Belarusian State Technological University
Department of Information Systems and Technology

Pavel Urbanovich

INFORMATION PROTECTION

Part 8: DIGITAL SIGNATURE

pav.urb@yandex.by, p.urbanovich@belstu.by

Assignment and Elements of Digital Signatures (DS)

Definition 1. DS is a binary string, which is added to the signed document *M*, depends on *M* and is equivalent to the manual way of signing documents.

Digital Signature System Model

*compute signature **s** on message **m** using (**Anna**'s or **Sender**'s) private key*

m, s → *to **B** (Borys)*

*verify whether **s** is the signature on **m** using A's (**Anna**'s or **Sender**'s) public key*

The most important DS Functions

- Ensuring the authenticity of a document - it allows to detect any unauthorized change of the content of a document,
- Sender identification - the document could be signed only by a person having a unique private key,
- Non-repudiation - the sender of the document can not refuse the signature placed under the document.

Types of DS

- Based on cryptography sym.,
- Based on cryptography sym. and Intermediary,
- Based on cryptography asym.,
- Based on cryptography asym. and hash-function.

DS based on Symmetric Cryptography

<u>DS creation:</u> $\mathbf{DS} = \mathbf{F}\ (m,k) \xrightarrow[\text{Implementation of a simple encryption}]{} \mathbf{c} = \mathbf{F}\ (m,k),$

k – a common key for the sender and recipient

it means: **DS ≡ c**; physically there is **no DS**.

<u>Signature verification:</u> $m = \mathbf{F}\ (c,k) \xrightarrow[\text{Implementation of a simple decryption}]{} \mathbf{PE} = \mathbf{F}\ (c,k)$

If ***m*** will be decrypted by key (***k***), which is known only to parties **A** and **B**, the result of the operation will be a confirmation of the <u>authenticity of m and the attribution of the authorship.</u>

Thus, all the basic functions of DS are performed.

DS based on Symmetric Cryptography and Intermediary (I)

1. **I** (pośrednik) generates different keys for **A** and **B**:$\mathbf{k_A}$ and $\mathbf{k_B}$.
2. **A** encrypts ***m*** by the key $\mathbf{k_A}$ and sends this (***c***) to **I**.
3. **I** decrypts ***c*** by the key $\mathbf{k_A}$, attaches to this information (***m***) new part of information (***m'***)
   (***m'***:'the sender of **m** is **A**'), thus **I** forms a new message:
   $m^I = m||m'$.
4. **I** encrypts m^I by the key k_B and sends it to ***B***.
5. **B** decrypts ***c*** by the key k_B, has ***m*** with a guarantee ('the sender of m is A') from **I**.

In this way:

- DS is real (**I**-guarantor),
- DS is not codified (only **A** and **B** know the keys: **I**-absolute trust),
- DS can not be re-used,
- The signed document ***M*** can not be changed,
- You can not deny the DS.

DS based on Asymmetric Cryptography

a) DS based on **RSA:**

- **A** encrypts **M** by own private key: $C = F(M; d_A, n_A)$,
- **B** decrypts **C** by the public key A: $M = F(C; e_A, n_A)$,

b) DS based on the **El-Gamal** algorithm:

1. Key generation:

•p - prime number;

•2 primes: **g** and **x** (**g, x < p**);

calculation: $y = g^x \bmod p$; public key - **p** and **g**, secret key - **x**.

2. **A** encrypts **M** by own secret key (**x**):

•takes a random number **k** - relatively primes of (**p-1**)

•PE generation: numbers **a** and **b**:

$a = g^k \bmod p$; **b** such that $M = (x\,a + k\,b) \bmod (p-1)$

3. **A** sends to **B**: **M**, **a**, **b**.

4. **B** receives **M**, **a**, **b** and verifies the signature:

DS belongs to **A**, if $(b^a a^b) \bmod p = g^M \bmod p$.

Example.

Let **p** = 11, **g** = 2, **x** = 8,
and **y = g^x mod p = 2^8 mod 11 = 3**;
public key: p = 11, g = 2, y = 3; secret key x = 8.

Signature (**M** = 5) notification:
A chooses **k = 9** (rel. prime of **p -1** = 10);
Calculation DS: **a = g^k mod p = 2^9 mod 11 = 6**;
on the basis of **M = (x * a + k * b) mod (p-1) A**
counts **b**:
5 = (8 * 6 + 9 * b) mod 10, b = 3.

A sends to **B: 5, 6, 3**
Verification of DS (by side **B**): **(3^6 * 6^3) mod 11 = 10;**
2^5 mod 11 = 10

<u>And finally</u>:
(3^6 * 6^3) mod 11 = 25 mod 11= 10.

DS based on Asymmetric Cryptography and Hash-function

Definition. A cryptographic hash-function H(M) takes as input a binary string of arbitrary length and returns a binary string of a fixed length.

Hash-functions which satisfy certain security properties are very important in cryptography and are widely used in cryptographic applications:

- digital signatures,
- public-key encryption systems,
- password protection schemes,
- conventional message authentication.

