Hash functions

Emil SIMION
e-mail: esimion@fmi.unibuc.ro

Agenda

General presentation of cryptological context of hash function usage;

Preliminary notions;

Hash functions;

Attacks on hash functions & Examples;

Applications;

References;

Q&A.

General presentation of cryptologic context of hash functions usage

Authentication versus confidentiality;

Need for authentication;

Digital signatures.

Authentication versus Confidentiality - 1

Authentication is a dual operator of the enciphering operator: in the authentication operation there is a redundancy of the message while in the enciphering operation this redundancy is reduced to minimum (can be ,,zero”).

Authentication is referring to:

- authentication of a received message, when the receiver want to be sure that the received message is coming form the real sender;

- to verify the identity of the user, process, or device often as a prerequisite to allowing access to resources in an information system.

Authentication versus Confidentiality -2

Order of cryptographic operations: first we sign the message then we encipher the message.

We may implement testing of verifying:

- *correct deciphering;*
- *signature (message authentication);*
- *integrity (protection against errors which can appear on communication channel during transmition of the message).*

Example of correct usage:

$E_k[M,MAC_k[M]]$, CRC $[E_k\ [M,MAC_k[M]]$.

Example of incorrect usage: if we encipher a message and we sign the result with the same RSA algorithm then this scheme is vulnerable;

Need for authentication

Assurance of authentication was imposed by:

- possibility of active eavesdropping when the attacker can manipulate the content of the intercepted message;

- electronic mail (e-mail) services where is more important to ensure the authentication rather the confidentiality of the message.

Beside the fact that the receiver of a message must be sure that the received message is coming from the real sender , an authentication system must solve eventually disagreements which can appear between the sender and receiver. In e-mail case the solution is based on CRIPTO protocols which involve the digital signature.

Digital signatures

Digital signature must have the following (theoretically) properties:

- the signature is unforgeable. The signature is proof that the signer, and no one else, deliberately signed the document;

- the signature is authentic. The signature convinces the document's recipent that the signer deliberately signed the document;

- the signature is not reusable. The signature is part of the document, an unscrupulous person cannot move the signature to a different document;

- the signed document is unalterable. After the document is signed, it cannot be alternated;

- the signature cannot be repudiated. The signature and the document are physical thinks. The signer cannot later claim that he or she didn't sign it.

The digital signature is applying on a hash (digest) of the document.

Digital signatures involve the following

1) Usage symmetric cryptosystems;
2) Usage asymmetric cryptosystems;
3) Need for one –way functions: data integrity code (DIC), manipulation detection code (MDC), message authentication code (MAC), data authentication code (DAC);
4) Conventional Digital signatures:
   - ElGamal;
   - Schnorr;
   - DSA (Digital Signature Algorithm) designed by NSA;
5) Other types of digital signatures:
   - invisible signature (only the legitimate user can see the signature);
   - fail-stop signature (in case of a forgery signer can proof data manipulation).
6) Standards & Legal Issues.

„Fathers” of PKI (incl. digital signatures)

From left to right: Adi Shamir, Ron Rivest, Len Adleman, Ralph Merkle, Martin Hellman, and Whitfield Diffie.

Diffie, Hellman and Merkle

Diffie, Hellman, Merkle. The first researchers to discover and publish the concepts of PKC were Whitfield Diffie and Martin Hellman from Stanford University, and Ralph Merkle from the University of California at Berkeley. As so often happens in the scientific world, the two groups were working independently on the same problem -- Diffie and Hellman on public key cryptography and Merkle on public key distribution -- when they became aware of each other's work and realized there was synergy in their approaches. In Hellman's words: "We each had a key part of the puzzle and while it's true one of us first said X, and another of us first said Y, and so on, it was the combination and the back and forth between us that allowed the discovery."

The first published work on PKC was in a groundbreaking paper by Whitfield Diffie and Martin Hellman titled "New Directions in Cryptography" in the November, 1976 edition of IEEE Transactions on Information Theory, and which also referenced Merkle's work. The paper described the key concepts of PKC, including the production of digital signatures, and gave some example algorithms for implementation. This paper revolutionized the world of cryptography research, which had been somewhat restrained up to that point by real and perceived Government restrictions, and galvanized dozens of researchers around the world to work on practical implementations of a public key cryptography algorithm.

Diffie, Hellman, and Merkle later obtained patent number 4200770 on their method for secure public key exchange.

Rivest, Shamir and Adleman

Rivest, Shamir, Adleman (RSA). The Diffie-Hellman-Merkle key exchange algorithm provided an implementation for secure public key distribution, but didn't implement digital signatures. After reading the Diffie-Hellman paper, three researchers at MIT named Ronald Rivest, Adi Shamir, and Leonard Adleman (RSA) began searching for a practical mathematical function to implement a complete PKC approach. After working on more than 40 candidates, they finally discovered an elegant algorithm based on the product of *two prime numbers* that exactly fit the requirement for a practical public key cryptography implementation.

In 1982, RSA formed the company RSA to market their PKC algorithm in electronic security products. They obtained patent number 4405829 on the RSA algorithm in the US, but could not obtain a patent internationally because they had already published the idea and most other countries bar retroactive patenting of open source concepts.

Because it provides secure communications over distances between parties that have not previously met, RSA provides the ideal mechanism required for private communications over electronic networks, and forms the basis of almost all of the security products now in use on the Internet for financial and other private communications, including most organizational level Public Key Infrastructure (PKI) systems.

In September 2000, the US patent for the RSA algorithm expired, for the first time enabling software developers everywhere to *freely* include this PKC standard in their products.

and GCHQ group: Ellis, Cocks and Wiliamson

Ellis, Cocks, Williamson. In December, 1997, it was revealed that researchers at the GCHQ organization did some work in the early 1970's in the field of "non-secret encryption", which is related to public key cryptography, but without inclusion of the concept of *digital signatures*. However, these claims are not verifiable since the work was not published, and there are no evidentiary artifacts available such as original copies of the papers (although modern transcriptions are linked below). Therefore, in keeping with a long tradition, credit for the development and publication of PKC must remain with the researchers who first published their work in the open scientific literature, as described above.

Preliminary notions on hash functions

One-way functions;

Examples.

One-Way Functions

Function f is called one-way if:

1) given the value x, it is easy to compute f(x);
2) given f(x), it is computationally difficult to compute the input x.

Function f is one-way with trap-door if:

1) given the value x, it is easy to compute f(x);
2) given f(x), it is computationally difficult to compute the input x;
3) based on a ,,secret” information y, it is easy to compute x from f(x).

