T-79.4501 Cryptography and Data Security

Summary 2006
- Life cycle of a cryptographic algorithm
- Searches and probabilities
- Highlights

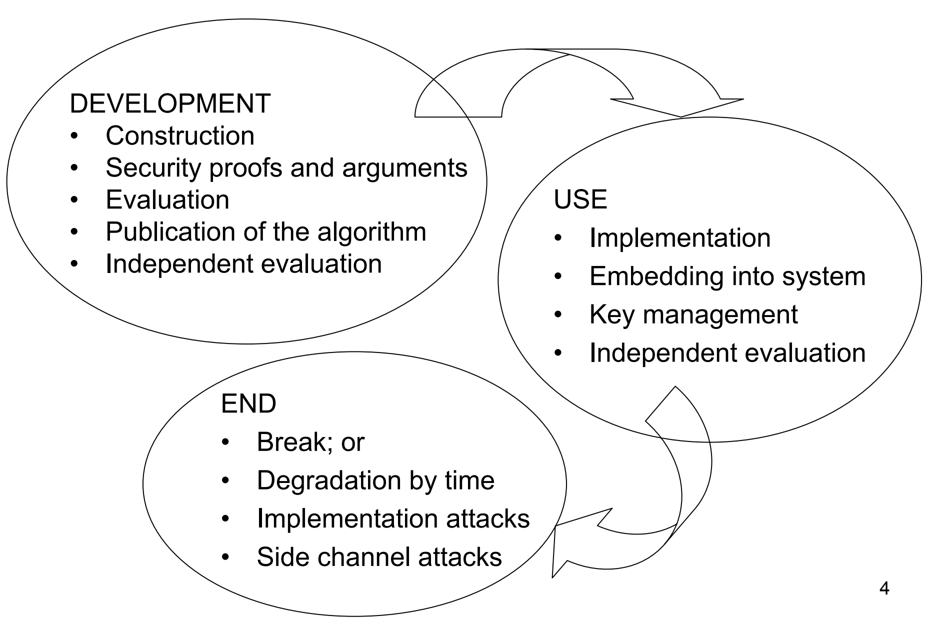
Building Blocks

- one-way functions
- one-way permutations
- one-way trapdoor permutations
- collision resistant compression (hash) functions
- pseudo-random functions
- (pseudo-) random number generators

Contemporary cryptographic primitives

- Secret key (symmetric) primitives
  - Block cipher
  - Stream cipher
  - Integrity primitives
    - Message authentication code (MAC)
    - Hash functions
- Public key (asymmetric) primitives
  - Public key encryption scheme
  - Digital signature scheme

Life Cycle of a Cryptographic Algorithm



One Time Pad

- Claude Shannon laid (1949) the information theoretic fundamentals of secrecy systems.
- Shannon's pessimistic inequality: For perfect secrecy you need as much key as you have plaintext.
- In practical ciphers the key is much shorter than the plaintext to be encrypted
- Practical ciphers never provide perfect secrecy

DES Data Encryption Standard 1977 - 2002

- Standard for 25 years
- Finally found to be too small. DES key is only 56 bits, that is, there are about 10^{16} different keys. By manufacturing one million chips, such that, each chip can test one million keys in a second, then one can find the key in about one minute.
- The EFF DES Cracker built in 1998 can search for a key in about 4,5 days. The cost of the machine is $250 000.
- The design was a joint effort by NSA and IBM. The design principles were not published until little-by-little. The complete set of design criteria is still unknown.

Relative key lengths

Source: S. Blake-Wilson et al, RFC 3278: Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS), (based on A. Lenstra and E.Verheul (J.Crypt 1999)

Valid until	Symmetric	ECC	DH/DSA/RSA
2010	80	163	1024
2030	112	233	2048
2045	128	283	3072
?	192	409	7680
?	256	571	15360

Different Approaches in Cryptography

- information theoretic (e.g. bounded-storage)
- complexity theoretic
- quantum cryptology
- system based

Different assumptions:

- capabilities of an opponent
- cryptanalytic success
- definition of security (e.g., unconditional security, computational security)

Open Competitions

USA
- DES (1974)
- AES (1997)

Europe
- RIPE (1988-1992)
- NESSIE (2000-2003)
- eSTREAM (2005-)

Japan
- CRYPTREC project (2000-2003)

Constraints

Why not all algorithms are secure?

- public – proprietary
- weak – strong
- crypto competence
- export control
- economic reasons
- degradation over time (Moore's Law, quantum threat)

Search workloads and success probabilities

- Exhaustive search
- Preimage search
- Collision search for one function
- Collision search for two functions

Exhaustive key search

- Searching for a secret value used in a cryptosystem: keys, passkeys, etc in a set of size N. E.g., $N = 2^L$, where L is the key length in bits.
- Test based on given input and output; workload is measured in the number of tests to be performed
- Sometimes called as Dictionary attack when the test results are precomputed for all values of the searched parameter
- Uniform distribution assumed
- Success probability p =1, average workload $w = N/2$ trials
- Success probability p, search over set of size Np, average workload

$$w = p(Np/2) + (1-p)Np = Np - ½\, Np^2 .$$

Pre-image search

- One-way hash function H, modelled as a "random oracle": given input x the output $y = H(x)$ is picked uniformly at random
- Number of possible outputs N
- Search problem: given y find x such that $y = H(x)$
- After k trials the success probability:

$$p = 1- (1-1/N)^k = 1- ((1-1/N)^N)^{k/N}$$

$$\approx 1- e^{-k/N} > 1/2 \text{ , for } k > N \ln 2 \approx 0.693N$$

Collision search for the same function

- One-way hash function H, modelled as a "random oracle": given input x the output $y = H(x)$ is picked uniformly at random
- Number of possible outputs N
- Search problem: Find x_1 and x_2 such that $H(x_1) = H(x_2)$
- After $H(x)$ has been computed for k values of x the probability p that some value H(x) has appeared at least twice is (see Lecture 2):

$$p \approx \frac{1}{2} = e^{-\ln 2} \quad \text{for} \quad k \approx \sqrt{2N \ln 2} \approx 1.17\sqrt{N}$$

Collision search for two different functions

- Two one-way hash functions $H_1$and H_2 with the same target set modelled as “random oracles”: given input x the outputs $y_1 = H_1(x)$ and $y_2 = H_1(x)$ are picked uniformly at random
- Number of possible outputs N for both functions
- Search problem: Find x_1 and x_2 such that $H_1(x_1) = H_2(x_2)$.
- Create two sets:

$$A_1 = \{H_1(x) \mid x\} \text{ and } A_2 = \{H_2(y) \mid y\}$$

- Assume (for simplicity) that A_1 has k different elements, and in A_2 the values have been computed for k different y.
- Then the probability p that the sets have at least one element in common is (see Stallings, Appendix 11A and HW1, Problem 6)