For a cryptographic hash-function **H**, the following problems are important:

- **Preimage:** given **y= H(x)**,find **x′** such that **H(x′) = y**
- **2nd preimage:** given **x**, find **x′ ≠ x** such **that H(x′) = H(x)**
- **Collision:** find **x** and **x′** such that **x′ ≠ x** and **H(x) = H(x′)**

Some hash-functions

System	Number of bits in hash code	Number of bits in message blocks	Designer	Year
MD4	128	512	Rivest	1990
MD5	128	512	Rivest	1991
SHA-0	160	512	US Gov.	1993
SHA-1	160	512	US Gov.	1995
SHA-256	256	512	US Gov.	2002
SHA-512	512	1024	US Gov.	2002
SHA-3	512	four variants	Four Belgians	2008

MD4

(Message Digest 4); R. Rivest

RFC 1320: MD4 Message-Digest Algorithm (1992)

https://www.ietf.org/rfc/rfc1320.txt

The entire algorithm can be divided into **five stages**:

- extension of the input message (append padding bits),
- breakdown of the expanded message into blocks,
- initialization of initial constants (constants **A, B, C, D** or MD buffer),
- message processing in blocks (main procedure of the hashing algorithm),
- taking the result.

•Extension of the input message

The message (**k** bits) is "padded" (extended) so that its length (in bits) is congruent to **448**,modulo **512**.

The message is expanded in such a way that its length (in bits) is congruent to 448 modulo 512. That is, the message is expanded to a size so that it lacks only 64 bits to have a length of 512. The message is always expanded, even if the length is already congruent to 448 modulo 512. This means that if 64 bits (= 512 - 448) are added to the size of the message in bits after the extension, then the resulting value should be a multiple of 512, i.e. should be divisible by 512 no residue.

Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Padding is performed as follows: a single „1” bit is appended to the message, and then „0” bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512.

In all, at least one bit and at most 512 bits are appended.

•Breakdown of the expanded message into blocks

Input: we have **n** blocks of **M** with a length of 512 bit each; each of these blocks are divided into 16 sub blocks with 32 bits (16 * 32 = 512).

•Initializationof initial constants

A four-word buffer (*A*,*B*,*C*,*D)* is used to compute the message digest. Here each of *A*,*B*, *C*,*D* is a 32-bit register.
These registers are initialized to the following values in hexadecimal, low-order bytes first):

word ***A*: 01 23 45 67**

word ***B*: 89 ab cd ef**

word ***C*: fe dc ba 98**

word ***D*: 76 54 32 10**

•Message processing in blocks

The entire algorithm consists of **3 rounds.**

We execute **16 steps in each round** (equal to the size of the sub-blocks).

At each step, the **non-linear function is calculated over the 3 variables from the set {*A*,*B*, *C*, *D*}:**

- **F(x,y,z) := xy ∨ (not x) z,**
- **G(x,y,z) := xy ∨ xz ∨ yz,**
- **H(x,y,z) := x ⊕ y ⊕ z.**

Each of the functions receives three 32-bit values on the input and forms one 32-bit result (Fig. 1).

Each of these steps uses a separate function: in the first round - **F (x, y, z)**, in the second round- **G (x, y, z)**, in the third round - **H (x, y, z)** (Fig. 2).

Truth Tables for the used Functions

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x	y	z	H
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