Examples of one-way functions

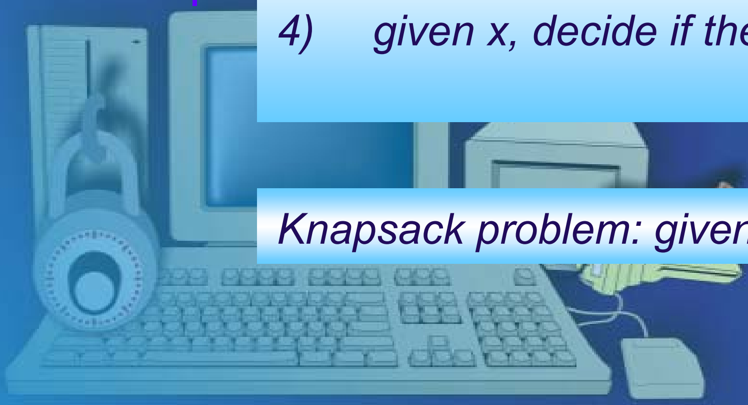
Discrete logarithm:

given p (prim number), g and y find the value x such that $g^x = y \mod p$;

Factorization: if N is the product of two prime numbers (unknown) then:

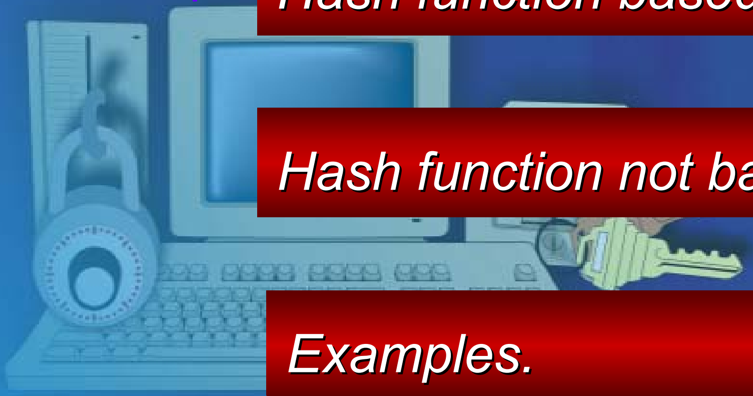
1) find the factors of N;
2) given e si C, find M such that $M^e = C \mod N$;
3) given M si C find the value d such that $C^d = M \mod N$;
4) given x, decide if there is a value y such that $x = y^2 \mod N$.

Knapsack problem: given a set of integer values find a subset with sum S.



Hash functions

- Definitions;
- Hashing algorithms;
- Hash function based on block ciphers;
- Hash function not based on block ciphers;
- Examples.



Definitions

Hash function is a function which input is an arbitrary length bit string and output a bit string of fixed length (generally output length is 64, 128 or 256 bits);

A function H is called one-way hash function if:

1) *H is hash function;*
2) *H is one-way function.*

For to be used in cryptographic applications (example in connection with digital signatures) one-way hash functions must provide:

1) *for every (given) M, it is difficult to find M' such that H(M)=H(M');*
2) *it is difficult to find a pair (M, M') such that H(M)=H(M').*

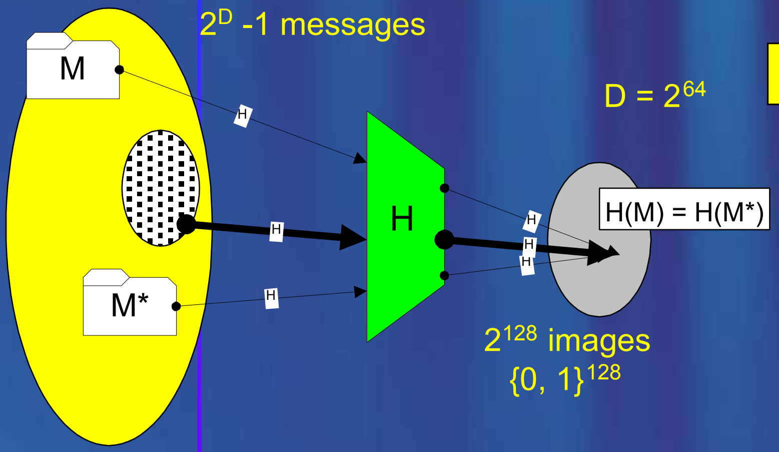
Hashing algorithms

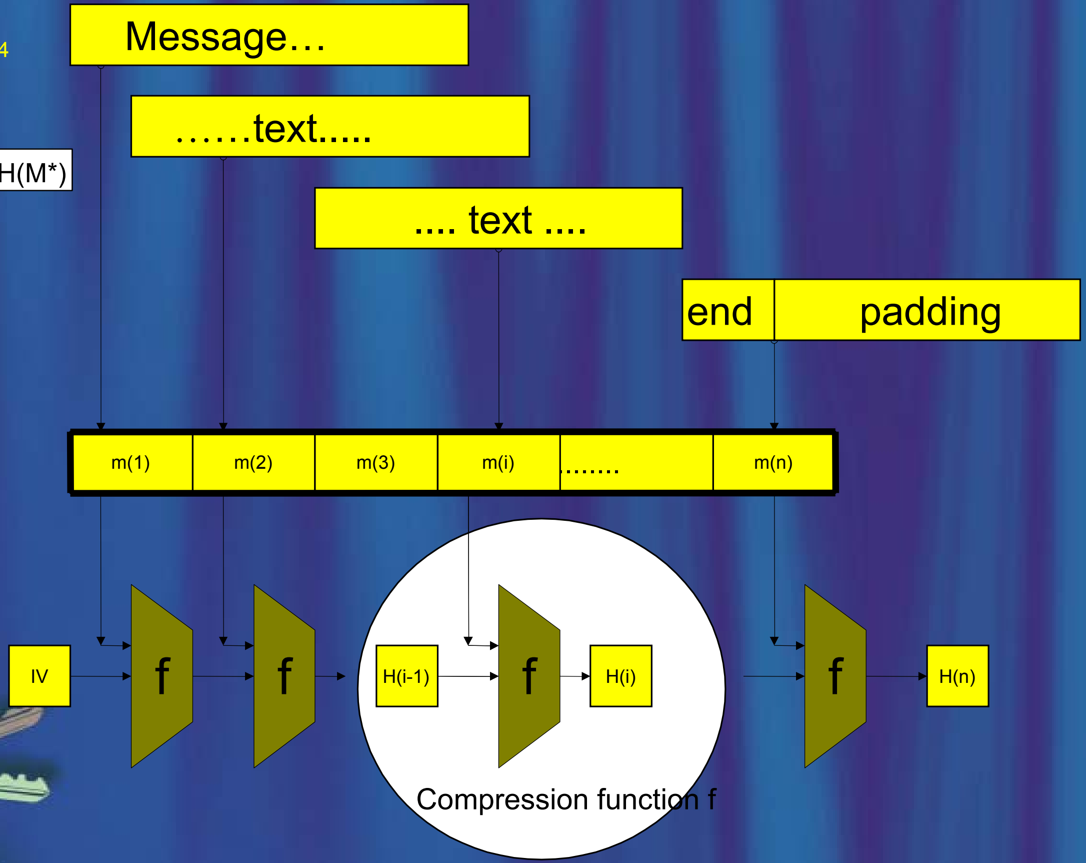
There is a variety of one-way hash functions design;

Some one-way hash functions produce the output of length n based on two inputs on the same length n.

In generally, in this case the input is a block of the message (part of a message) and the previous block hash, that is $h_i=f(M_i,h_{i-1})$.

General hash scheme





Hash functions are:
- one-way and collision-free;
- robust and complex;
- the problem is to find collisions.

Hash functions based on block ciphers

Let us denote by:

E(K,M) a block cipher algorithm which operates in first variable with the key and in the second with plain text;

IV – initialisation block;

t – number of block in which the message has been divided ;

In the following we have (if no other specify) H_0=IV and H(M) = H_t thus we specify only the recurrence for i=1,…,t.