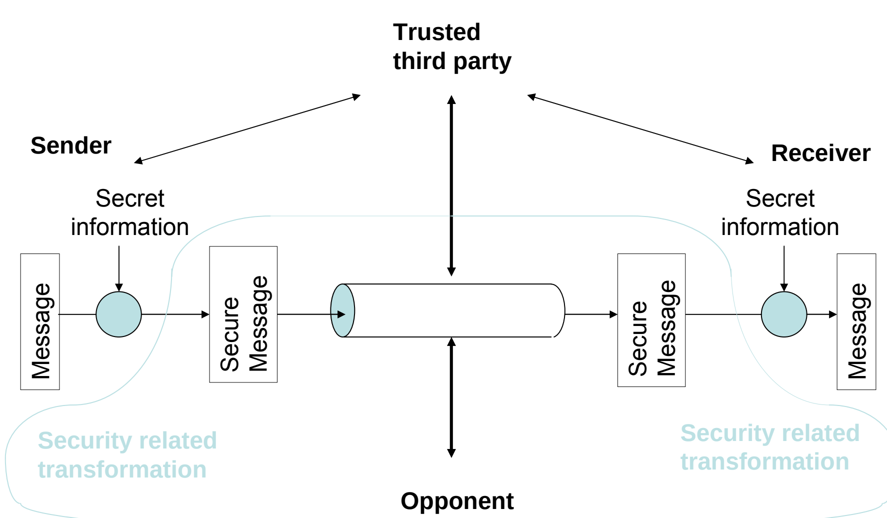
$$p \approx \frac{1}{2} \quad \text{for} \quad k \approx 0.87\sqrt{N}$$

Highlights

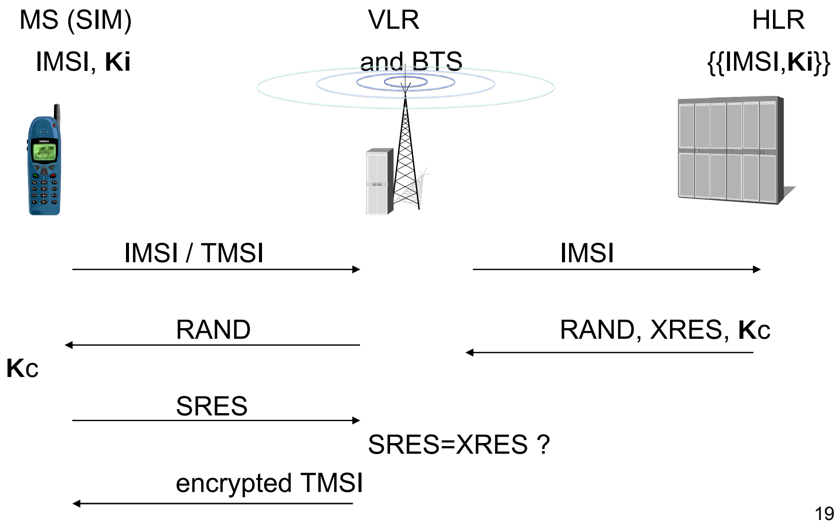
Contents of the course

- Introduction to data security
- Classical cryptosystems
- Introduction to modern cryptography
- Block ciphers: DES, IDEA, AES
- Stream ciphers: RC4, Snow2.0
- Block cipher modes of operation
- Hash-functions and MACs
- Mathematical tools: Modular arithmetic, Euclid's algorithm, Chinese Remainder Theorem, Euler's totient function, Euler's theorem
- Public key cryptosystems: RSA
- Prime number generation
- Polynomial arithmetic
- Public key cryptosystems: Diffie-Hellman, El Gamal, DSA
- Authentication and Digital signatures
- Random number generation
- Authentication and key agreement protocols in practise: Bluetooth, PGP, SSL/TLS, IPSEC, IKEv2 and EAP

Model for network security



GSM Authentication



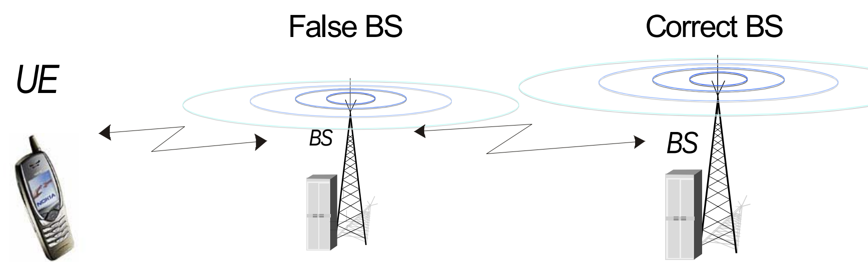
Criticism

Active attacks
- this refers to somebody who has the required equipment to masquerade as a legitimate network element and/or legitimate user terminal

Missing or weak protection between networks
- control data, e.g. *keys* used for radio interface ciphering, are sometimes sent unprotected between different networks

Secret design
- some essential parts of the security architecture are kept secret, e.g. the cryptographic algorithms

Active Attack



2.1 Classical Cryptosystems

Ceasar Cipher, or Shift Cipher

Plain: `meet me after the toga party`

Cipher: `PHHW PH DIWHU WKH WRJD SDUWB`

Monoalphabetic substitution

Alphabets

Plain: `abcdefghijklmnopqrstuvwxyz`

Cipher: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Key = permutation of the 26 characters

Size of key space 26! $\cong 4 \times 10^{26}$

Cryptanalysis based on statistical properties of the plaintext

Relative Frequency of Letters in English Text

A	8.167
B	1.492
C	2.782
D	4.253
E	12.702
F	2.228
G	2.015
H	6.094
I	6.996
J	0.153
K	0.772
L	4.025
M	2.406

N	6.749
O	7.507
P	1.929
Q	0.095
R	5.987
S	6.327
T	9.056
U	2.758
V	0.978
W	2.360
X	0.150
Y	1.974
Z	0.074

Playfair Cipher

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

The encryption rules

Plaintext formatting

`oo -> oxo`

Same row or column

`ar -> RM`

`mu -> CM`

Regular case

`hs -> BP`

`ea -> IM`

Polyalphabetic ciphers: Vigenère

Plain and Cipher:

finite sequences of characters in {0,1,2,…,25}

Key of period q : $k_1 k_2 k_3 \dots k_{q-1} k_q$

sequences of length q of characters in {0,1,2,…,25}

Encryption:

$c_1 = (p_1 + k_1)\bmod 26 \qquad c_{q+1} = (p_{q+1} + k_1)\bmod 26$

$c_2 = (p_2 + k_2)\bmod 26 \qquad c_{q+2} = (p_{q+2} + k_2)\bmod 26$

... ...

$c_q = (p_q + k_q)\bmod 26 \qquad c_{2q} = (p_{2q} + k_q)\bmod 26$

and so on..

Kasiski’s method to determine the period

- Many strings of characters repeat themselves in natural languages.
- Assume the interval between occurence of a string is a multiple of the period length.
- Then a repetition of a character string of the same length occurs in the ciphertext.
- By detecting repetitions of strings in the ciphertext one can find the period as the greatest common divisor (GCD) of the repetition intervals
- Their may be false repetitions. The longer the repeating string the more significant it is. Repeating strings of length ≥ 3 are the most significant.

