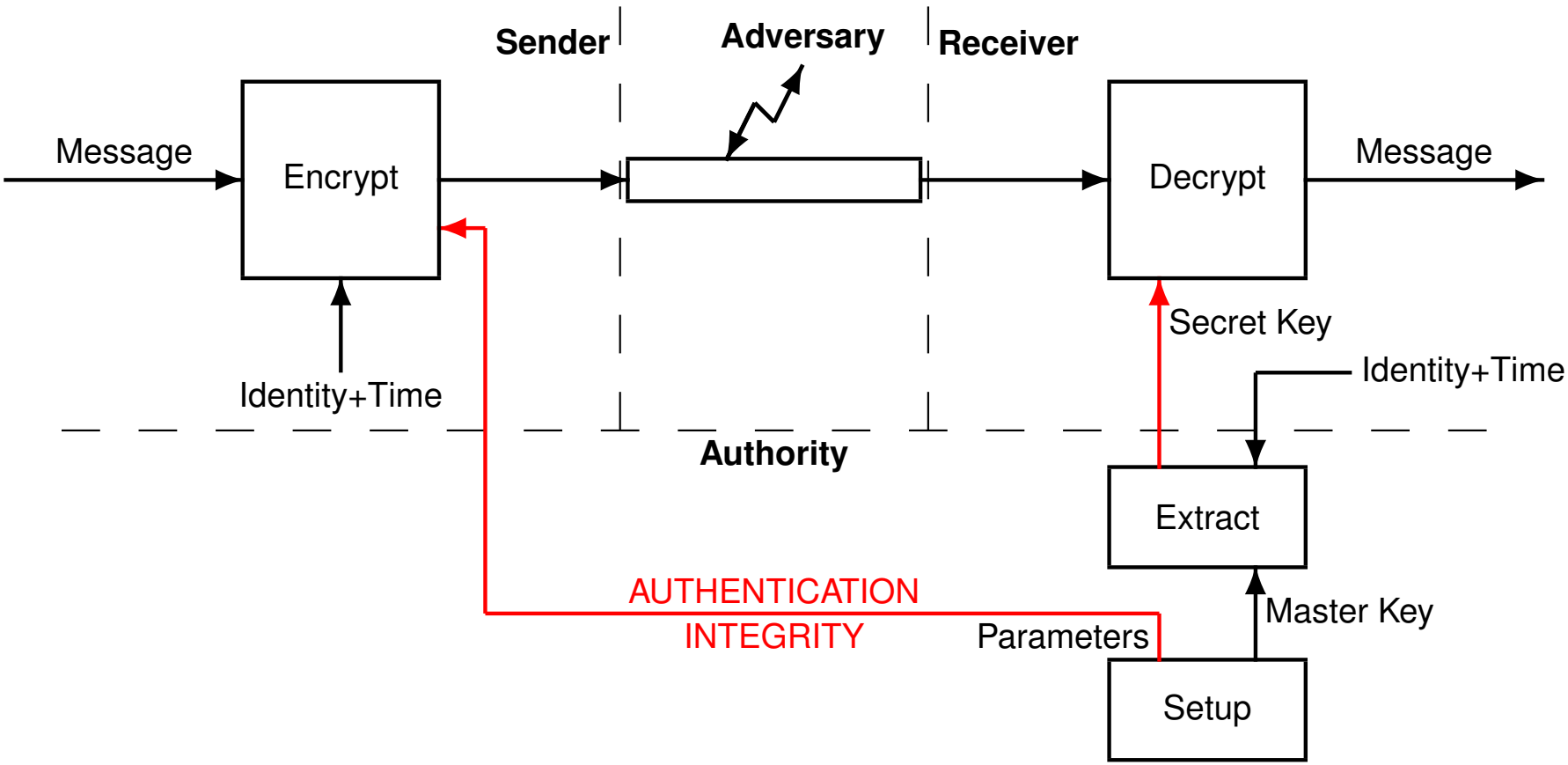
Several 3rd-Party Based Trust Infrastructure

- **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
  $\approx$ equivalent to certificaless encryption (name is confusing)

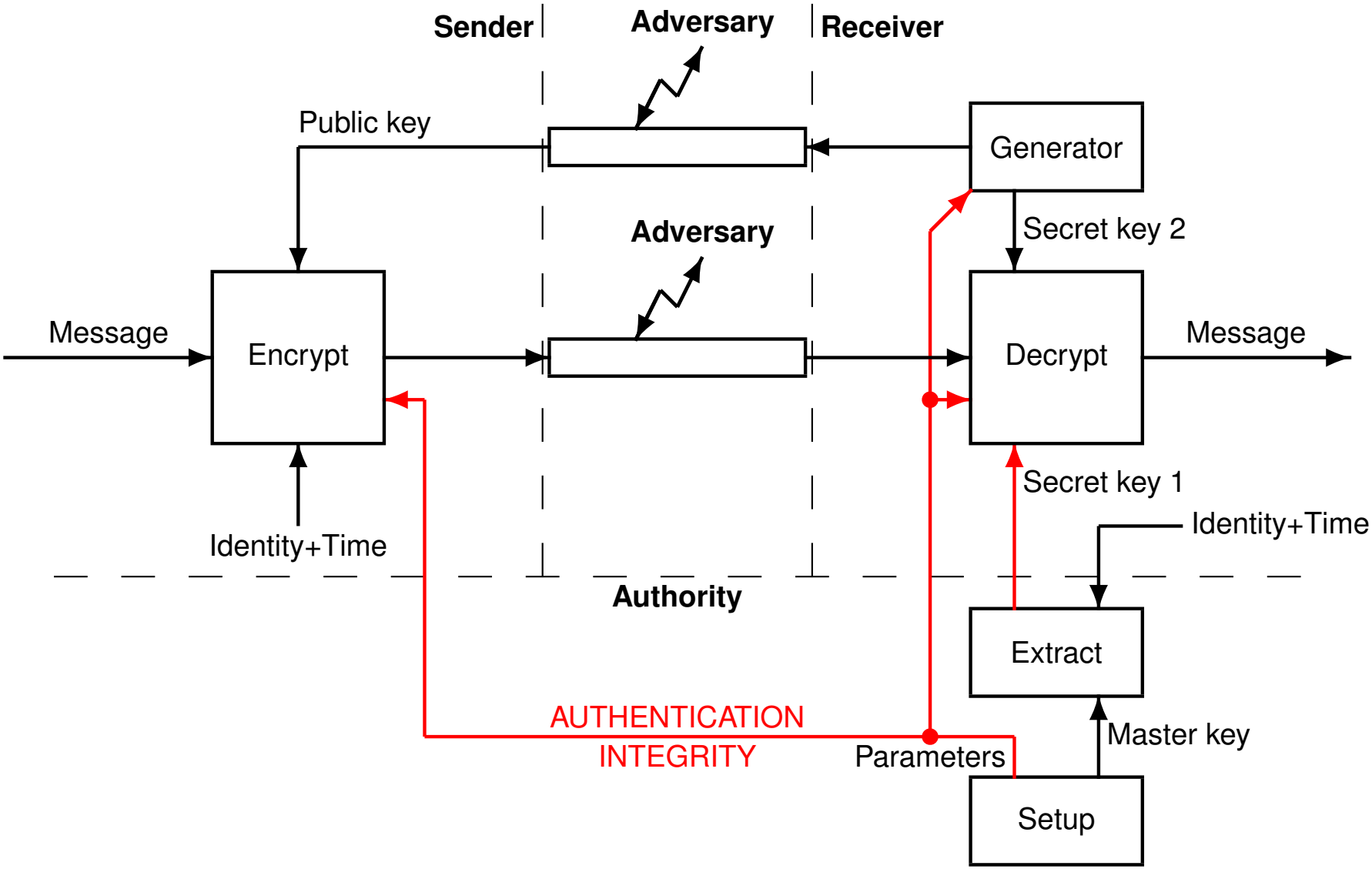
Public-Key Infrastructure



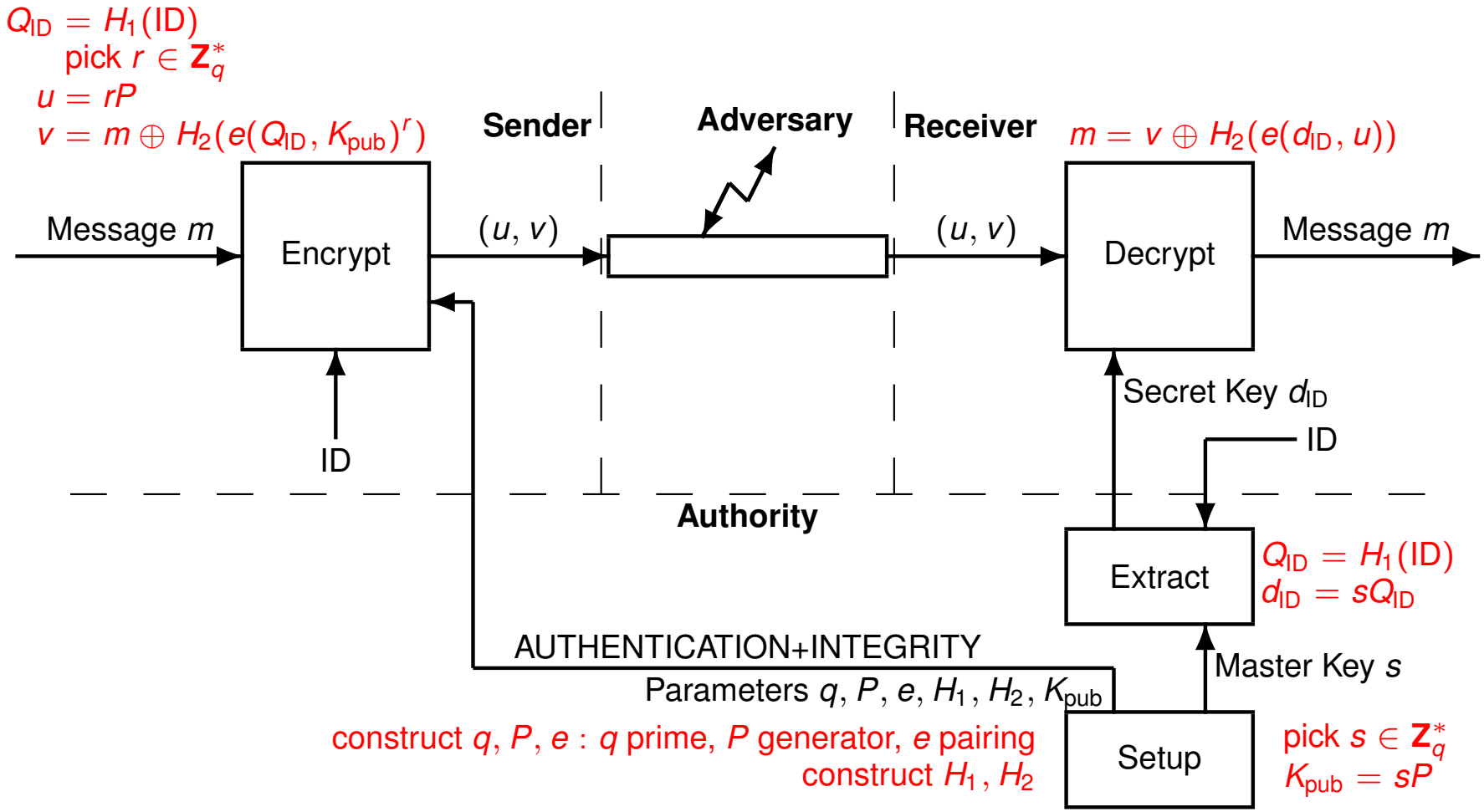
Identity-Based Encryption



Certificateless Encryption



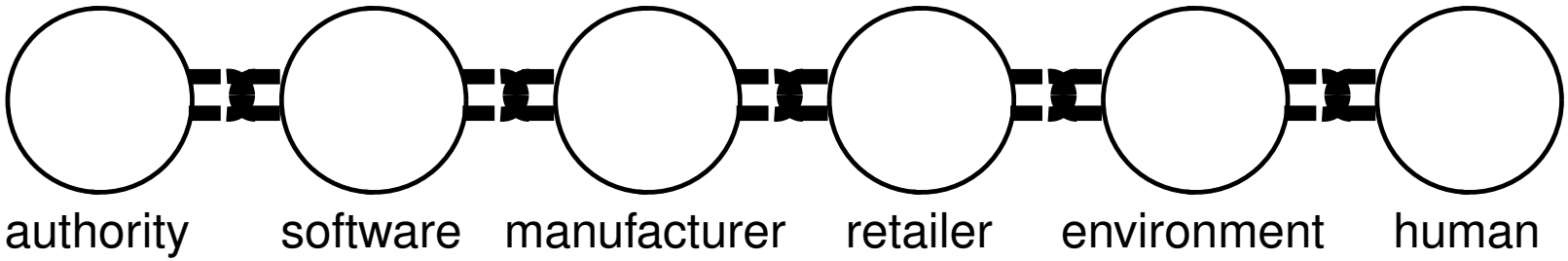
Boneh-Franklin Identity-Based Encryption



8 Trust Establishment

- Access Control
- Password-Based Cryptography
- 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

Chain of Trust in the PKI Model



- CA must issue correct certificate
- sofware must include correct CA public keys
- harware 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
$\longrightarrow$ **education**, **psychology**, **ergonomy**, **technology**

Several Approaches to Certificate Verification

- **TLS**: trust model based on a PKI
  verify a certificate every time the public key is used
  clients hold a list of CA public keys and retrieve server certificates
- **SSH**: trust model based on cache
  verify that a public key has not changed since the last time
  clients keep in cache the public key of servers
  (first connection may be insecure)
- **PGP**: trust model monitored by users
  use a public key ring set up by the user
  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)

More References

- **Gentry**. Certificate-Based Encryption and the Certificate Revocation Problem. *EUROCRYPT 2003*, LNCS 2656.

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
  $\longrightarrow$ **need for crypto-diversity**
- academic system failure:
  crypto results are done under pressure: too many conferences, too many papers, too many beans to get
  $\longrightarrow$ many results are wrong
  $\longrightarrow$ **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
  $\longrightarrow$ **need for good composability models**
- quantum threat
  $\longrightarrow$ **need for post-quantum cryptography**

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

Must be Known

- techniques for access control
- password-based cryptography
- secure channels
- SAS-based cryptography
- Kerberos
- public-key cryptography and man-in-the-middle attacks
- PKI, certificate validation model

Train Yourself

- secure channel:
  final exam 2012–13 ex3
  final exam 2009–10 ex2
- mass surveillance:
  final exam 2016–17 ex2
- bad EKE variant:
  final exam 2014–15 ex4
- SAS-based crypto:
  final exam 2017–18 ex3
- Covid certificate:
  final exam 2021–22 ex2

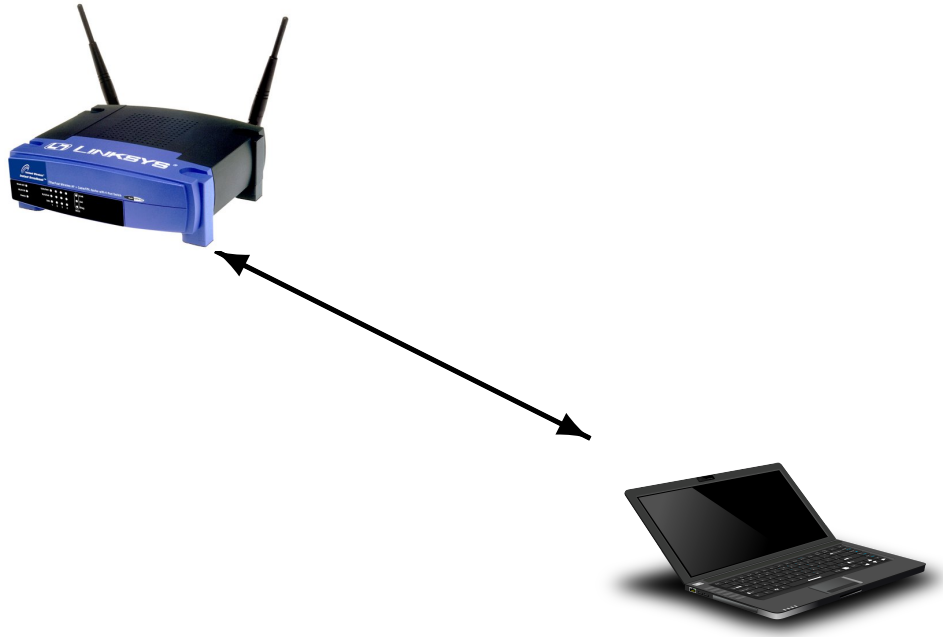
Roadmap

- WiFi
- blockchains
- mobile telephony
- Signal
- NFC creditcard payment
- Bluetooth
- Biometric passport
- TLS

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

IEEE 802.11 in a Nutshell



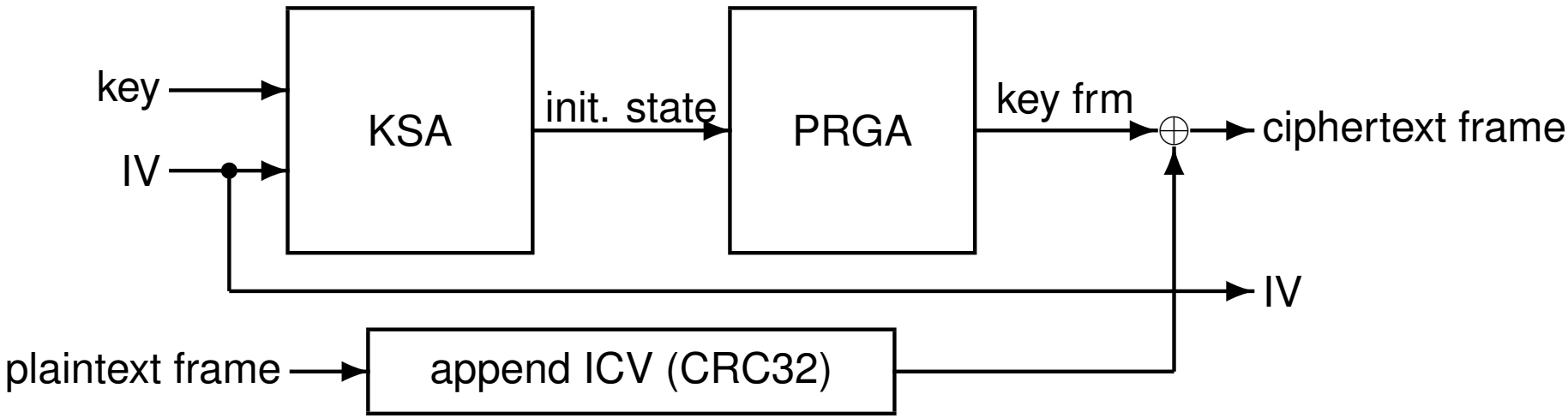
- wireless local area network (WLAN)
- since 1997
- secure communication by **wired equivalent privacy** (WEP)
- ~~station authentication by **Shared Key Authentication** (SKA)~~
- since 2003: interim **Wi-Fi Protected Access** (WPA)
  due to security issues
- since 2004: added WPA2 (complete change)

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
  $\rightarrow$ key not frequently changed
  $\rightarrow$ key not too long (40 or 104 bits)
  $\rightarrow$ key stored at many places
- entirely based on RC4 stream cipher

WEP Encryption



$\rightarrow$ self-synchronizing stream cipher (24-bit IV sent in clear)

(Terrible) Integrity Protection using CRC32

$$\mathsf{Enc}_{\mathsf{key}}(\mathsf{IV}, \mathsf{pt}) = [\mathsf{pt} \| \mathsf{CRC32}(\mathsf{pt})] \oplus \mathsf{keyframe}(\mathsf{key}, \mathsf{IV})$$

packets are easily malleable (Borisov-Goldberg-Wagner 2001):

$$\begin{aligned}
& \mathsf{Enc}_{\mathsf{key}}(\mathsf{IV}, \mathsf{pt}) \oplus [\Delta \| \mathsf{CRC32}(\Delta)] \\
= \quad & [\mathsf{pt} \| \mathsf{CRC32}(\mathsf{pt})] \oplus [\Delta \| \mathsf{CRC32}(\Delta)] \oplus \mathsf{keyframe}(\mathsf{key}, \mathsf{IV}) \\
= \quad & [\mathsf{pt} \oplus \Delta \| \mathsf{CRC32}(\mathsf{pt}) \oplus \mathsf{CRC32}(\Delta)] \oplus \mathsf{keyframe}(\mathsf{key}, \mathsf{IV}) \\
= \quad & [\mathsf{pt} \oplus \Delta \| \mathsf{CRC32}(\mathsf{pt} \oplus \Delta)] \oplus \mathsf{keyframe}(\mathsf{key}, \mathsf{IV}) \\
= \quad & \mathsf{Enc}_{\mathsf{key}}(\mathsf{IV}, \mathsf{pt} \oplus \Delta)
\end{aligned}$$

WEP 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 ☹

an example not to follow

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 MICHAEL, a broken MAC...)~~
- check IV increases to protect against nonce repetition
- set up master key using EAP (Extensible Authentication Protocol)
  - PSK (Pre-Shared Key)
  - one of the possible authentication protocols from 802.1x using an authentication server (e.g. RADIUS)
    TLS (+ two certificates), TTLS (one certificate, one password), PEAP, SIM (using GSM), AKA (using UMTS), FAST (Cisco)

WPA2

RC4 replaced by AES CCMP (CCM Protocol = AES in CCM mode)
128 or 256 bit key

severe mistake:

- there is an option in the handshake to reset the key to a previously used one
  (to save computation, to ask to resend a lost packet, ...)
- but this resets the nonce counter as well...
- exploit by Vanhoef and Piessens in 2016:
  KRACK (Key Reinstallation Attack)
- patched implementations disabled this
  but have a less reliable connectivity...