Rabin: $H_i=E(M_i,H_{i-1})$

Cipher Block Chaining: $H_i=E(K, M_i \oplus H_{i-1})$

Combined Plaintext - Ciphertext Chaining: $H_i=E(K, M_i \oplus M_{i-1} \oplus H_{i-1})$, $M_0=H_0=0$, $M_{t+1}=IV$, $H(M)=H_{t+1}$

Key chaining: $H_i=E(M_i \oplus H_{i-1}, H_{i-1})$

Davies-Meyer: $H_i=E(M_i,H_{i-1}) \oplus H_{i-1}$

Matyas: $H_i=E(H_{i-1},M_i) \oplus M_i$

N-hash: $H_i=E(M_i,H_{i-1}) \oplus M_i \oplus H_{i-1}$

Miyaguchi: $H_i=E(H_{i-1},M_i) \oplus M_i \oplus H_{i-1}$

$H_i=E(M_i, M_i \oplus H_{i-1}) \oplus H_{i-1}$

$H_i=E(H_{i-1}, M_i \oplus H_{i-1}) \oplus M_i$

$H_i=E(M_i, M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$

$H_i=E(H_{i-1}, M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$

$H_i=E(M_i \oplus H_{i-1}, M_i) \oplus M_i$

$H_i=E(M_i \oplus H_{i-1}, M_{i-1}) \oplus H_{i-1}$

Hash function non-based block ciphers

1) RSA type: $H_i = (M_i \oplus H_{i-1})^e$ mod N, where e and N are public;
2) Quadratic type: H_i extracts m bits from $(00111111||H_i||M_i)^2$ mod N;
3) There are hashing schemes based on cellular automata, Fourier transform etc;
4) From hash function non-based on block ciphers we remember MD2, MD4 and MD5 designed by Ron Rivest, SHA designed by NSA (also FIPS standard), RIPEMED designed by den Boer (RACE european project) and MDC2 designed by IBM.

Examples of hash functions

1) SNERFU: hash function designed by Ralph Merkle, process blocks of 512 bits length, output hashing value is on 128 or 256 bits;

2) N-HASH: hashing algorithm designed by Nippon Telephone and Telegraph, process blocks of 128 bits length, output hashing value is on 128 bits;

3) MD2, MD4 si MD5: hashing algorithms designed by Ron Rivest, process blocks of 512 bits length, output hashing value is on 128 bits. Used in PEM protocols;

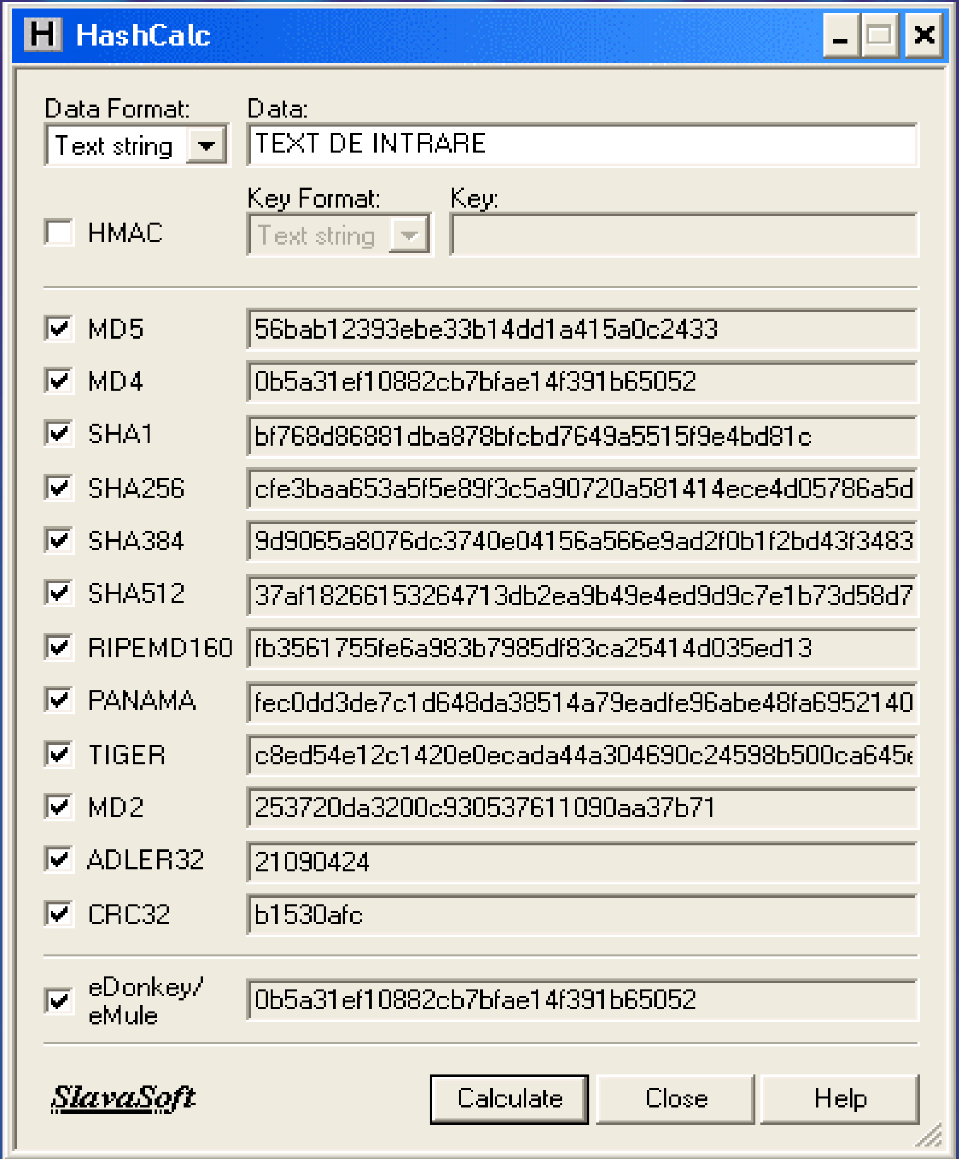
4) SHA: hash algorithm designed by NSA to be used with Digital Signature Standard, process blocks of 512 bits length, output hashing value is on 160 bits;

5) RIPE-MD: hash algorithm designed for UE in RIPE project, algorithm is a variation of MD4;

6) HAVAL: modification of MD5, process blocks of 1024 bits length, ouput hashing value is on 128, 160, 192, 224 or 256 bits.

Software of hash computing

For testing implementation and the results we may use OpelSSL or others software designed to run on Windows OS such as HashCalc (desiged by SlavaSoft) .



Data Format: Text string
Data: TEXT DE INTRARE

☐ HMAC — Key Format: Text string — Key:

☑ MD5	56bab12393ebe33b14dd1a415a0c2433
☑ MD4	0b5a31ef10882cb7bfae14f391b65052
☑ SHA1	bf768d86881dba878bfcbd7649a5515f9e4bd81c
☑ SHA256	cfe3baa653a5f5e89f3c5a90720a581414ece4d05786a5d
☑ SHA384	9d9065a8076dc3740e04156a566e9ad2f0b1f2bd43f3483
☑ SHA512	37af1826615326471 3db2ea9b49e4ed9d9c7e1b73d58d7
☑ RIPEMD160	fb3561755fe6a983b7985df83ca25414d035ed13
☑ PANAMA	fec0dd3de7c1d648da38514a79eadfe96abe48fa6952140
☑ TIGER	c8ed54e12c1420e0ecada44a304690c24598b500ca645
☑ MD2	253720da3200c930537611090aa37b71
☑ ADLER32	21090424
☑ CRC32	b1530afc
☑ eDonkey/ eMule	0b5a31ef10882cb7bfae14f391b65052

Attacks on hash funtions

Types of attacks on hash functions;

Example: MD5 attack (description, Wang’s attack).

