INF3510 Information Security
University of Oslo
Spring 2014

Lecture 5
Cryptography

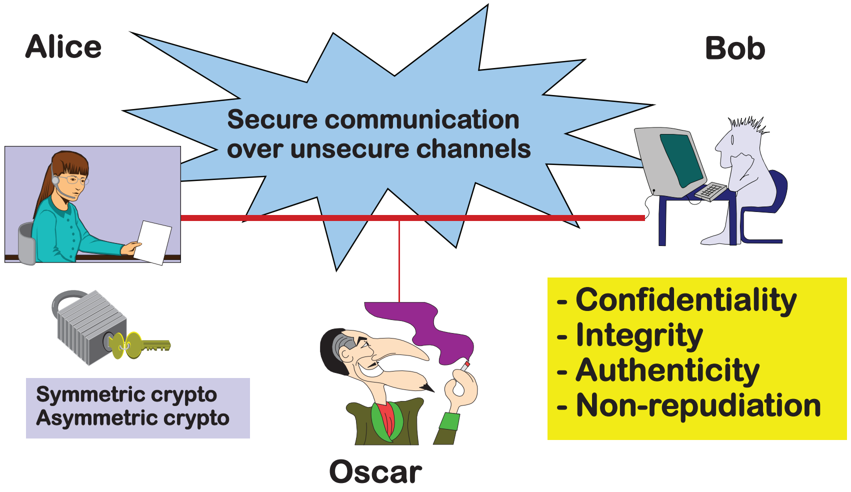
University of Oslo, spring 2014
Leif Nilsen

Outline

- What is cryptography?
- Brief crypto history
- Security issues
- Symmetric cryptography
  - Stream ciphers
  - Block ciphers
  - Hash functions
- Asymmetric cryptography
  - Factoring based mechanisms
  - Discrete Logarithms
  - Digital signatures

Want to learn more?
Look up UNIK 4220

What is cryptology?



Alice

Bob

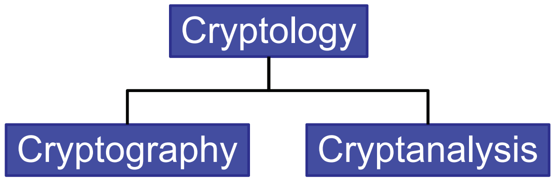
Secure communication over unsecure channels

Symmetric crypto
Asymmetric crypto

Oscar

- **- Confidentiality**
- **- Integrity**
- **- Authenticity**
- **- Non-repudiation**

Terminology



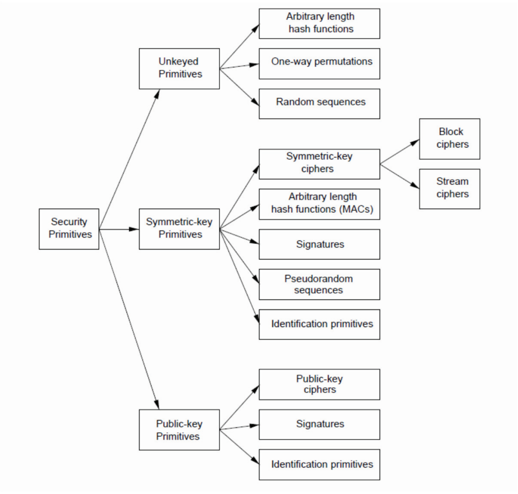
Cryptology

Cryptography

Cryptanalysis

- **Cryptography** is the science of secret writing with the goal of hiding the meaning of a message.
- **Cryptanalysis** is the science and sometimes art of *breaking* cryptosystems.

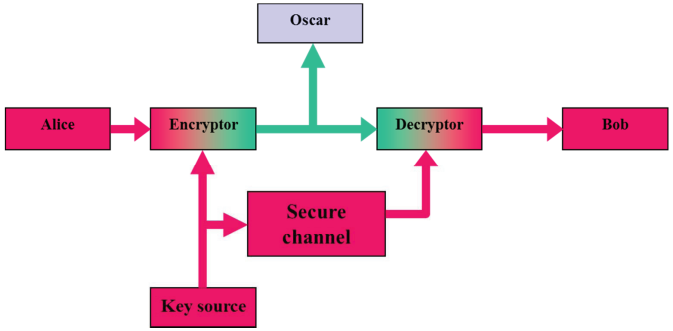
Taxonomy of cryptographic primitives



When is cryptography used?

- Some example situations:
  - **Historically**, the military and spy agencies were the main users of cryptology
    - Situation: transmitting messages over insecure channels
  - **Now**, it is used in many other areas, especially in electronic information processing and communications technologies:
    - Banking: your financial transactions, such as EFTPOS
    - Communications: your mobile phone conversations
    - Info stored in databases: hospitals, universities, etc.
- Cryptography can be used to protect information in storage or during transmission

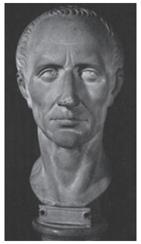
Model of symmetric cryptosystem



Terminology

- Encryption: plaintext (clear text) *M* is converted into a ciphertext *C* under the control of a key *k*.
  - We write $C = E(M, k)$.
- Decryption with key *k* recovers the plaintext *M* from the ciphertext *C*.
  - We write $M = D(C, k)$.
- Symmetric ciphers: the secret key is used for both encryption and decryption.
- Asymmetric ciphers: Pair of private and public keys where it is computationally infeasible to derive the private decryption key from the corresponding public encryption key.

Caesar cipher



Example: Caesar cipher

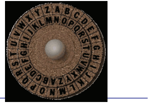
$\mathcal{P}$ = {`abcdefghijklmnopqrstuvwxyz`}

$\mathcal{C}$ = {`DEFGHIJKLMNOPQRSTUVWXYZABC`}

Plaintext: `kryptologi er et spennende fag`

Chiphertext: `NUBSWRORJL HU HT VSHQQHQGH IDJ`

Note: Caesar chipher in this form does not include a variable key, but is an instance of a “shift-cipher” using key $K = 3$.



Numerical encoding of the alphabet

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

p	q	r	s	t	u	v	w	x	y	z	æ	ø	å
14	16	17	18	19	20	21	22	23	24	25	26	27	28

Using this encoding many classical crypto systems can be expressed as algebraic functions over $\mathbb{Z}_{26}$ (English alphabet) or $\mathbb{Z}_{29}$ (Norwegian alphabet)

Shift cipher

Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{29.}$ For $0 \leq K \leq 28$, we define

$$E(x, K) = x + K \pmod{29}$$

and

$$D(y, K) = y - K \pmod{29}$$

$(x, y \in \mathbb{Z}_{29})$

Question: What is the size of the key space?

Puzzle: ct =

LAHYCXPAJYQHRBWNNMNMOXABNLDANLXVVDWRLJCRXWB

Find the plaintext!

Exhaustive search

```
For[i=0, i<26, i++, Print["Key = ", i, " Plain = ", decrypt[ct,1,i]]]
Key = 0 Plain = LAHYCXPAJYQHRBWNNMNMOXABNLDANLXVVDWRLJCRXWB
Key = 1 Plain = KZGXBWOZIXPGQAVMMLMLNWZAMKCZMKWUUCVQKIBQWVA
Key = 2 Plain = JYFWAVNYHWOFPZULLKLKMVYZLJBYLJVTTBUPJHAPVUZ
Key = 3 Plain = IXEVZUMXGVNEOYTKKJKJLUXYKIAXKIUSSATOIGZOUTY
Key = 4 Plain = HWDUYTLWFUMDNXSJJIJIKTWXJHZWJHTRRZSNHFYNTSX
Key = 5 Plain = GVCTXSKVETLCMWRIIHIHJSVWIGYVIGSQQYRMGEXMSRW
Key = 6 Plain = FUBSWRJUDSKBLVQHHGHGIRUVHFXUHFRPPXQLFDWLRQV
Key = 7 Plain = ETARVQITCRJAKUPGGFGFHQTUGEWTGEQOOWPKECVKQPU
Key = 8 Plain = DSZQUPHSBQIZJTOFFEFEGPSTFDVSFDPNNVOJDBUJPOT
Key = 9 Plain = CRYPTOGRAPHYISNEEDEDFORSECURECOMMUNICATIONS
Key = 10 Plain = BQXOSNFQZOGXHRMDDCDCENQRDBTQDBNLLTMHBZSHNMR
Key = 11 Plain = APWNRMEPYNFWGQLCCBCBDMPQCASPCAMKKSLGAYRGMLQ
Key = 12 Plain = ZOVMQLDOXMEVFPKBBABACLOPBZROBZLJJRKFZXQFLKP
```

- .
-

Substitution cipher - example

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
U	D	M	I	P	Y	Æ	K	O	X	S	N	Å	F	A

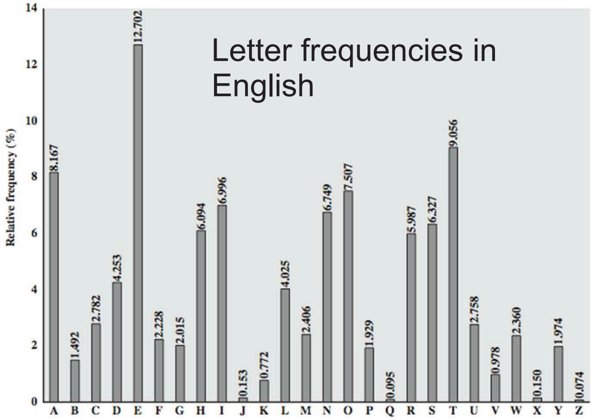
p	q	r	s	t	u	v	w	x	y	z	æ	ø	å	
E	R	T	Z	B	Ø	C	Q	G	W	H	L	V	J	

Plaintext: fermatssisteteorem
Ciphertext: YPTÅUBZZOZBPBPATPÅ

What is the size of the key space?