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

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* $\text{link}_1, \ldots, \text{link}_n$ *pay* x_1 *to* pk_1*, ..,* x_m *to* pk_m*"*
[signature]

- UTXO = unspent transaction output
- requirement: $x_1 + \cdots + x_m$ equals sum of given UTXO
- then, amounts from [link_1], ..., [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

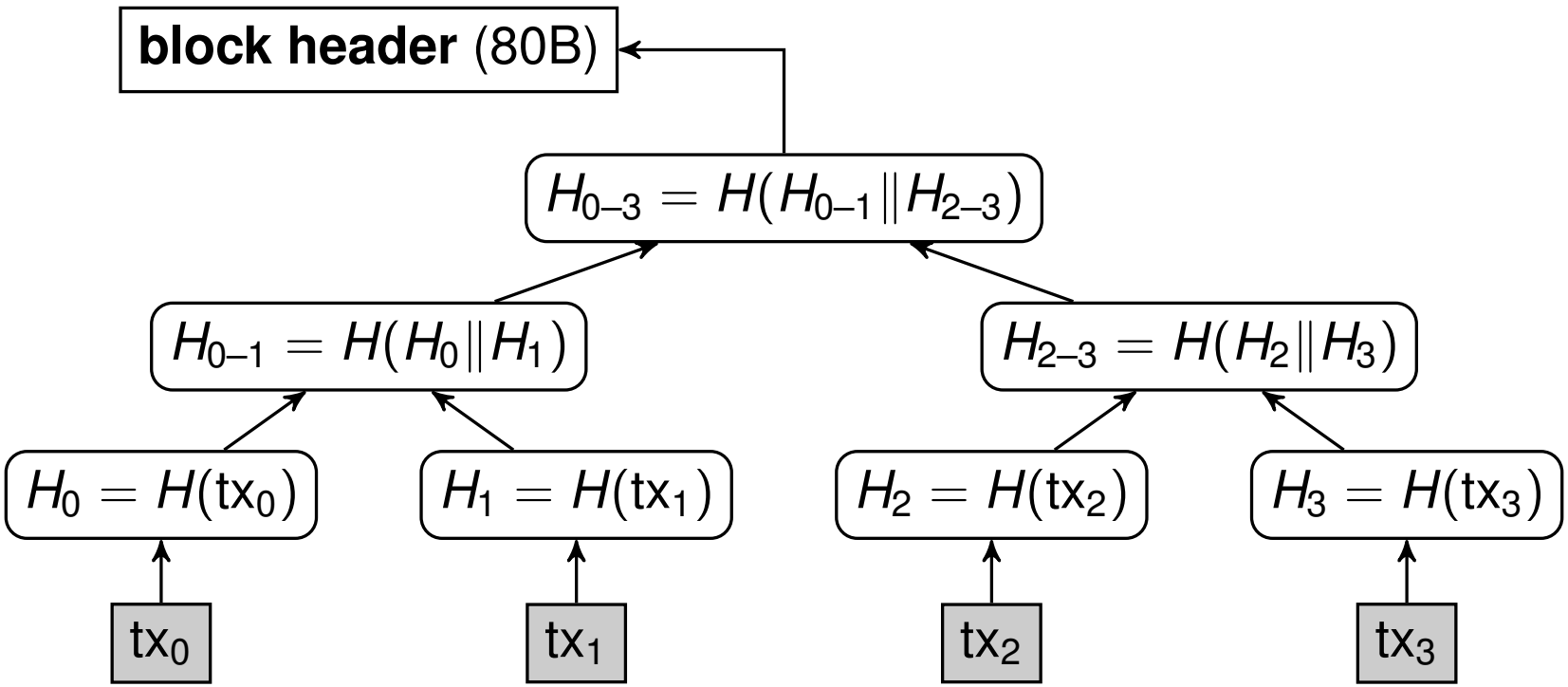
a block from the block chain:

- 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

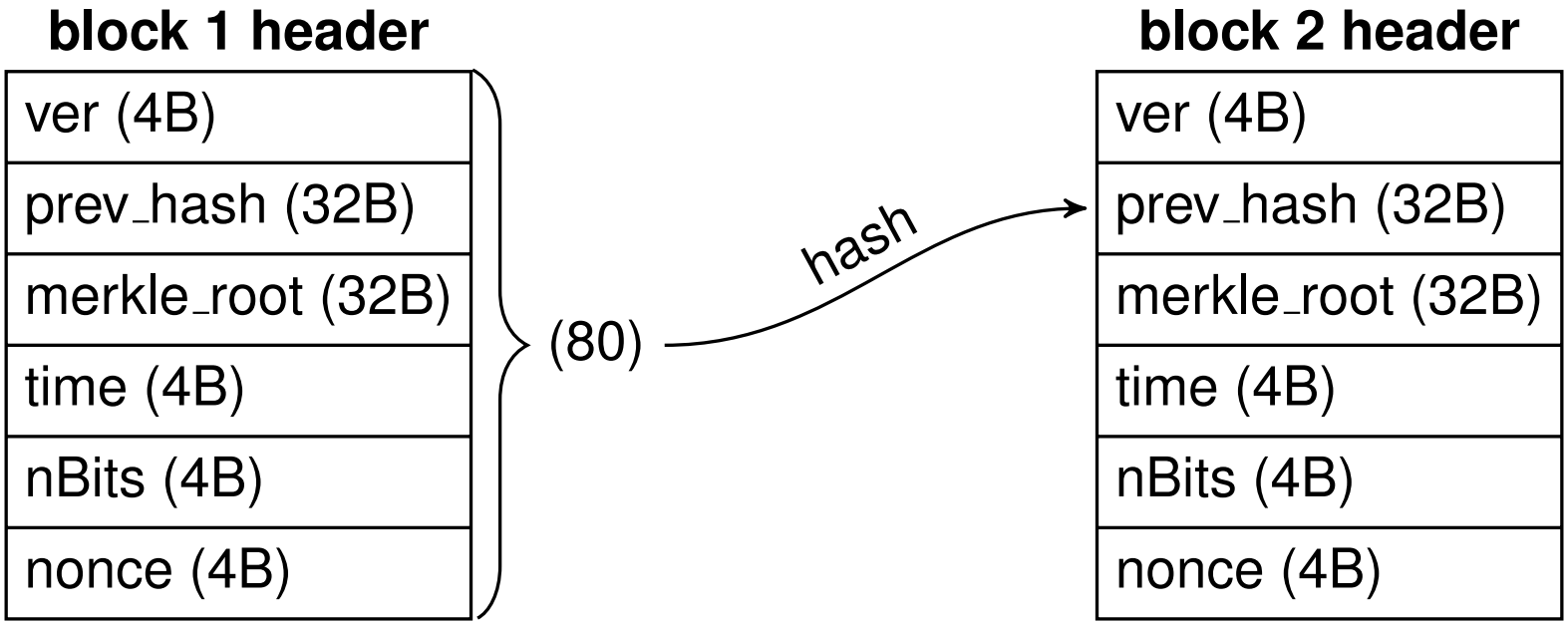
scheme for miners (every 10 minutes):

- 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 first transaction (the coinbase transaction) rewards the miner

Bitcoin Block



Bitcoin Blockchain



$$\text{prev\_hash} = \text{SHA256}\left(\underbrace{\text{SHA256}\left(\underbrace{\text{prev\_block}}_{80\text{B}}\right)}_{32\text{B}}\right)$$

Proof-of-Work

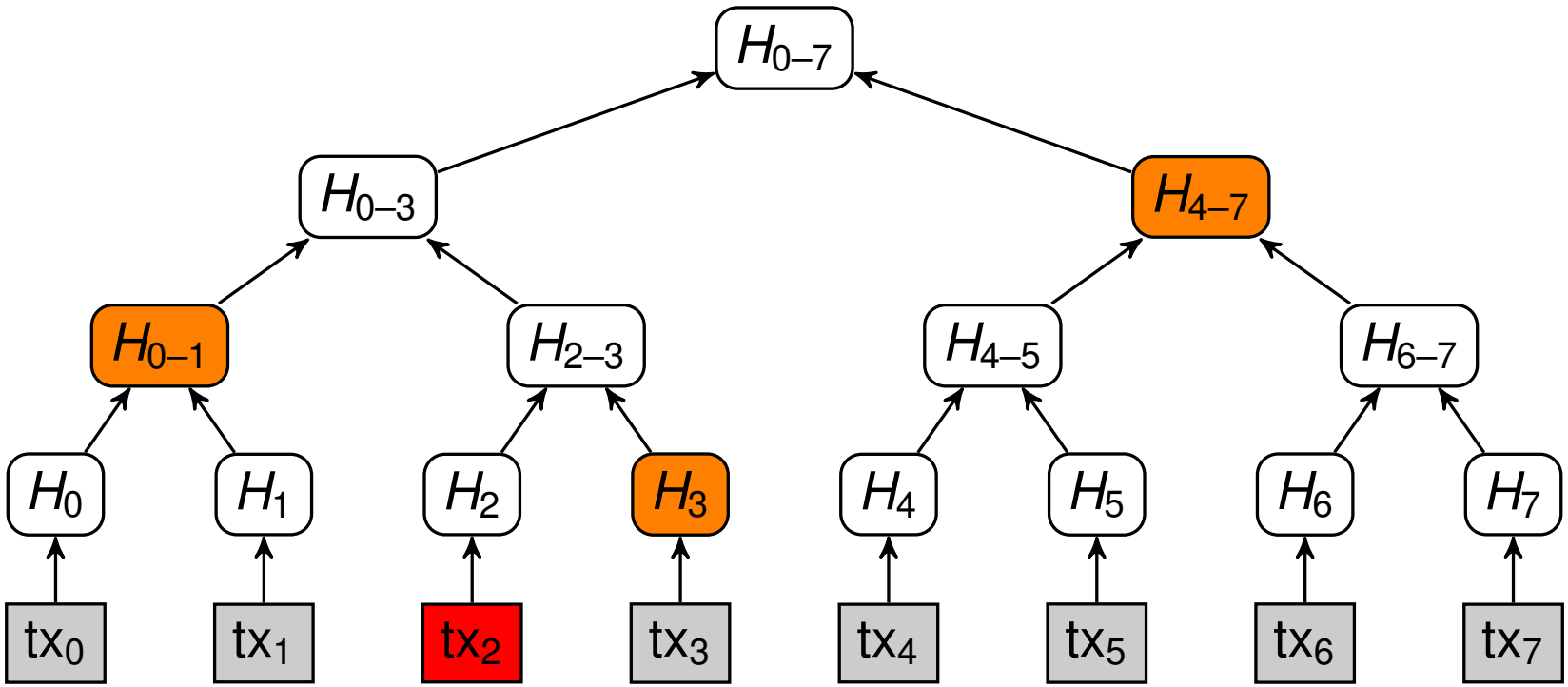
block shall contains for PoW value such that

SHA256(block) starts with 69 zero bits

69 is the difficulty of June 2016
it is constantly calibrated
nonce is only 32-bit long but more data is used in PoW:

- another nonce in the coinbase tx
- timestamp (in msec)
- new transactions
- their order

Merkle Authentication Tree (Hash Tree)



Assuming $H_{0–7}$ is authenticated, to authenticate tx_2, just give $H_3, H_{0–1}, H_{4–7}$.

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

GSM Architecture

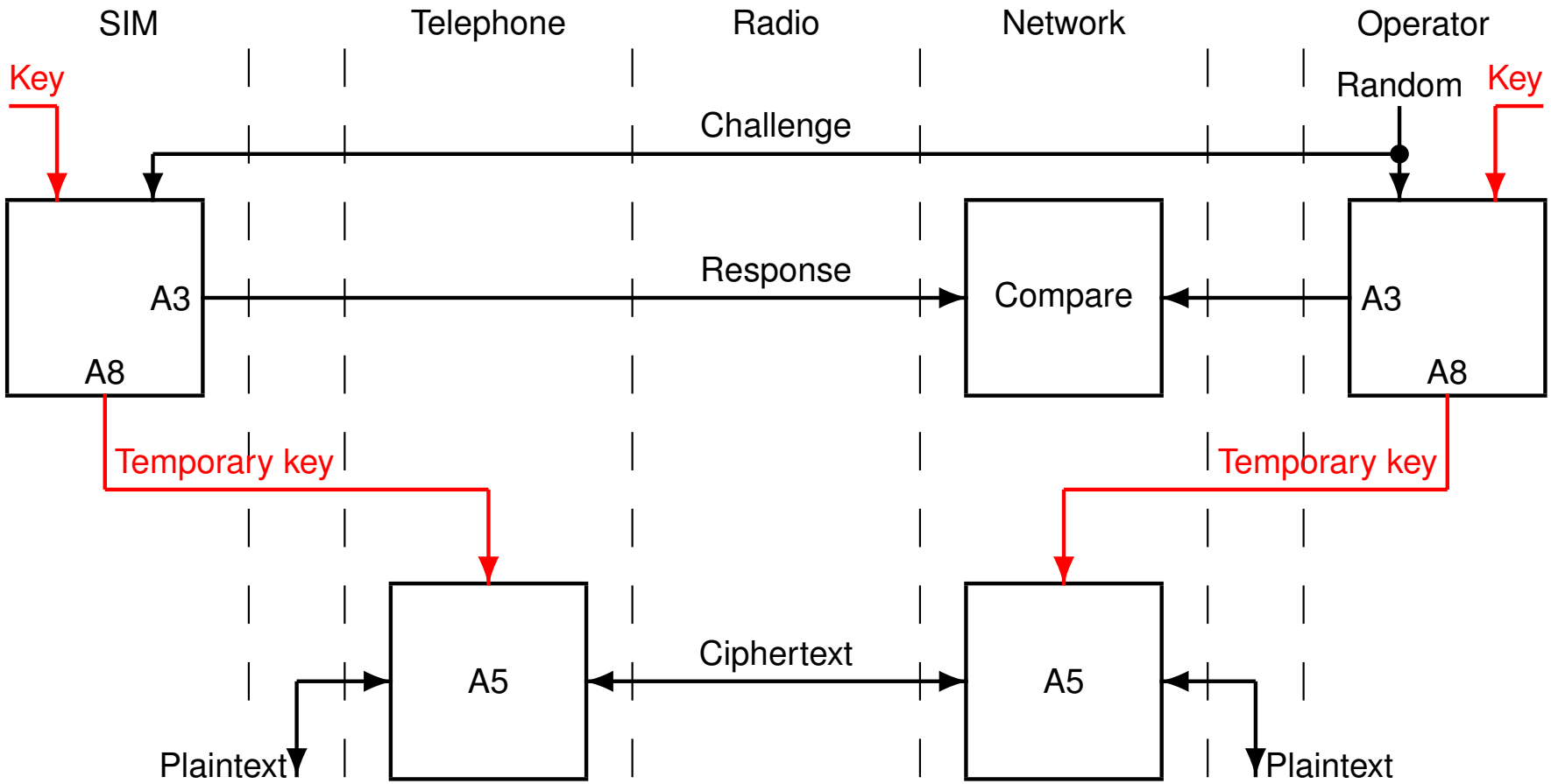
- principle 1: authentication of mobile system
- principle 2: privacy protection in the wireless link

- 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

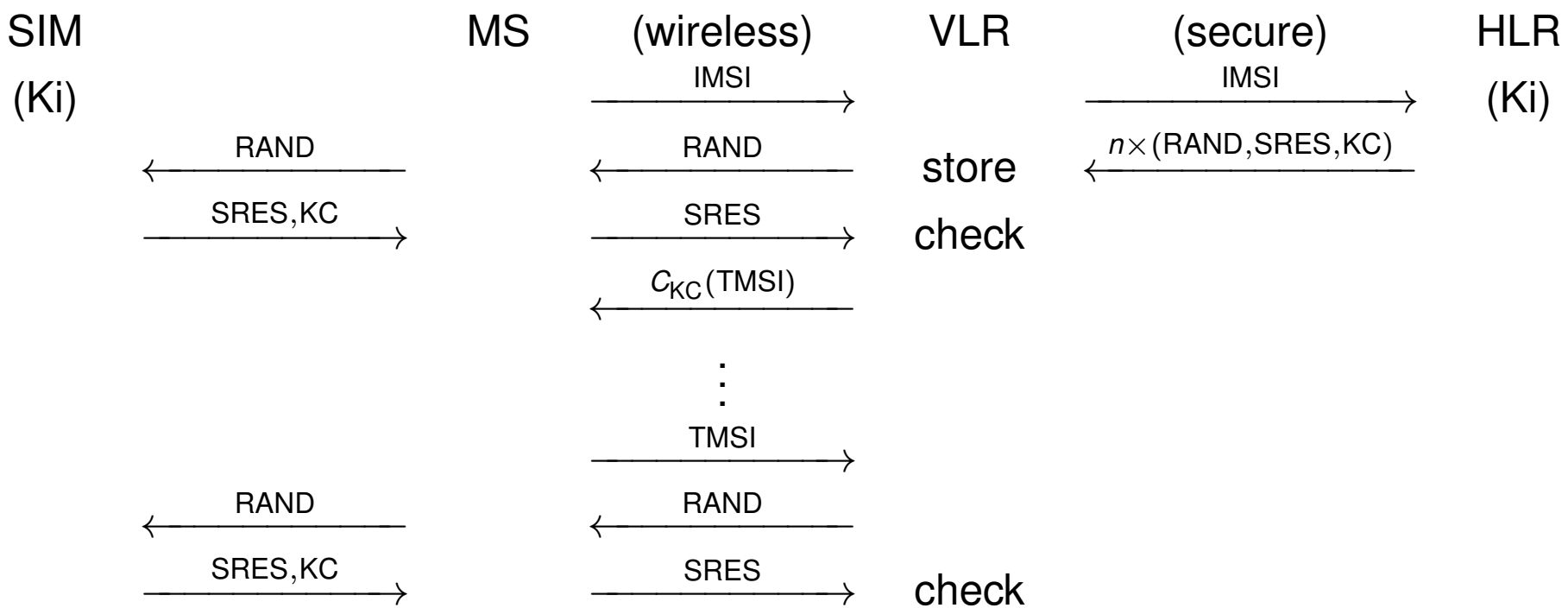
- GSM: Global System for Mobile telecommunications
- MS: Mobile Station
- SIM: Subscriber Identity Module (part of MS)
- HLR: Home Location Register
- VLR: Visitor Location Register
- IMSI: International Mobile Subscriber Identity (stored in SIM)
- Ki: subscriber Integrity Key (securely stored in SIM)

GSM Protocol



GSM Peer Authentication

$$\text{A3/8}(\text{Ki}, \text{RAND}) = (\text{SRES}, \text{KC})$$



Security of Peer 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 473)