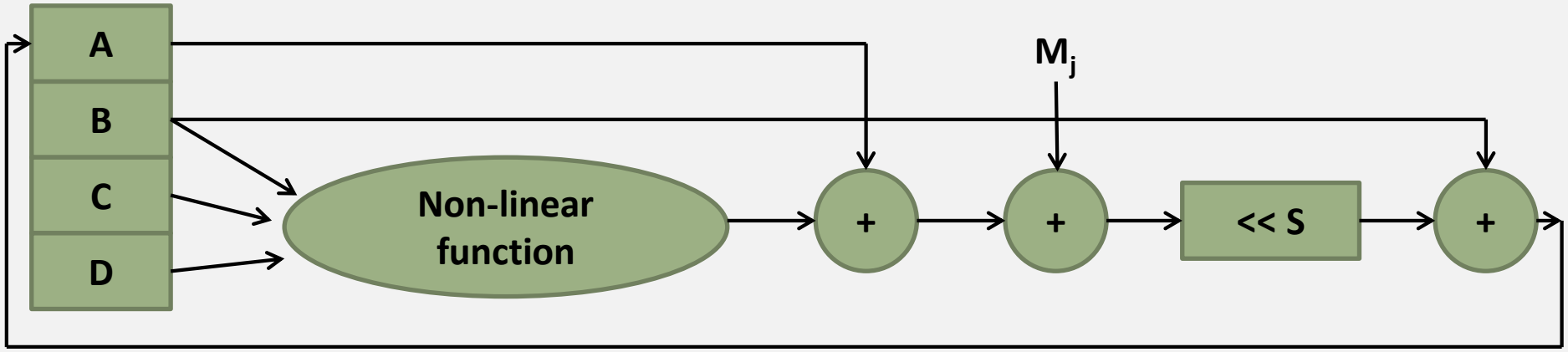

A
B
C
D
Non-linear function
+
+
<< S
+
M_j

Fig. 1

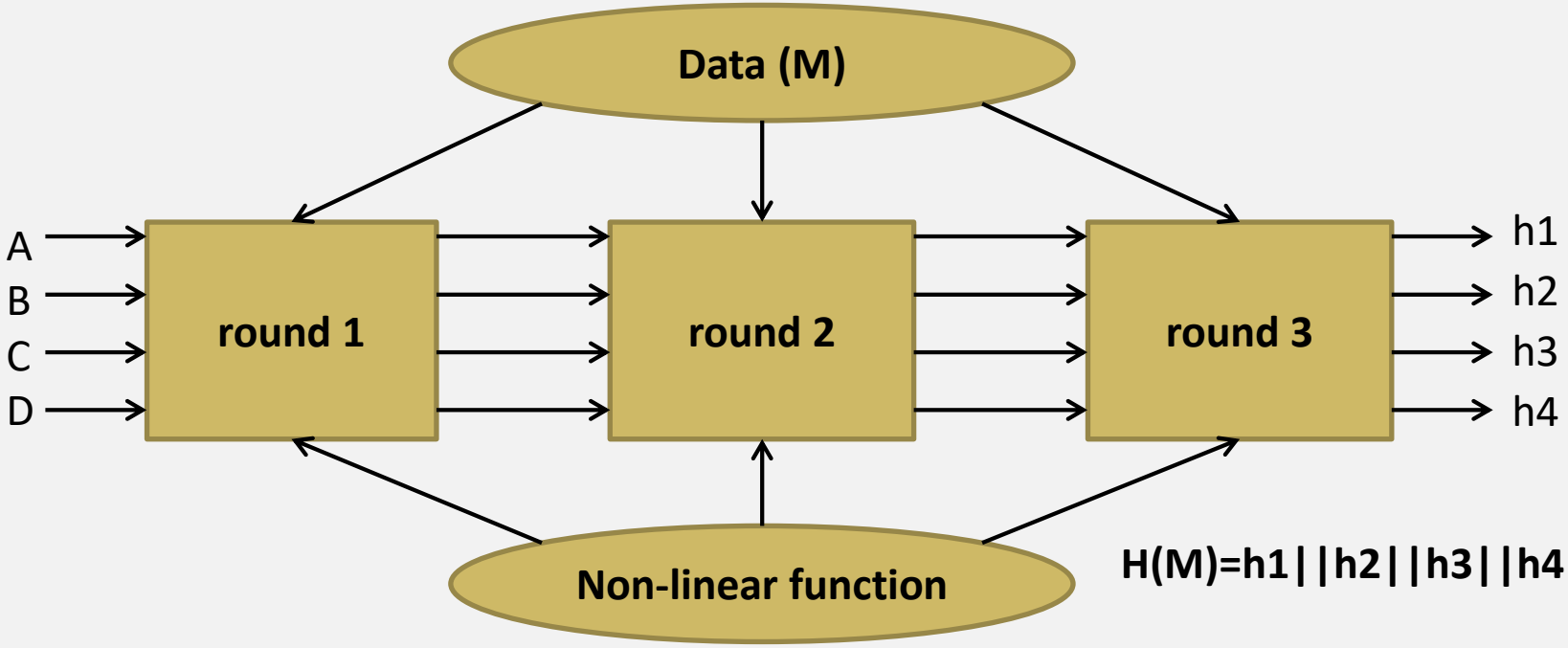

Data (M)
A
B
C
D
round 1
round 2
round 3
h1
h2
h3
h4
Non-linear function
H(M)=h1||h2||h3||h4

Fig. 2

The processor (we call it a block or a program module that performs the analyzed processing) also performs three basic transformation operations.
These operations use the functions **F, G** and **H** defined above and are expressed by the following formulas:

A = (A + F(B,C,D) + X[k]) <<< s,

A = (A + G(B,C,D) + X[k] + 5A827999) <<< s,

A = (A + H(B,C,D) + X[k] + 6ED9EBA1) <<< s

the symbol **<<<** is a cyclic shift operation to the left; **s** is the number of shift bits; **X [k]** is a 32-bit word, k = 0,1,2, ..., 15;

Example. The following form:

A = (A + F (B, C, D) + X [0]) <<< 3

means that the processor - based on the contents of registers A, B, C and D,

and also the word "0" (the first in the account of 16:
from 0 to 15) 32-bit

word of the processed 512-bit block - builds a new (current) value of register **A.**

Example.

Consider the features of the operations in question in the **1st** round.

Let ***[ABCD k s]*** denote the basic operation:

A= (A+ F(B,C,D) + X[k]) <<< s.

Then this round is, according to the algorithm, the following sequence of elementary transformations for each of the 16 words:

[ABCD0 3] [DABC1 7] [CDAB2 11] [BCDA3 19],

[ABCD4 3] [DABC5 7] [CDAB6 11] [BCDA7 19],

[ABCD8 3] [DABC9 7] [CDAB10 11] [BCDA11 19],

[ABCD12 3] [DABC13 7] [CDAB14 11] [BCDA15 19].

- **Output of the result**

Hash is a concatenation of the 4th 32-bit strings: **h1||h2||h3||h4**, so it is 128-bit hash (Fig. 2).

MD5

Differences between MD4 and MD5:

1. A fourth round has been added.
2. Each step now has a unique additive constant.
3. The function **g** in round 2 was changed from (**XY v XZ v YZ**) to (**XZ v Y not(Z)**) to make **g** less symmetric.
4. Each step now adds in the result of the previous step. This promotes a faster „avalanche effect” (efekt lawiny).
5. The order in which input words are accessed in rounds 2 and 3 is changed, to make these patterns less like each other.
6. The shift amounts in each round have been approximately optimized, to yield a faster „avalanche effect”. The shifts in different rounds are distinct.