$88417619937397019545436160000000 \approx 2^{103}$

Letter Frequencies→ statistical attacks

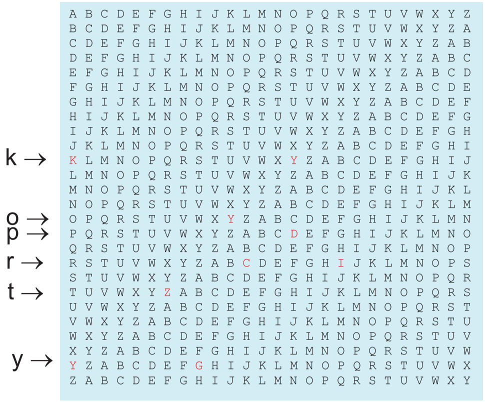


- Encryption must hide statistical patterns in data
- Achieved with a series of primitive functions

Lessons learned

- A cipher with a small keyspace can easily be attacked by *exhaustive search*
- A *large keyspace* is necessary for a secure cipher, but it is by itself not suffcient
- Monoalphabetical substitution ciphers can easily be broken

Vigenére (1523-1596)



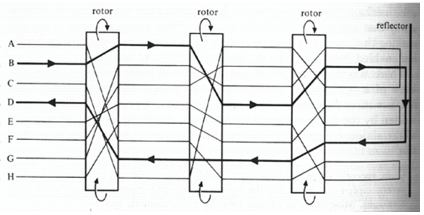
```
     A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
     B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
     C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
     D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
     E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
     F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
     G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
     H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
     I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
     J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
k →  K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
     L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
     M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
     N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
o →  O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
p →  P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
     Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
r →  R S T U V W X Y Z A B C D E F G H I J K L M N O P S
     S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
t →  T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
     U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
     V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
     W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
     X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
y →  Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
     Z 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
```

Key: **kryptokry**
Plaintext: **OLAOGKARI**
Chiphertext: **ycydzykig**

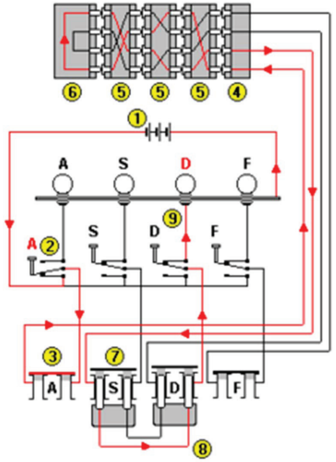
Polyalphabetical, but completely insecure

Enigma

- **German WW II crypto machine**
- **Many different variants**
- **Analysed by Polish and English mathematicians**

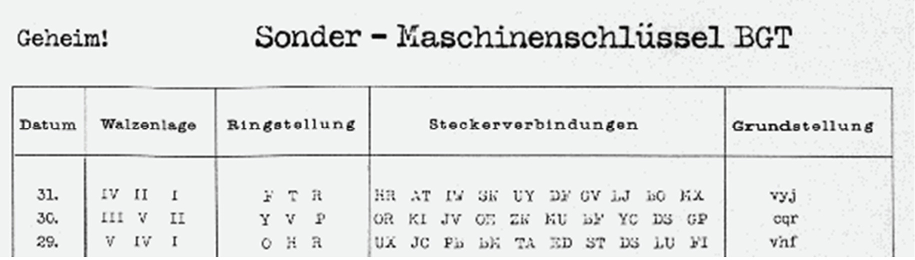


Operating principles



1 - Battery
2 - Keyboard
3 - Stecker board
4 - Entry ring
5 - Rotors (L M R)
6 - Reflector
7 - Conductor
8 - Connector
9 - Lamp

Enigma key list



Geheim! Sonder - Maschinenschlüssel BGT

Datum	Walzenlage	Ringstellung	Steckerverbindungen	Grundstellung
31.	IV II I	F T R	HR AT IW SK UY DF GV LJ BO MX	vyj
30.	III V II	Y V P	OR KI JV QE ZN MU BF YC DS GP	cqr
29.	V IV I	O H R	UX JC PB DK TA ED ST DS LU FI	vhf

Enigma encryption example

Message: “Ich bin sicher, daß unser Führer eine lose Schraube hat”

Enigma Simulator For Windows. ©1995-1999 Geoff Sullivan. Norway build 002
Thu Mar 06 15:46:40 2008
Rotor Order: B V I III Ringstellung: T E K [20 05 11]
Steckers:
Message Key: A A A [01 01 01]

Plaintext: ICHBI NSICH ERDAS SUNSE RFUHR EREIN ELOSE SCHRA UBEHA T
Ciphertext: OVKWR IZXJE OXFNR YPBJZ DBVCG SWLFR TGHPF KEOQL KKRLQ I

Practical complexity for attacking Enigma

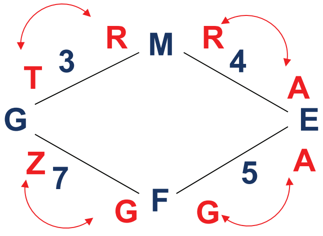
Cryptoanalytical assumptions during WW II:

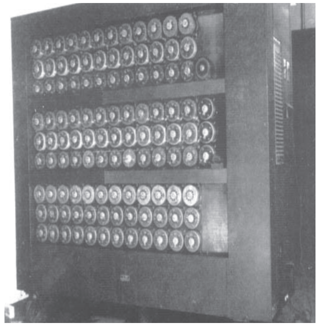
- 3 out of 5 rotors with known wiring
- 10 stecker couplings
- Known reflector

N = 150 738 274 937 250 · 60 · 17 576 · 676 =
107458687327250619360000 (77 bits)

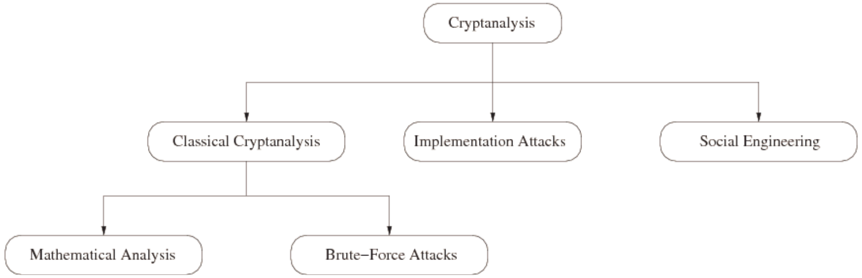
Attacking ENIGMA

Posisjon: 1 2 3 4 5 6 7
Chiffertekst: J T G E F P G
Crib: R O M M E L F





Cryptanalysis: Attacking Cryptosystems



- **Classical Attacks**
  - Mathematical Analysis
  - Brute-Force Attack
- **Implementation Attack**: Try to extract the key through reverse engineering or power measurement, e.g., for a banking smart card.
- **Social Engineering**: E.g., trick a user into giving up her password

---

Brute-Force Attack (or Exhaustive Key Search)

- Treats the cipher as a black box
- Requires (at least) 1 plaintext-ciphertext pair (x_0, y_0)
- Check all possible keys until condition is fulfilled:

$$d_K(y_0) = x_0$$

- How many keys to we need ?

Key length in bit	Key space	Security life time (assuming brute-force as best possible attack)
64	2^{64}	**Short term** (few days or less)
128	2^{128}	**Long-term** (several decades in the absence of quantum computers)
256	2^{256}	**Long-term** (also resistant against quantum computers – note that QC do not exist at the moment and might never exist)

Kerckhoff's principles

- The system should be, if not theoretically unbreakable, unbreakable in practice.
- The design of a system should not require secrecy and compromise of the system should not inconvenience the correspondents (Kerckhoffs' principle).
- The key should be rememberable without notes and should be easily changeable
- The cryptograms should be transmittable by telegraph
- The apparatus or documents should be portable and operable by a single person
- The system should be easy, neither requiring knowledge of a long list of rules nor involving mental strain

---

Attack models:

Known ciphertext
Known plaintext
Chosen plaintext (adaptive)
Chosen ciphertext (adaptive)

What are the goals of the attacker?

- Find the secret plaintext or part of the plaintext
- Find the encryption key
- Distinguish the encryption of two different plaintexts

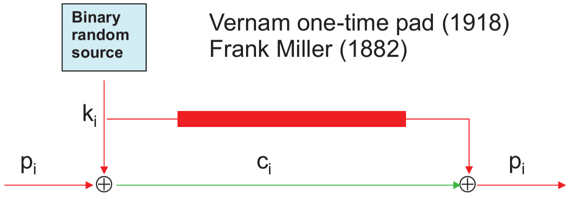
How clever is the attacker?