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 (covered in WEP)

security: ☹

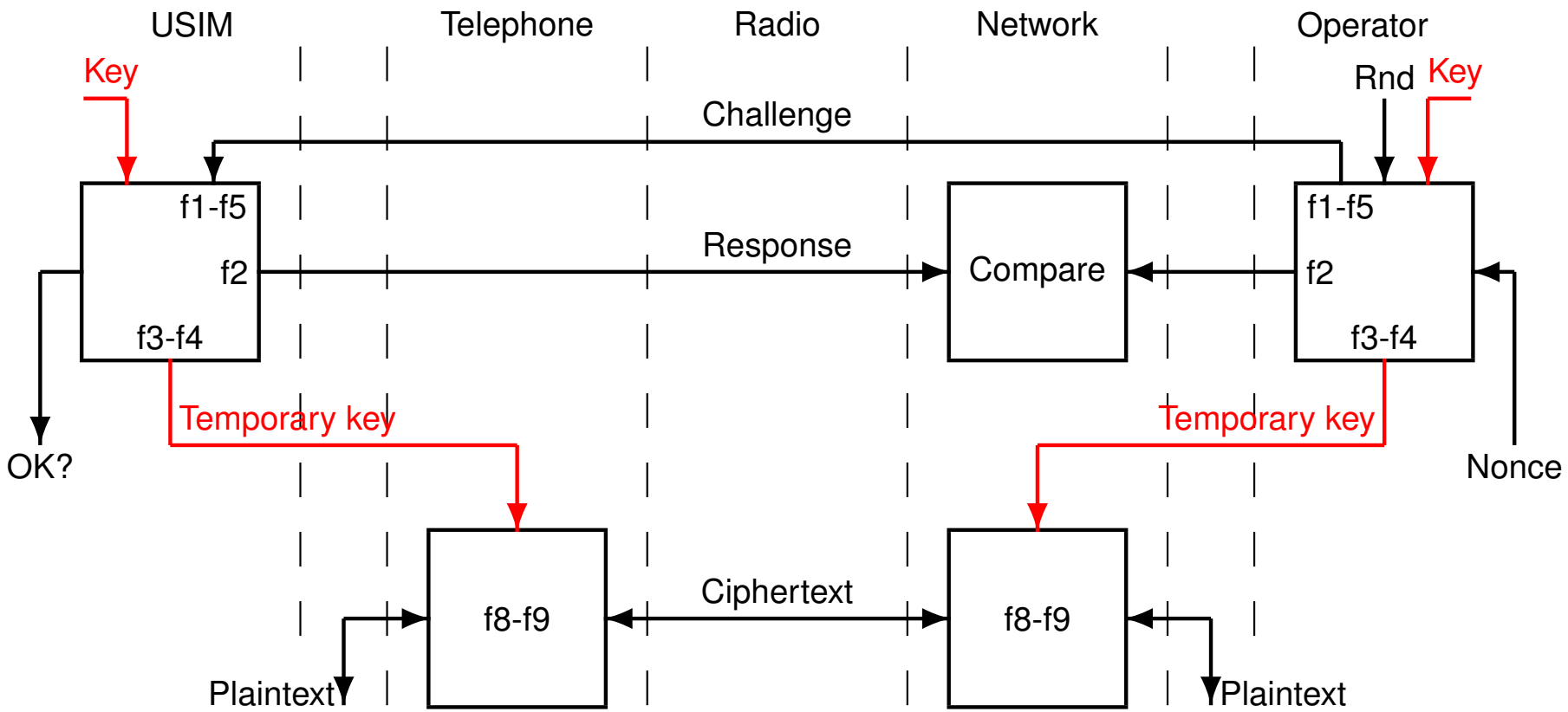
Improvements in 3G Mobile Telephony

- 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

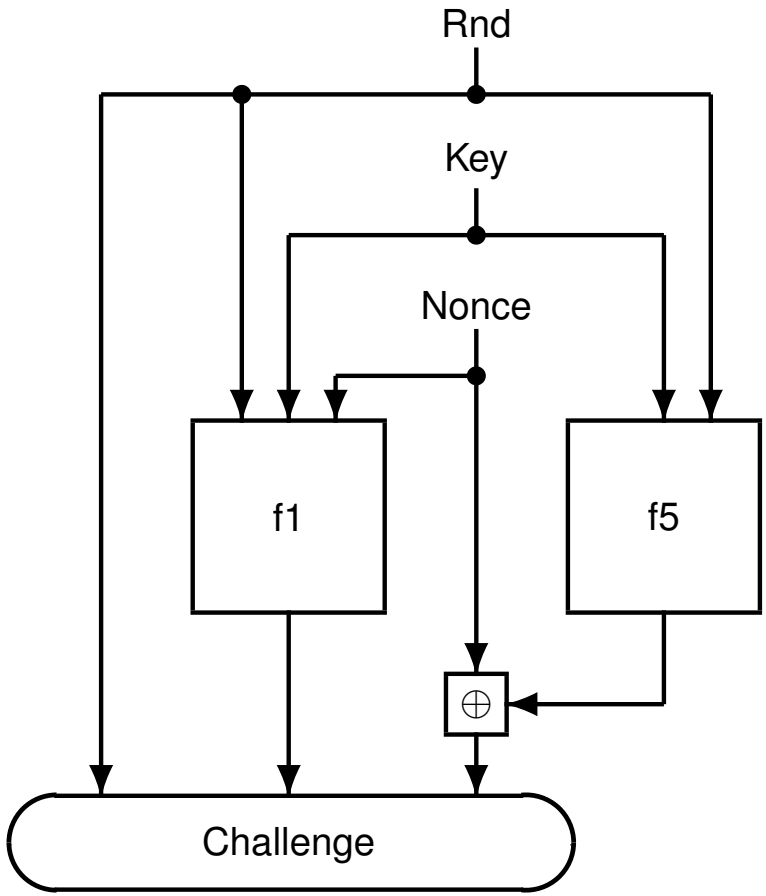
The UMTS Crypto Menagery

- 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

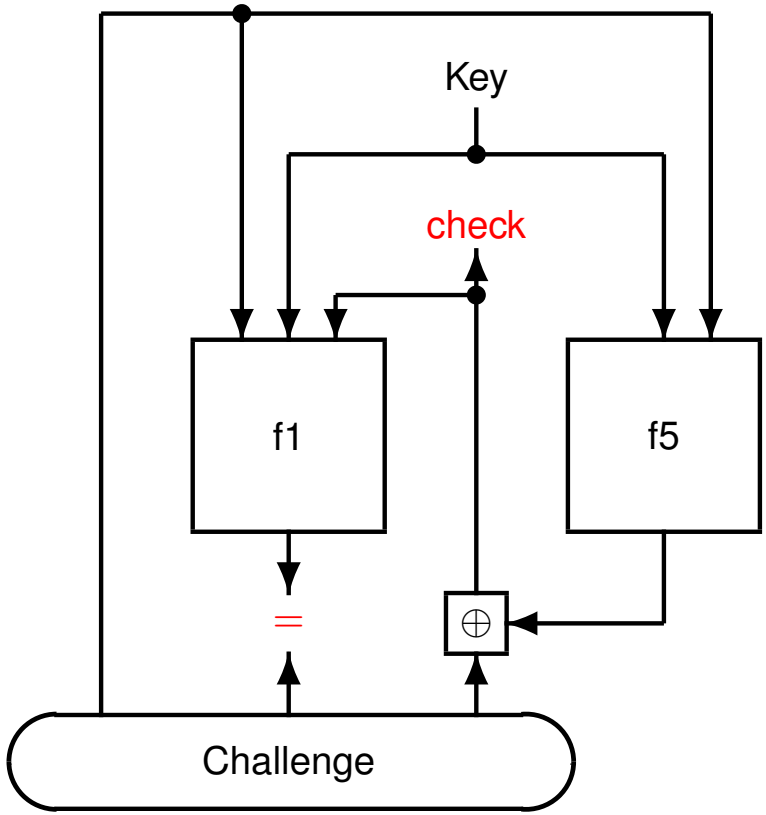


MILENAGE Challenges



- 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(Key, Rnd)
3. check authentication (f1)
4. check Nonce is correct

Security Misses

- network is not authenticated (network only proves he received authorization from operator)
  $\rightarrow$ 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

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

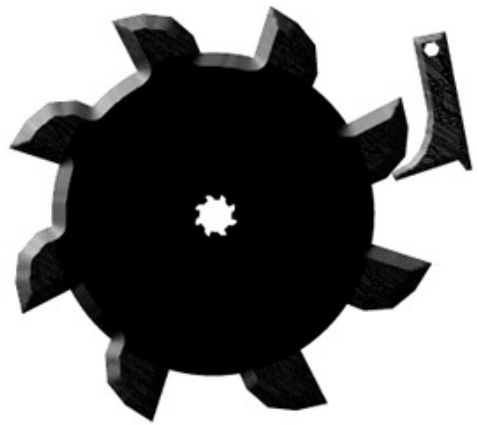
Signal

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
- X3DH + Double Ratchet

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

X3DH: Initial Key Agreement

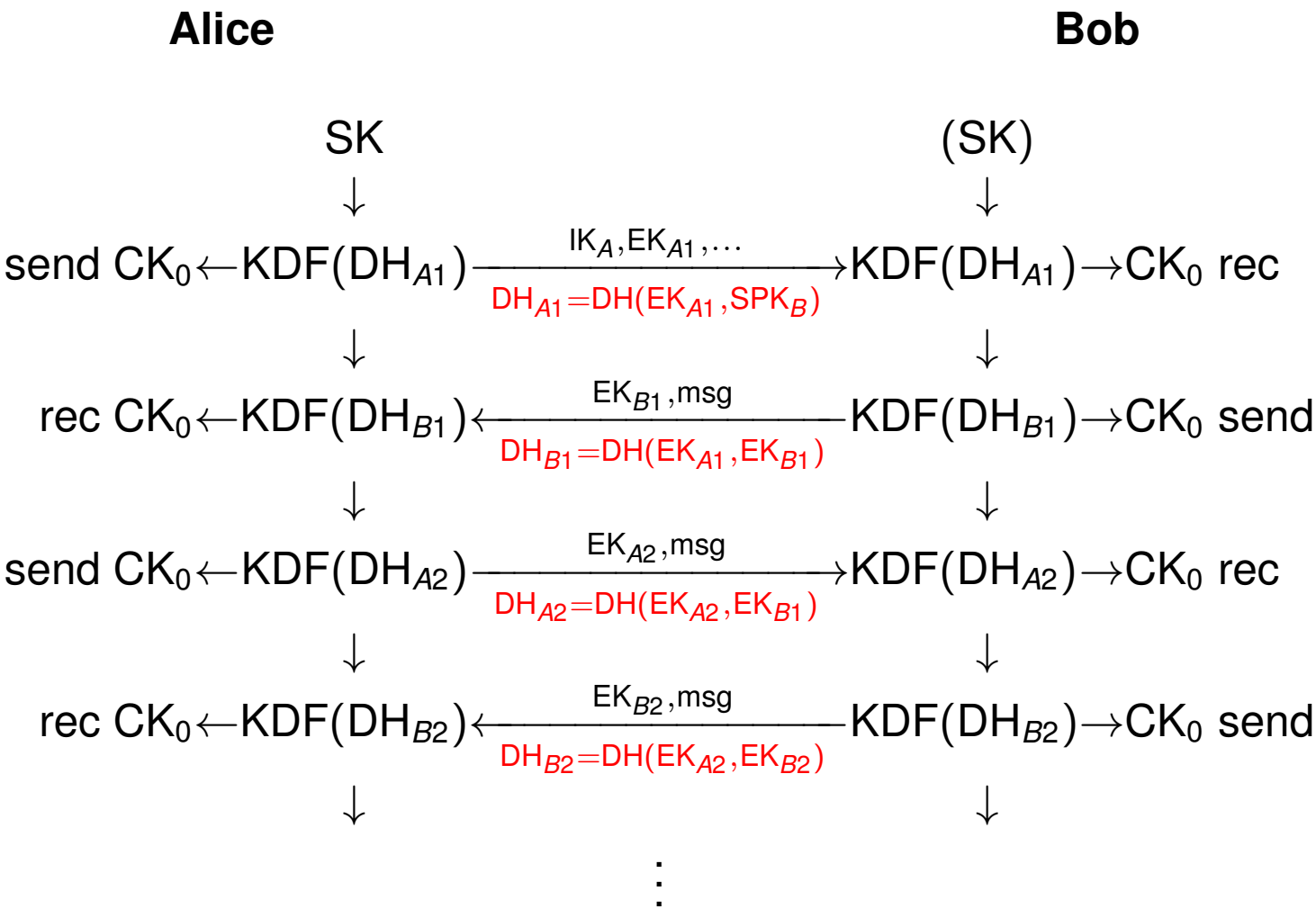
(keys are in Curve25519)

Alice	**Server**	**Bob**
IK_A: identity key		IK_B: identity key SPK_B: signed prekey $[OPK_B]$: one-time prekey $\sigma_B \leftarrow \text{Sign}(SPK_B)$
	register $\xleftarrow{IK_B, SPK_B, \sigma_B, [OPK_B^i]}$	$i = 1, \ldots$
$\xrightarrow{\text{Bob?}}$		(σ_B signed with IK_B)
$\xleftarrow{IK_B, SPK_B, \sigma_B, [OPK_B^i]}$	[erase OPH_B^i]	
EK_A: ephemeral key $DH1 \leftarrow DH(IK_A, SPK_B)$ $DH2 \leftarrow DH(EK_A, IK_B)$ $DH3 \leftarrow DH(EK_A, SPK_B)$ $[DH4 \leftarrow DH(EK_A, OPK_B^i)]$ $SK \leftarrow KDF(DH1 \Vert DH2 \Vert DH3 \Vert [DH4])$ $AD \leftarrow IK_A \Vert IK_B$ state: SK, AD send first message with AD		OPK avoids replay attacks making SK reused
$\xrightarrow{IK_A, EK_A, \text{used prekeys of Bob}, AEAD_{SK}(\text{first message})}$		
...		...

Double Ratchet

- DH: a ratchet for every time the direction of exchange changes
  ratchet message indicates the new ephemeral key to use in DH
  good forward and future secrecy
- symmetric-key ratchet: two ratchets (one for each direction)
  no real future secrecy
  plausible deniability

Diffie-Hellman Ratchet

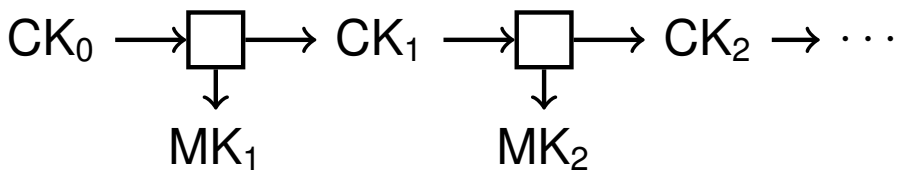


Symmetric-Key Ratchet

given CK_0, derive chain keys CK_i, message keys MK_i

$$(\mathrm{MK}_i, \mathrm{CK}_{i+1}) = \mathrm{KDF\_CK}(\mathrm{CK}_i)$$

the message i is encrypted using MK_i with AD (AEAD)

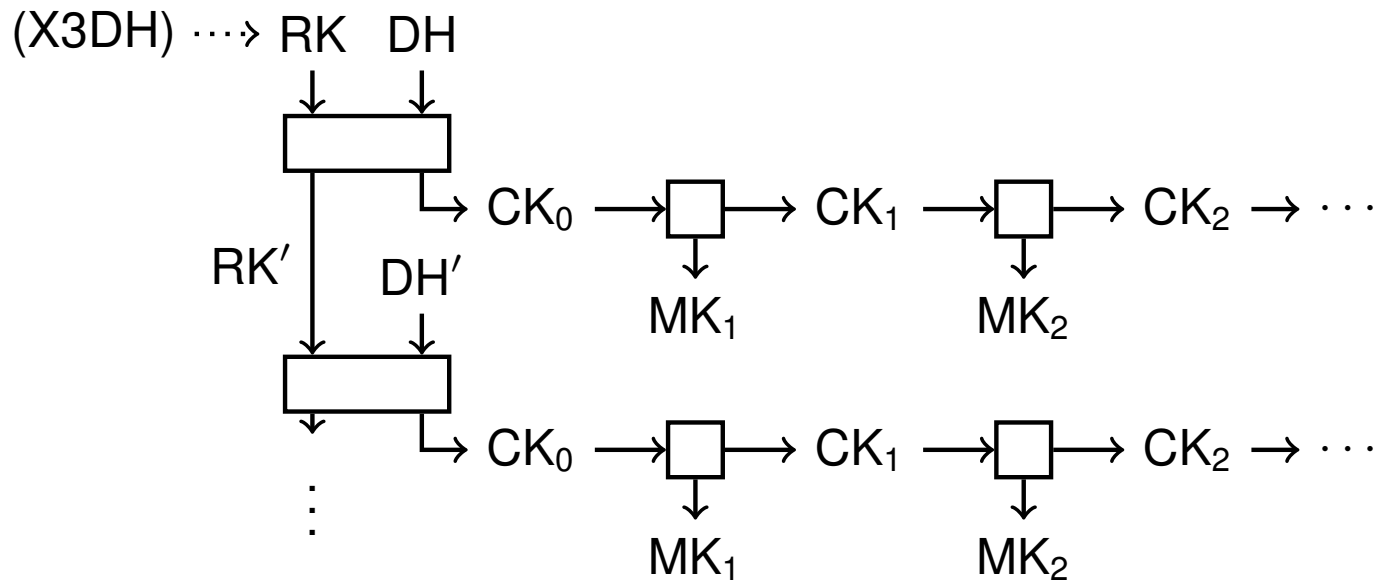


Double Ratchet

given SK, derive root keys RK_i, chain keys CK_i, message keys MK_i

$$
\begin{aligned}
RK_{\text{initial}} &= SK \qquad \text{(from X3DH)} \\
(RK_{\text{new}}, CK_0) &= \text{KDF\_RK}(RK_{\text{old}}, DH_{\text{new}}) \\
(MK_i, CK_{i+1}) &= \text{KDF\_CK}(CK_i) \\
DH &= DH(EK_{A\text{current}}, EK_{B\text{current}})
\end{aligned}
$$

the message i is encrypted using MK_i with AD (AEAD)



Involved Cryptography

- ECDH on Curve25519
- HMAC-SHA256
- AES256 CBC
- HKDF with SHA2
- AEAD:

$$
\begin{aligned}
\mathsf{MK} &= K_{\mathsf{enc}}^{(32)} \| K_{\mathsf{mac}}^{(32)} \| \mathsf{IV}^{(16)} \\
\mathsf{ct} &= \text{AES-CBC}_{K_{\mathsf{enc}},\mathsf{IV}}(\mathsf{pt} \| \text{padding}) \\
\tau &= \text{HMAC}_{K_{\mathsf{mac}}}(\mathsf{AD} \| \mathsf{ct}) \\
\text{AEAD}_{\mathsf{MK}}(\mathsf{AD}, \mathsf{pt}) &= \mathsf{ct} \| \tau
\end{aligned}
$$

Out-of-Band Authentication

safety number $H(\mathsf{IK}_A \| \mathsf{IK}_B \| \cdots)$

- can be viewed numerically (60 decimal digits!) or with QR code
- can cross-check numbers or cross scan QR codes
- used to manually authenticate identity keys and more

Contact Discovery

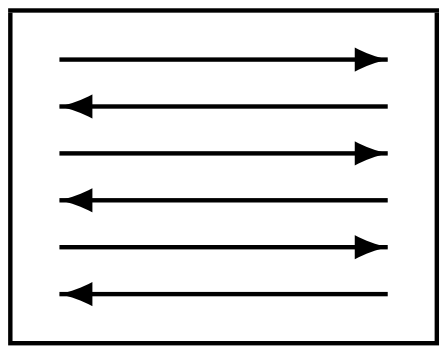
To determine if some of the contacts in the phone are Signal users, the app sends the hash of every phone number in the contact list to the central server...