One Time Pad

- Claude Shannon laid (1949) the information theoretic fundamentals of secrecy systems.
- Shannon’s pessimistic inequality: For perfect secrecy you need as much key as you have plaintext.
- An example of a cipher which achieves perfect secrecy is the One Time Pad

$$c_i = (p_i + k_i) \bmod 26$$

where the key is a string of characters $k_1 k_2 k_3 \dots k_i$ chosen uniformly at random.

- Practical ciphers do not provide perfect secrecy

Block ciphers, security

- Security is measured in terms of time: How long it takes to break the cipher using available resources.
- Upperbound of security: The time complexity of exhaustive key search, which is equal to 2^k, with key length of k bits.
- A second upperbound: $2^{n/2}$, with block length *n* (due to Birthday paradox, to be explained later)
- If an attack leads to a break, in time 2^t, where $t < k$, then the cipher is said to be *theoretically broken,* and that the *effective key length* of the cipher is reduced to *t*. (This does not mean that the cipher is broken in practise unless *t* is very small.)

Block ciphers, design principles

- The ultimate design goal of a block cipher is to use the secret key as efficiently as possible.
- Confusion and diffusion (Shannon)
- New design criteria are being discovered as response to new attacks.
- A state-of-the-art block cipher is constructed taking into account all known attacks and design principles.
- But no such block cipher can become provably secure, it may remain open to some new, unforeseen attacks.
- Common constructions with iterated round function
  - Substitution permutation network (SPN)
  - Feistel network

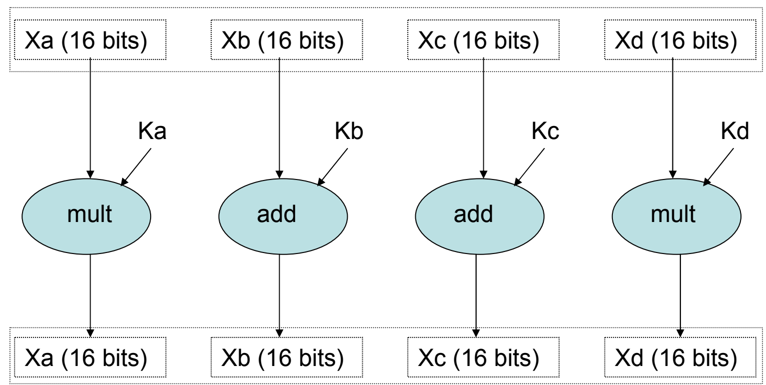
DES Data Encryption Standard 1977 - 2002

- Standard for 25 years
- Finally found to be too small. DES key is only 56 bits, that is, there are about 10^{16} different keys. By manufacturing one million chips, such that, each chip can test one million keys in a second, then one can find the key in about one minute.
- The EFF DES Cracker built in 1998 can search for a key in about 4,5 days. The cost of the machine is $250 000.
- DES has greately contributed to the development of cryptologic research on block ciphers.
- The design was a joint effort by CIA and IBM. The design principles were not published until little-by-little. The complete set of design criteria is still unknown.
- Differential cryptanalysis 1989
- Linear cryptanalysis 1993

The Security of IDEA

- IDEA has been around almost 15 years
- Designed by Xuejia Lai and Jim Massey
- Its only problem so far is its small block size
- Numerous analysis has been published, but nothing substantial
- It is not available in public domain, except for research purposes
- It is available under licence
- It is widely used, e.g in PGP (see Lecture 11)

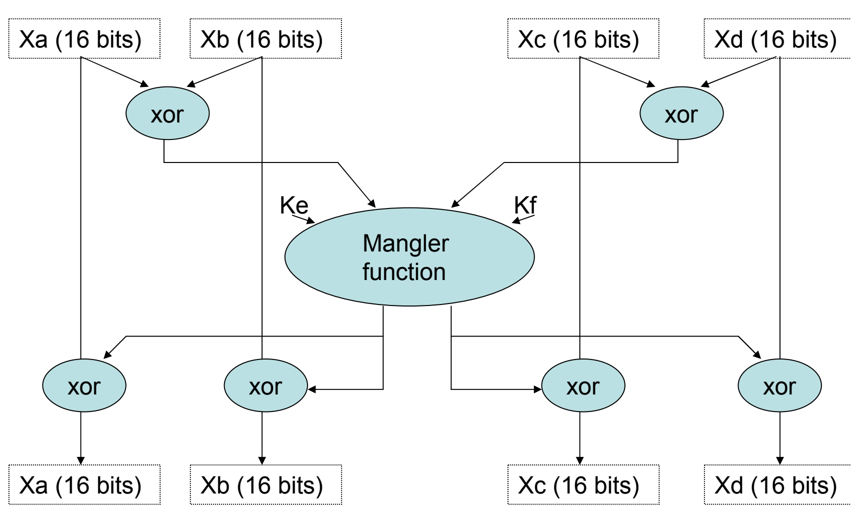
One round of IDEA: odd round



Legend: mult — Multiplication modulo $2^{16}+1$, where input 0 is replaced by 2^{16}, and result 2^{16} is encoded as 0

add — Addition modulo 2^{16}

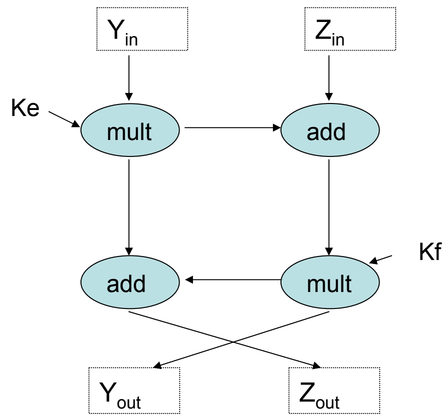
One round of IDEA: even round



The mangler function

Y_{out} = (Ke mult Y_{in}) add Z_{in}) mult Kf

Z_{out} = (Ke mult Y_{in}) add Y_{out}



AES

AES

- Candidates due June 15, 1998: 21 submissions, 15 met the criteria
- 5 finalists August 1999: MARS, RC6, Rijndael, Serpent, and Twofish, (along with regrets for E2)
- October 3, 2000, NIST announces the winner: Rijndael
- FIPS 197, November 26, 2001

  Federal Information Processing Standards Publication 197, ADVANCED ENCRYPTION STANDARD (AES)

AES(Rijndael) - Internal Structure

- First Initial Round Key Addition
- 9 rounds, numbered 1-9, each consisting of
  - Byte Substitution transformation
  - Shift Row transformation
  - Mix Column transformation
  - Round Key Addition
- A final round (round 10) consisting of
  - Byte Substitution transformation
  - Shift Row transformation
  - Final Round Key Addition

The Security of AES

- Designed to be resistant against differential and linear cryptanalysis
  - S-boxes optimal
  - Wide Trail Strategy
- Has quite an amazing algebraic structure (see the next slide)
- Algebraic cryptanalysis tried but not yet (!) successful
- Algebraic cryptanalysis: given known plaintext – ciphertext pairs construct algebraic systems of equations, and try to solve them.