Attacks on hash functions

1) preimage attack : Given only a message digest y, find any message (or *preimage*) x that generates that digest i.e. $h(x)=y$. Roughly speaking, the hash function must be one-way;

2) second preimages attack: Given one message *x*, find another message *x'* that has the same message digest $h(x)=h(x')$. An attack that finds a second message with the same message digest is a *second pre-image* attack.
-It would be easy to forge new digital signatures from old signatures if the hash function used weren't second preimage resistant.

3) Generation of collisions: Find any two different messages *x* and *x'* with the same message digest $h(x)=h(x')$.
- Collision resistance implies second preimage resistance;
- Collisions, if we could find them, would give signatories a way to repudiate their signatures.

4) Generation of pseudocollisions: *find x si x' such that* $h_1(x)=h_2(x')$ *with initial values IV different for hash functions* h_1 *and* h_2;

5) birthday attack : *attack algorithm independent on hash functions which imples generation of random uniform input variables. Attack is based on birthday paradox;*

6) meet in the middle: *techniques used in the case of iterative usage of hash functions;*

7) Attack with fixed points: *fix point of a compression function f is a pair* (H_{i-1},x_i) *for which* $f(H_{i-1},x_i)=H_i$.

Example: Yuval's attack (Criptologia, 1979)

Is birthday attack type:

- can be applied to every function with m bit input;

- processing time $O(2^{m/2})$, suitable in paralell processing;

- used on digital signatures attack (if we use a has function);

- require storage space but using Floyd's cycle searching algorithm, we can reduce this storage requirements;

INPUT: legitimate message x_1; fraudulent message x_2; hash function h with m - bit input;

OUTPUT: x_1' si x_2' resultsing by minor changes on x_1 respectively x_2 such that $h(x_1)=h(x_2)$;

STEP 1: generate $t=2^{m/2}$ minor modifications x_1' of x_1;

STEP 2: Hash each such modified message, and store the hash-values (grouped with corresponding message) such that they can be subsequently searched on hash-value. Processing time O(t);

STEP 3: Generate minor modifications x_2' of x_2 computing $h(x_2')$ for each and checking for matches with any x_1' above; continue until a match is found. (Each table lookup will require constant time; a match can be expected after about t candidates x_2'.

Floyd's searching cycle algorithm

Floyd's searching algorithm is:

- iterative;

- used for elimination storage requirements;

- algorithm is described in D. Knuth, Semi-numerical algorithms, vol. 2

INPUT: pair (x_1 ,x_2) of integer numbers between 0 and p-1 h iteration function which takes values between 0 and p-1;

OUTPUT: value m for which $x_m=x_{2m}$;

STEP 1: compute iteratively using function h the pair (x_i,x_{2i}) form the precedent pair (x_{i-1},x_{2i-2}) until $x_m=x_{2m}$;

Remarks: a) if the queue of the sequence has length l and the cycle has length t then the first time when $x_m=x_{2m}$ is achieved for t(1+[l/t]);

b) let us note that l<m<l+t thus the running time of this algorithm is O($n^{1/2}$).

Important papers in cracking hash functions

Important progress in the field of cryptanalysis of hash functions was presented in the papers of the Chinese researchers Wang & Yu:

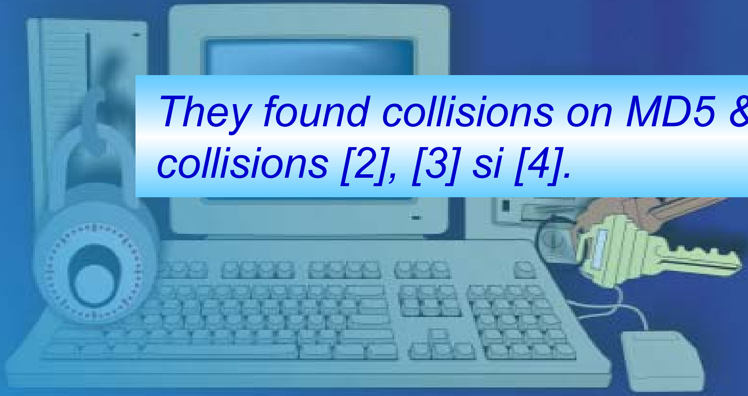
[1] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", rump session, CRYPTO 2004, Cryptology ePrint Archive, Report 2004/199, http://eprint.iacr.org/2004/199.

[2] X. Wang, X. Lai, H. Yu, "How to Break MD5 and Other Hash Functions".

[3] X. Wang, X. Lai, H. Yu, "Collision Search Attacks on SHA1".

and independently by Vlastimil Klima:

[4] "Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications".



They found collisions on MD5 & SHA functions and methods for generating this collisions [2], [3] si [4].

Collisions on MD5/SHA0 and other functions generated by Wang's team

42 years old prof. Xiaoyun Wang and her research group had cracked major U.S. government algorithm used in digital signatures in year 2004. Prof. Wang is a mathematician, expertising in number theory. Her team consists of eight Chinese researchers, out of which six are female mathematicians/computer scientists. Prof. Wang said: "We are used to thinking in the way of mathematics. Once mathematics became our instincts, we view numbers as beautiful music notes. Our research is as interesting and creative as composing music. "

Collisions on MD5 functions generated by Wang & Yu

Presented in the paper X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", rump session, CRYPTO 2004, Cryptology ePrint Archive, Report 2004/199, http://eprint.iacr.org/2004/199.

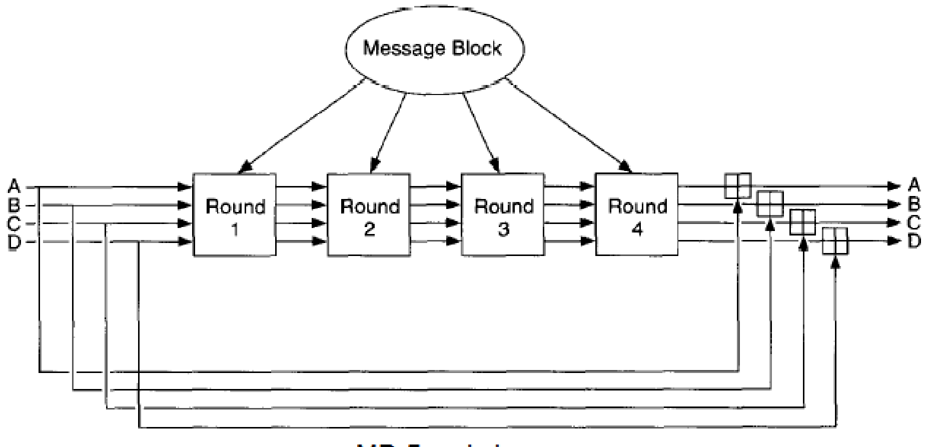
They present collisions on MD5 function… but no given details (at that time) how to generate them.