Inversion attack is easy

What is needed: private set intersection:

Smartphone
(small) set: contacts

Server
(big) set: users



output: contacts ∩ users

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

Typical Requirements in Secure Browsing

- unidirectional **authentication**
- **confidentiality** of communication
- **integrity** of communication

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 in 2018 [RFC8446]

Goal: secure any communication (e.g. HTTP) based on TCP/IP

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)

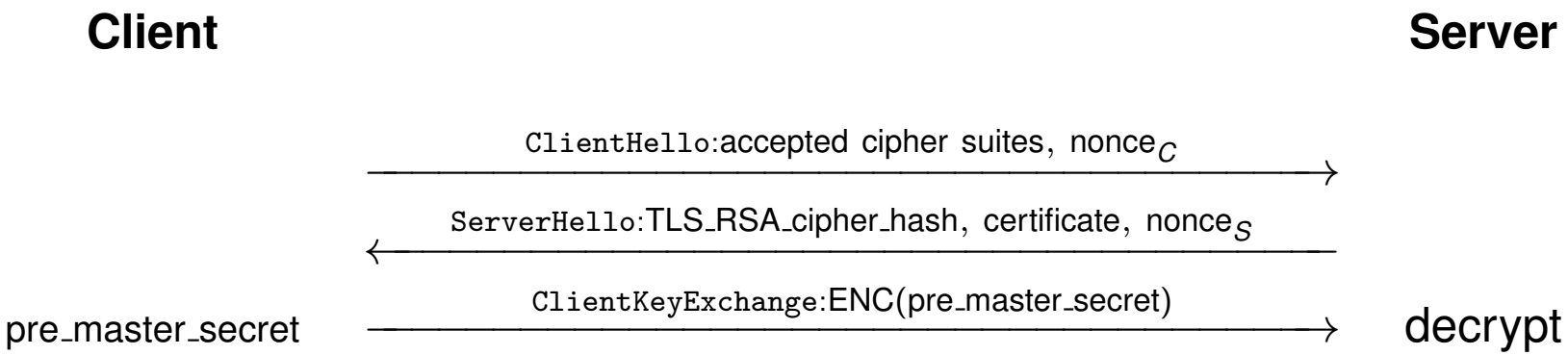
Cipher Suites (from 1.0 to 1.2)

- cipher: RC4 or DES/3DES in CBC mode
  (key could be limited to 40 bits for “export”)
  - RC4 has biases
  - CBC mode has padding oracles

  added in 1.2: AES_GCM, AES_CCM, CAMELLIA, ARIA
- hash: MD5 or SHA1
  - both have collisions
  - maybe less a problem with HMAC

  added in 1.2: SHA2
- key exchange: RSA or DH (in several variants)
  DH_anon: ephemeral DH
  DH_sig: static DH with a certificate
  DHE_sig: ephemeral DH with a signed ephemeral public key
  added in 1.2: ECDSA, PSK

RSA Key Exchange (Old TLS)



- RSA encryption is PKCS#1v1.5
- the RSA public key must be authenticated (with a certificate)

TLS 1.3

new: better handshake protocol (less round-trips)

cipher suite in the form

`TLS_KEA_AUTH_WITH_CIPHER_HASH`

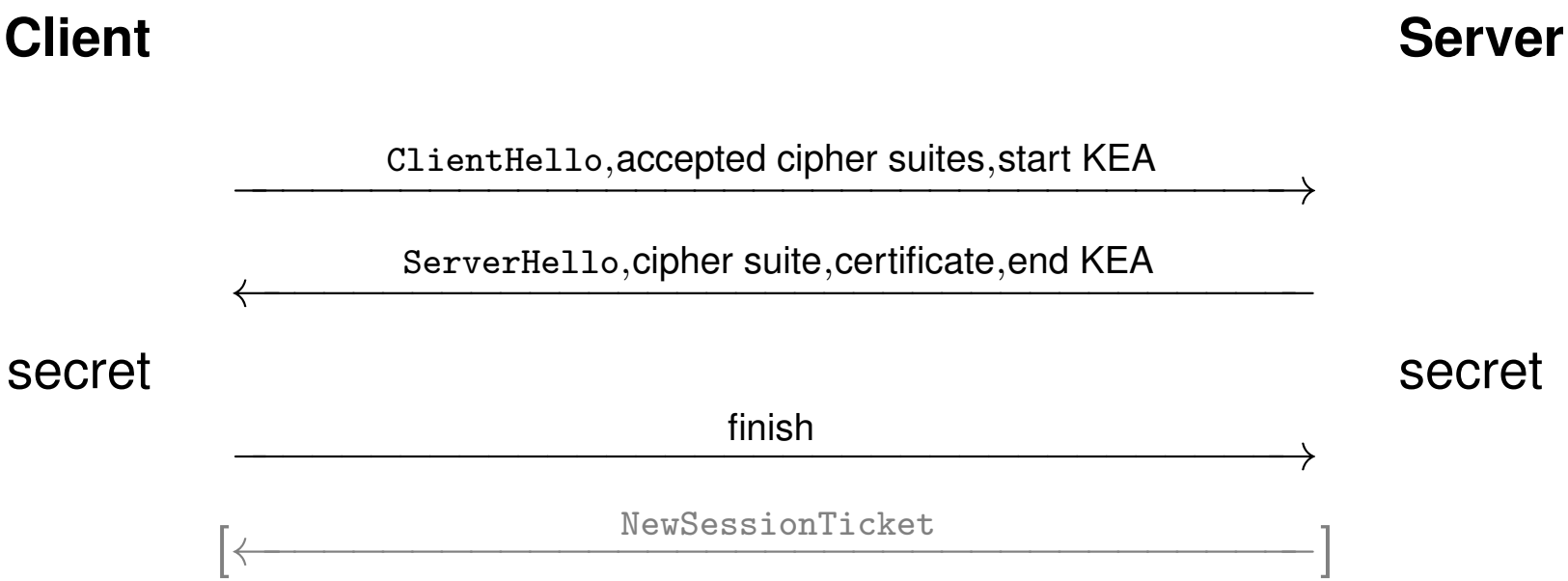
- key exchange (KEA) and authentication (AUTH) are separate
- KEA is ephemeral Diffie-Hellman only: DHE or ECDHE or PSK
  (for Diffie-Hellman: forward secrecy)
  keys are derived using HKDF
- AUTH is the way to authenticate peers, it can be with a certificate (RSA or ECDSA) or PSK
- CIPHER: AES-GCM, AES-CCM, CHACHA20-POLY1305
  (AEAD: Authenticated Encryption with Associated Data)
- 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]

TLS 1.3 Handshake



`ClientHello` includes supported groups, a first KEA message for some of those groups, supported signatures (to verify certificates), a list of identifiers for PSK known keys and the PSK mode to be used

if the server selects a supported group with no first KEA message, it requests an extra round trip

TLS 1.3 Resumption

0RTT Handshake (0 round-trip time)

- a client can re-establish (resume) a previous connection
- client sends a `ClientHello` with PSK and a `SessionTicket`
- resume + send encrypted messages at the same time
- `SessionTicket` includes a validity
- `SessionTicket` includes a way to recover `ResumptionSecret`
- continue normal handshake + define next `ResumptionSecret`

session cache

- server stores `ResumptionSecret`
- `SessionTicket` has lookup key
- delete `ResumptionSecret` after
- drawback: memory on server
- forward secure
- immune against replay attacks

session ticket

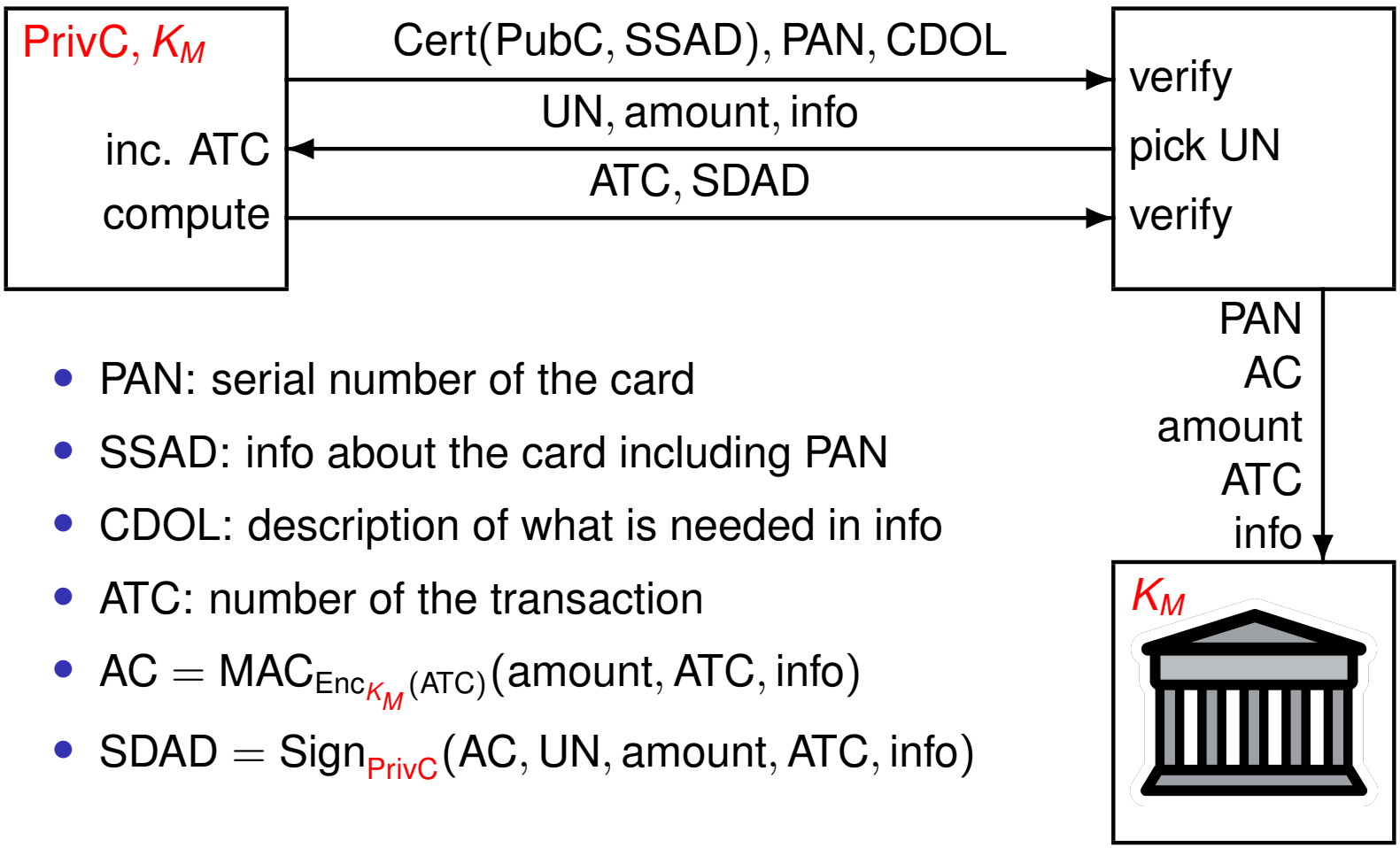
- $\texttt{SessionTicket} = \text{Enc}_{\text{STEK}}(\texttt{ResumptionSecret})$
- STEK: session ticket encryption key
- drawback: no forward secrecy
- drawback: replay attack

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

(Simplified) EMV PayPass Protocol (NFC)

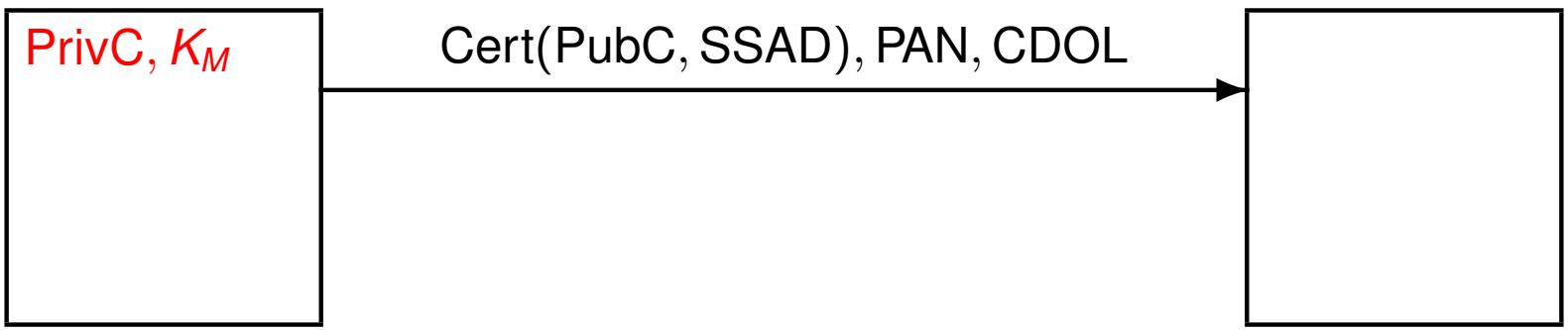


- PAN: serial number of the card
- SSAD: info about the card including PAN
- CDOL: description of what is needed in info
- ATC: number of the transaction
- $\mathrm{AC} = \mathrm{MAC}_{\mathrm{Enc}_{K_M}(\mathrm{ATC})}(\mathrm{amount}, \mathrm{ATC}, \mathrm{info})$
- $\mathrm{SDAD} = \mathrm{Sign}_{\mathrm{PrivC}}(\mathrm{AC}, \mathrm{UN}, \mathrm{amount}, \mathrm{ATC}, \mathrm{info})$

From Paper to Bits...

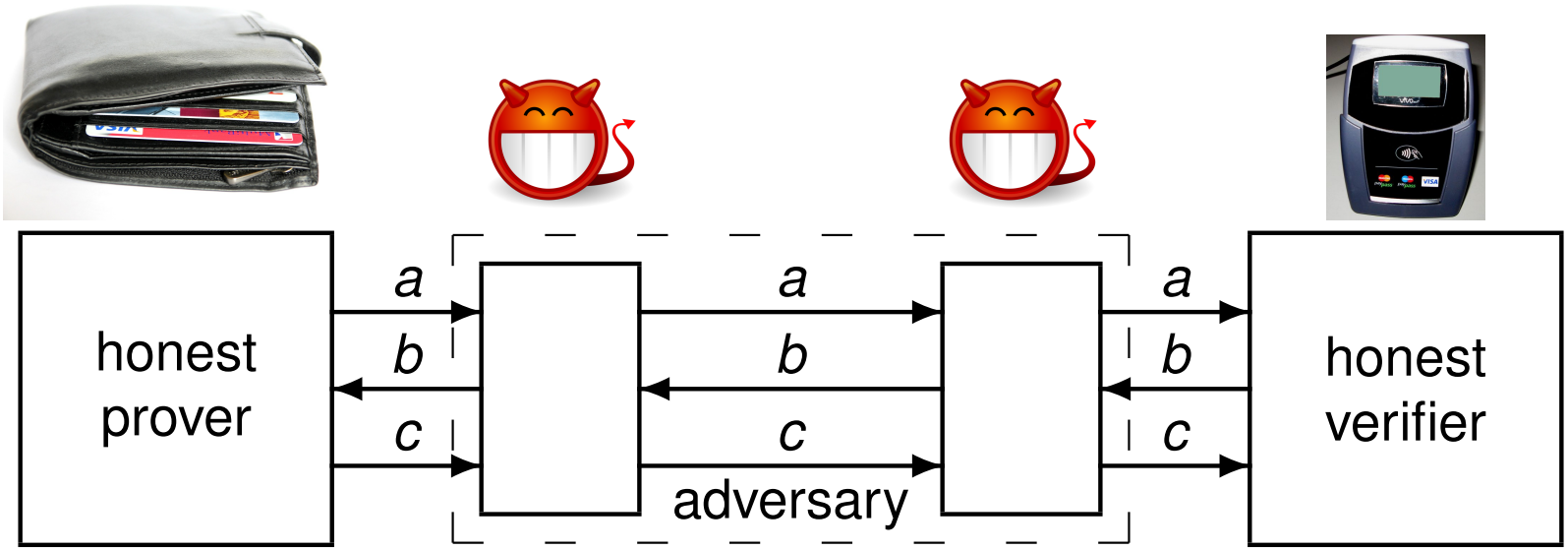
- holder is not aware a payment is happening
- holder is not aware of the payment amount
- no access control of the payment terminal (no PIN)
- payee is not authenticated (info could be anyone)
- privacy issue (SSAD leaks)

Skimming



get name on card, credit card number, expiration date, etc

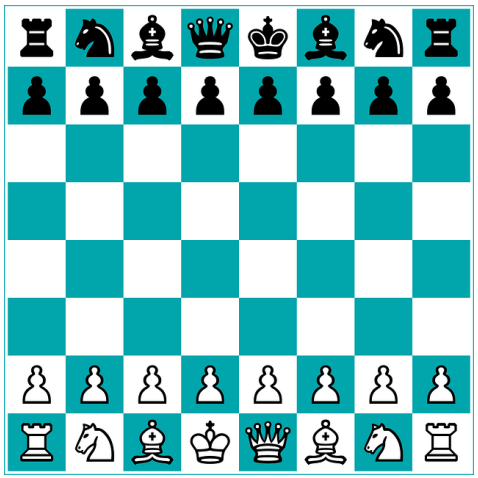
Relay Attacks

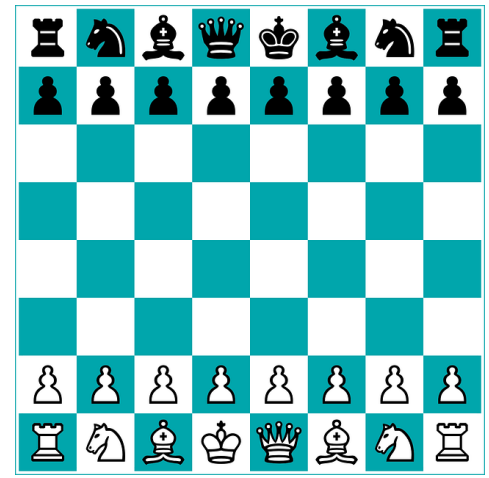


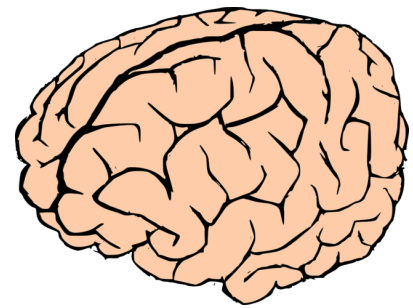
Relay Attacks in Real

- opening cars and ignition (key with no button)
- RFID access to buildings or hotel rooms
- toll payment system
- NFC credit card (for payment with no PIN)
- access to public transport
- ...

Playing against two Chess Grandmasters







back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport

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 fequency
- robustness, low complexity, low power, low cost

Bluetooth History

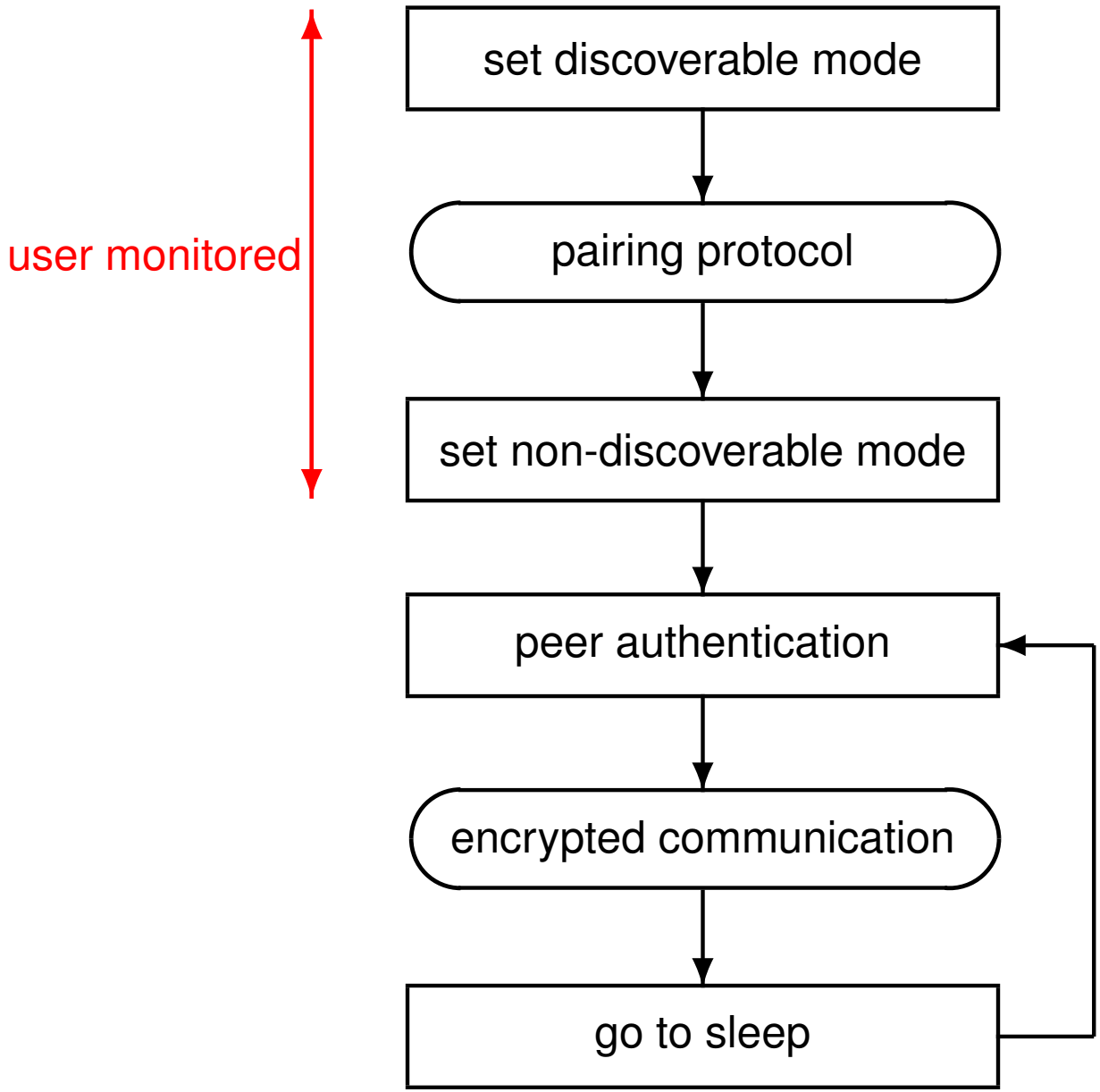
- 10th Century: Viking King Harald Blåtand (Harold Bluetooth) tried to unify Denmark, Norway, and Sweden
- 1994: Ericsson initiated a study to investigate the feasibility
- May 20, 1998: Bluetooth announced, controled by the Special Interest Group (SIG) formed by