Does secure ciphers exist?

- What is a secure cipher?
  - Perfect security
  - Computational security
  - Provable security

"I'm sorry, we already have a director of security..."

A perfect secure crypto system



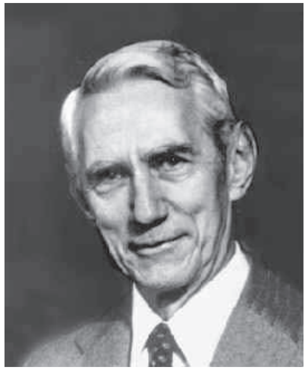
Vernam one-time pad (1918)
Frank Miller (1882)

$c_i = p_i \oplus k_i$
$p_i = c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i$
Note: $a \oplus b = a + b \pmod 2$

Offers perfect security assuming the key is perfectly random, of same length as The Message; and only used once. Proved by Claude E. Shannon in 1949.

Claude Shannon (1916 – 2001)
The Father of Information Theory – MIT / Bell Labs

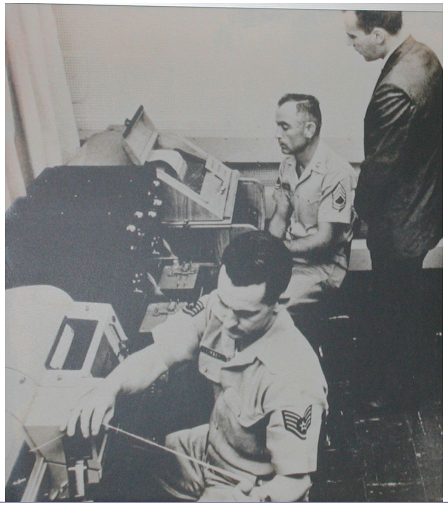
- Information Theory
  - Defined the „binary digit“ (bit) as information unit
  - Definition of „entropy“ as a measure of information amount
- Cryptography
  - Model of a secrecy system
  - Definition of perfect secrecy
  - Designed S-P networks, i.e. a series of substitution & permutation functions



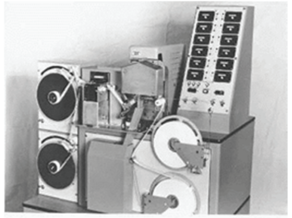
ETCRRM

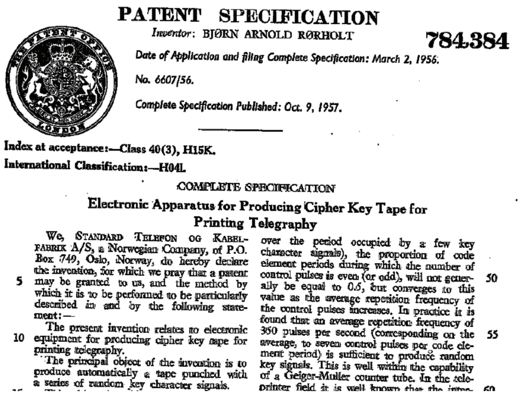
- Electronic Teleprinter Cryptographic Regenerative Repeater Mixer (ETCRRM)
- Invented by the Norwegian Army Signal Corps in 1950
- Bjørn Rørholt, Kåre Mesingseth
- Produced by STK
- Used for ”Hot-line” between Moskva and Washington
- About 2000 devices produced

White House Crypto Room 1960s



Producing key tape for the one-time pad





PATENT SPECIFICATION

Inventor: BJØRN ARNOLD RØRHOLT

784,384

Date of Application and filing Complete Specification: March 2, 1956.

No. 6607/56.

Complete Specification Published: Oct. 9, 1957.

Index at acceptance:—Class 40(3), H15K.

International Classification:—H04L.

COMPLETE SPECIFICATION

Electronic Apparatus for Producing Cipher Key Tape for Printing Telegraphy

We, STANDARD TELEFON OG KABELFABRIK A/S, a Norwegian Company, of P.O. Box 749, Oslo, Norway, do hereby declare the invention, for which we pray that a patent may be granted to us, and the method by which it is to be performed to be particularly described in and by the following statement:—

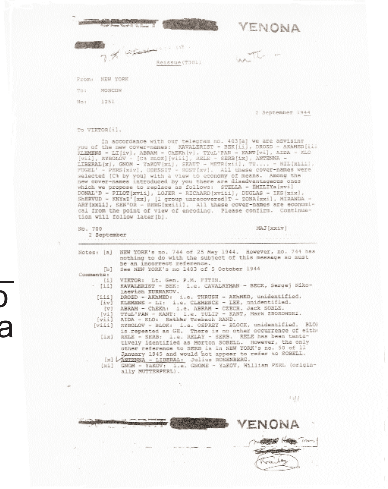
The present invention relates to electronic equipment for producing cipher key tape for printing telegraphy.

The principal object of the invention is to produce automatically a tape punched with a series of random key character signals.

over the period occupied by a few key character signals), the proportion of code element periods during which the number of control pulses is even (or odd), will not generally be equal to 0.5, but converges to this value as the average repetition frequency of the control pulses increases. In practice it is found that an average repetition frequency of 350 pulses per second (corresponding on the average, to seven control pulses per code element period) is sufficient to produce random key signals. This is well within the capability of a Geiger-Muller counter tube. In the teleprinter field it is well known that the inter-

Venona

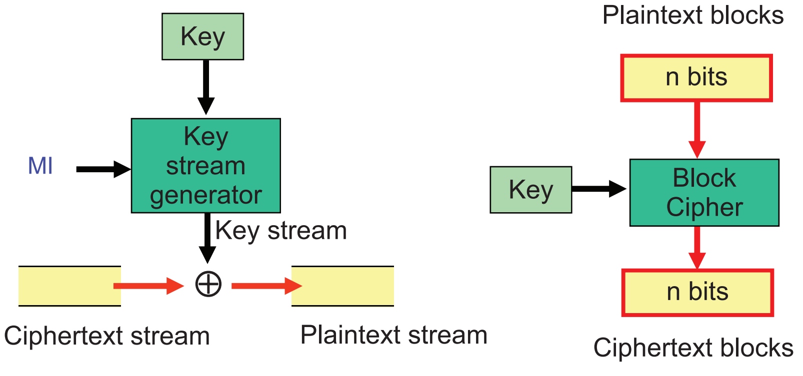
- US attack on encrypted SovjetUnion traffic due to re-use of one-time pads
- 1943-1980
- Ca. 3000 messages decrypted
- http://www.nsa.gov/about/_files/cryptologic_heritage/publications/coldwar/venona_story.pdf



Symmetric encryption

Is it possible to design secure and practical crypto?

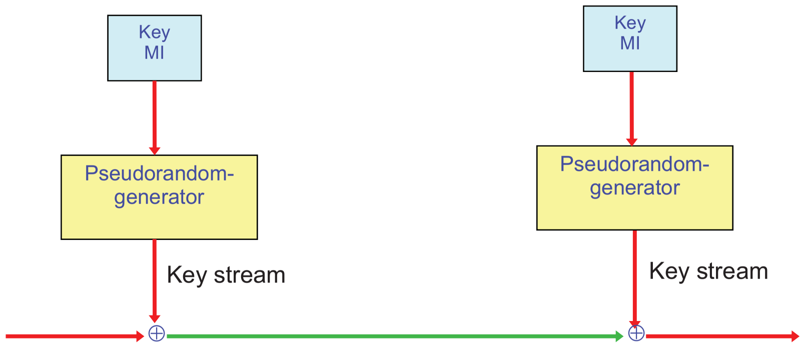
Stream Cipher vs. Block Cipher



Stream cipher

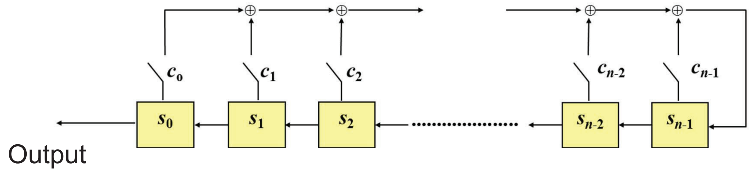
Block cipher

Symmetric stream cipher



LFSR

Linear feedback shift register



Using *n* flip-flops we may generate a binary sequence of period 2^n -1

$$s_{n+i} = c_0 s_i + c_1 s_{i+1} + \cdots + c_{n-1} s_{i+n-1}$$

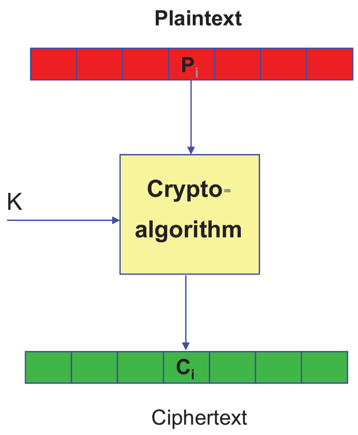
Note: The stream cipher is stateful

LFSR - properties

- Easy to implement in HW, offers fast clocking
- The output sequence is completely determined of the initial state and the feedback coefficients
- Using “correct” feedback a register of length *n* may generate a sequence with period 2^n-1
- The sequence will provide good statistical properties
- Knowing 2*n* consecutive bits of the key stream, will reveal the initial state and feedback
- The linearity means that a single LFSR is completely useless as a stream cipher, but LFSRs may be a useful building block for the design of a strong stream cipher