Stream ciphers

- Stream ciphers are generally faster than block ciphers, especially when implemented in hardware.
- Stream ciphers have less hardware complexity.
- Stream ciphers can be adapted to process the plaintext bit by bit, or word by word, while block ciphers require buffering to accumulate the full plaintext block.
- Synchronous stream ciphers have no error propagation; encryption is done character by character with keys K_i that are independent of the data

  $C_i = E_{Ki}(P_i)$
- Function E is simple, the function which computes the key sequence is complex
- Example: Vigenère cipher, One Time Pad

  $C_i = (P_i + K_i)\text{mod } 26$

Stream ciphers: Security

- Known plaintext gives known key stream. Chosen plaintext gives the same but nothing more.
- Chosen ciphertext attack may be a useful method for analysing a self-synchronising stream cipher.
- The attacker of a stream cipher may try to find one internal state of the stream cipher to obtain a functionally equivalent algorithm without knowing the key.
- Distinguishing a key stream sequence from a truly random sequence allows also the keystream to be predicted with some accuracy. Such attack is also called prediction attack.

Requirements:

- Long period
- A fixed initialisation value the stream cipher generates a different keystream for each key.

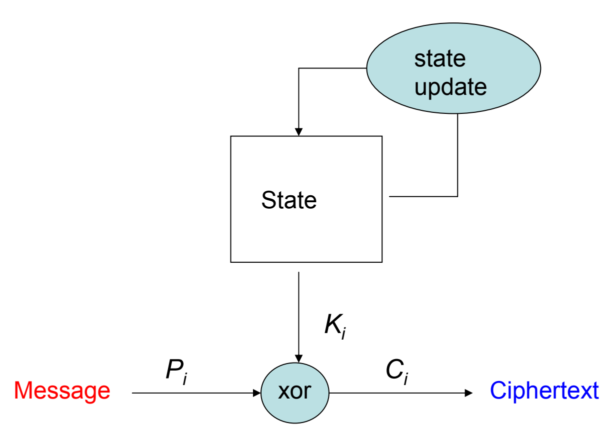
Stream ciphers: Designs

Linear feedback shift register (LFSR). LFSRs are often used as the running engine for a stream cipher.

Stream cipher design based on LFSRs uses a number of different LFSRs and nonlinear Boolean functions coupled in different ways. Three common LFSR-based types of stream cipher can be identified:

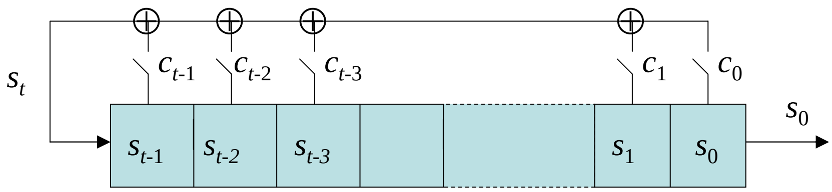
- *Nonlinear combination generators*: The keystream is generated as a nonlinear function of the outputs of multiple LFSRs
- *Nonlinear filter generators*: The keystream is generated as a nonlinear function of stages of a single LFSR.
- *Clock controlled generators*: In these constructions, the necessary nonlinearity is created by irregular clocking of the LFSRs. The GSM encryption algorithm A5/1 is an example of a stream cipher of this type.

Synchronous stream cipher: encryption



IV picks a different starting state for each new message

Linear Feedback Shift Register (LFSR)



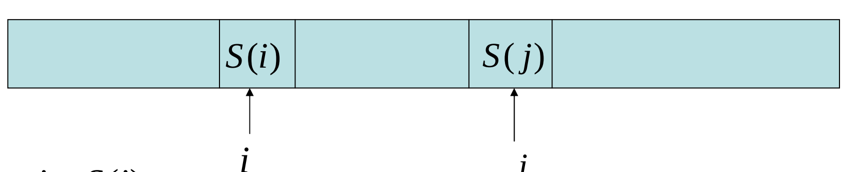
$$s_t = \sum_{i=0}^{t-1} c_i s_i = c_{t-1}s_{t-1} + c_{t-2}s_{t-2} + \ldots + c_0 s_0$$

The taps c_i are defined be giving the *feedback polynomial*

$$f(x) = x^t + c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \ldots + c_1 x + c_0$$

RC4

Register of 256 octets initialised using the key.
Counter i is set to zero. Then:



$j = S(i)$

$S(i)$ are $S(j)$ swapped

$k = (j + S(j)) \bmod 256$

$\text{ouput} = S(k)$

$i = (i+1) \bmod 256$

4.2 Block cipher confidentiality modes of operation

Block ciphers (in general) not good as such

- AES modes of operations:
  - ELECTRONIC CODEBOOK MODE, ECB
  - CIPHER BLOCK CHAINING, CBC
  - CIPHER FEEDBACK , CFB
  - OUTPUT FEEDBACK , OFB
  - COUNTER MODE, CTR

standardised by NIST, Special Publication 800-38A ,see:
http://csrc.nist.gov/publications/nistpubs/index.html

DES algorithm not good as such (small key size)

- Triple DES Special Publication 800-67

Triple DES (TDEA)

DES algorithm not good as such (small key size)

Double DES with two different keys K_1 and K_2 not good either (security not more than single DES) due to the Meet-in-the-Middle Attack (see next slide):

Triple DES Special Publication 800-67, see

http://csrc.nist.gov/publications/nistpubs/index.html

Triple DES with two keys

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

reduces to single DES, in case $K_1 = K_2$.

Meet in the Middle

Double DES with two different keys K_1 and K_2 not good either (security is not more than single DES due to the Meet-in-the-Middle Attack. Such attack can be launched when the attacker has two known plaintext-ciphertext pairs (P,C) and (P',C'). For such pairs obtained using the secret keys K_1 and K_2 the attacker has $C = E_{K_2}(E_{K_1}(P))$ and $C' = E_{K_2}(E_{K_1}(P'))$ or what is the same: $D_{K_2}(C) = E_{K_1}(P)$ and $D_{K_2}(C') = E_{K_1}(P')$.

Now we make a table T with a complete listing of all possible pairs $K_2, D_{K_2}(C)$ as K_2 runs through all possible 2^{56} values. The table has 2^{56} rows with 120 bits on each row. We make one more column to this table, and fill it with K_1 values as follows: For each K_1 we compute the value $E_{K_1}(P)$ and search in the table T for a match $D_{K_2}(C) = E_{K_1}(P)$. For each K_2 we expect to find a (almost) unique K_1 such that such a match occurs. Now we go through all key pairs K_1, K_2 suggested by table T, and test against the equation $D_{K_2}(C') = E_{K_1}(P')$ we have based on the second plaintext – ciphertext pair (P',C'). The solution is expected to be unique. The size of table T is 2^{56} (56 + 64 + ~56 bits) < 2^{64} bits, which is the memory requirement of this attack. The number of encryptions (decryptions) needed is about $4 \cdot 2^{56} = 2^{58}$.