  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)

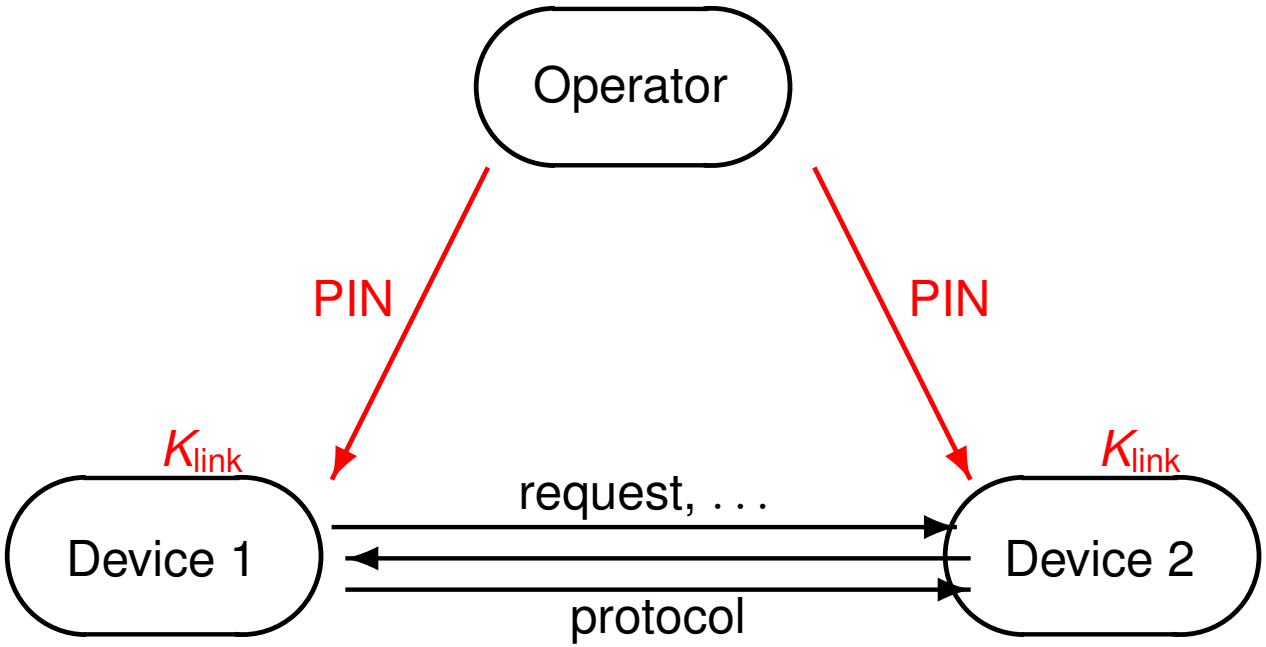
Bluetooth Security Basics (Link Level)

- 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

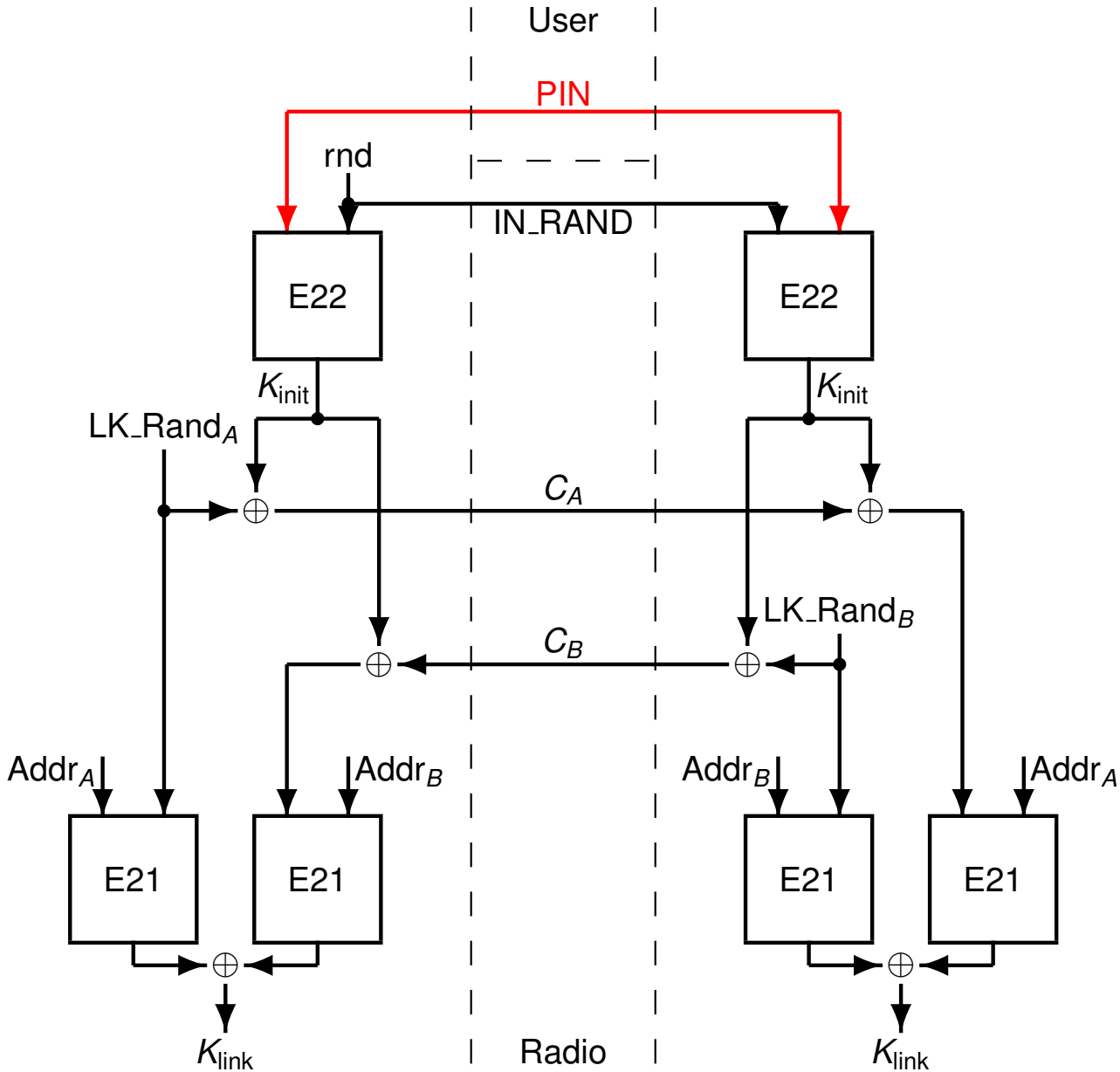
Cycles in Bluetooth



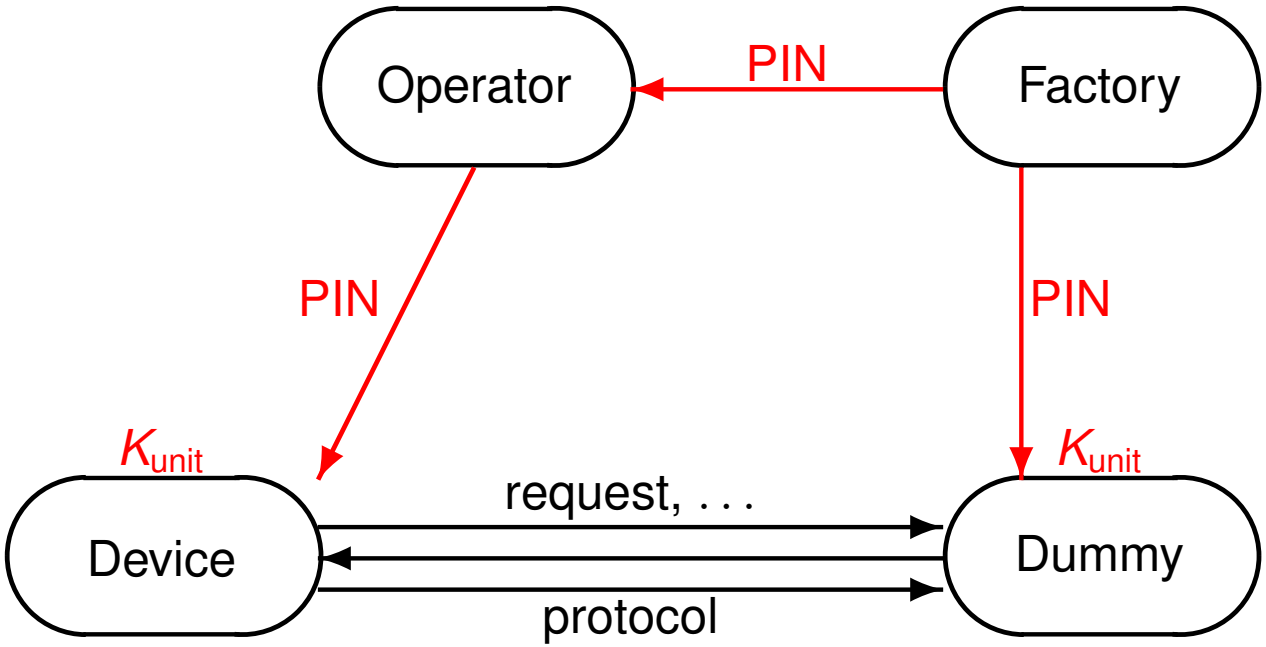
Device Pairing



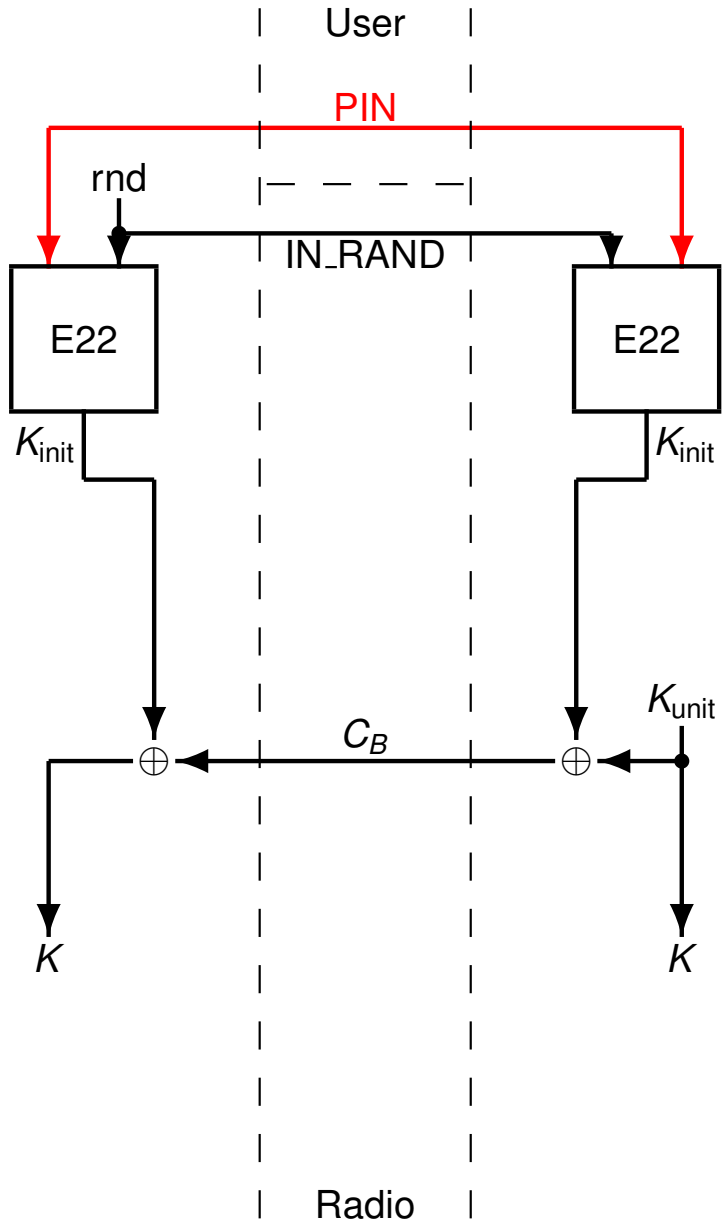
Legacy Pairing Protocol



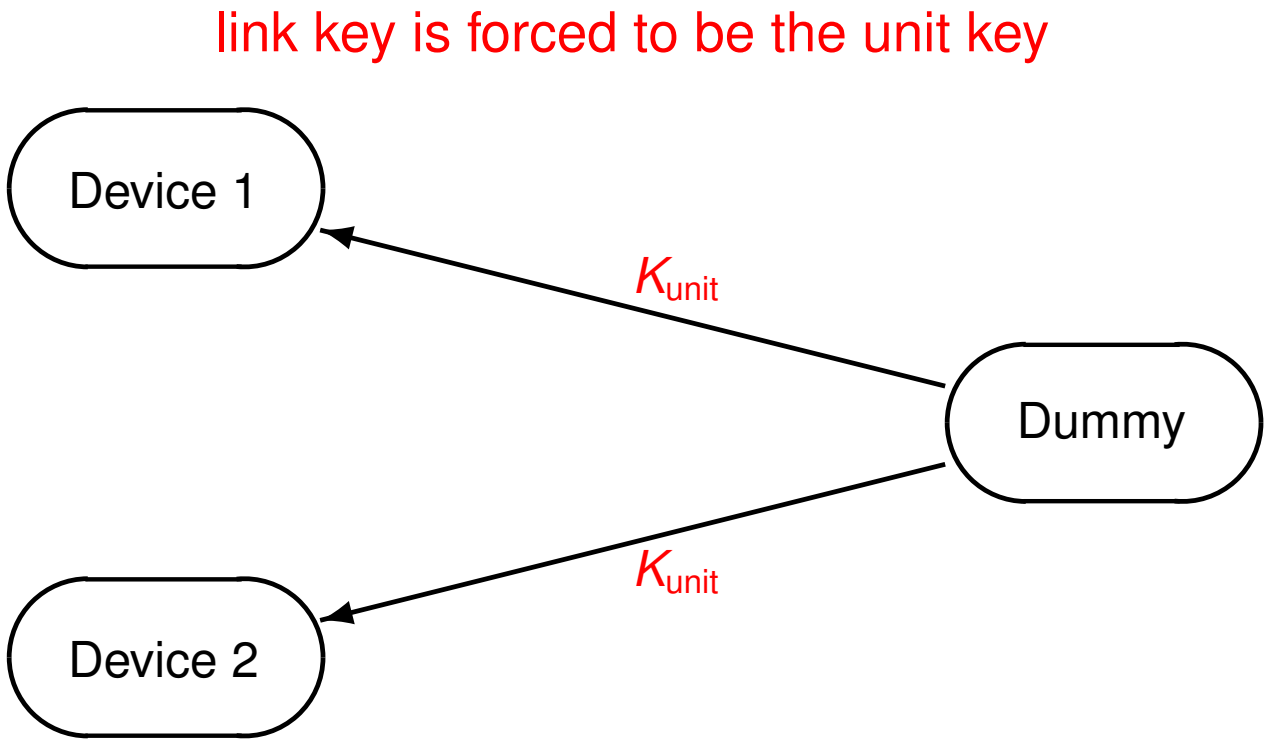
Pairing with a Dummy Device



Legacy Pairing with a Dummy Device

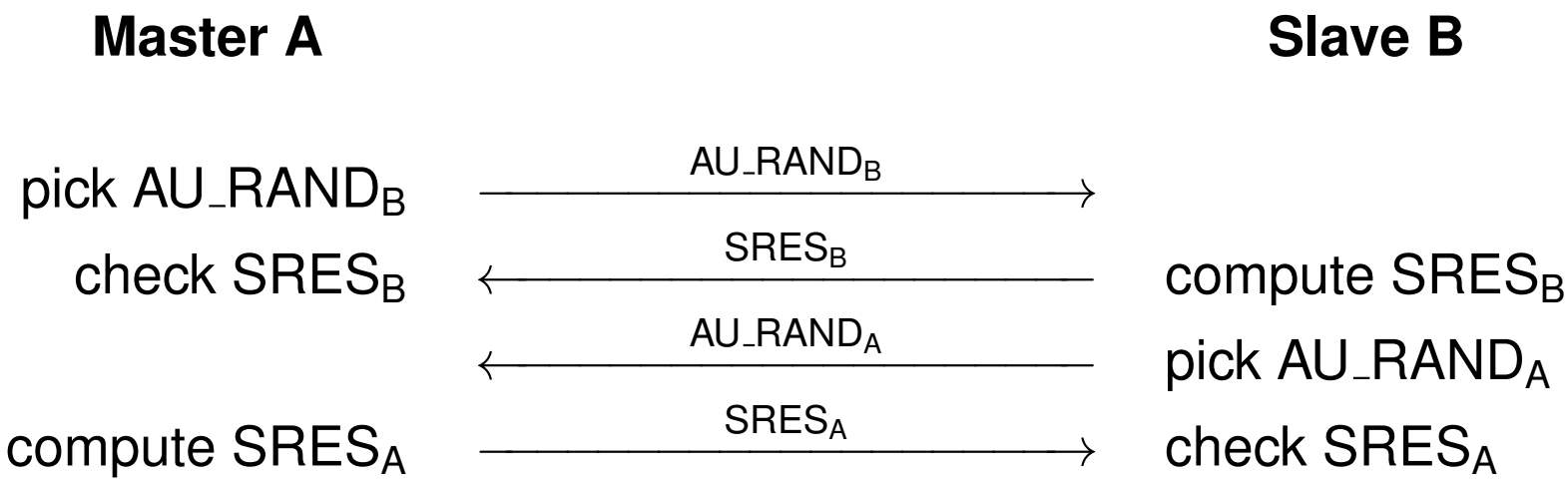


Dummy Devices: Unit Key is Shared with Many Devices



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



$$\mathrm{SRES}_d = \mathrm{E1}(K, \mathrm{AU\_RAND}_d, \mathrm{BD\_ADDR}_d)$$

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

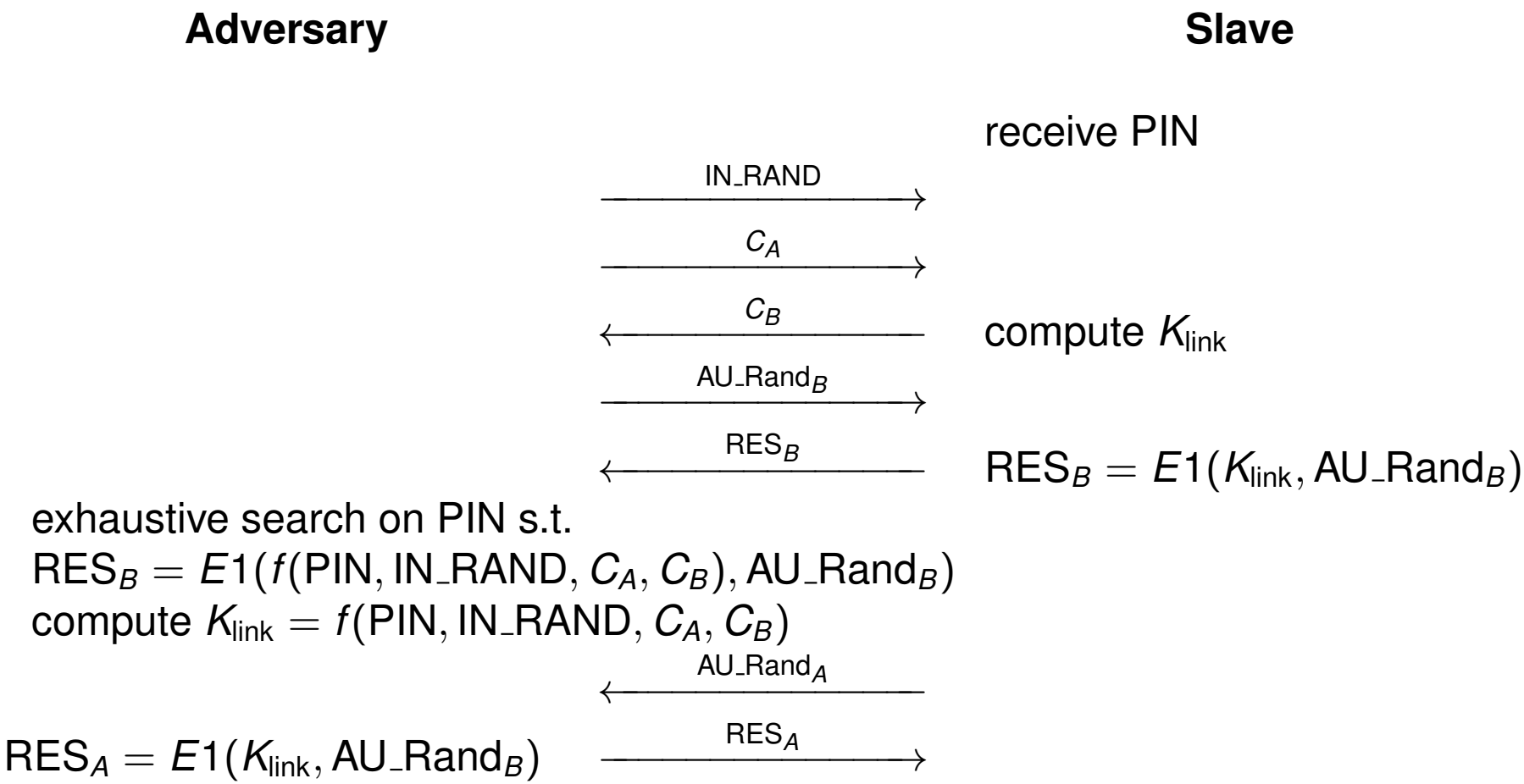
devastating sniffing attacks in other cases! (Jakobsson-Wetzel 2001)