Operation Diagram on one Step MD5

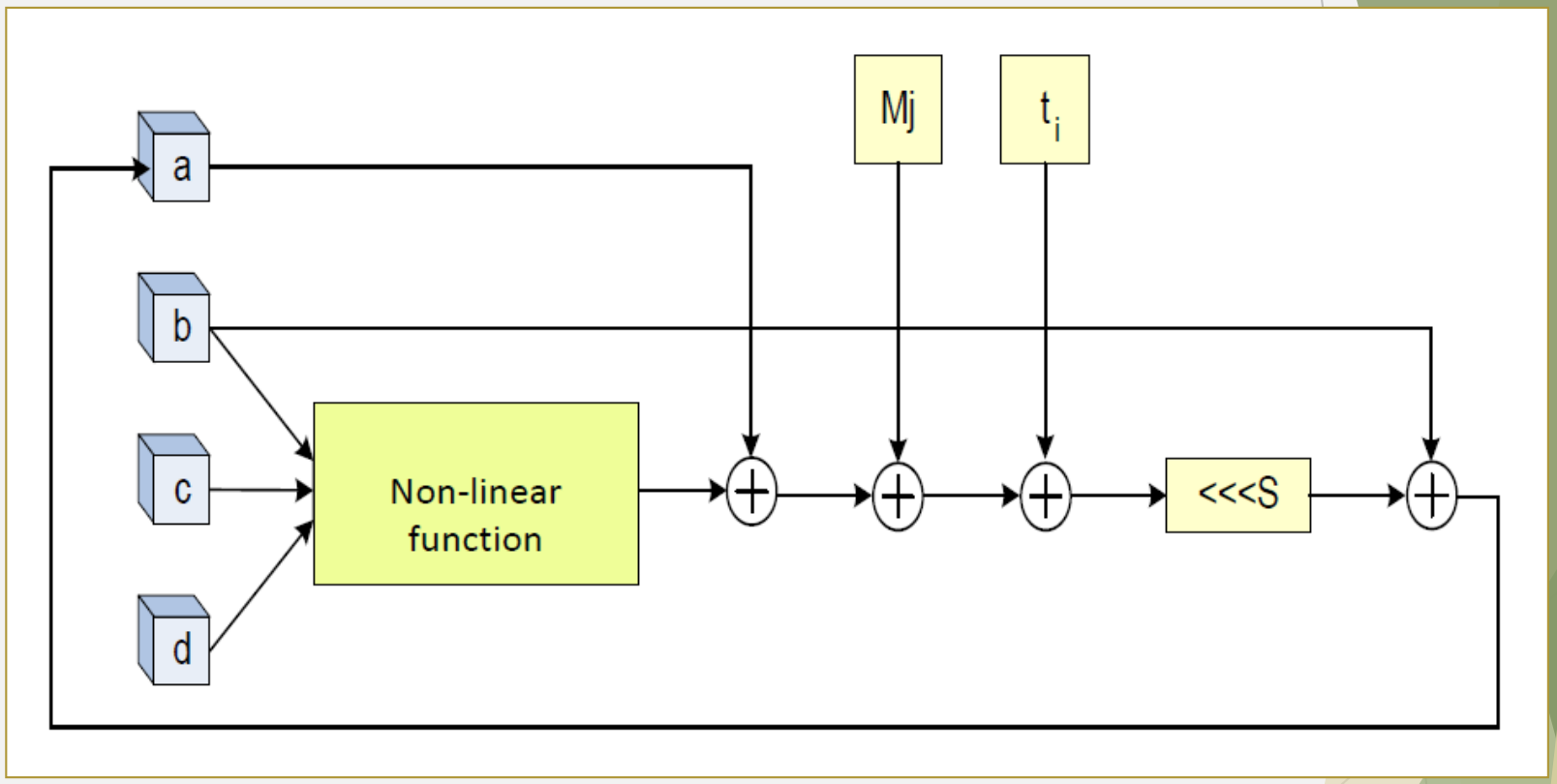


Fig. 3

The SHA Family Algorithms

- The **SHA** (Secure Hash Algorithm) family of functions is a collection of cryptographic hash functions created by the United States National Security Agency and has been approved by NIST as the Federal Information Processing Standard.
- This family was created when it turned out that the algorithm from the MD family and exactly the MD4 algorithm is not secure.
- The SHA family includes SHA-0, SHA-1 and SHA-2, SHA-3.
- There are several types of SHA-2 they have one algorithm but they differ in key so we differentiate them as SHA-224, SHA-256, SHA-384 and SHA-512.

Algorithm	Size of the Message (bits)	Block Size (bits)	Lenght of the Word (bits)
SHA-1	$< 2^{64}$	512	32
SHA-256	$< 2^{64}$	512	32
SHA-384	$< 2^{128}$	1024	64
SHA-512	$< 2^{128}$	1024	64

The Method of Generating a Hash-function using SHA-1

- **Input**: bit string **x** (message **m**)
- **Output**: 160-bit hash of string **x**

Steps of the algorithm:

- **Extension of the input message** (see MD4)
- The purpose of message padding is to make the **total length** of a padded message a **multiple of 512.**
- SHA-1 sequentially processes **blocks of 512 bits** when computing the message digest.
- The following specifies how this padding shall be performed. As a summary, a "1" followed by **m** "0"s followed by a **64-bit integer** are appended to the end of the message to produce a padded message of length **512 * n**.