MD5 inventor

Ronald Linn Rivest *(born 1947, Schenectady, New York) is a cryptographer. He is also the inventor of the symmetric key encryption algorithms RC2, RC4, RC5, and co-inventor of RC6. The "RC" stands for "Rivest Cipher", or alternatively, "Ron's Code". (RC3 was broken at RSA Security during development; similarly, RC1 was never published.) He also authored the MD2, MD4 and MD5 cryptographic hash functions.*

Schematics of MD 5 hash function – main loop

1) After some initial processing, MD5 processes the input text in 512-bit blocks, divided into 16 32-bit sub-blocks. The output of the algorithm is a set of four 32-bit blocks, which concatenate to form a single 128bit hash value.
2) First, the message is padded so that its length is just 64 bits short of being a multiple of 512.
3) Initialised chaining variables A, B, C and D.
4) Now, the main loop of the algorithm begins. This loop continues for as many 5 12 bit blocks as are in the message:



MD 5 main loop

The main loop has four rounds (MD4 had only three rounds), all very similar. Each round uses a different operation 16 times. Each operation performs a non-linear function on three of a, b, c, and d. Then it adds that result to the fourth variable, a sub block of the text and a constant. Then it rotates that result to the right a variable number of bits and adds the result to one of a, b, c, or d. There are four non-linear functions (depending of variables X, Y and Z), F,G,H, and I one used in each operation (a different one for each round). These functions are designed so that if the corresponding bits of X, Y and Z are independent and unbiased, then each bit of the result will also be independent and unbiased. If M_i represents the j-th sub-block of the message (from 0 to 15) and <<<s represents a left circular shift of s bits, the four operations are:

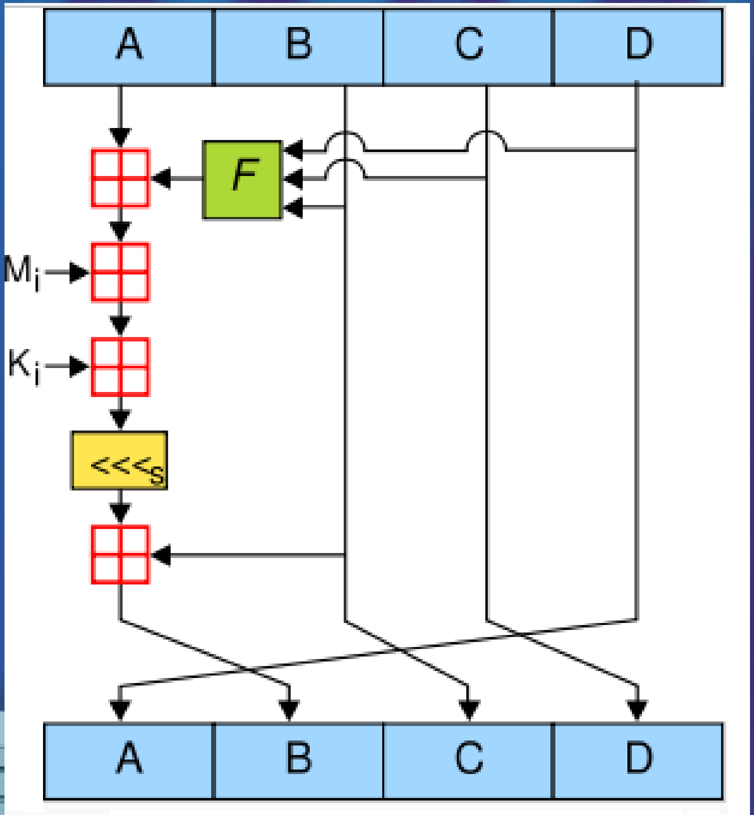
FF(a,b,c,d, M_i,s,t_i) denotes $a = b + ((a + F(b,c,d) + M_i + t_i) <<< s)$

GG(a,b,c,d, M_i,s,t_i) denotes $a = b + ((a + G(b,c,d) + M_i + t_i) <<< s)$

HH a,b,c,d, M_i,s,t_i) denotes $a = b + ((u + H(b,c,d) + M_i + t_i) <<< s)$

II(a,b,c,d, M_i,s,t_i) denotes $a = b + ((a + I(b,c,d) + M_i + t_i) <<< s)$

Schematics of MD 5 hash function – main operations

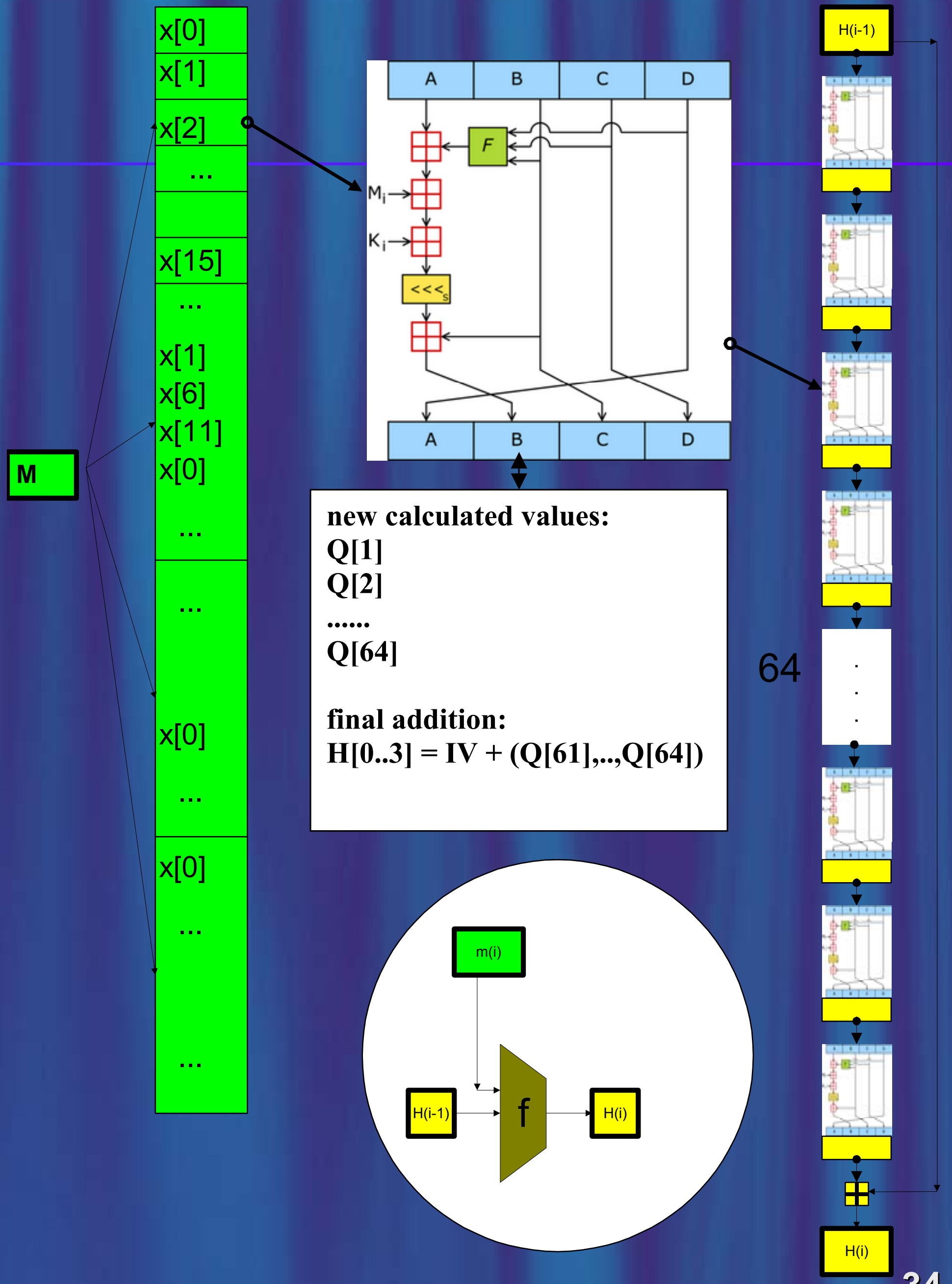


One MD5 operation-MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. *F* is a nonlinear function; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation.