---

Symmetric block cipher



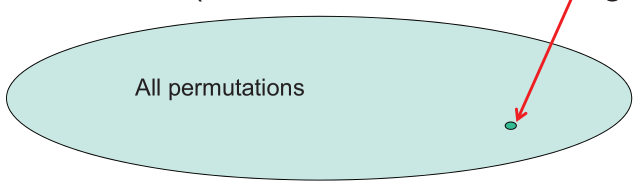
- The algorithm represents a family of permutations of the message space
- Normally designed by iterating a less secure round function
- May be applied in different operational modes
- Must be impossible to derive K based on knowledge of P and C

Block cipher and random permutations

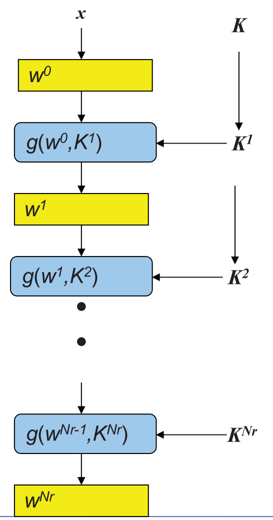
- Given block size *m* = 64 and key length *l* = 56 bit
- Number of different DES-permutations is

  2^{56} = 72057594037927936
- Number of possible permutations of 2^{64} elements is

  2^{64}! = ?? (more than 2^{71} decimal digits)



Itrerated block cipher design



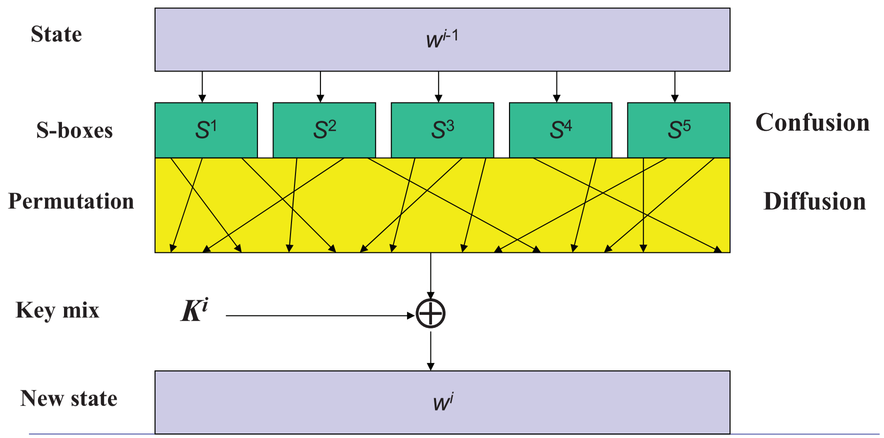
Algorithm:

$w^0 \leftarrow x$
$w^1 \leftarrow g(w^0,K^1)$
$w^2 \leftarrow g(w^1,K^2)$
•
•
$w^{Nr-1} \leftarrow g(w^{Nr-2},K^{Nr-1})$
$w^{Nr} \leftarrow g(w^{Nr-1},K^{Nr})$
$y \leftarrow w^{Nr}$

NB! For a fixed *K*, *g* must be injective in order to decrypt *y*

Substitusjon-Permutasjon nettverk (SPN):

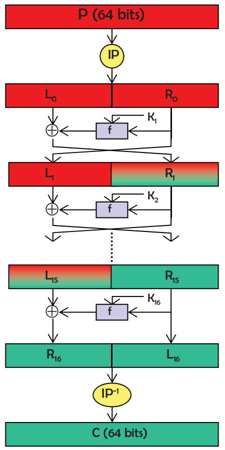
Round function *g* :



Data Encryption Standard

- Published in 1977 by the US National Bureau of Standards for use in unclassified government applications with a 15 year life time.
- 16 round Feistel cipher with 64-bit data blocks, 56-bit keys.
- 56-bit keys were controversial in 1977; today, exhaustive search on 56-bit keys is very feasible.
- Controversial because of classified design criteria, however no loop hole was ever found.

DES architecture

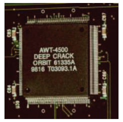


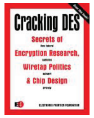
DES(P):
$(L_0, R_0) = IP(P)$
FOR i = 1 TO 16
$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(R_{i-1},K_i)$
$C = IP^{-1}(R_{16},L_{16})$

64 bit data block
56 bit key
72.057.594.037.927.936

EFF DES-cracker

- Dedicated ASIC with 24 DES search engines
- 27 PCBs housing 1800 circuits
- Can test 92 billion keys per second
- Cost 250 000 $
- DES key found July 1998 after 56 hours search
- Combined effort DES Cracker and 100.000 PCs could test 245 billion keys per second and found key after 22 hours

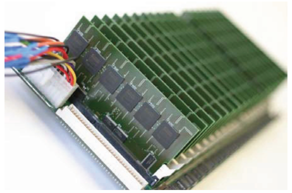


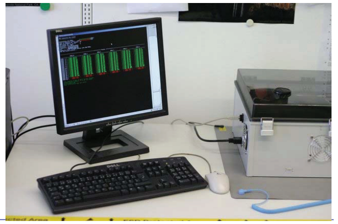


Copacobana

COPACOBANA, the Cost-Optimized Parallel COde Breaker, is an FPGA-based machine which is optimized for running cryptanalytical algorithms. COPACOBANA is suitable for parallel computation problems which have low communication requirements. DES cracking is such a parallelizable problem: an exhaustive key search of the Data Encryption Standard (DES) takes no longer than a week on average with COPACOBANA. Other ciphers can be attacked too, and COPACOBANA can also be used for parallel computing problem outside cryptography.

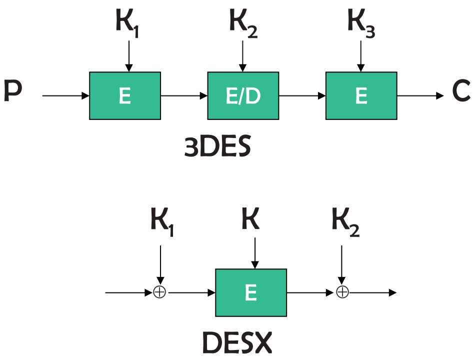
(And yes, we know, Rio de Janeiro's famous beach is spelled slightly differently, Copacabana ;)





DES Status

- DES er the “work horse” which over 30 years have inspired cryptographic research and development
- “Outdated by now”!
- Single DES can not be considered as a secure block cipher
- Use 3DES (ANSI 9.52) or DESX



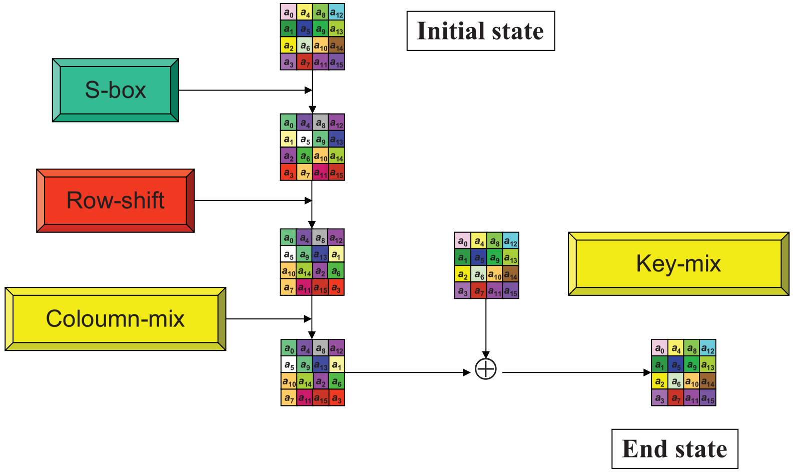
Advanced Encryption Standard

- Public competition to replace DES: because 56-bit keys and 64-bit data blocks no longer adequate.
- Rijndael nominated as the new Advanced Encryption Standard (AES) in 2001 [FIPS-197].
- Rijndael (pronounce as “Rhine-doll”) designed by Vincent Rijmen and Joan Daemen.
- 128-bit block size (Note error in Harris p. 809)
- 128-bit, 196-bit, and 256-bit key sizes.
- Rijndael is <u>not</u> a Feistel cipher.