•The **64-bit integer is the length of the original message.** The padded message is then processed by the SHA-1 as **n 512-bit blocks.**

•The padded message will contain **16 * n** words for some n > 0.

•Breakdown of the expanded message into blocks

Input: we have n blocks of M with a length of 512 bit each; each of these blocks are divided into16 blocks with 32 bits (16 * 32 = 512)

•Initialization of initial constants

•A: = 67452301

•B: = efcdab89

•C: = 98badcfe

•D: =10325476

•E: = c3d2e1f0

•Functions and constants used

•A sequence of logical functions $\mathbf{F_0,F_1,\ldots,F_{79}}$ is used in SHA-1.

•Each $\mathbf{F_t}$, 0 <= t<= 79,operates on three 32-bit words **B,C,D** and produces a**32-bit word as output.**

•**F (t;B,C,D)** is defined as follows:

each of the *four rounds uses* a separate nonlinear function for words **B,C,D** (**x, y, z** – in MD4),

F(t;B,C,D)= (B AND **C)** OR **((**NOT **B)** AND **D),** **(0<= t <= 19)**

G(t;B,C,D) = B XOR **C** XOR **D,** **(20<= t<= 39)**

H(t;B,C,D)= (B AND **C)** OR **(B** AND **D) OR (C** AND **D),** **(40<= t<= 59)**

I(t;B,C,D) = B XOR **C** XOR **D,** **(60<= t<= 79).**

A sequence of constant words
K_0, K_1, ... , K_{79} is used in the SHA-1.

Kt = 5A827999 (0 <= t <= 19)

Kt = 6ED9EBA1 (20 <= t <= 39)

Kt = 8F1BBCDC (40 <= t <= 59)

Kt = CA62C1D6 (60 <= t <= 79).

- **the main cycle iteratively processes each 512-bit block**

The iteration consists of four rounds of twenty operations in each.

The message block is converted from 16 of 32-bit words $\mathbf{m_i}$ into 80 of 32-bit words $\mathbf{w_t}$ according to the following rule:

- ✓ for **t** from 0 to15 calculate

  $\mathbf{w_t = m_j}$,

- ✓ for **t** from16 do 79 calculate

  $\mathbf{w_t = Shift1(m_{t-3} \oplus m_{t-8} \oplus m_{t-14} \oplus m_{t-16})}$,

  Shift1 - is acyclic left shift

✓ for **t** from 0 to 79: $_t$

temp = (A<<5) + F_t(B,C,D) + E+ w+ K_t

H5= D

H4= C

H3= B<<30

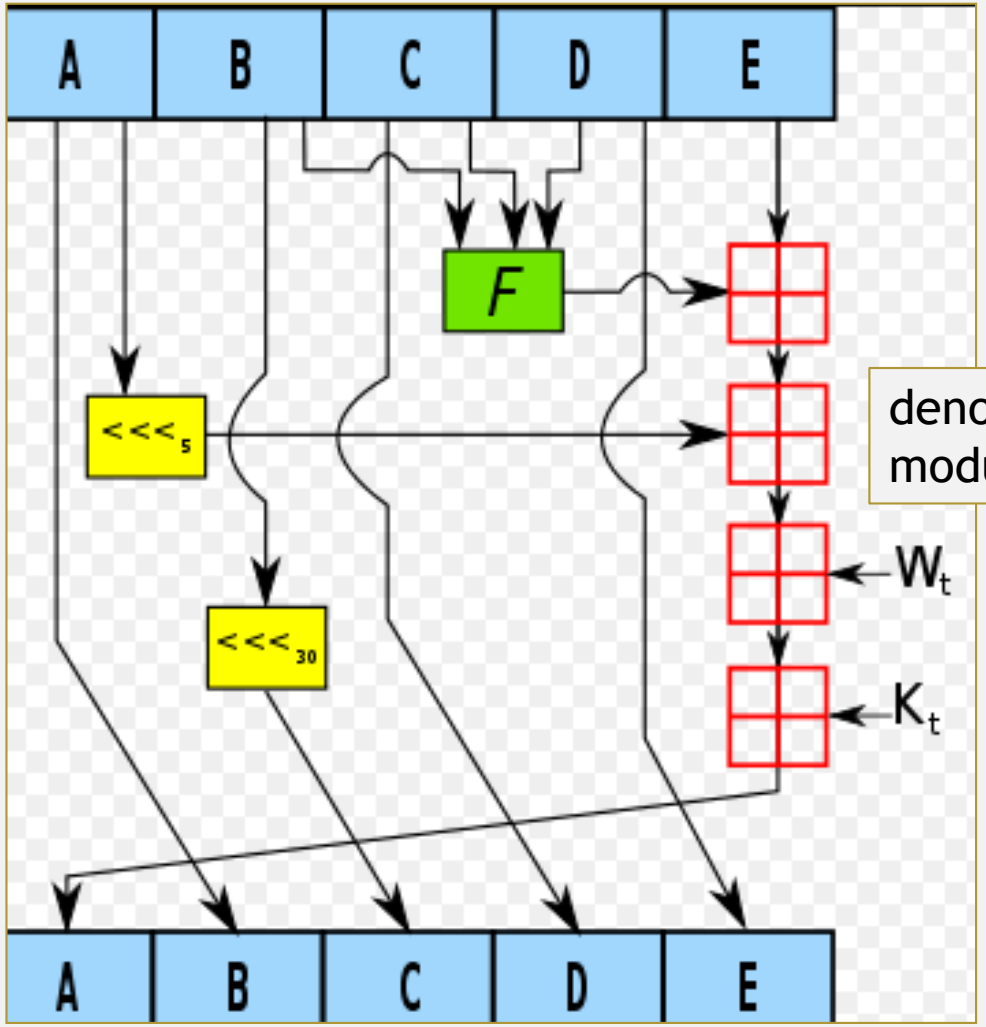
H2= A

H1= temp

After that **H1, H2, H3, H4, H5** are added to **A, B, C, D, E,** respectively (see Fig.4).