Sniffing + Offline Attack

Assumption: pairing not made in a private environment (channel not confidential) and guessable PIN (lazzy operator)

1. sniff the pairing protocol, get IN_RAND, C_A, C_B
2. $\longrightarrow$ can compute K_{link} from PIN
3. sniff a peer-authentication protocol, get rand, $F(\text{rand}, K_{\text{link}})$
4. $\longrightarrow$ can check a guess on K_{link}
5. run an offline exhaustive search on PIN

Online Impersonation Attack



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 ☹
- 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

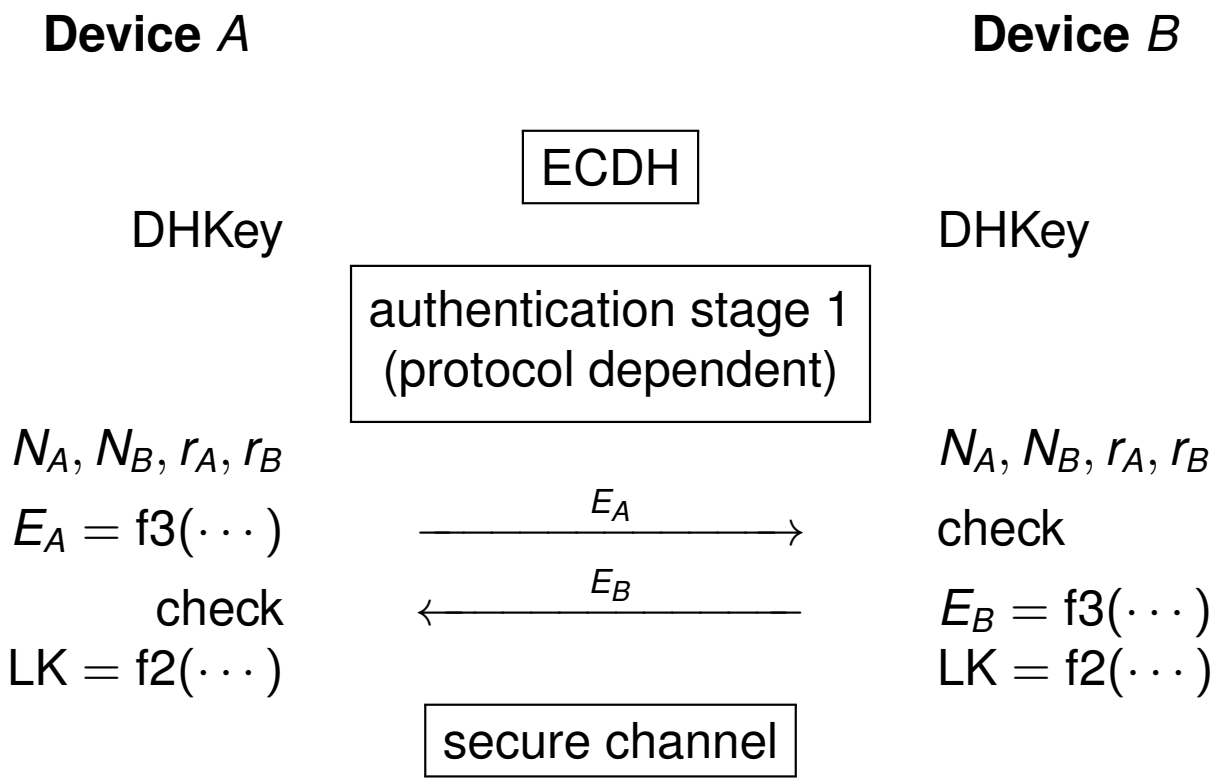
Bluetooth v2.1: Secure Simple Pairing

4 variants

- numeric comparison (assumes secure comparison by humans)
- passkey entry (assumes a secure PIN input by human)
- just works (assumes no active attack)
- out-of-band (assumes a secure channel, e.g. cable or near field communication)

resist active adversary
resist passive adversary only (out-of-band resists to active adversaries if no attack is possible 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

$$
\begin{aligned}
E_A &= \mathrm{f3}(\mathrm{DHKey}, N_A, N_B, r_B, \mathrm{IOcap}_A, \mathrm{BD\_ADDR}_A, \mathrm{BD\_ADDR}_B) \\
E_B &= \mathrm{f3}(\mathrm{DHKey}, N_B, N_A, r_A, \mathrm{IOcap}_B, \mathrm{BD\_ADDR}_B, \mathrm{BD\_ADDR}_A)
\end{aligned}
$$

- **step 12**: link key calculation
  key derivation from DHKey, Na, Nb, and the addresses

$$\mathrm{LK} = \mathrm{f2}(\mathrm{DHKey}, N_{\mathrm{master}}, N_{\mathrm{slave}}, \mathrm{btlk}, \mathrm{BD\_ADDR}_{\mathrm{master}}, \mathrm{BD\_ADDR}_{\mathrm{slave}})$$

- **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:

$$
\begin{aligned}
p &= 2^{192} - 2^{64} - 1 \\
a &= -3 \bmod p \\
b &= 2455155546008943817740293915197451784769108058161191238065 \\
r &= 6277101735386680763835789423176059013767194773182842284081 \\
G_x &= \texttt{188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012} \\
&= 602046282375688656758213480587526111916698976636884684818 \\
G_y &= \texttt{07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811} \\
&= 174050332293622031404857552280219410364023488927386650641
\end{aligned}
$$

  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

$$\text{DHKey} = \text{P192}(\text{SK}_A, \text{PK}_B) = \text{P192}(\text{SK}_B, \text{PK}_A)$$

The New Bluetooth Menagery

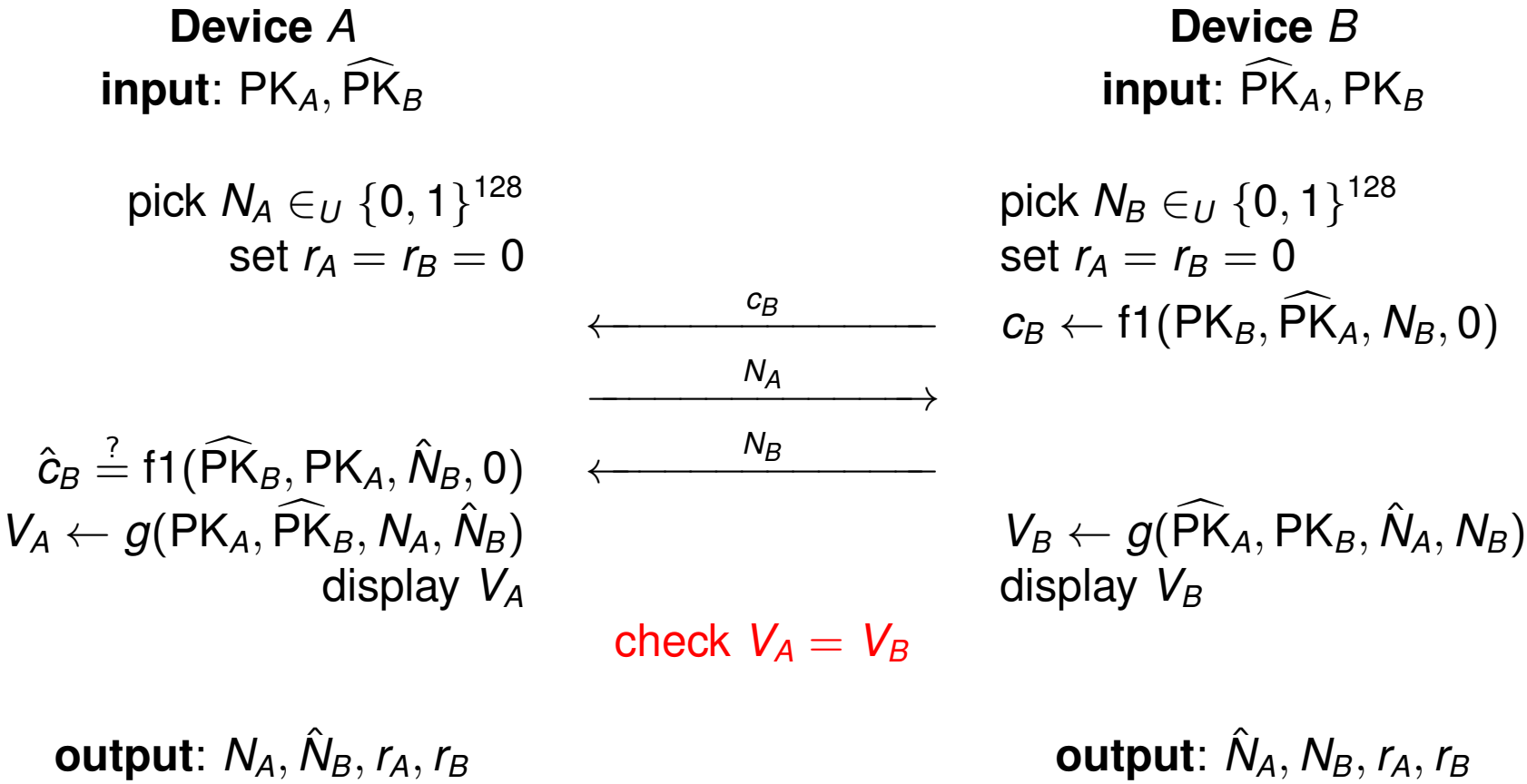
$$\begin{aligned}
\mathrm{f1}(U, V, X, Z) &= \mathrm{trunc}_{128}\left(\mathrm{HMAC}_X(U \| V \| Z)\right) \\
\mathrm{g}(U, V, X, Y) &= \mathrm{SHA256}(U \| V \| X \| Y) \bmod 2^{32} \\
\mathrm{f2}(W, N_1, N_2, \mathrm{keyID}, A_1, A_2) &= \mathrm{trunc}_{128}\left(\mathrm{HMAC}_W(N_1 \| N_2 \| \mathrm{keyID} \| A_1 \| A_2)\right) \\
\mathrm{f3}(W, N_1, N_2, R, \mathrm{IOcap}, A_1, A_2) &= \mathrm{trunc}_{128}\left(\mathrm{HMAC}_W(N_1 \| N_2 \| R \| \mathrm{IOcap} \| A_1 \| A_2)\right)
\end{aligned}$$

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



Numeric Comparison Analysis

Device A	**Adversary**	**Device** B
input: $\mathrm{PK}_A, \widehat{\mathrm{PK}}_B$		**input**: $\widehat{\mathrm{PK}}_A, \mathrm{PK}_B$
pick $N_A \in_U \{0,1\}^{128}$		pick $N_B \in_U \{0,1\}^{128}$
set $r_A = r_B = 0$		set $r_A = r_B = 0$
	$\xleftarrow{\hat{c}_B}$?? $\xleftarrow{c_B}$	$c_B \leftarrow \mathrm{f1}(\mathrm{PK}_B, \widehat{\mathrm{PK}}_A, N_B, 0)$
	$\xrightarrow{N_A}$?? $\xrightarrow{\hat{N}_A}$	
$\hat{c}_B \stackrel{?}{=} \mathrm{f1}(\widehat{\mathrm{PK}}_B, \mathrm{PK}_A, \hat{N}_B, 0)$	$\xleftarrow{\hat{N}_B}$?? $\xleftarrow{N_B}$	
$V_A \leftarrow g(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B, N_A, \hat{N}_B)$		$V_B \leftarrow g(\widehat{\mathrm{PK}}_A, \mathrm{PK}_B, \hat{N}_A, N_B)$
display V_A		display V_B
	check $V_A = V_B$	
output: $N_A, \hat{N}_B, r_A, r_B$		**output**: $\hat{N}_A, N_B, r_A, r_B$

if $(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B) \neq (\widehat{\mathrm{PK}}_A, \mathrm{PK}_B)$, due to commitment c_B and $\hat{c}_B$:
Adversary does not know V_B before he receives N_B
Adversary cannot influence V_A after sending $\hat{c}_B$

Note on Numerical Comparison

- presumably, not many human users will carefully compare the 32-bit strings V_A and V_B
- "**just works**" is a variant where no check is made (vulnerable to active attacks)

Bluetooth Simple Secure Pairing Variants — ii

Passkey Entry

Device A		**Device** B
input: $\mathrm{PK}_A, \widehat{\mathrm{PK}}_B$		**input**: $\widehat{\mathrm{PK}}_A, \mathrm{PK}_B$
	type $r_1 \cdots r_k$	
	FOR $i = 1$ to k	
pick $N_A \in_U \{0,1\}^{128}$		pick $N_B \in_U \{0,1\}^{128}$
$c_A \leftarrow \mathrm{f1}(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B, N_A, r_i)$	$\xrightarrow{c_A}$	
	$\xleftarrow{c_B}$	$c_B \leftarrow \mathrm{f1}(\mathrm{PK}_B, \widehat{\mathrm{PK}}_A, N_B, r_i)$
	$\xrightarrow{N_A}$	$\hat{c}_A \stackrel{?}{=} \mathrm{f1}(\widehat{\mathrm{PK}}_A, \mathrm{PK}_B, \hat{N}_A, r_i)$
$\hat{c}_B \stackrel{?}{=} \mathrm{f1}(\widehat{\mathrm{PK}}_B, \mathrm{PK}_A, \hat{N}_B, r_i)$	$\xleftarrow{N_B}$	
	ENDFOR	
	keep the last N_A and N_B	
output: $N_A, \hat{N}_B, r, r$		**output**: $\hat{N}_A, N_B, r, r$

Collision Attack on Passkey Entry

find $\hat{c}_B, N_0, N_1$ s.t. $\mathrm{f1}(\widehat{\mathrm{PK}}_B, \mathrm{PK}_A, N_0, 0) = \mathrm{f1}(\widehat{\mathrm{PK}}_B, \mathrm{PK}_A, N_1, 1) = \hat{c}_B$ (collision)

(2^{64} complexity)

Device A		**Device** B
input: $\mathrm{PK}_A, \widehat{\mathrm{PK}}_B$		**input**: $\widehat{\mathrm{PK}}_A, \mathrm{PK}_B$
	(type $r_1 \cdots r_k$) FOR $i = 1$ to k	
pick $N_A \in_U \{0,1\}^{128}$	pick $\hat{N}_A$	pick $N_B \in_U \{0,1\}^{128}$
$c_A \leftarrow \mathrm{f1}(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B, N_A, r_i)$	$\xrightarrow{c_A}$	$c_B \leftarrow \mathrm{f1}(\mathrm{PK}_B, \widehat{\mathrm{PK}}_A, N_B, r_i)$
	$\xleftarrow{\hat{c}_B}$	
	$\xrightarrow{N_A}$	
$\hat{c}_B \stackrel{?}{=} \mathrm{f1}(\widehat{\mathrm{PK}}_B, \mathrm{PK}_A, \hat{N}_B, r_i)$	$\xleftarrow{N_{r_i}}$	
	deduce	$\xrightarrow{\hat{c}_A}$
		$\xleftarrow{c_B}$
		$\xrightarrow{\hat{N}_A}$
		$\hat{c}_A \stackrel{?}{=} \mathrm{f1}(\widehat{\mathrm{PK}}_A, \mathrm{PK}_B, \hat{N}_A, r_i)$
		$\xleftarrow{N_B}$
	ENDFOR	

deduce:
deduce bit r_i s.t. $c_A = \mathrm{f1}(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B, N_A, r_i)$
set $\hat{c}_A = \mathrm{f1}(\widehat{\mathrm{PK}}_A, \mathrm{PK}_B, \hat{N}_A, r_i)$

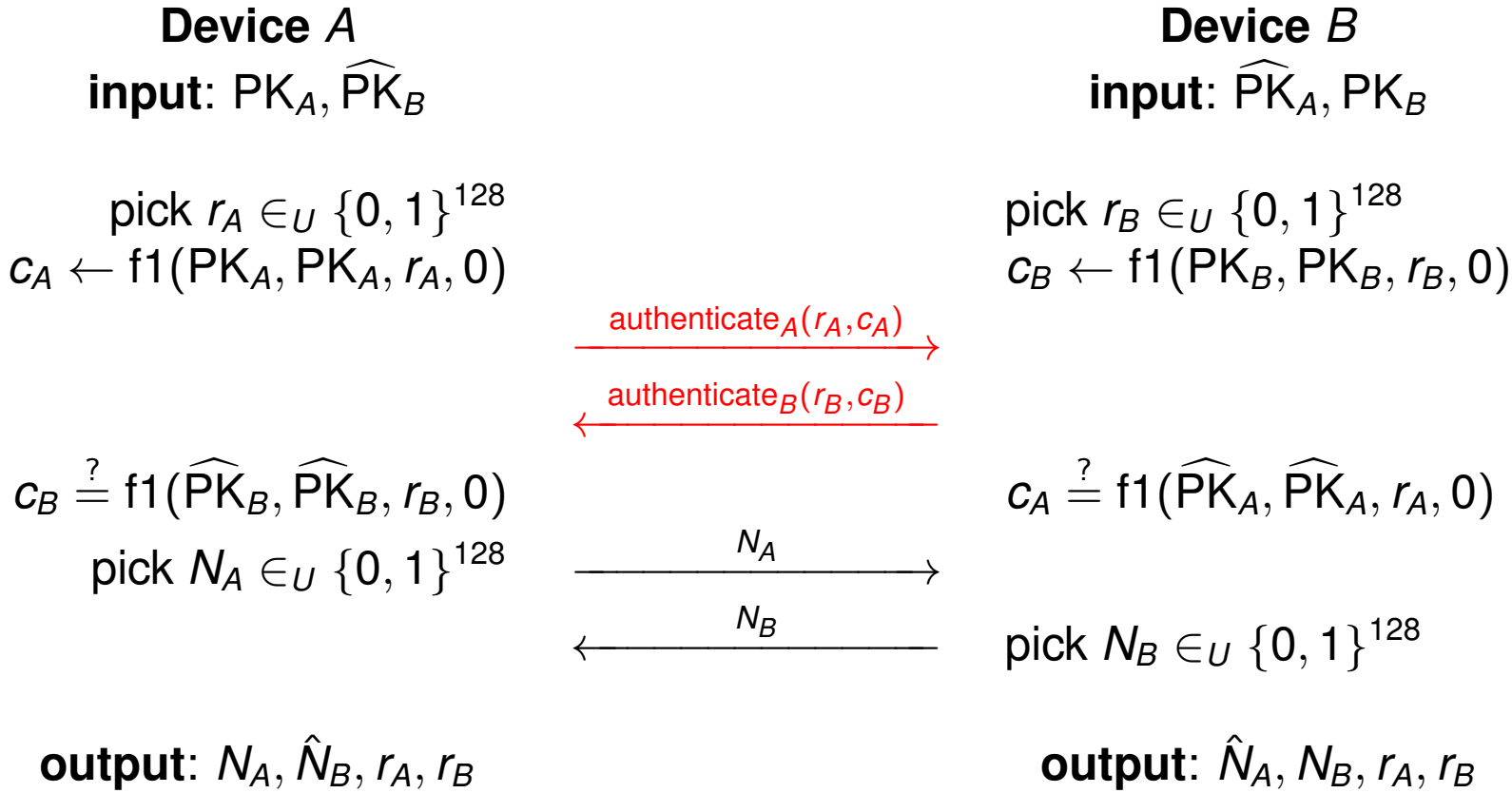
Pass Entry Analysis

If $(\mathrm{PK}_A, \widehat{\mathrm{PK}}_B) \neq (\widehat{\mathrm{PK}}_A, \mathrm{PK}_B)$ and f is collision-resistant:

- 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



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

back to chapter

9 Case Studies

- WiFI: WEP/WPA/WPA2
- Block Chains
- Mobile Telephony
- Signal
- TLS
- NFC Creditcard Payment
- Bluetooth
- The Biometric Passport


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

- $\rightarrow$ biometrics
- $\rightarrow$ contactless IC chip
- $\rightarrow$ digital signature + PKI

maintained by UN/ICAO (International Civil Aviation Organization)