Rijndael, the selected AES cipher

Designed by Vincent Rijmen and Joan Daemen from Belgium

Rijndael round function



Rijndael encryption

1. Key mix (round key K_0)
2. N_r-1 rounds containing:
    a) Byte substitution
    b) Row shift
    c) Coloumn mix
    d) Key mix (round key K_i)
3. Last round containing:
    a) Byte substitution
    b) Row shift
    c) Key mix (round key K_{Nr})

Key	Rounds
128	10
192	12
256	14

Using encryption for real

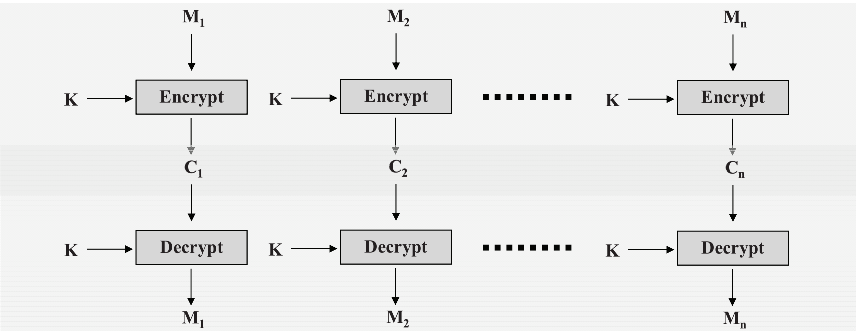
- With a block cipher, encrypting a *n*-bit block *M* with a key *k* gives a ciphertext block *C* = E(*M*,*k*).
- Given a well designed block cipher, observing *C* would tell an adversary nothing about *M* or *k*.
- What happens if the adversary observes traffic over a longer period of time?
    - The adversary can detect if the same message had been sent before; if there are only two likely messages “buy” and “sell” it may be possible to guess the plaintext without breaking the cipher.

Block Ciphers: Modes of Operation

- Block ciphers can be used in different modes in order to provide different security services.
- Common modes include:
  - **E**lectronic **C**ode **B**ook (ECB)
  - **C**ipher **B**lock **C**haining (CBC)
  - **O**utput **F**eed**b**ack (OFB)
  - **C**ipher **F**eed**b**ack (CFB)
  - **C**oun**t**e**r** Mode (CTR)
  - **G**alois **C**ounter **M**ode (GCM) {Authenticated encryption}

Electronic Code Book

- ECB Mode encryption
  - Simplest mode of operation
  - Plaintext data is divided into blocks $M_1, M_2, \ldots, M_n$
  - Each block is then processed separately
    - Plaintext block and key used as inputs to the encryption algorithm

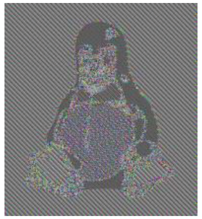


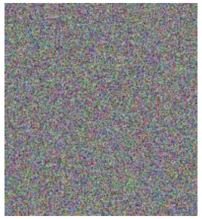
ECB Mode

- ECB Mode Issues
  - Problem: For a given key, the same plaintext block always encrypts to the same ciphertext block.
    - This may allow an attacker to construct a code book of known plaintext/ciphertext blocks.
    - The attacker could use this codebook to insert, delete, reorder or replay data blocks within the data stream without detection
  - Other modes of operation can prevent this, by not encrypting blocks independently
    - For example, using the output of one block encryption as input to the next (chaining)

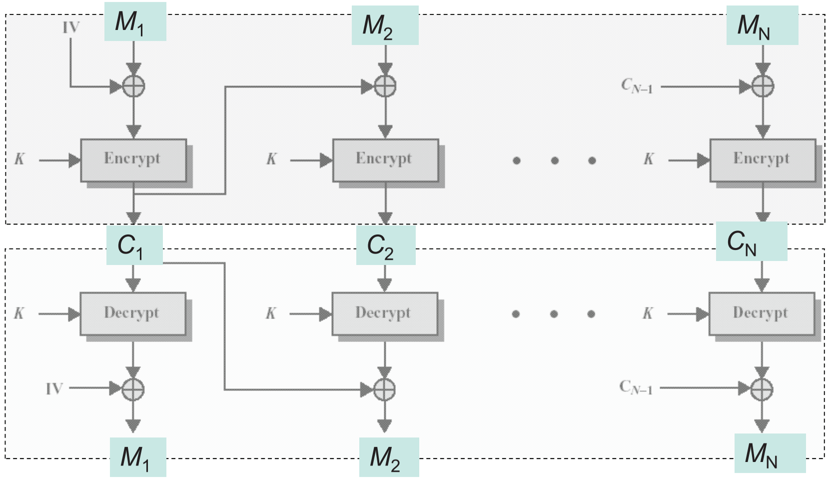
Use a secure mode!


Plaintext

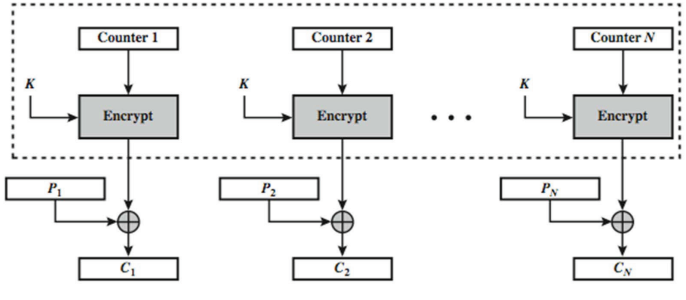

Ciphertext using ECB mode


Ciphertext using secure mode

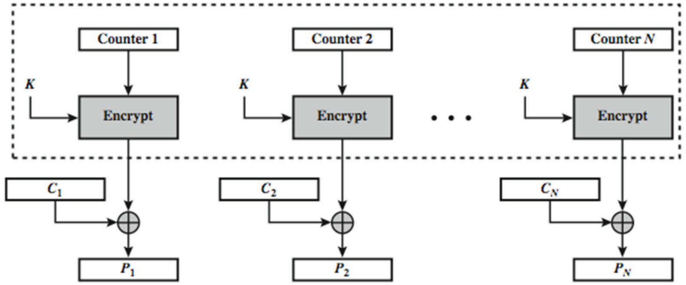
Cipher Block Chaining Mode



CTR Counter Mode



(a) Encryption



(b) Decryption

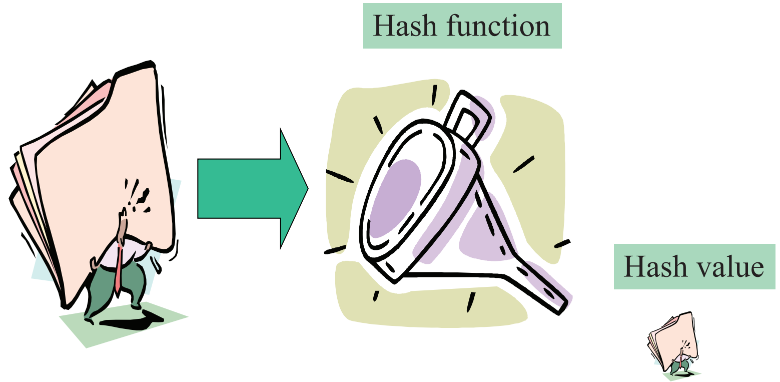
Block cipher: Applications

- Block ciphers are often used for providing confidentiality services
- They are used for applications involving processing large volumes of data, where time delays are not critical.
  - – Examples:
    - Computer files
    - Databases
    - Email messages
- Block ciphers can also be used to provide integrity services, i.e. for message authentication

---

Integrity Check Functions

Hash functions



Applications of hash functions

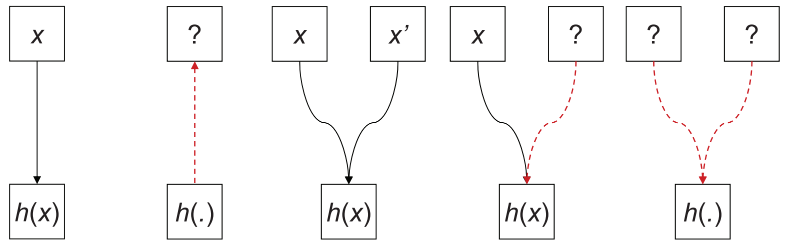
- Protection of password
- Comparing files
- Authentication of SW distributions
- Bitcoin
- Generation of Message Authentication Codes (MAC)
- Digital signatures
- Pseudo number generation/Mask generation functions
- Key derivation

Hash functions (message digest functions)

Requirements for a one-way hash function *h*:

1. Ease of computation: given x, it is easy to compute $h(x)$.
2. Compression: h maps inputs x of arbitrary bitlength to outputs $h(x)$ of a fixed bitlength n.
3. One-way: given a value y, it is computationally infeasible to find an input x so that $h(x)=y$.
4. Collision resistance: it is computationally infeasible to find x and x', where $x \neq x'$, with $h(x)=h(x')$ (note: two variants of this property).

Properties of hash functions



Ease of computation

Pre-image resistance

Collision

Weak collision resistance (2nd pre-image resistance)

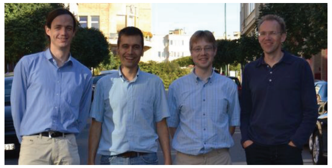
Strong collision resistance

Frequently used hash functions

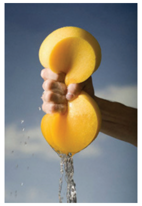
- MD5: 128 bit digest. Broken. Often used in Internet protocols but no longer recommended.
- SHA-1 (Secure Hash Algorithm):160 bit digest. Potential attacks exist. Designed to operate with the US Digital Signature Standard (DSA);
- SHA-256, 384, 512 bit digest. Still secure. Replacement for SHA-1
- RIPEMD-160: 160 bit digest. Still secure. Hash function frequently used by European cryptographic service providers.
- NIST competition for new secure hash algorithm, announcement of winner expected in 2012.