5.1.Message authentication codes (MAC)

(Secret key , Message) ⟶ MAC

- A MAC of a message P of arbitrary length is computed as a function $H_K(P)$ of P under the control of a secret key K. The MAC is appended to the message by the sender.
- Given a message P and its MAC value M, the MAC can be verified by anybody in possession of the secret key K and the MAC computation algorithm.
- The MAC length m is fixed.
- Security requirement: it must be infeasible, without the knowledge of the secret key, to determine the correct value of $H_K(P)$ with a success probability larger than $1/2^m$. This is the probability of simply guessing the MAC value correctly at random. It should not be possible to increase this probability even if a large number of correct pairs P and $H_K(P)$ is available to the attacker.

Security requirements

The requirement: It must be infeasible, without the knowledge of the secret key, to determine the correct value of $H_K(P)$ with a success probability larger than $1/2^m$.

This means, in particular, that the following are satisfied

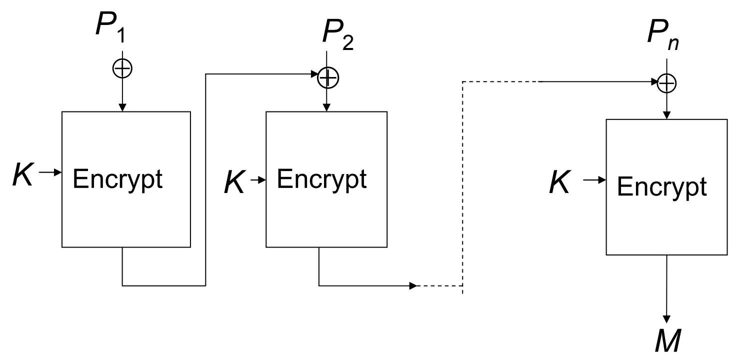
- Given a message P and $M = H_K(P)$ it should be infeasible to produce a modified message P' such that $H_K(P') = M$ without the knowledge of the key
- For each K, the function $P \rightarrow H_K(P)$ is one-way
- Given known MACs for a number of known (or chosen or adaptively chosen) messages, it should be infeasible to derive the key.

MAC Designs

- Similarly as block ciphers, MAC algorithms operate on relatively large blocks of data.
- Most MACs are iterated constructions. The core function of the MAC algorithm is a compression function. At each round the compression function takes a new data block and compresses it together with the compression result from the previous rounds. Hence the length of the message to be authenticated determines how many iteration rounds are required to compute the MAC value.

CBC MAC

A MAC mode of operation of any block cipher



- CBC encryption with fixed IV = `00…0`. The last ciphertext block (possibly truncated) is taken as the MAC.

Polynomial MAC

- Another MAC for stream ciphers
- Idea: An (cryptographically unsecure) error detecting code is encrypted using non-repeating keystream (ideally, a one-time pad)

An n-block message $P = P_0, P_1, \ldots, P_{n-1}$ with block size m bits is associated with the polynomial with m-bit coefficients:

$$P(x) = P_0 + P_1 x + P_2 x^2 + \ldots + P_{n-1} x^{n-1}$$

Also the value of the polynomial is assumed to be expressed as an m-bit string.

The secret key K consists of a point $x = X$ and an m-bit one-time key stream string $(k_0, k_1, k_2, \ldots, k_{m-1})$.

First the message polynomial is evaluated at the point X. Let us denote the value by $(c_0, c_1, c_2, \ldots, c_{m-1})$. The MAC is computed as the xor of the key stream string and the value as

$$(c_0 \oplus k_0, c_1 \oplus k_1, c_2 \oplus k_2, \ldots, c_{m-1} \oplus k_{m-1})$$

Note: The point X can be reused for different messages

Combined modes of operation

- CCM: Counter mode encryption and CBC MAC , see:

  1) IETF RFC 4309 (and RFC 3610)

  2) NIST Special Publication SP800-38C (with consideration to the IEEE 802.11i)

  (see Exercise 3.5)

- GCM: Counter mode encryption and a Polynomial-based MAC over Galois Field, see: http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/

Hash functions

Message → Hash code

- A hash code of a message *P* of arbitrary length is computed as a function *H*(*P*) of *P* . The hash length *m* is fixed.
- Hash function is public: Given a message *P* anybody can compute the hash code of *P*.
- Security requirements:
  1. *Preimage resistance:* Given *h* it is impossible to find *P* such that *H*(*P*) = *h*
  2. *Second preimage resistance:* Given *P* it is impossible to find *P'* such that *H*(*P'*) = *H*(*P*)
  3. *Collision resistance:* It is impossible to find *P* and *P'* such that *P* ≠ *P'* and *H*(*P'*) = *H*(*P*)

Design Principles

- Similarly as MAC algorithms, hash functions operate on relatively large blocks of data.
- Most hash functions are iterated constructions. The core function in a hash function is a compression function. At each round the compression function takes a new data block and compresses it together with the compression result from the previous rounds. Hence the length of the message to be authenticated determines how many iteration rounds are required to compute the MAC value.

SHA-1

- Designed by NSA
- FIPS 180-1 Standardi 1995 –
  www.itl.nist.gov/**fips**pubs/**fip**180-1.htm

February 2005:

Professor Xiaoyun Wang (Shandong University) announce an algorithm which finds collisions for SHA-1 with complexity 2^{69}

Recommendation: Use 256- or 512-bit versions of SHA:
csrc.nist.gov/publications/**fips**/**fips**180-2/**fips**180-2.pdf

Prime Numbers

Definition: An integer $p > 1$ is a prime if and only if its only positive integer divisors are 1 and p.

Fact: Any integer $a > 1$ has a unique representation as a product of its prime divisors

$$a = \prod_{i=1}^{t} p_i^{e_i} = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$$

where $p_1 < p_2 < \ldots < p_t$ and each e_i is a positive integer.

Some first primes: 2,3,5,7,11,13,17,… For more primes, see:

www.utm.edu/research/**primes/**

Composite (non-prime) numbers and their factorisations:

$18 = 2\times3^2$, $27 = 3^3$, $42 = 2\times3\times7$, $84773093 = 8887\times9539$

Extended Euclidean Algorithm and computing a modular inverse

Fact: Given two positive integers a and b there are integers u and v such that

$$u\times a + v\times b = \gcd(a,b)$$

In particular, if $\gcd(a,b) = 1$, there is a positive integer u such that

$$u\times a = 1 \pmod{b},$$

and similarly, there is a positive integer v such that

$$v\times b = 1 \pmod{a}.$$

u and v can be computed using the Extended Euclidean Algorithm, which iteratively finds integers r_i , u_i and v_i such that

$$r_{i-2} - q_i\times r_{i-1} = r_i \quad \text{and} \quad u_i\times a + v_i\times b = r_i$$

$$u_i = u_{i-2} - q_i \times u_{i-1} \quad \text{and} \quad v_i = v_{i-2} - q_i \times v_{i-1}$$

The index $i = n$ for which $r_n = \gcd(a,b)$ gives $u_n = u$ and $v_n = v$.