Differential scheme

A change in one word x[i] leads to four changes, what leads to many changes in the next steps…

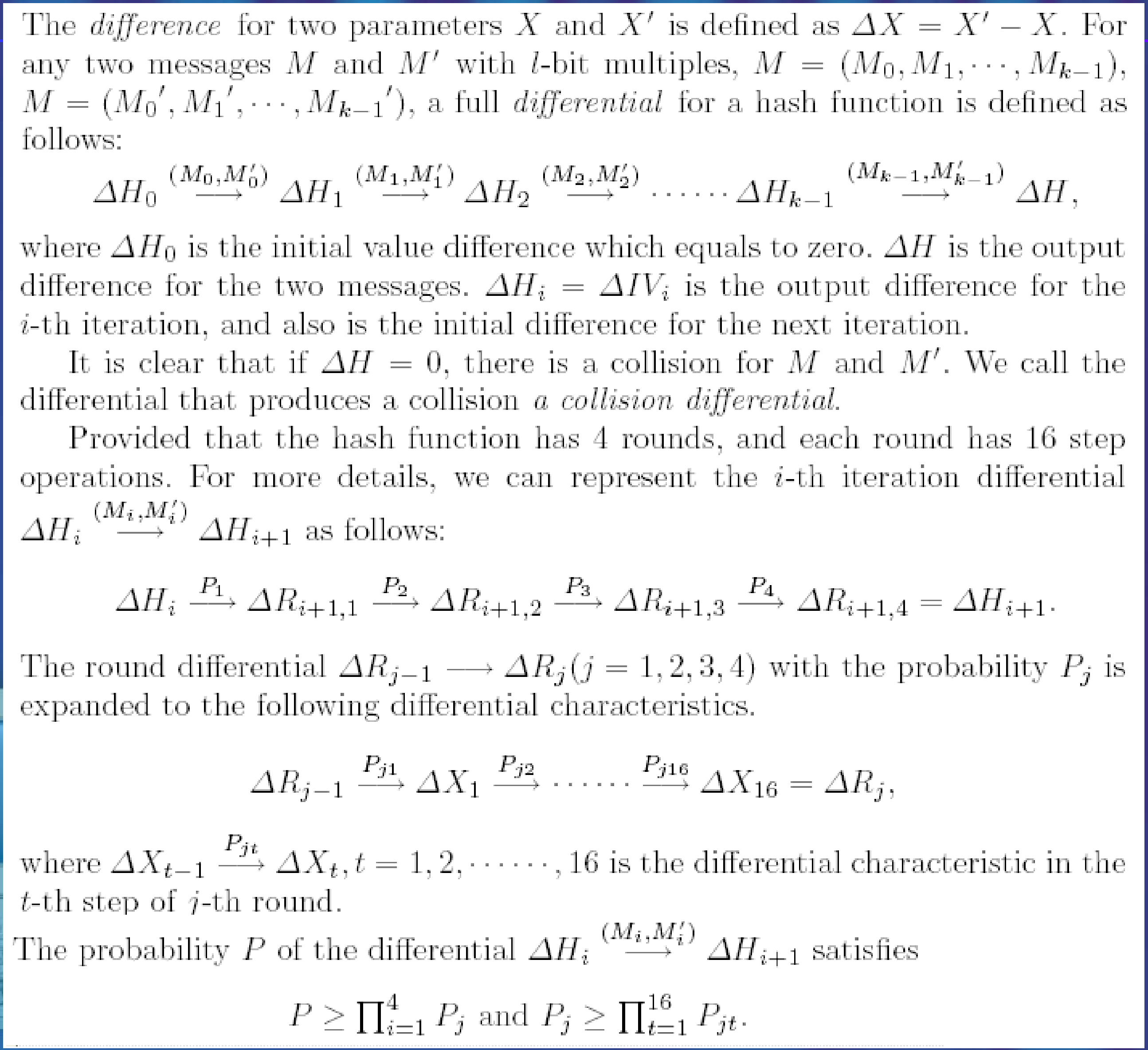
Presetting values of some bits of Q[…] enables to control the changes through the scheme – this are called stationary conditions.



new calculated values:
Q[1]
Q[2]
……
Q[64]

final addition:
H[0..3] = IV + (Q[61],..,Q[64])

Differential attack on hash functions



The *difference* for two parameters X and X' is defined as $\Delta X = X' - X$. For any two messages M and M' with l-bit multiples, $M = (M_0, M_1, \cdots, M_{k-1})$, $M = (M_0', M_1', \cdots, M_{k-1}')$, a full *differential* for a hash function is defined as follows:

$$\Delta H_0 \overset{(M_0, M_0')}{\longrightarrow} \Delta H_1 \overset{(M_1, M_1')}{\longrightarrow} \Delta H_2 \overset{(M_2, M_2')}{\longrightarrow} \cdots\cdots \Delta H_{k-1} \overset{(M_{k-1}, M_{k-1}')}{\longrightarrow} \Delta H,$$

where ΔH_0 is the initial value difference which equals to zero. ΔH is the output difference for the two messages. $\Delta H_i = \Delta IV_i$ is the output difference for the i-th iteration, and also is the initial difference for the next iteration.

It is clear that if $\Delta H = 0$, there is a collision for M and M'. We call the differential that produces a collision *a collision differential*.

Provided that the hash function has 4 rounds, and each round has 16 step operations. For more details, we can represent the i-th iteration differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ as follows:

$$\Delta H_i \xrightarrow{P_1} \Delta R_{i+1,1} \xrightarrow{P_2} \Delta R_{i+1,2} \xrightarrow{P_3} \Delta R_{i+1,3} \xrightarrow{P_4} \Delta R_{i+1,4} = \Delta H_{i+1}.$$

The round differential $\Delta R_{j-1} \longrightarrow \Delta R_j (j = 1, 2, 3, 4)$ with the probability P_j is expanded to the following differential characteristics.