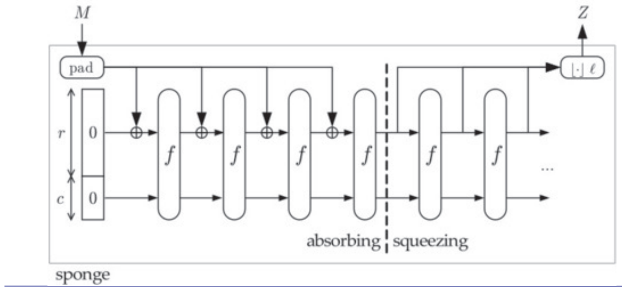
And the winner is?

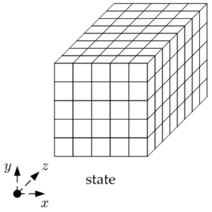
- NIST announced Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition on October 2, 2012, and ended the five-year competition.
- Keccak was designed by a team of cryptographers from Belgium and Italy, they are:
  - Guido Bertoni (Italy) of STMicroelectronics,
  - Joan Daemen (Belgium) of STMicroelectronics,
  - Michaël Peeters (Belgium) of NXP Semiconductors, and
  - Gilles Van Assche (Belgium) of STMicroelectronics.



Keccak and sponge functions



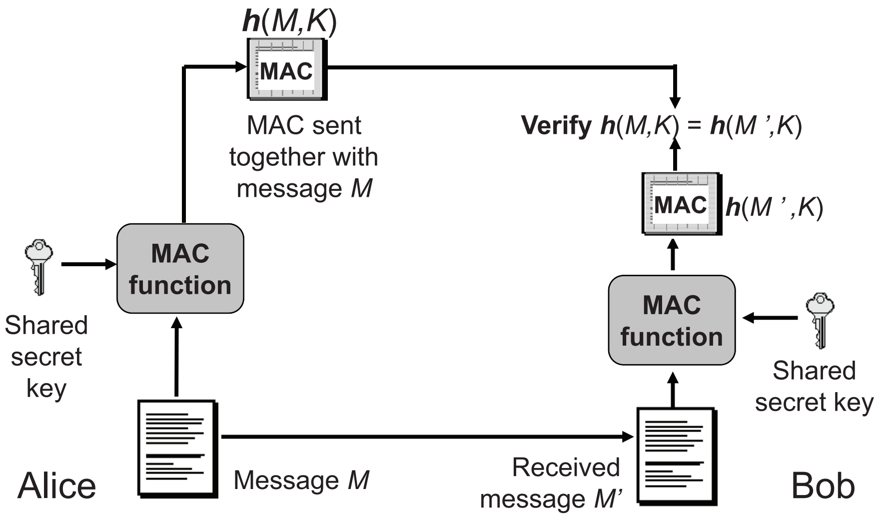




MAC and MAC algorithms

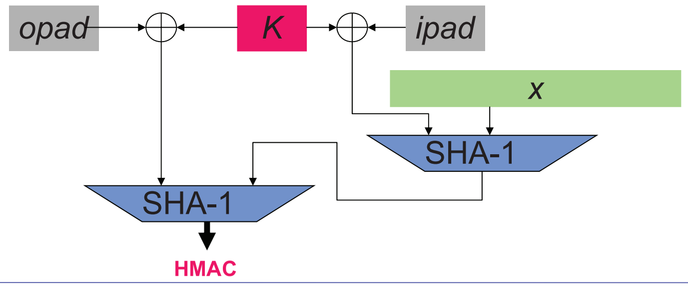
- MAC means two things:
  1. The computed message authentication code *h*(*M, k*)
  2. General name for algorithms used to compute a MAC
- In practice, the MAC algorithm is e.g.
  - HMAC (Hash-based MAC algorithm))
  - CBC-MAC (CBC based MAC algorithm)
  - CMAC (Cipher-based MAC algorithm)
- MAC algorithms, a.k.a. keyed hash functions, support data origin authentication services.

Practical message integrity with MAC



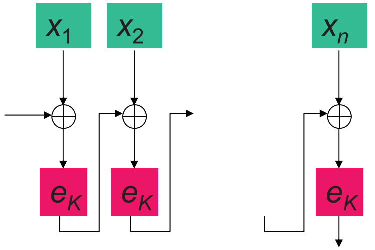
HMAC

- Define: *ipad* = 3636….36 (512 bit)
- *opad* = 5C5C…5C (512 bit)
- $\text{HMAC}_K(x) = \text{SHA-1}((K \oplus opad) \,||\, \text{SHA-1}((K \oplus ipad) \,||\, x))$



CBC-MAC

```
• CBC-MAC(x, K)
• sett $x = x_1 || x_2 || …. || x_n$
• IV ← 00 … 0
• $y_0$ ← IV
• for $i$ ← 1 to $n$
•   do $y_i \leftarrow e_K(y_{i-1} \oplus x_i)$
• return ($y_n$)
```

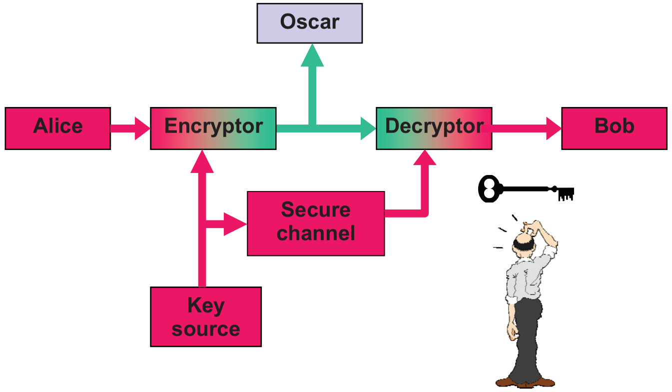


Hash functions and Message Authentication

- Shared secret key is used with a MAC
- When used during message transmission, this provides Message Authentication:
  - A correct MAC value confirms the sender of the message is in possession of the shared secret key
  - Hence, much like a password, it confirms the authenticity of the message sender to the receiver.
- Indeed, message integrity is meaningless without knowing who sent the message.

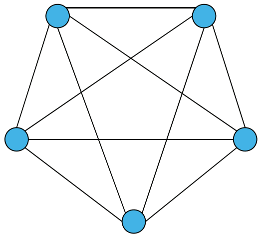
Public-Key Cryptography

Symmetric cryptosystem

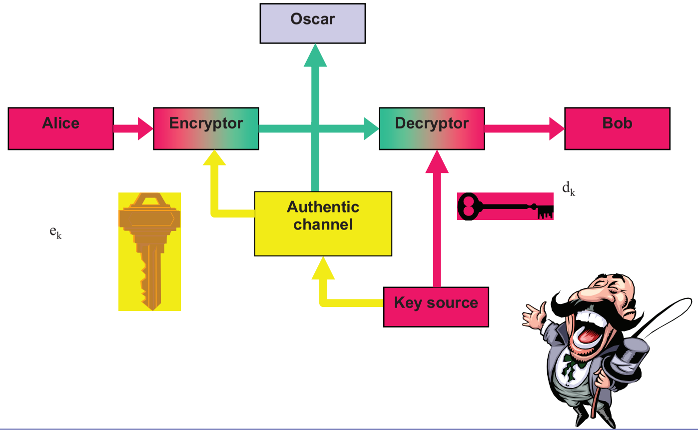

Oscar
Alice
Encryptor
Decryptor
Bob
Secure channel
Key source

Symmetric key distribution

- Shared key between each pair
- In network of *n* users, each participant needs *n*-1 keys.
- Total number of exchanged keys: = (*n*-1) + (*n*-2) + … + 2 + 1 = *n*(*n*-1)/2
- Grows quadraticly, which is problematic.
- Is there a better way?

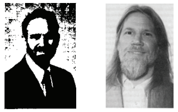

Network of 5 nodes

Asymmetrisk kryptosystem


Oscar
Alice
Encryptor
Decryptor
Bob
Authentic channel
Key source
e_k
d_k

Public key inventors?

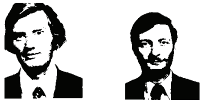
Marty Hellman and Whit Diffie, Stanford 1976



R. Rivest, A. Shamir and L. Adleman, MIT 1978

James Ellis, CESG 1970

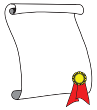
C. Cocks, M. Williamson, CESG 1973-1974



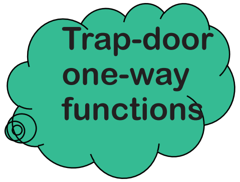
Asymmetric crypto

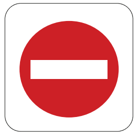
Public key cryptography was born in May 1975, the child of two problems and a misunderstanding!

Key Distribution!



Digital signing!





One-way functions

Modular power function

Given $n = pq$, where p and q are prime numbers. No efficient algoritms to find p and q.