Chinese Remainder Theorem (general case)

Theorem: Assume $m_1, m_2, \ldots, m_t$ are mutually coprime. Denote $M = m_1 \times m_2 \times \ldots \times m_t$. Given $x_1, x_2, \ldots, x_t$ there exists a unique x, $0 < x < M$, such that

$x = x_1 \bmod m_1$

$x = x_2 \bmod m_2$

…

$x = x_t \bmod m_t$

x can be computed as

$x = (x_1 \times u_1 \times M_1 + x_2 \times u_2 \times M_2 + \ldots + x_t \times u_t \times M_t) \bmod M,$

where $M_i = (m_1 \times m_2 \times \ldots \times m_t)/ m_i$ and $u_i = M_i^{-1} \pmod{m_i}$

Euler's Totient Function $\phi(n)$

Definition: Let $n > 1$ be integer. Then

$\phi(n) = \#\{ a \mid 0 < a < n, \gcd(a,n) = 1\}$, that is, $\phi(n)$ is the number of positive integers less than n which are coprime with n.

For prime p, $\phi(p) = p-1$. We set $\phi(1) = 1$.

For a prime power, we have $\phi(p^e) = p^{e-1}(p-1)$

Given m,n, $\gcd(m,n) = 1$, we have $\phi(m \times n) = \phi(m) \times \phi(n)$.

Now Euler's totient function can be computed for any integer using its prime factorisation.

Example: $\phi(18) = \phi(2 \times 3^2) = \phi(2) \times \phi(3^2) = (2-1) \times (3-1)3^1 = 6$, that is, the number of invertible numbers modulo 18 is equal to 6. These numbers are: 1,5,7,11,13,17.

Euler's Theorem

$$Z_n^* = \{a \mid 0 < a < n, \gcd(a,n) = 1\}, \text{and } \# Z_n^* = \phi(n)$$

Euler's Theorem: For any integers n and a such that $a \neq 0$ and gcd(a,n) = 1 the following holds:

$$a^{\phi(n)} \equiv 1 (\bmod n)$$

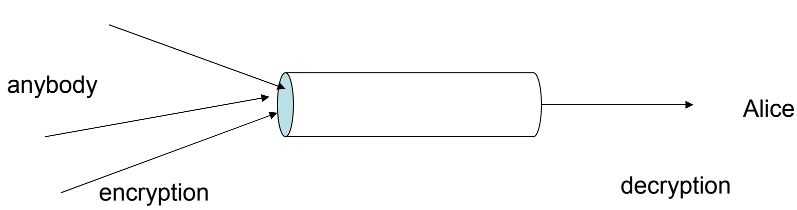
Fermat's Theorem: For a prime p and any integer a such that $a \neq 0$ and a is not a multiple of p the following holds:

$$a^{p-1} \equiv 1 (\bmod p)$$

The Principle of Public Key Cryptosystems

Encryption operation is public

Decryption is private



Alice's key for a public key cryptosystem is a pair: (K_{pub},K_{priv}) where K_{pub} is public and K_{priv} is cannot be used by anybody else than Alice.

Setting up the RSA

- Generate two different odd primes p and q
- Compute n = pq and compute $\phi(n) = (p-1)(q-1)$
- Select a public exponent e such that $\gcd(e, \phi(n)) = 1$
- Using Extended Euclidean Algorithm compute the multiplicative inverse of e modulo $\phi(n)$. Denote $d = e^{-1} \bmod \phi(n)$.

Public key: K_{pub} =(n,e)

Private key: K_{priv} = (n,d)

(or K_{priv} = (p,q,d). This is needed if private computations make use of the CRT)

n is called the RSA modulus; e is the public encryption exponent; d is the private decryption exponent.

Miller-Rabin Primality test

1. Let $n \geq 3$ be odd, consider the even number n -1, and write it as
$$n-1 = 2^k q, \text{ with } q \text{ odd}$$
2. Select a random integer a, 1< a < n -1.
3. If $a^q \bmod n = 1$ then return: n maybe a prime.
4. For j = 0 to k -1 do
5. if $a^{2^j q} \bmod n = n-1$ then return: n may be a prime
6. Return: n is composite

RSA encryption and decryption

Let M be a message, $0 \le M < n$. Then

Encryption of M is $C = M^{e} \bmod n$

Decryption of C is $M = C^{d} \bmod n$

This works, because $(M^{e})^{d} \bmod n = M$.

Proof (For M in Z_n^*): By Euler's theorem,

$M^{\phi(n)} \equiv 1 (\bmod n)$. On the other hand, $ed \equiv 1 (\bmod \phi(n))$

It follows:

$$(M^e)^d = M^{ed} = M^{1+k\phi(n)} = M \cdot (M^{\phi(n)})^k = M (\bmod n)$$

Polynomial Arithmetic

- Modular arithmetic with polynomials
- We limit to the case where polynomials have binary coefficients, that is, 1+1 = 0, and + is the same as -.

Example:

$$(x^2 + x + 1)(x^3 + x + 1) =$$

$$x^5 + x^3 + x^2 + x^4 + x^2 + x + x^3 + x + 1 =$$

$$x^5 + x = x \cdot (x^4 + 1) = x \cdot x = x^2 (\bmod (x^4 + x + 1))$$

Computation $\bmod (x^4 + x + 1)$ means that everywhere we take $x^4 + x + 1 = 0$,which means, for example, that $x^4 + 1 = x$.

Galois Field

Given a binary polynomial f(x) of degree n, consider a set of binary polynomials with degree less than n. This set has 2^n polynomials. With polynomial arithmetic modulo f(x) this set is a ring.

Faxt: If f(x) is irreducible, then this set with 2-ary (binary) polynomial arithmetic is a field denoted by $GF(2^n)$.

In particular, every nonzero polynomial has a multiplicative inverse modulo f(x). We can compute a multiplicative inverse of a polynomial using the Extended Euclidean Algorithm.

Example: Compute the multiplicative inverse of x^2 modulo $x^4 + x + 1$

Example: Modulo 2^3 arithmetic compared to $GF(2^3)$ arithmetic (multiplication).