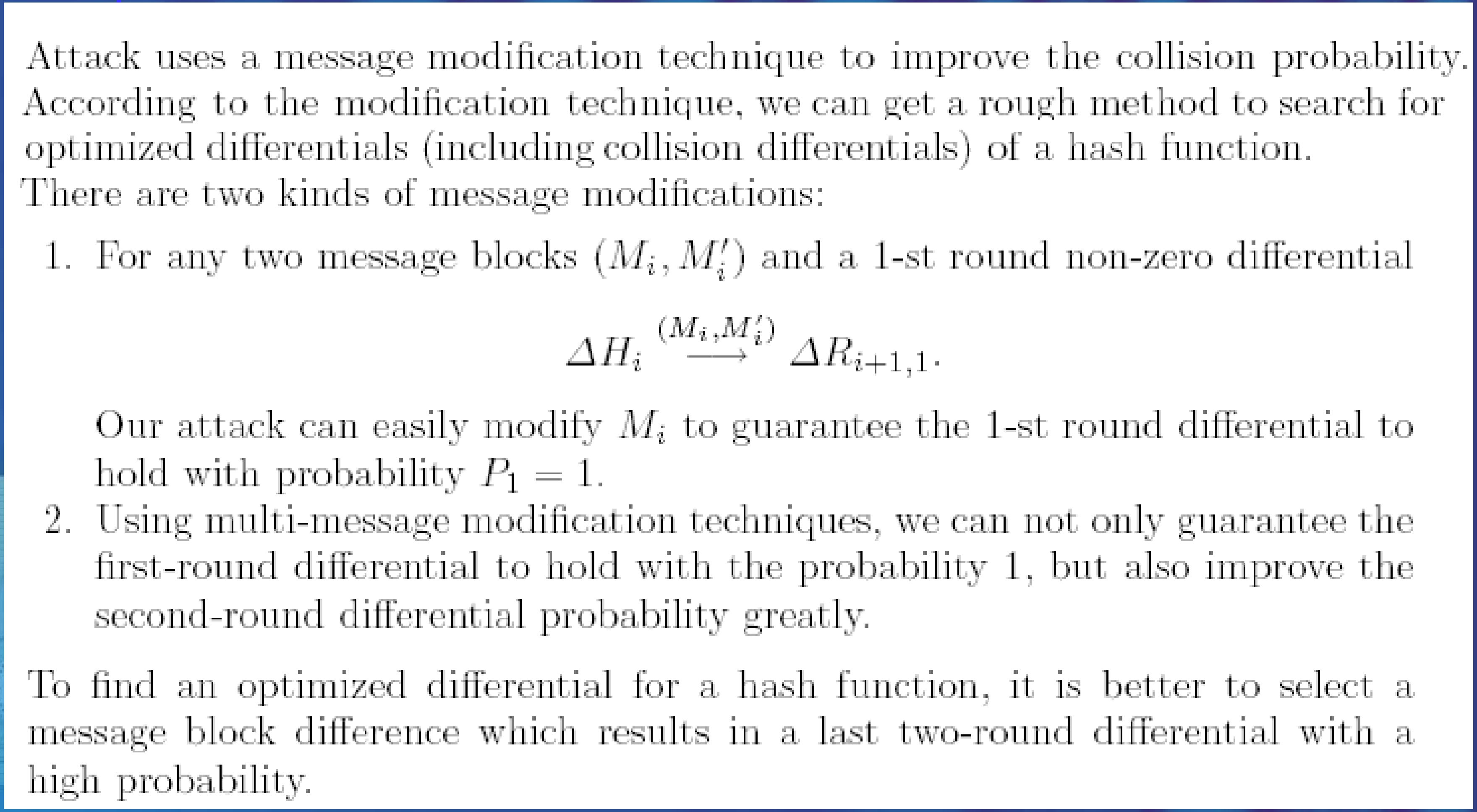
$$\Delta R_{j-1} \xrightarrow{P_{j1}} \Delta X_1 \xrightarrow{P_{j2}} \cdots\cdots \xrightarrow{P_{j16}} \Delta X_{16} = \Delta R_j,$$

where $\Delta X_{t-1} \xrightarrow{P_{jt}} \Delta X_t, t = 1, 2, \cdots\cdots, 16$ is the differential characteristic in the t-th step of j-th round.

The probability P of the differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ satisfies

$$P \geq \prod_{i=1}^{4} P_j \text{ and } P_j \geq \prod_{t=1}^{16} P_{jt}.$$

Optimised collision differentials for hash functions



Attack uses a message modification technique to improve the collision probability. According to the modification technique, we can get a rough method to search for optimized differentials (including collision differentials) of a hash function. There are two kinds of message modifications:

1. For any two message blocks (M_i, M_i') and a 1-st round non-zero differential

$$\Delta H_i \xrightarrow{(M_i,M_i')} \Delta R_{i+1,1}.$$

   Our attack can easily modify M_i to guarantee the 1-st round differential to hold with probability $P_1 = 1$.
2. Using multi-message modification techniques, we can not only guarantee the first-round differential to hold with the probability 1, but also improve the second-round differential probability greatly.

To find an optimized differential for a hash function, it is better to select a message block difference which results in a last two-round differential with a high probability.

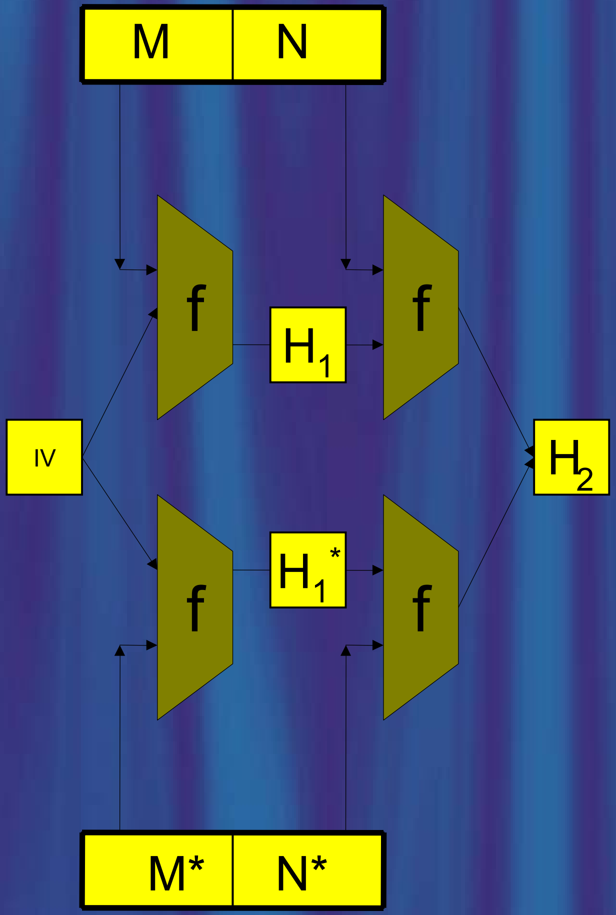
Technical data on MD5 collisions generated by Wang & Yu

M = x[0…15], M = x*[0…15] x*[i] = x[i], except x*[4] = x[4] +2^{31}, x*[11] = x[11] +2^{15} , x*[14] = x[14] +2^{31}*

N = y[0…15], N =y*[0…15], y*[i] = y[i], except y*[4] = y[4] +2^{31}, y*[11] = y[11] -2^{15}, y*[14] = y[14] +2^{31}*

H1 - H1 = (0, +2^{31}+2^{25}, +2^{31}+2^{25}, +2^{31}+2^{25}) , collision generated on one hour on an IBM P690 machine*