Then the next 512-bitblock is processed.

One iteration within the SHA-1 compression function



denotes addition modulo 2^{32}

Fig. 4

Source: https://ru.wikipedia.org/wiki/SHA-1

Parameters that are differ between **MD4** and **SHA-1**:

- the hash of the message is **160 bits** (SHA-1),
- five (instead of four) 32-bit variables are used to generate a hash,
- a hash is generated in four rounds (not three), each of 20 steps,
- the same functions **F, G, H** are used only in a different order:
  F in the first round, **G** in the third, **H** in the second and fourth,
- after extending the message to a length that is a multiple of 512, each of the 16-word blocks is further extended to 80 words of 32 bits; these 80 words are input sequences for the next 80 steps (4 rounds of 20 steps),
- SHA-1 uses four non-zero constants, and MD4 three, of which only two are non-zero.

Digital Signatures using the Hash-function

1. The digital signature standard, DSS

- In **1991** the US The National Institute of Standards and Technology (NIST) proposed a digital signature standard, **DSA (Digital Signature Algorithm).**
- It is the first digital signature that has **the same legal force as a handwritten signature.**
- This algorithm is a **variant of the ElGamal signature method.**
- The DSS standard uses the **SHA-1hash-function.**
- The **sender signs the hash function**, not the document.
- Thus, he does not have to reveal the content of the document.

DSA Parameters:

- **p** - a prime, where $2^{L-1} < p < 2^{L}$ for $512 \le L \le 1024$
and **L** is a multiple of **64**,
so **L** will be one member of the set
{512, 576, 640, 704, 768, 832, 896, 960, 1024}

- **q** - a prime divisor of **p-1**, where $2^{159} < q < 2^{160}$

- **g** = $b^{(p-1)/q} \bmod p$, where **b** is any integer with $1 < b < p-1$
such that $b^{(p-1)/q} \bmod p > 1$
(**g** has order **q mod p**)

the parameters **p, q**, and **g** are made public.

Next steps:

- **x** - a randomly or pseudorandomly generated integer with **0 < x < q**

- calculation: $y = g^x \bmod p$

- **k** - a randomly or pseudorandomly generated integer with **0 < k < q**

the parameters **x** and **k** are used for signature generation and must be kept private

k will be randomly or pseudorandomly **generated for each signature**

the private key is **x**,

the public key are **y**, **(p, q, g)**.

The **signature of the message m** will be a pair of the numbers: **r and s**,
which will be computed from the following equations:

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

$$s = (k^{-1}(h(m) + xr)) \bmod q,$$

k^{-1} is the multiplicative inverse of **k (mod q).**

Signature Verification

Before getting the digitally signed message the receiver must know the parameters **y, p, q, g**.

Let **m′, r′, s′** be the received versions of **m, r, and s**.
To verify the signature the verifying program must check to see that $0 < r' < q$ and $0 < s' < q$
and if either fails the signature should be rejected.

If both of the conditions are satisfied then we will compute:

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

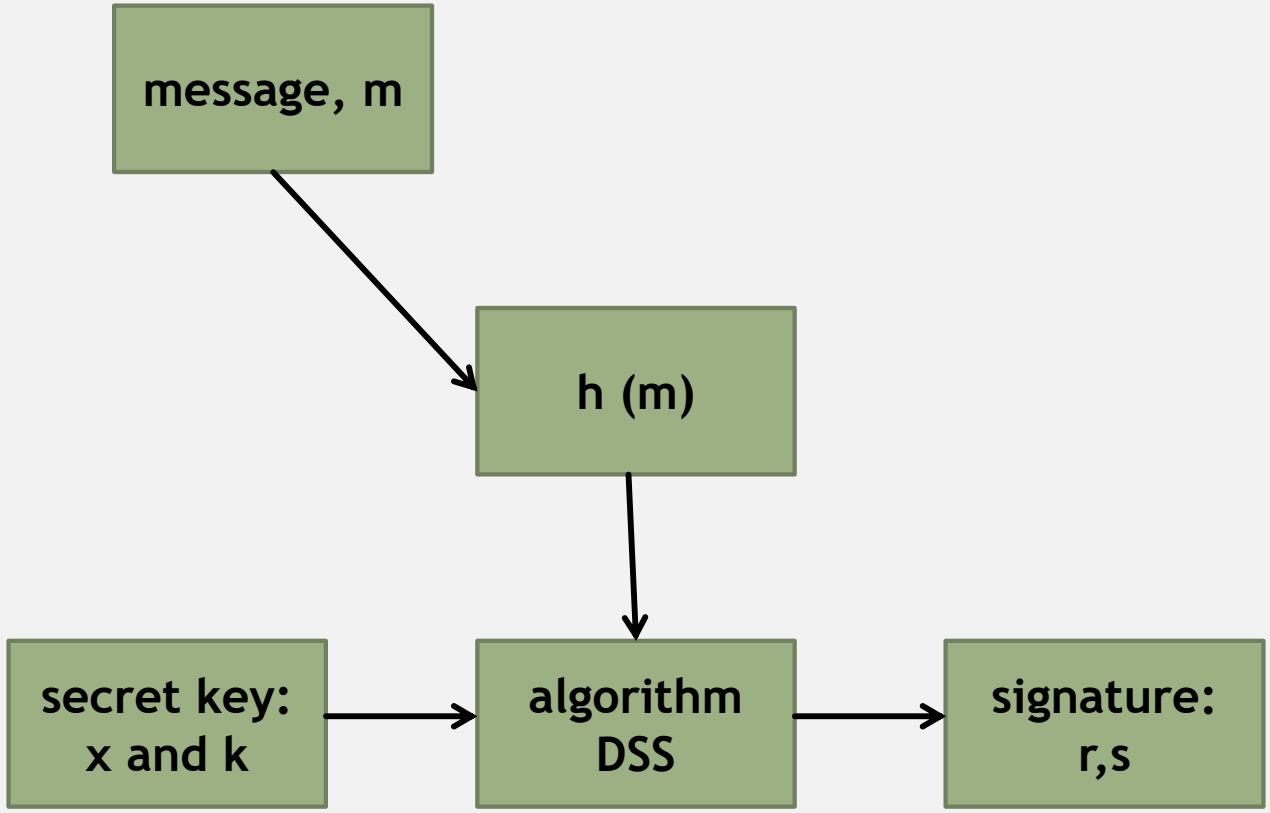
$u_1 = ((h(m'))w) \bmod q$

$u_2 = ((r')w) \bmod q$

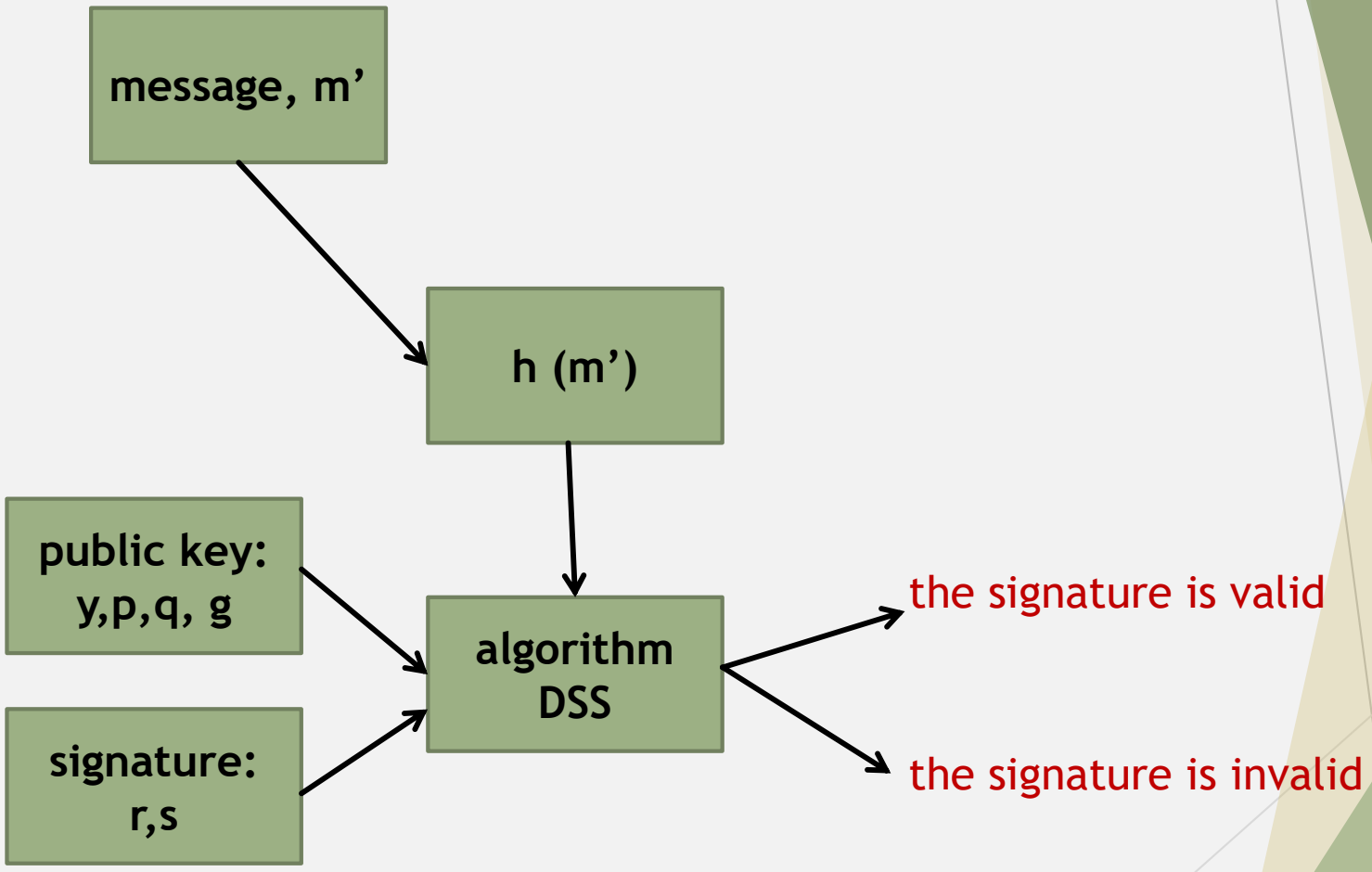
$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q$

Then if **v = r′ then the signature is valid**
and if not then it can be assumed that the data may have been changed or the message was sent by an impostor.