In $GF(2^n)$ arithmetic, we identify polynomials of degree less than n:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

with bit strings of length n: $(a_0, a_1, a_2, \ldots, a_{n-1})$

and further with integers less than 2^n:

$$a_0 + a_1 2 + a_2 2^2 + \cdots + a_{n-1} 2^{n-1}$$

Example: In $GF(2^3)$ arithmetic with polynomial x^3+x+1 (see next slide) we get:

$4 \cdot 3 = (100) \cdot (011) = x^2 \cdot (x+1) = x^3 + x^2 = (x+1) + x^2 = x^2 + x+1 = (111) = 7$

Multiplication tables

modulo 8 arithmetic

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

$GF(2^3)$ Polynomial arithmetic

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	6
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Generated elements

Example: Finite field Z_{19}

$g = 2$
$g^i \bmod 19,\ i = 0,1,2,\ldots$

Element a = 2 generates all nonzero elements in Z_{19}. Such an element is called primitive.

i	g^i	i	g^i
0	1	10	17
1	2	11	15
2	4	12	11
3	8	13	3
4	16	14	6
5	13	15	12
6	7	16	5
7	14	17	10
8	9	18	1
9	18		

Example: Cyclic group in Galois Field

$GF(2^4)$ with polynomial $f(x) = x^4 + x + 1$

$g = 0011 = x+1$
$g^2 = x^2+1=0101$
$g^3 = (x+1)(x^2+1) = x^3 + x^2 + x + 1 = 1111$
$g^4 = (x+1)(x^3 + x^2 + x + 1) = x^4 + 1 = x = 0010$
$g^5 = (x+1)(x^4 + 1) = x^5 + x^4 + x + 1 = x^2 + x = 0110$
$g^6 = (x+1)(x^2 + x) = x^3 + x = 1010$
$g^7 = (x+1)(x^3 + x) = x^4 + x^3 + x^2 + x = x^3 + x^2 +1= 1101$
$g^8 = (x+1)(x^3 + x^2 +1) = x^4 + x^2 +x+1= x^2 =0100$
$g^9 = (x+1)x^2 = x^3 + x^2 = 1100$
$g^{10} = (x+1)(x^3 + x^2)= x^2 + x + 1= 0111$
$g^{11} = (x+1)(x^2 + x +1) = x^3 + 1 = 1001$
$g^{12} = (x+1)(x^3 + 1) = x^3 = 1000$
$g^{13} = (x+1)x^3 = x^3 + x + 1 = 1011$
$g^{14} = (x+1)(x^3 + x + 1) = x^3 + x^2 +x = 1110$
$g^{15} = (x+1)(x^3 + x^2 +x) = 1= 0001$

Discrete logarithm

Given $a \in \langle g \rangle = \{1,g^1,g^2,\ldots,g^{r-1}\}$, there is x, $0 \le x < r$ such that $a = g^x$. The exponent x is called the discrete logarithm of a to the base g.

Example: Solve the equation

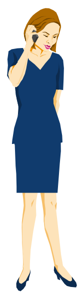
$$2^x = 14 \bmod 19$$

We find the solution using the table (slide 13): x =7.

Without the precomputed table the discrete logarithm is often hard to solve. Cyclic groups, where the discrete logarithm problem is hard, are used in cryptography.

Diffie-Hellman Key Exchange

ALICE



a secret

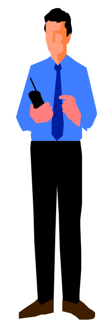
$A = g^a \mod p$

$K = B^a \mod p$

A →

← B

BOB



b secret

$B = g^b \mod p$

$K = A^b \mod p$

Man-in-the-Middle in the DH KE

Alice

a

g^a

g^a →

← g^{c2}

$K_2= (g^a)^{c2}$

Carl
(man-in-the-middle)

c1

g^{c1}

g^{c1} →

← g^b

c2

g^{c2}

$K_1= (g^b)^{c1}$

$K_2= (g^a)^{c2}$

Bob

b

g^b

$K_1= (g^b)^{c1}$

Protection using K_2 ↔

Protection using K_1 ↔

Setting up the ElGamal public key cryptosystem

- Alice selects a primitive element g in Z_p^* .
- Alice generates a, 0< a < p-1, and computes g^a mod p = A.
- Alice's public key: K_{pub} = (g, A)
- Alice's private key: K_{priv} = a
- Encryption of message m $\in Z_p^*$: Bob generates a secret, unpredictable k, 0< k < p-1. The encrypted message is the pair (g^kmod p, ($A^k \cdot$m) mod p).
- Decryption of the ciphertext: Alice computes $(g^{k)a} = A^k$ mod p, and the multiplicative inverse of A^k mod p. Then m = $(A^k)^{-1} \cdot (A^k \cdot m)$ mod p.

Diffie-Hellman Key Exchange and ElGamal Cryptosystem can be generalised to any cyclic group, where the discrete logarithm problem is hard.

Authentication functions

- Authentication functions are cryptographic primitives which are used by message authentication protocols between two parties, sender and receiver. Sender attaches to the message an authenticator. Receiver uses the authenticator to verify authenticity of the message.
- Authentication functions:
  - Message encryption
  - Message authentication code (MAC function)
  - Hash function

Message Authentication Protocols

Messages are sent from Alice to Bob:

Authenticity requirements:

1. Bob can verify that Alice sent the message
2. Bob can verify that the contents of the message is as it was when Alice sent it.
3. Bob can prove to Carol that Alice sent the message
4. Bob can prove to Carol what the message contents was when Alice sent it.
5. Alice cannot deny that she sent the message.

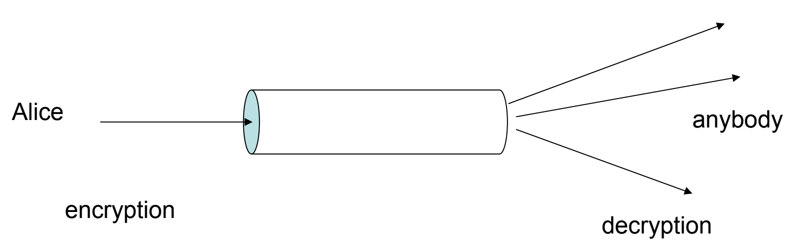
Requirements 1 and 2 can be fulfilled using protocols based on symmetric key authentication functions.

Requirements 3-5 can be fulfilled only using protocols based on asymmetric (public key) cryptosystems: Digital Signatures

Asymmetric encryption as authentication function

Encryption operation is private

Decryption is a public operation



Alice's key for a public key cryptosystem is a pair: (K_{pub},K_{priv}) where K_{pub} is public and K_{priv} is cannot be used by anybody else than Alice.

Digital Signature

Two types

- Digital signature with message recovery: the entire message is encrypted using the private key; before encryption some verifiable redundancy must be added to the message. The message authenticator is the entire ciphertext.
- Digital signature with appendix: First a hash code is computed from the message. Then the hash code encrypted using private key. The encrypted hash code is the authenticator, which is appended to the cleartext message.

The RSA Digital Signature

- Key derivation: the same as in RSA encryption:

  $n = pq$, p, q two different primes, e public exponent, d private exponent, $ed \bmod \phi(n) = 1$
- RSA authenticator generation function: given D the authenticator is computed as $S = D^d \bmod n$
- RSA verification function: given S, the RSA verification function is computed as $S^e \bmod n$
- Hash function: any hash function allowed
- Formatting of D is specified in PKCS#1 (octet string):

  $D = 0 \,||\, 1 \,||\, \{\text{at least eight octets of } ff_{16}\} \,||\, 0 \,||\, A$,

where A is the ASN.1 encoding of the hash type and the hash code of the message. The number of all-one octets in the middle is chosen to adjust the length of D at most equal to the length of the modulus n.