MRTD History

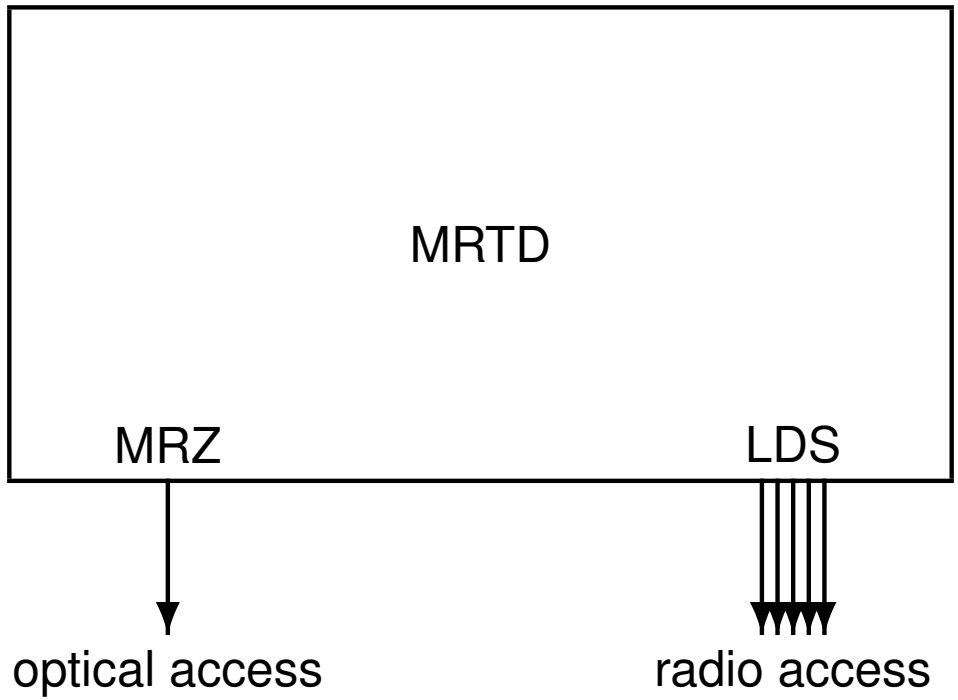
- 1968: ICAO starts working on MRTD
- 1980: first standard (OCR-B **Machine Readable Zone (MRZ)**)
- 1997: ICAO-NTWG (New Tech. WG) starts working on biometrics
- 2001 9/11: US want to speed up the process
- 2002 resolution: ICAO adopts **facial recognition** (+ optional fingerprint and iris recognition)
- 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

MRZ Example

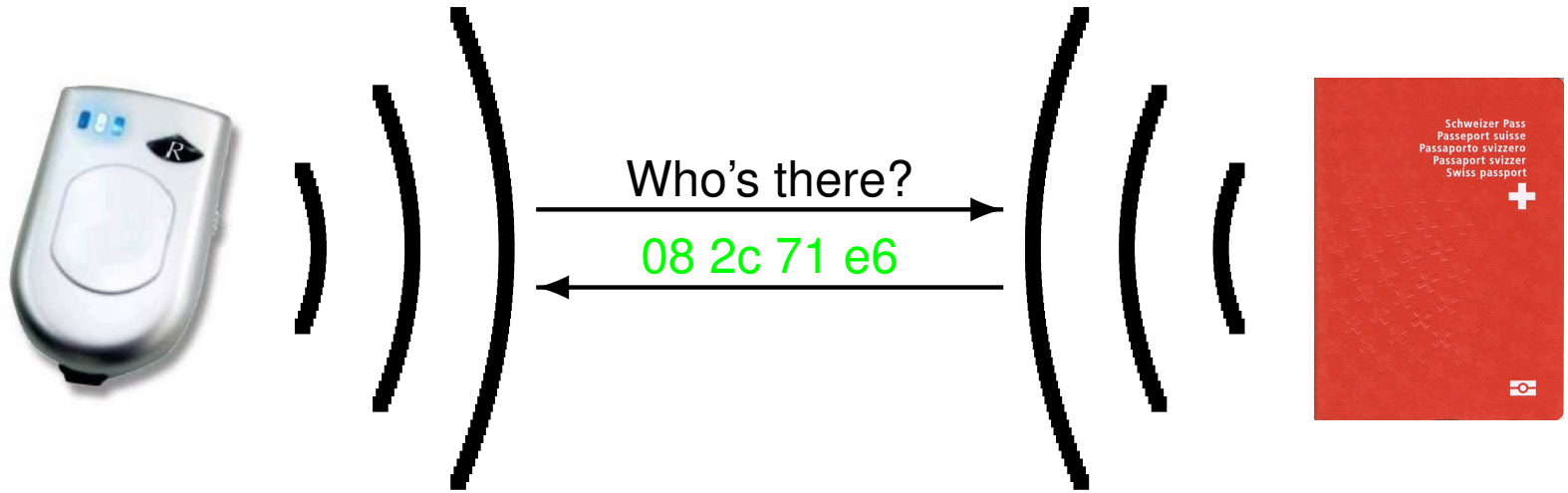
```
PMFRADUPONT<<<<JEAN<<<<<<<<<<<<<<<<<<<<<<<<
74HK8215<6CHE7304017M0705121<<<<<<<<<<<<<<03
```

- document type
- issuing country
- holder name
- doc. number + CRC
- nationality
- date of birth + CRC
- gender
- date of expiry + CRC
- options + CRC

MRTD in a Nutshell

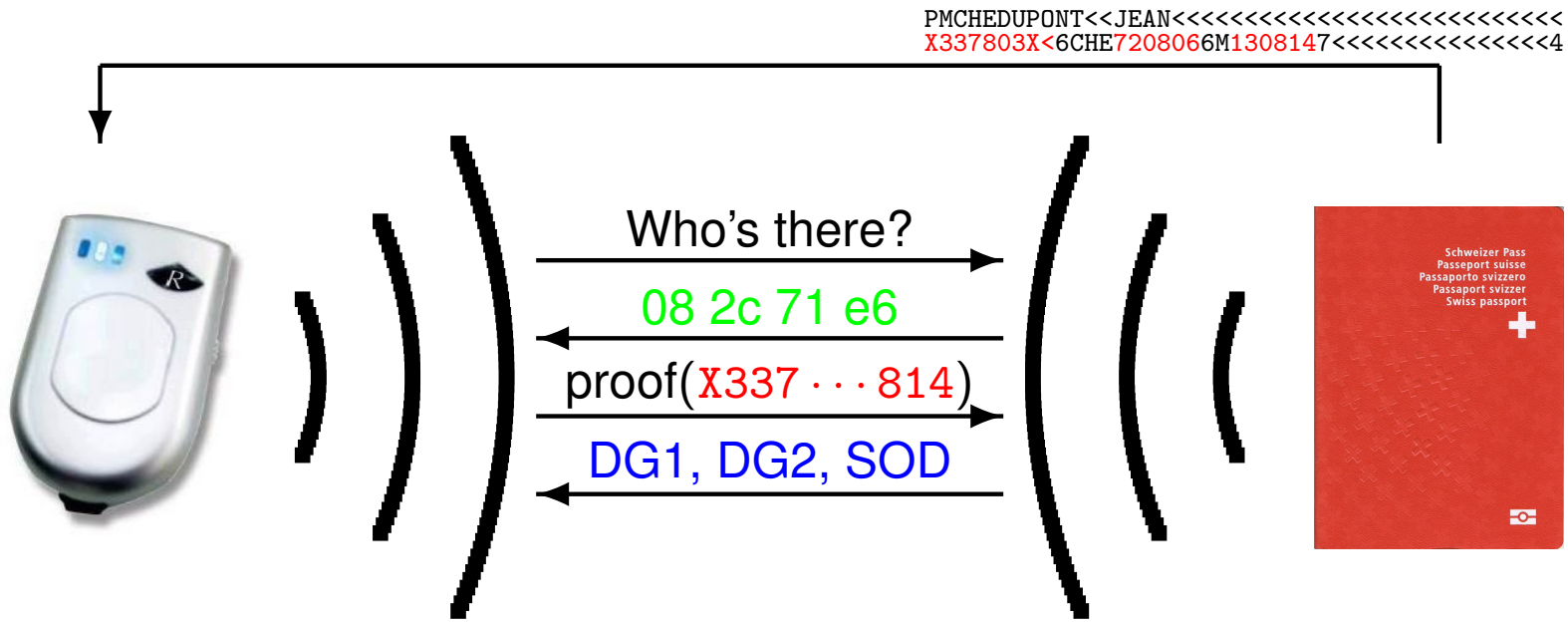


ISO 14443 (RFID)



- frequency: 13.56MHz
- typical range: 2cm
- reported range (with legal equipment): 12m

ICAO (MRTD): BAC and Passive Authentication



- DG1: official name, citizenship, X337 · · · 814, gender
- DG2: facial picture
- SOD: signature by authorities of the hash of DG's

Identity Example

DG1

```
PMCHEDUPONT<<JEAN<<<<<<<<<<<<<<<<<<<<<<<<<<<<
X337803X<6CHE7208066M1308147<<<<<<<<<<<<<<4
```

DG2

SOD

Hashes:

DG1: `4e1249fb72c8e70ba72f488dc1f91394e57f9f83`

DG2: `a3853c3c7261c2788fc2c4b9db372c5875f5c91d`

Signature:

```
54a4 a626 4ee1 c0ab e022 3f1d e673 75d4
7c89 7e7f d8fb acd6 abbf d568 b178 7171
652d e730 43c2 9495 6134 680c 7070 9028
1caa 2364 17e8 ffa0 9ee7 c8be 4c32 908c
```

Certificate:

```
MIIECTCCA5GgAwIBAgIBFDAJBgcqhkjOPQQBMHExCzAJBgNVBAYTAkNIMQ4wDAYD
VQQKEwVBZG1pbjERMA8GA1UECxMIU2VydmljZXMxIjAgBgNVBAsTGUNlcnRpZmlj
YXRpb24gQXV0aG9yaXRpZXMxGzAZBgNVBAMTEmNzY2Etc3dpdHplcmxhbmQtMTAe
Fw0wODA1MTkwODA4NDVaFw0xNDA2MjEwODA4NDVaMG0xCzAJBgNVBAYTAkNIMQ4w
DAYDVQQKEwVBZG1pbjERMA8GA1UECxMIU2VydmljZXMxGTAXBgNVBAsTEFNpZ25h
dHVyZS1TZXJ2ZXIxDzANBgNVBAsTBlBhc3MwNjEPMA0GA1UEAxMGZHMtMDAxMIIB
MzCB7AYHKoZIzj0CATCB4AIBATAsBgcqhkjOPQEBAiEA/////wAAAAEAAAAAAAAA
AAAAAD///////////////8wRAQg/////wAAAAEAAAAAAAAAAAAAAD////////
//////wEIFrGNdiqOpPns+u9VXaYhrxlHQawzFOw9jvOPD4n0mBLBEEEaxfR8uEs
Qkf4vOblY6RA8ncDfYEt6zOg9KE5RdiYwpZP4OLi/hp/m47n60p8D54WK84zV2sx
Xs7LtkBoN79R9QIhAP////8AAAAA//////////+85vqtpxeehPO5ysL8YyVRAgEB
A0IABO8J8UthgahfN1JQKIq9a1ll/L3er54mUd1SZMnKQ2pQTbX5JwHc9ByEgw3G
5kucfGw1k2uAts+Ck+WSovy7k7GjggFBMIIBPTArBgNVHRAEJDAigA8yMDA4MDUx
OTA4MDg0NVqBDzIwMDgwODIwMDgwODQ1WjBgBgNVHSAEWTBXMFUGCGCFdAERAzQB
MEkwRwYIKwYBBQUHAgEWO2h0dHA6Ly93d3cucGtpLmFkbWluLmNoL3BvbGljeS9D
UFNfMl8xNl83NTZfMV8xN18zXzUyXzEucGRmMIGbBgNVHSMEgZMwgZCAFE7InZjJ
tOCQ9StbhZdQVr/oJOt2oXWkczBxMQswCQYDVQQGEwJDSDEOMAwGA1UEChMFQWRt
aW4xETAPBgNVBAsTCFNlcnZpY2VzMSIwIAYDVQQLExlDZXJ0aWZpY2F0aW9uIEF1
dGhvcml0aWVzMRswGQYDVQQDExJjc2NhLXN3aXR6ZXJsYW5kLTGCAQEwDgYDVR0P
AQH/BAQDAgeAMAkGByqGSM49BAEDZwAwZAIwGYMbTqjlYQnJ1DSpb//5WtQthjoy
pGrbBZW1Rqa7TXffzQX818OjQCdQOn9tZEDlAjBPtMdS9OymxywZpXZj9Os2qO6M
6htXJKXpdKSWq75ZhQRet/or3pT2MQ56n69hqGw=
```

MRTD

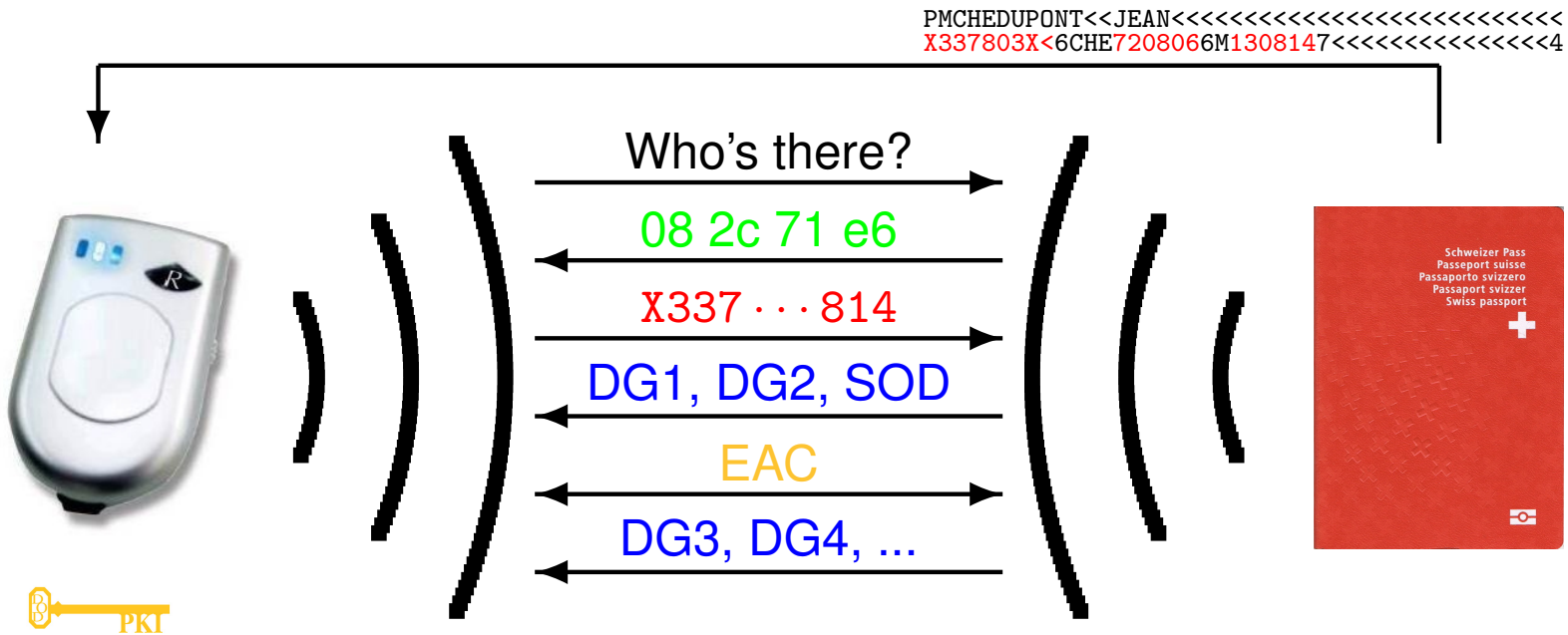
Advantages

- impossible to forge an identity
- protect against non-organized illegal immigration

Problems

- encourage identity theaft
- facial recognition is weakly reliable
- passeport cloning
- tracking people
- leakage of evidence
  - proof of official name
  - proof of age
  - proof of gender
- anonymity loss

EAC: Access Control and Active Authentication



- EAC: chip authentication
- EAC: terminal authentication
- DG3...: fingerprint, other data

EAC

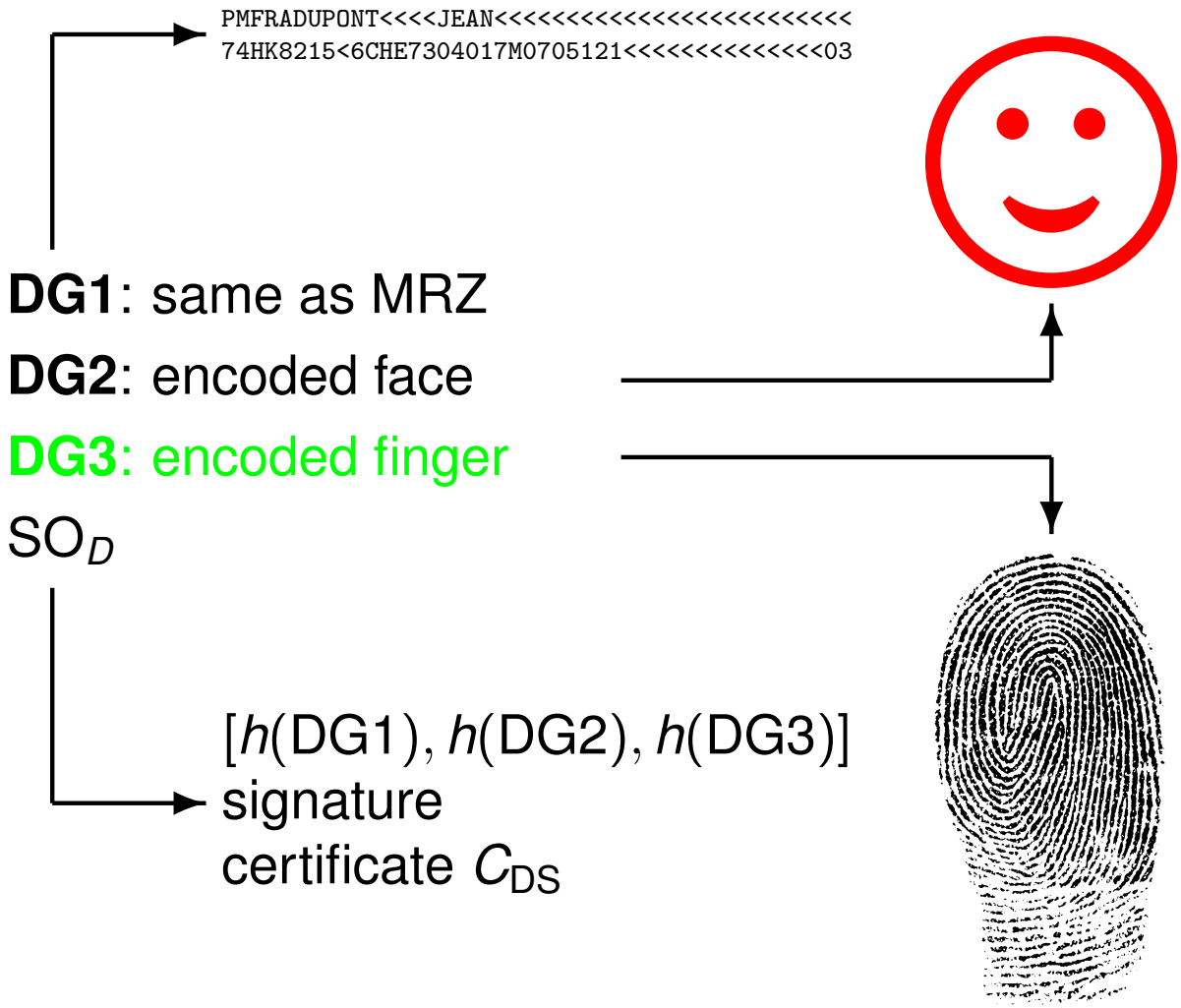
Advantages

- anti-cloning
- better access control
- better identification

Problems

- only where EAC is available
- still evidence leakages
- a new PKI

LDS Example


PMFRADUPONT<<<<JEAN<<<<<<<<<<<<<<<<<<<<<<<<
74HK8215<6CHE7304017M0705121<<<<<<<<<<<<<<03
DG1: same as MRZ
DG2: encoded face
DG3: encoded finger
SO_D
[h(DG1), h(DG2), h(DG3)]
signature
certificate C_DS