DS Creation



DS verification



2. Digital signature based on RSA

1) The signature creation:

- the sender calculates **h(m)**,
  and **$s = (h(m))^d \bmod n$** (where **d** is the sender's private key),
- the sender sends the message **m** and its signature **s**.

2) A person (recipient) who wants to verify the signature must use the following algorithm:

Let **m', s'** be the received versions of **m, s**.

Input: **m', s'**, as well as (**e, n**) - the sender's public key
the recipient calculates **m''= h (m')**,
if congruence **$m'' = (s')^e \bmod n$** (where **e** is the public key of the sender) is true, then the signature is accepted, otherwise rejected.

3) Proof of the correctness of the signature

We know that **$de \equiv 1 \bmod \varphi(n)$ (n)**.
Based on Euler's theorem, we get:
$(s)^e = (m)^d)^e \bmod n = (m)^{de} \bmod n = m$.

Attacks on Digital Signature Systems

The purpose of the attack on a digital signature is counterfeit (falsification) it.

This is possible if:

- an unauthorized person (an intruder) is able to counterfeit a signature for at least one message,
- the intruder is able to generate a signature for specific messages or a class of selected messages,
- the intruder is able to generate a signature for each message or find an effective algorithm being functionally equivalent to the message signing algorithm used.

There are two basic methods of a digital signature attacking :

1) an intruder only knows the sender's public key,
2) an intruder can analyze the signatures corresponding to either known or selected messages.

In the case of 2) we can differentiate three classes of attacks:

- an intruder has signatures for a set of messages that are known to him,

- an intruder is able to obtain valid signatures for selected messages before attempting to completely break the schema; this type of attack is non-adaptive in the sense that the message was selected before intruder came into possession of their signature,

- adaptive attack through the selected text – an intruder can receive the signatures of electronic documents, which he chooses himself, this is the most dangerous type of attack.

Electronic Signature Certificate

ESC - an electronic certificate that assigns data for validation of an electronic signature to an individual and confirms at least the name or pseudonym of that person

•The ESC consists of *data identifying the* user(eg.name and surname, e-mail address) and *electronic cryptographic key* (owner‘s public key) *used to verify the signed information and to encrypt it* (exact use is specified in the additional parameters included in the certificate).

•All information contained in the certificate is signed by the Certification Authority (CA), which gives the authenticity and the appropriate level of information verification.

- Thanks to the use of the certificate, the recipient of the message is sure that the sender of the message is who he or she claims to be and in the second case – that the integrity of the message is not violated.

- The digital certificate also contains, among others: *information on the validity period, the name of the issuing body and the serial number of the certificate*.

X.509 Public Key Certificates

- The **X.509** Public Key Infrastructure (**PKI**) standard identifies the requirements for robust public key certificates.
- A certificate is a signed data structure that binds a public key to a person, computer, or organization.
- Certificates are issued by *certification authorities (CA)*. All who are party to secure communications that make use of a public key rely on the CA to adequately verify the identities of the individuals, systems, or entities to which it issues certificates.
- The level of verification typically depends on the level of security required for the transaction. If the CA can suitably verify the identity of the requester, it signs (encrypts), encodes, and issues the certificate.

References:

1. C.E. Shannon. The Communication Theory of Secrecy Systems, Bell Sys. Tech. J. – V.28. – №4. – 1949. – P. 656-715.

2. J. Pieprzyk, T. Harjono, J. Seberry. Fundamentals of Computer Security, Springer-Verlag, 2003

3. B. Schneier. Applied Cryptography, Second Edition, John Wiley & Sons, 1996

4. Урбанович, П. П. Защита информации методами криптографии, стеганографии и обфускации: учеб.-метод. пособие для студ.. - Минск: БГТУ, 2016. - 220 с. (URL: http://elib.belstu.by/handle/123456789/23763)

5. Rivest, R. The MD4 Message Digest Algorithm, RFC 1320, MIT and RSA Data Security, Inc., April 1992.

6. Rivest, R. The MD4 message Digest Algorithm, in: A.J. Menezes and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90 Proceedings, Springer-Verlag, 1991, p. 303-311.

7. RFC 1321: MD5 Message-Digest Algorithm, [Electronic Resource], URL:https://tools.ietf.org/html/rfc1321

8. US Secure Hash Algorithm 1 (SHA1), [Electronic Resource], URL : https://tools.ietf.org/html/rfc3174

9. DSS: Digital Signature Standard, [Electronic Resource], URL : http://home.pacbell.net/tpanero/crypto/dsa.html

10. Закон Республики Беларусь Об электронном документе и электронной цифровой подписи от 28 декабря 2009 г. № 113-З, [Electronic Resource], URL: https://kodeksy-by.com/zakon_rb_ob_elektronnom_dokumente_i_elektronnoj_tsifrovoj_podpisi.htm

11. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, [Electronic Resource], URL: https://tools.ietf.org/html/rfc5280