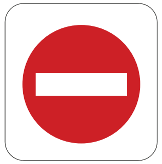
Chose a positive integer b and define $f : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$

$f(x) = x^b \bmod n$

Modular exponentiation

Given prime p, generator g and a modular power $a = g^x \pmod{p}$. No efficient algoritms to find x. $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$

$f(x) = g^x \bmod p$



Public Key Encryption

- Proposed in the open literature by Diffie & Hellman in 1976.
- Each party has a public encryption key and a private decryption key.
- Reduces total number of exchanged keys to n
- Computing the private key from the public key should be computationally infeasible.
- The public key need not be kept secret but it is not necessarily known to everyone.
- There can be applications where even access to public keys is restricted.

Ralph Merkle, Martin Hellman and Whitfield Diffie

Merkle, Hellman and Diffie

- Merkle invented (1974) and published (1978) Merkle's puzzle, a key exchange protocol which was unpractical
- Diffie & Hellman invented (influenced by Merkle) a practical key exchange algorithm using discrete exponentiation.
- D&H defined public-key encryption (equiv. to non-secret encryption)
- Defined digital signature
- Published 1976 in "*New directions in cryptography*"

Diffie-Hellman key agreement (key exchange)

(provides no authentication)

Alice picks random integer a

g^a mod p →

← g^b mod p

Computationally impossible to compute discrete logarithm

Bob picks random integer b

Alice computes the shared secret $(g^b)^a = g^{ab}$ mod p

Bob computes the same secret $(g^a)^b = g^{ab}$ mod p.

Example

- $\mathbb{Z}_{11}$ using $g = 2$:
  - $2^1 = 2 \pmod{11}$ $2^6 = 9 \pmod{11}$
  - $2^2 = 4 \pmod{11}$ $2^7 = 7 \pmod{11}$
  - $2^3 = 8 \pmod{11}$ $2^8 = 3 \pmod{11}$
  - $2^4 = 5 \pmod{11}$ $2^9 = 6 \pmod{11}$
  - $2^5 = 10 \pmod{11}$ $2^{10} = 1 \pmod{11}$

- $\log_2 5 = 4$
- $\log_2 7 = 7$
- $\log_2 1 = 10$ ($\equiv 0$ mod 10)

Example (2)

p =
3019662633453665226674644411185277127204721722044543980521881984280643980698016315342127777985323
7655786915947633907457862442472144616346714598423225826077976000905549946633556169688641786953396
0040623713995997295449774004045416733136225768251717475634638402409117911722715606961870076297223
4159137526583857970362142317237148068590959528891803802119028293828368386437223302582405986762635
8694772029533769528178666567879514981999272674689885986300092124730492599541021908208672727813714
8522572014844749083522090193190746907275606521624184144352256368927493398678089550310568789287558
75522700141844883356351776833964003

g =
1721484410294542720413651217788953849637988183467987659847411571496616170507302662812929883501017
4348250308006877834103702727269721499966768323290540216992770986728538508742382941595672248624817
9949179397494476750553747868409726540440305778460006450549504248776668609868201521098873552043631
7965394509849072406890541468179263651065250794610243485216627272170663501147422628994581789339082
7991578201408649196984764863302981052471409215846871176739109049866118609117954454512573209668379
5760420560620966283259002319100903253019113331521813948039086102149370446134117406508009893347295
86051242347771056691010439032429058

Finn *a* når

g^a (mod p) =
4411321635506521515968448863968324914909246042765028824594289876687657182492169027666262097915382
0952830455103982849705054980427000258241321067445164291945709875449674237106754516103276658256727
2413603372376920980338976048557155564281928533840136742732489850550648761094630053148353906425838
5317698361559907392252360968934338558269603389519179121915049733353702083721856421988041492207985
6566434665604898681669845852964624047443239120501341277499692338517113201830210812184500672101247
2700988032756016626566167579963223042395414267579262222147625965023052419869061244027798941410432
68551743878130988606078310881106 17

Solution

a =

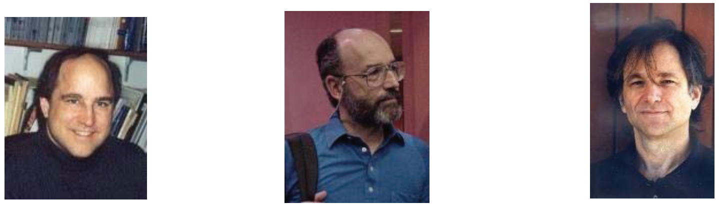
71893136149709653804503478677866573695060790720621260648699193249561437588126371185
81694154929099396752251787268346548051895320171079663652680741564200286881487888963
19895353311170236034836658449187117723820644855184055305945501710227615558093657781
93109639893698220411548578601884177129022057550866690223052160523604836233675971504
25938247630127368253363295292024736143937779912318142315499711747531882501424082252
28164641111954587558230112140813226698098654739025636607106425212812421038155501562
37005192231836155067262308141154795194735834753570104459663325337960304941906119476
18181858300094662765895526963615406

It is easy to compute g^a (mod p) {0.016 s}, but it is computaionally infeasable to compute the exponent a from the g^a.

Diffie-Hellman Applications

- IPSec (IP Security)
  - IKE (Internet Key Exchange) is part of the IPSec protocol suite
  - IKE is based on Diffie-Hellman Key Agreement
- SSL/TLS
  - Several variations of SSL/TLS protocol including
    - Fixed Diffie-Hellman
    - Ephemeral Diffie-Hellman
    - Anonymous Diffie-Hellman

Ron Rivest, Adi Shamir and Len Adleman



- Read about public-key cryptography in 1976 article by Diffie & Hellman: *"New directions in cryptography"*
- Intrigued, they worked on finding a practical algorithm
- Spent several months in 1976 to re-invent the method for non-secret/public-key encryption discovered by Clifford Cocks 3 years earlier
- Named RSA algorithm

RSA parametre (textbook version)

- Bob generates two large prime numbers p and q and computes $n = p \cdot q$.
- He then computes a public encryption exponent e, such that
- (e, (p-1)(q-1))) = 1 and computes the corresponding decryption exsponent d, by solving:

$$\boxed{d \cdot e \equiv 1 \pmod{(p-1)(q-1)}}$$

- Bob's public key is the pair $P_B = (e, n)$ and the corresponding private and secret key is $S_B = (d, n)$.

$$\boxed{\begin{array}{l} \text{Encryption: } C = M^e \pmod{n} \\ \text{Decryption: } M = C^d \pmod{n} \end{array}}$$

RSA toy example

- Set p = 157, q = 223. Then $n = p \cdot q$ =157 · 223 = 35011 and (p-1)(q-1) = 156 ·222 = 34632
- Set encryption exponent: e = 14213 {gcd(34632,14213) = 1}
- Public key: (14213, 35011)
- Compute: $d = e^{-1} = 14213^{-1}$ (mod 34632) = 31613
- Private key: (31613, 35011)

- Encryption:
- Plaintext M = 19726, then C = 19726^{14213} (mod 35011) = 32986

- Decryption:
- Cipherertext C = 32986, then M = 32986^{31613}(mod 35011) = 19726

Factoring record– December 2009

- Find the product of
- p = 33478071698956898786044169848212690817704794983713768568
- 912431388982883793878002287614711652531743087737814467999489
- and
- q= 36746043666799590428244633799627952632279158164343087642
- 676032283815739666511279233373417143396810270092798736308917?

Answer:

n= 123018668453011775513049495838496272077285356959533479219732
245215172640050726365751874520219978646938995647494277406384592
519255732630345373154826850791702612214291346167042921431160222
1240479274737794080665351419597459856902143413

Computation time ca. 0.0000003 s on a fast laptop!
RSA768 - Largest RSA-modulus that have been factored (12/12-2009)
Up to 2007 there was 50 000$ prize money for this factorisation!

Computational effort?

- Factoring using NFS-algorithm (Number Field Sieve)
- 6 mnd using 80 cores to find suitable polynomial
- Solding from August 2007 to April 2009 (1500 AMD64-år)
- 192 796 550 * 192 795 550 matrise (105 GB)
- 119 days on 8 different clusters
- Corresponds to 2000 years processing on one single core 2.2GHz AMD Opteron (ca. 2^{67} instructions)

Asymmetric Ciphers: Examples of Cryptosystems

- RSA: best known asymmetric algorithm.
  - RSA = Rivest, Shamir, and Adleman (published 1977)
  - Historical Note: U.K. cryptographer Clifford Cocks invented the same algorithm in 1973, but didn't publish.
- ElGamal Cryptosystem
  - Based on the difficulty of solving the discrete log problem.
- Elliptic Curve Cryptography
  - Based on the difficulty of solving the EC discrete log problem.
  - Provides same level of security with smaller key sizes.