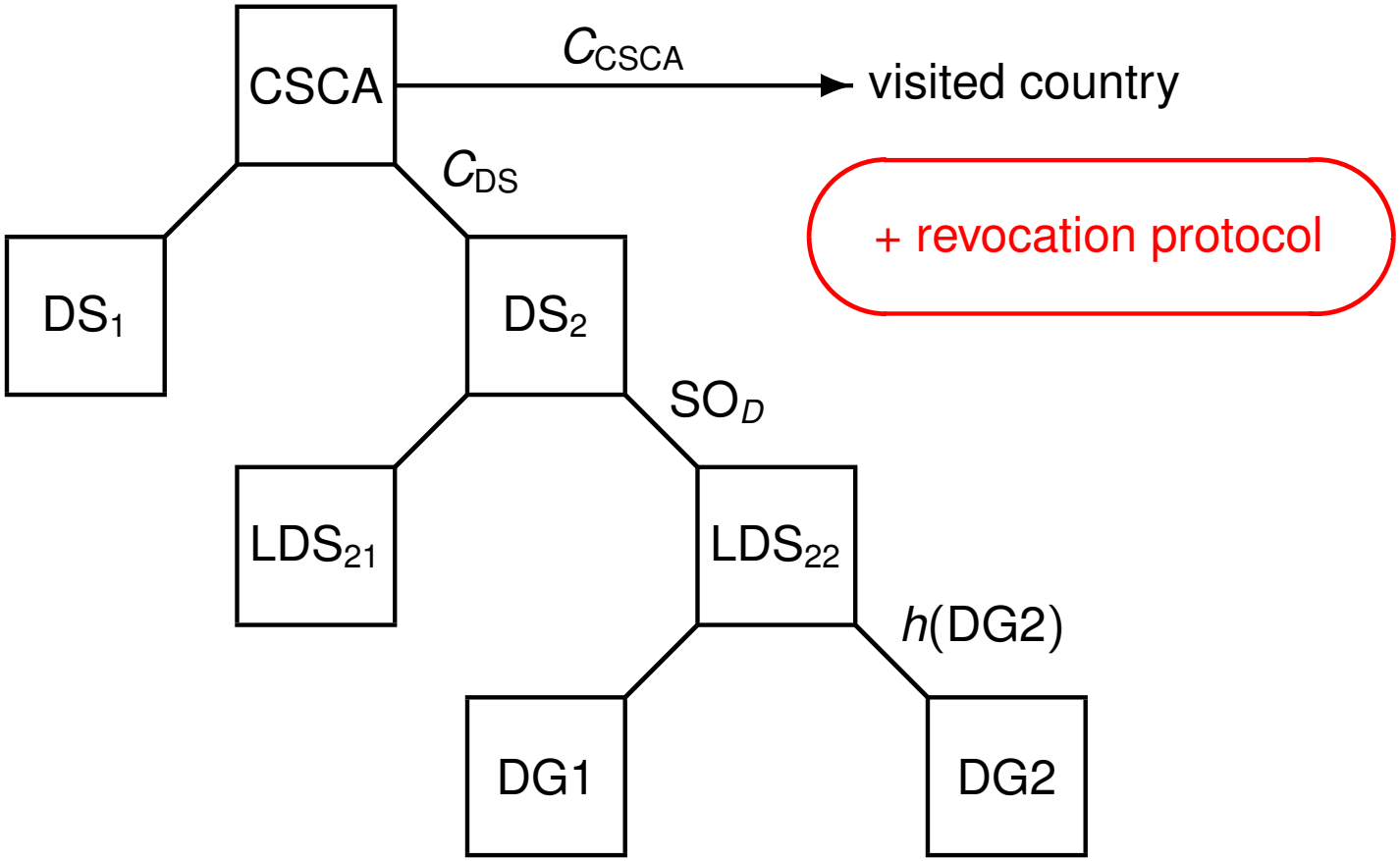
LDS Structure

- 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: KPu_{AA}
- *DG16: person(s) to notify*
- SO_D

SO_D 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

Some MRTD Security Notions

- **Passive authentication**: authentication of the DG by means of a digital signature and a PKI
- **Basic access control**: access control to the chip based on a (printed) MRZ_info
- **Secure messaging**: secure communication between chip and terminal
- **Active authentication**: interactive authentication of the chip using a public key
- **Terminal authentication**: authentication of the terminal by means of a PKI
- **Chip authentication**: replacement of active authentication
- **Extended access control**: use of teminal authentication, chip authentication, and PACE

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

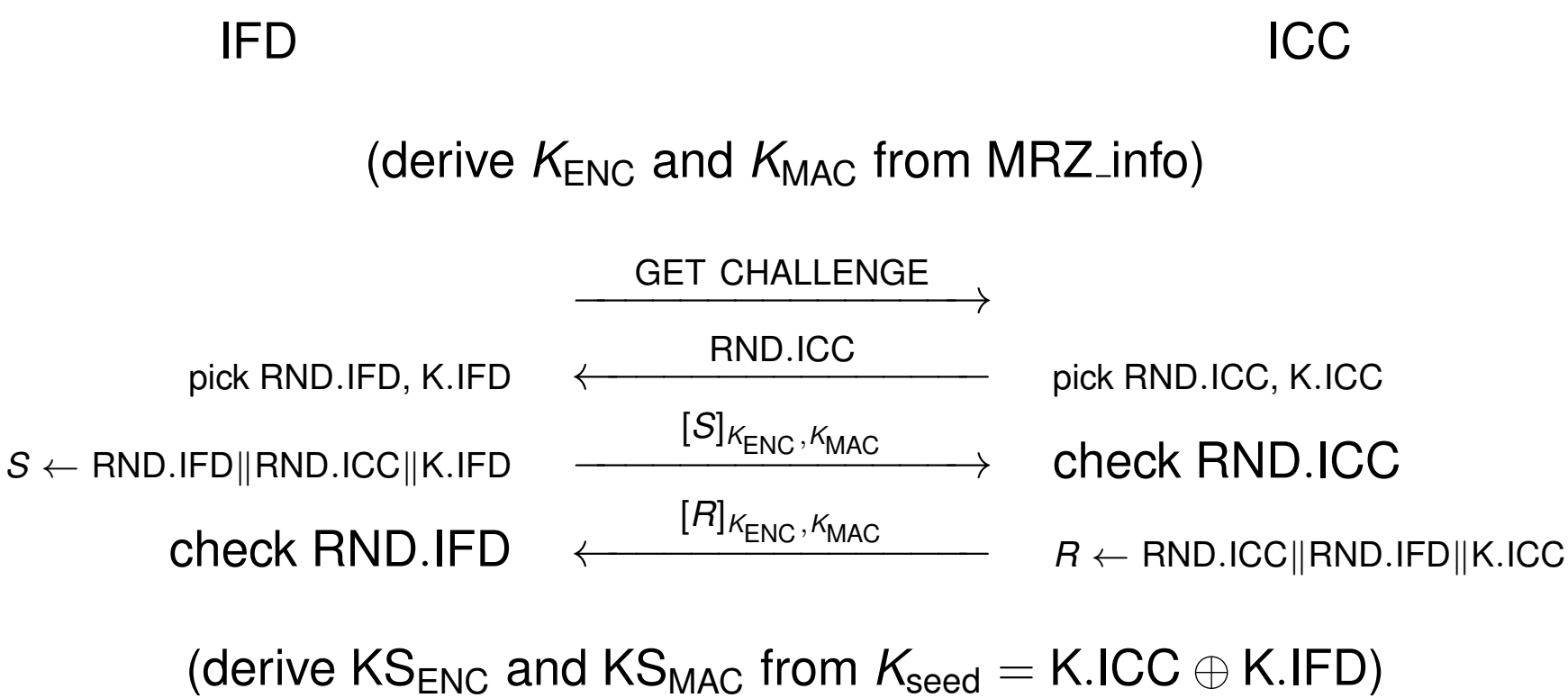
MRZ_info

```
PMFRADUPONT<<<<JEAN<<<<<<<<<<<<<<<<<<<<<<<<
74HK8215<6CHE7304017M0705121<<<<<<<<<<<<<<03
```

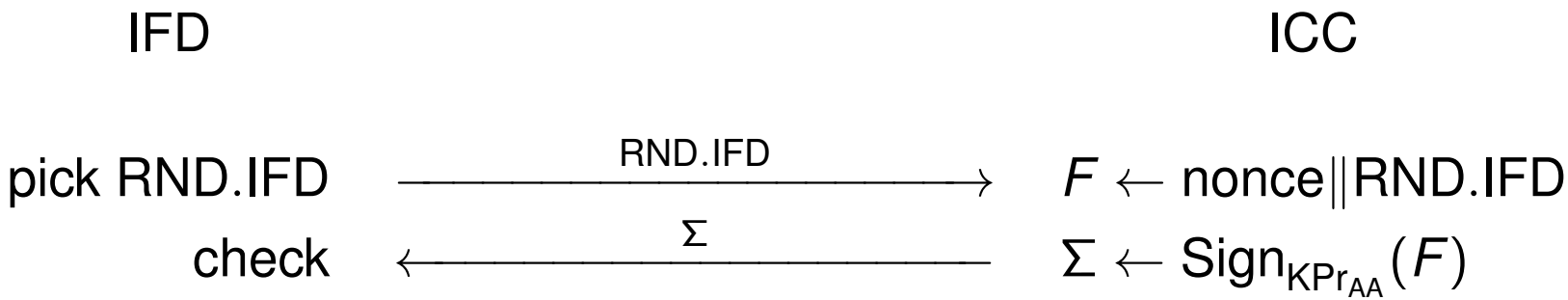
- document type
- issuing country
- holder name
- doc. number + CRC
- nationality
- date of birth + CRC
- gender
- date of expiry + CRC
- options + CRC

Basic Access Control

Authenticated Key Exchange Based on MRZ_info



Active Authentication Protocol



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`

Security and Privacy Issues

- collision avoidance discrepancies
  $\rightarrow$ deviating from standard induce leakages
- MRZ_info entropy
  $\rightarrow$ online attack or offline decryption from skimming
- underestimated wireless range limits
  $\rightarrow$ claimed to be possible at a distance of 25m
- identity theft (by stealing/cloning MRTD)
  $\rightarrow$ facial recognition is weak
- remote passport detection
  $\rightarrow$ nice to find passports to steal
- relay attacks
- denial of services
- ...

Identity Theft

biometry

picture

steal

identity

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 Inspection

Advanced Inspection

run PACE (or BAC)
start secure messaging
↓
(if not in PACE) run Chip Auth.
restart secure messaging
↓
passive auth. of SO_D
↓
(optional) run AA
↓
run Terminal Authentication v1
↓
read and verify data

Basic Inspection

run PACE (or BAC)
start secure messaging
↓
passive auth. of SO_D
↓
(optional) run AA
↓
read and verify basic data

PACE (GM v2)

- better protocol (than BAC) based on π = MRZinfo
- can play the role of Chip Authentication

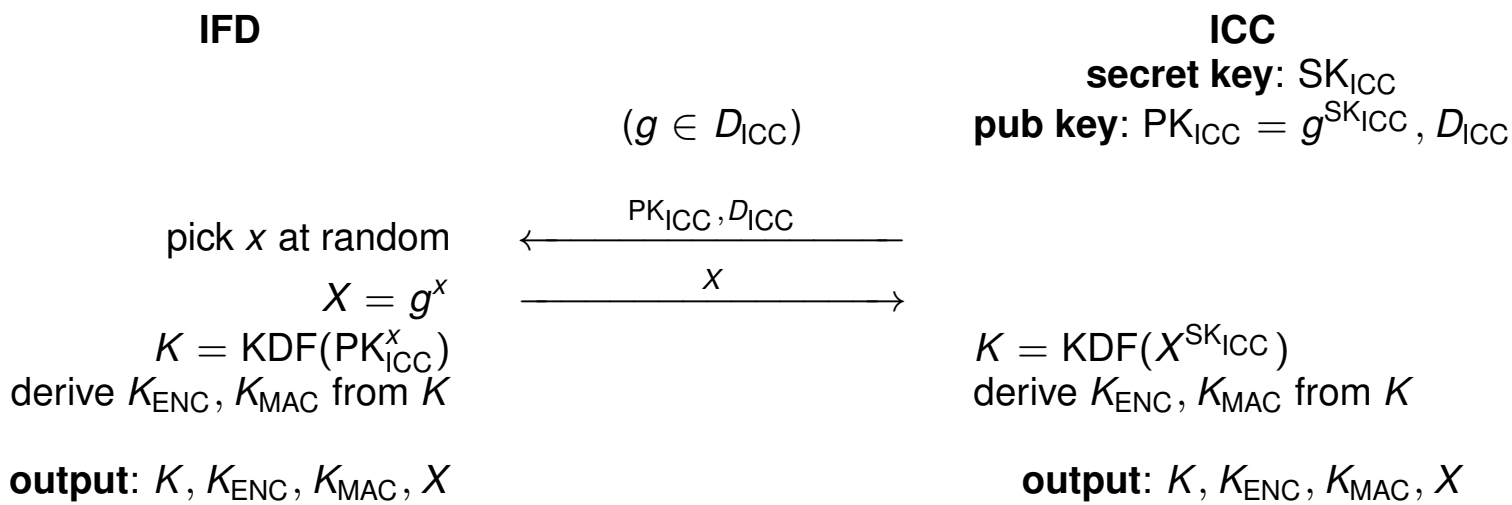
PCD **password**: π		**IC** **password**: π **secret key**: $\mathrm{SK}_{\mathrm{IC}}$ **pub key**: $\mathrm{PK}_{\mathrm{IC}} = g^{\mathrm{SK}_{\mathrm{IC}}}, D_{\mathrm{IC}}$
	$(g \in D_{\mathrm{ICC}})$	
pick s at random	$\xleftarrow{\mathrm{PK}_{\mathrm{IC}}, D_{\mathrm{IC}}}$	
$z = \mathrm{ENC}_{K_\pi}(s)$	$\xrightarrow{z}$	$s = \mathrm{DEC}_{K_\pi}(z)$
pick $\mathrm{SK}_{\mathrm{MAP,PCD}}$, $\mathrm{PK}_{\mathrm{MAP,PCD}} = g^{\mathrm{SK}_{\mathrm{MAP,PCD}}}$	$\xrightarrow{\mathrm{PK}_{\mathrm{MAP,PCD}}}$	pick $\mathrm{SK}_{\mathrm{MAP,IC}}$, $\mathrm{PK}_{\mathrm{MAP,IC}} = g^{\mathrm{SK}_{\mathrm{MAP,IC}}}$
$\hat{g} = g^s \mathrm{PK}_{\mathrm{MAP,IC}}^{\mathrm{SK}_{\mathrm{MAP,PCD}}}$	$\xleftarrow{\mathrm{PK}_{\mathrm{MAP,IC}}}$	$\hat{g} = g^s \mathrm{PK}_{\mathrm{MAP,PCD}}^{\mathrm{SK}_{\mathrm{MAP,IC}}}$
pick $\mathrm{SK}_{\mathrm{DH,PCD}}$, $\mathrm{PK}_{\mathrm{DH,PCD}} = \hat{g}^{\mathrm{SK}_{\mathrm{DH,PCD}}}$	$\xrightarrow{\mathrm{PK}_{\mathrm{DH,PCD}}}$	pick $\mathrm{SK}_{\mathrm{DH,IC}}$, $\mathrm{PK}_{\mathrm{DH,IC}} = \hat{g}^{\mathrm{SK}_{\mathrm{DH,IC}}}$
$K = \mathrm{PK}_{\mathrm{DH,IC}}^{\mathrm{SK}_{\mathrm{DH,PCD}}}$	$\xleftarrow{\mathrm{PK}_{\mathrm{DH,IC}}}$	$K = \mathrm{PK}_{\mathrm{DH,PCD}}^{\mathrm{SK}_{\mathrm{DH,IC}}}$
derive $\mathrm{KS}_{\mathrm{ENC}}, \mathrm{KS}_{\mathrm{MAC}}$ from K		derive $\mathrm{KS}_{\mathrm{ENC}}, \mathrm{KS}_{\mathrm{MAC}}$ from K
$T_{\mathrm{PCD}} = \mathrm{MAC}_{\mathrm{KS}_{\mathrm{MAC}}}(\mathrm{PK}_{\mathrm{DH,PCD}})$	$\xrightarrow{T_{\mathrm{PCD}}}$	check T_{PCD}
check T_{IC}	$\xleftarrow{T_{\mathrm{IC}}}$	$T_{\mathrm{IC}} = \mathrm{MAC}_{\mathrm{KS}_{\mathrm{MAC}}}(\mathrm{PK}_{\mathrm{DH,IC}})$
$\mathrm{CA}_{\mathrm{IC}} = \mathrm{DEC}_{\mathrm{KS}_{\mathrm{ENC}}}(A_{\mathrm{IC}})$, check $\mathrm{CA}_{\mathrm{IC}}$	$\xleftarrow{A_{\mathrm{IC}}}$	$\mathrm{CA}_{\mathrm{IC}} = \frac{\mathrm{SK}_{\mathrm{MAP,IC}}}{\mathrm{SK}_{\mathrm{IC}}}$, $A_{\mathrm{IC}} = \mathrm{ENC}_{\mathrm{KS}_{\mathrm{ENC}}}(\mathrm{CA}_{\mathrm{IC}})$
output: $\mathrm{KS}_{\mathrm{ENC}}, \mathrm{KS}_{\mathrm{MAC}}, X = \mathrm{PK}_{\mathrm{DH,PCD}}$		**output**: $\mathrm{KS}_{\mathrm{ENC}}, \mathrm{KS}_{\mathrm{MAC}}, X = \mathrm{PK}_{\mathrm{DH,PCD}}$

check $\mathrm{CA}_{\mathrm{IC}}$: $\mathrm{PK}_{\mathrm{IC}}^{\mathrm{CA}_{\mathrm{IC}}} \stackrel{?}{=} \mathrm{PK}_{\mathrm{MAP,IC}}$

Chip Authentication (if not in PACE)

- chip has a static Diffie-Hellman key in DG14 (SOD-authenticated)
- semi-static ECDH with domain parameters D_{ICC}
- replace the secure messaging keys

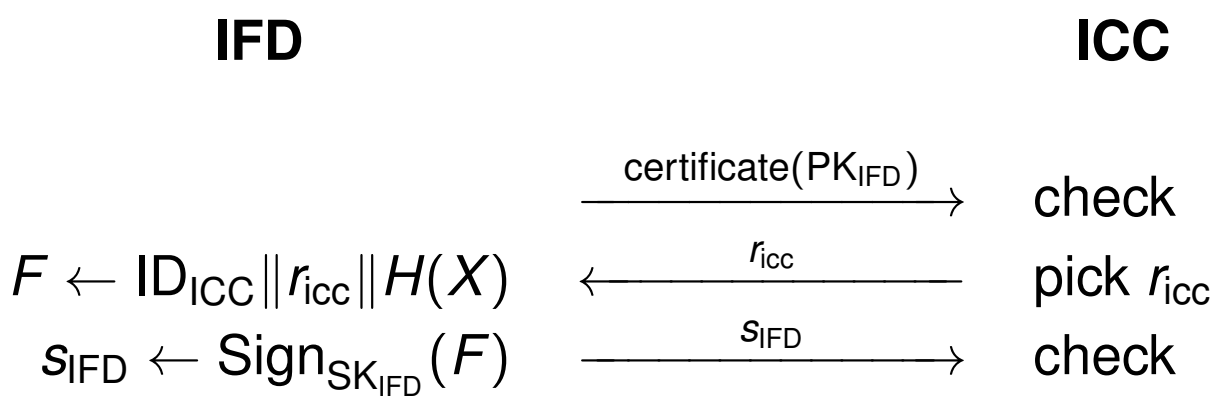
$\rightarrow$ resists passive attacks



Terminal Authentication

- terminal sends a certificate to chip (ECDSA)
- terminal signs a challenge + ephemeral key X from Chip Authentication
- $\mathrm{ID}_{\mathrm{ICC}}$ set to serial number (for BAC) or to ephemeral key of ICC (for PACE)

$\rightarrow$ strong access control



Terminal Authentication Issues

Terminal revocation issue:

- MRTDs are not online!
- MRTDs have no reliable clock

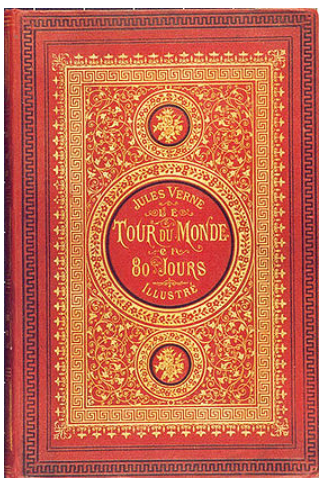
$\longrightarrow$ MRTD must trust readers to revoke themselves

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

back to chapter

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
- 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

- **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.
- **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.

Must Be Known

- GSM security infrastructure
- mobile telephony security
- Bluetooth pairing
- foward secrecy

Train Yourself

- stealing bitcoins using Shor: final exam 2024–25 ex3
- protecting bitcoins against Shor: final exam 2020–21 ex4
- biometric passport: final exam 2015–16 ex3