(|| denotes concatenation of octet strings)

The Digital Signature Algorithm DSA

- FIPS 186-2 (2000)
- DSA is a digital signature with appendix
- The complete specification defines:
  - The asymmetric cryptosystem: Key derivation, private key operation (for signature creation), public key operation (for signature verification)
  - Prime number generation
  - The hash function
  - Pseudo-random number generator

The DSA public key cryptosystem

Global public key components

p (old: prime number where $2^{L-1} < p < 2^L$, for $512 \leq L \leq 1024$ and L is a multiple of 64) changed in 2001 to: p is a 1024-bit prime

q a prime divisor of p-1, where q is a 160-bit number

g $= h^{(p-1)/q} \bmod p$, where h is any integer such that $1 < h < p-1$ and $h^{(p-1)/q} \bmod p \neq 1$. (Then the order of the group <g> generated by g in Z_p^* is equal to q.)

User's private key

x random or pseudo-random integer with $0 < x < q$

User's public key

$y = g^x \bmod p$

DSA: Signature generation

Message M; H = SHA-1(M) (considered as integer)

per-message randomizer:

k secret random or pseudorandom integer $0 < k < q$

The first part of the signature:

$r = (g^k \bmod p) \bmod q$

The second part of the signature:

$s = k^{-1} \cdot (H + r \cdot x) \bmod q$ ← Private key used here!

The signed message:

M,(r,s), where (r,s) is the authenticator appended to the message M

DSA: Signature verification

Verifier receives: M’,(r’,s’) and computes:

H’ = SHA-1(M’)

$w = s^{-1} \bmod q$

$u_1 = w \cdot H' \bmod q$

$u_2 = w \cdot r' \bmod q$

$v = g^{u_1} y^{u_2} \bmod p$ ← Public key used here!

and checks if v = r’.

The Use of Random Numbers

- Random numbers are needed in cryptographic protocols: there is no security without apparent randomness and unpredictability; things must look random to an external observer.
- Cryptographic keys
  - symmetric keys
  - Keys for asymmetric cryptosystems, random numbers with some additional properties
- Cryptographic nonces (= **n**umbers used **once**) to guarantee freshness

Random and pseudorandom numbers

Random numbers are characterised using the following statistical properties:

- Uniformity: Random numbers are uniformly distributed
- Independence: generated random numbers cannot be derived from other generated random numbers
- Generated using physical devices, e.g, quantum random number generator

Pseudorandom numbers are nonrandom numbers that cannot be distinguished from random numbers:

- Statistical distribution cannot be distinguished from the uniform distribution
- Independent-looking: pseudorandom numbers should be unpredictable, given a sequence of previously generated pseudorandom numbers
- Generated using deterministic algorithms from a short truly random or pseudorandom seed.

Cryptographic PRNGs

The security requirements for a cryptographically secure pseudorandom number generator are similar than those for a keystream generator. In practice, the difference lies in the fact that keystream generators are used for encryption and must be fast, and consequently, security is traded off to achieve the required speed. Random number generators are used for key and nonce generation, and therefore security is more important than speed.

Some standard PRNGs:

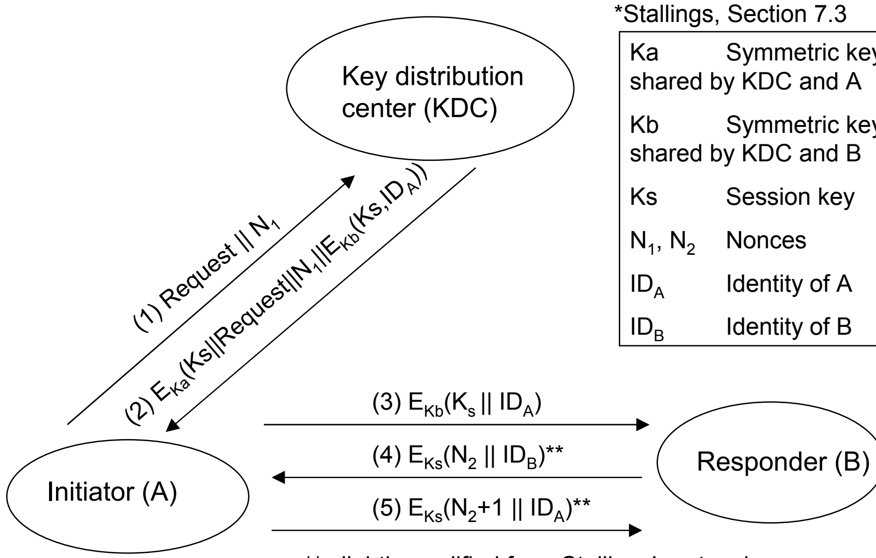
- Counter mode keystream generator is a cryptographically strong PRNG
- ANSI X9.17 PRNG based on Triple DES with two keys in encryption-decryption-encryption mode.
- FIPS 186-2 specifies a random number generator based on SHA-1 for generation of the private keys and per-message nonces for siganture generation
- Blum-Blum-Shub generator is provably secure if factoring is hard

Key Hierarchy

1. Master Keys
   - long term secret keys
   - used for authentication and session key set up
   - Distributed using physical security or public key infrastructure
2. Session Keys
   - short term secret keys
   - used for protection of the session data
   - distributed under protection of master keys
3. Separated session keys
   - short term secrets
   - to achieve cryptographic separation: Different cryptographic algorithms should use different keys. Weaknesses in one algorithm should not endanger protection achieved by other algorithms
   - derived from the main session key

A Key Management Scenario*

*Stallings, Section 7.3



Ka	Symmetric key shared by KDC and A
Kb	Symmetric key shared by KDC and B
Ks	Session key
N_1, N_2	Nonces
ID_A	Identity of A
ID_B	Identity of B

** slightly modified from Stallings' protocol

Authenticated Diffie-Hellman Key Exchange

Recall: Diffie-Hellman Key Exchange provides confidentiality against passive wiretapper. Active man-in-the-middle attack can be prevented using authentication, e.g. as follows:

K	Authentication key shared by A and B
a	private exponent of A
ID_A	Identity of A
ID_B	Identity of B

Distribution of Public Keys

- Public announcement
  - Just appending one's public key, or the fingerprint (hash) of the public key in one's signed email message is not secure
  - PGP public key fingerprints need to be truly authenticated based on face-to-face or voice contact
- Publicly available directory
  - An authorised directory, similar to phone directory that is published in print
- Public-key Authority
  - Public keys obtained from an online service. Communication needs to be secured
- Public-key Certificates
  - Public keys bound to user's identities using a certificate signed by a Certification Authority (CA)

CA and Registration Authority

Certification Authority
- E.g. in Finland: Population Register Center
- The certificate is stored in the subject's Electronic Identity Card

Registration Authority
- Identifies the user based on user's true identity and establishes a binding between the public key and the subject's identity

Management of private keys
- Private keys generated by the user
- Private key generated by a tusted authority
- Private key generated inside a smart card from where it is never taken out. The public key is taken out.

Certificate Revocation List
- Black list for lost or stolen private keys
- CRL must be available online for certificates with long validity period