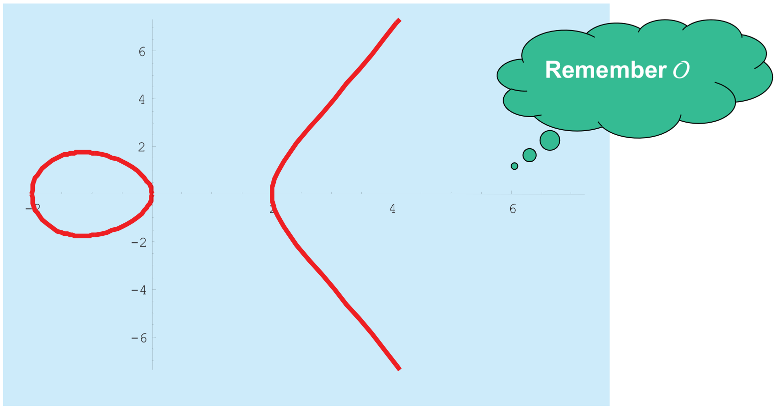
Elliptic curves

- Let $p > 3$ be a prime. An elliptic curve $y^2 = x^3 + ax + b$ over GF(p) = $\mathbb{Z}_p$ consist of all solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ to the equation

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

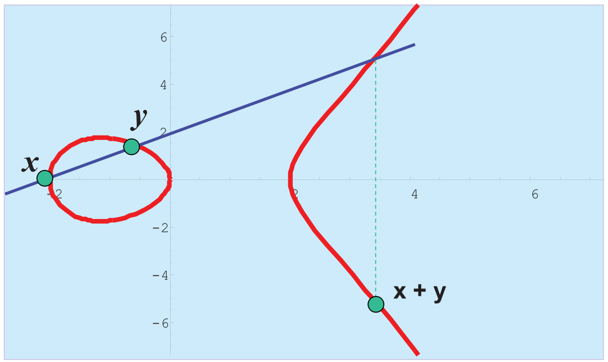
- where $a, b \in \mathbb{Z}_p$ are constants such that $4a^3 + 27b^2 \neq 0 \pmod{p}$, together with a special point O which is denoted as *the point at infinity*.

Elliptic curve over ℝ

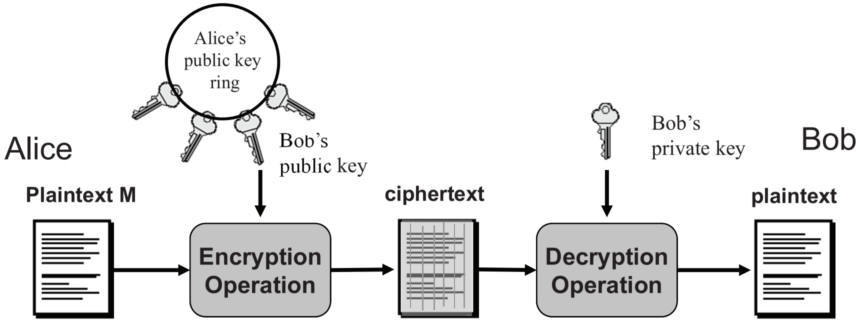


$$y^2 = x^3 - 4x$$

Point addition



Asymmetric Encryption: Basic encryption operation



$C = E(M,K_{pub})$

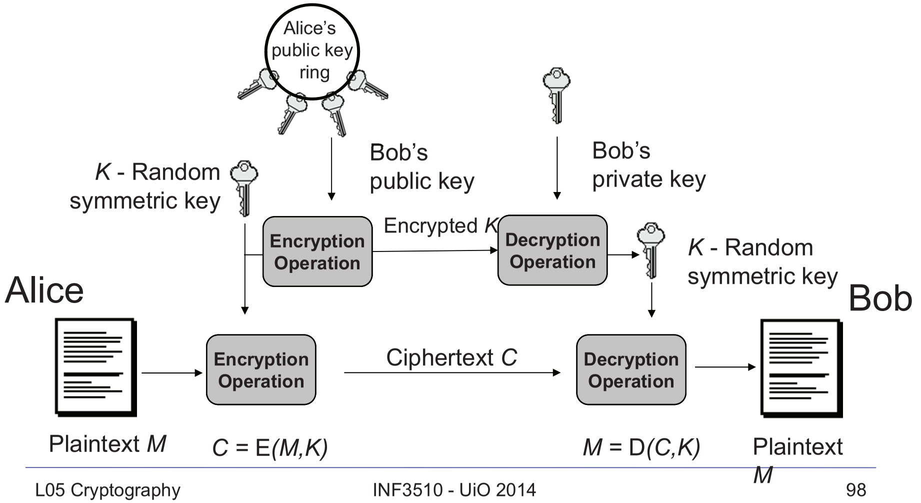
$M = D(C,K_{priv})$

- In practice, large messages are not encrypted directly with asymmetric algorithms. Hybrid systems are used, where only symmetric session key is encrypted with asymmetric alg.

Hybrid Cryptosystems

- Symmetric ciphers are faster than asymmetric ciphers (because they are less computationally expensive), but ...
- Asymmetric ciphers simplify key distribution, therefore ...
- a combination of both symmetric and asymmetric ciphers can be used – a hybrid system:
  - The asymmetric cipher is used to distribute a randomly chosen symmetric key.
  - The symmetric cipher is used for encrypting bulk data.

Confidentiality Services: Hybrid Cryptosystems

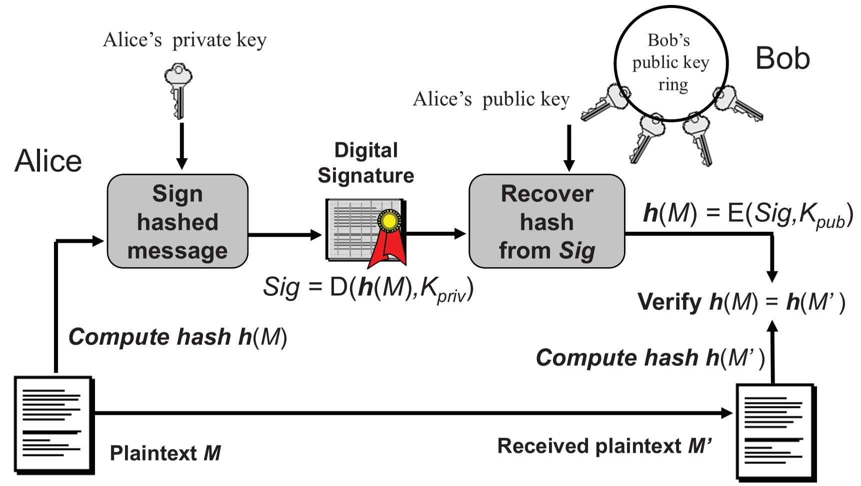


Digital Signatures

Digital Signature Mechanisms

- A MAC cannot be used as evidence that should be verified by a third party.
- Digital signatures used for non-repudiation, data origin authentication and data integrity services, and in some authentication exchange mechanisms.
- Digital signature mechanisms have three components:
  - key generation
  - signing procedure (private)
  - verification procedure (public)
- Algorithms
  - RSA
  - DSA and ECDSA

Practical digital signature based on hash value



Digital Signatures

- To get an authentication service that links a document to *A*'s name (identity) and not just a verification key, we require a procedure for *B* to get an authentic copy of *A*'s public key.
- Only then do we have a service that proves the authenticity of documents 'signed by *A*'.
- This can be provided by a PKI (Public Key Infrastructure)
- Yet even such a service does not provide non-repudiation at the level of persons.

Difference between MACs & Dig. Sig.

- MACs and digital signatures are both authentication mechanisms.
- MAC: the verifier needs the secret that was used to compute the MAC; thus a MAC is unsuitable as evidence with a third party.
  - The third party does not have the secret.
  - The third party cannot distinguish between the parties knowing the secret.
- Digital signatures can be validated by third parties, and can in theory thereby support both non-repudiation and authentication.

Key length comparison:

Symmetric and Asymmetric ciphers offering comparable security

AES Key Size	RSA Key Size	Elliptic curve Key Size
-	1024	163
128	3072	256
192	7680	384
256	15360	512

Another look at key lengths

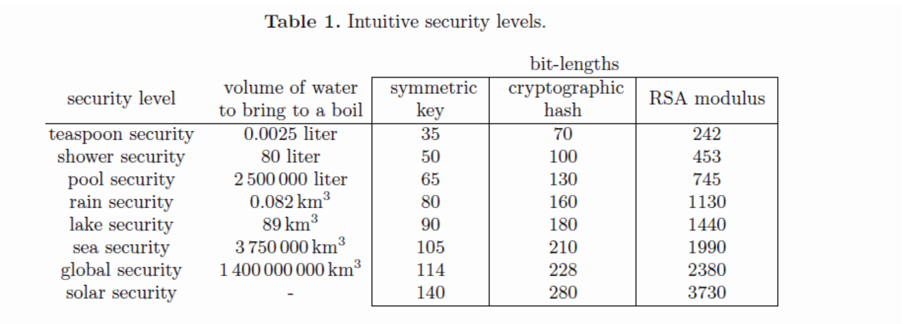
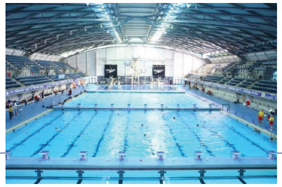


Table 1. Intuitive security levels.

security level	volume of water to bring to a boil	bit-lengths symmetric key	 cryptographic hash	 RSA modulus
teaspoon security	0.0025 liter	35	70	242
shower security	80 liter	50	100	453
pool security	2 500 000 liter	65	130	745
rain security	$0.082\,\text{km}^3$	80	160	1130
lake security	$89\,\text{km}^3$	90	180	1440
sea security	$3\,750\,000\,\text{km}^3$	105	210	1990
global security	$1\,400\,000\,000\,\text{km}^3$	114	228	2380
solar security	-	140	280	3730



End of lecture