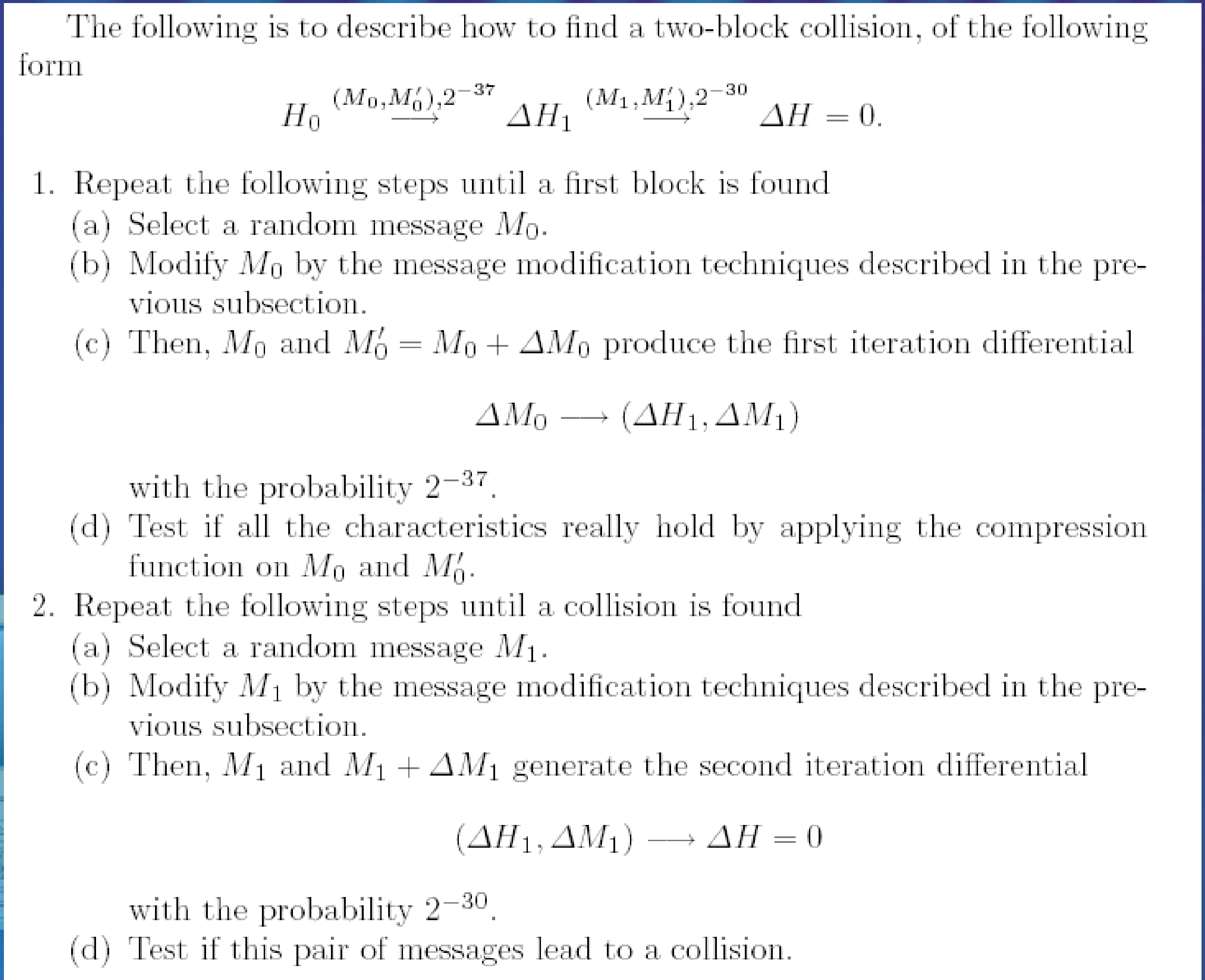
H2 - H2 = (0,0,0,0), second collision generated in about 15s-5min.*



Sufficient conditions for the characteristics to hold

Derive a table of sufficient conditions for the characteristics to hold: first iteration differential and second iteration differential;

The differential attack on MD5



The following is to describe how to find a two-block collision, of the following form

$$H_0 \xrightarrow{(M_0,M_0'),2^{-37}} \Delta H_1 \xrightarrow{(M_1,M_1'),2^{-30}} \Delta H = 0.$$

1. Repeat the following steps until a first block is found
   (a) Select a random message M_0.
   (b) Modify M_0 by the message modification techniques described in the previous subsection.
   (c) Then, M_0 and $M_0' = M_0 + \Delta M_0$ produce the first iteration differential

   $$\Delta M_0 \longrightarrow (\Delta H_1, \Delta M_1)$$

   with the probability 2^{-37}.
   (d) Test if all the characteristics really hold by applying the compression function on M_0 and M_0'.
2. Repeat the following steps until a collision is found
   (a) Select a random message M_1.
   (b) Modify M_1 by the message modification techniques described in the previous subsection.
   (c) Then, M_1 and $M_1 + \Delta M_1$ generate the second iteration differential

   $$(\Delta H_1, \Delta M_1) \longrightarrow \Delta H = 0$$

   with the probability 2^{-30}.
   (d) Test if this pair of messages lead to a collision.

Input values which generates collisions using Wang & Yu algorithm

Message of 1024 bits length composed from blocks M and N of 512 bits length each:

313838DD FC2932C7 C030B717 BAFC1BAE 6673A8D7 9DDCF416 85D70859 99403DB0
0634ADD1 C0736004 9558BD1F 21E10982 CA94C90B 6AAE6E69 CBF61BF1 06B0E615
2E82D48B 16BDF161 CE10BD62 C3C6809D B6745639 FC0E06C7 6573A914 BEF0D753
537B8755 497B92E8 46F559C2 7D7A347A 0511D8B1 98EBEB68 C9CA4559 EB10E037

Message of 1024 bits length composed from blocks M and N* of 512 bits length each:*

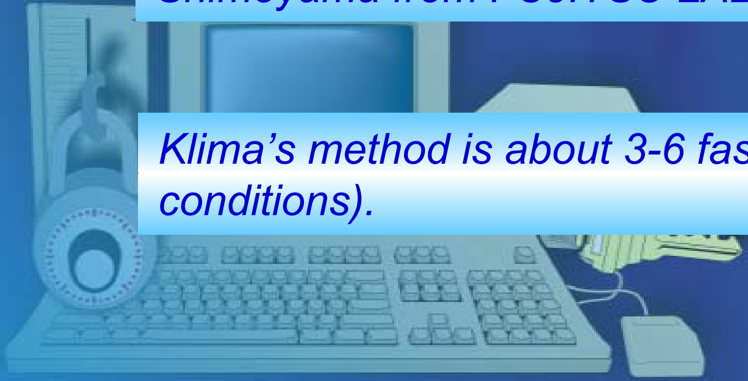
313838DD FC2932C7 C030B717 BAFC1BAE E673A8D7 9DDCF416 85D70859 99403DB0
0634ADD1 C0736004 9558BD1F 21E18982 CA94C90B 6AAE6E69 4BF61BF1 06B0E615
2E82D48B 16BDF161 CE10BD62 C3C6809D 36745639 FC0E06C7 6573A914 BEF0D753
537B8755 497B92E8 46F559C2 7D79B47A 0511D8B1 98EBEB68 49CA4559 EB10E037

Common hash value: 91F0F9DB8D73D703C99C9881BDEED0E9

New collisions generated by Vlastimil Klima

Worked for Czech National Security Authority, chief of R&D group (1981-1992) now independent cryptologist.

Collisions on MD5 hash functions generated by V. Klima



Presented in the paper of Vlastimil Klima “Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications”;

He present a different method of generating collisions on MD5 hash function (the set of Wang’s conditions were not sufficient, corrected by Jun Yajima and Takeshi Shimoyama from FUJITSU LABORATORIES LTD.

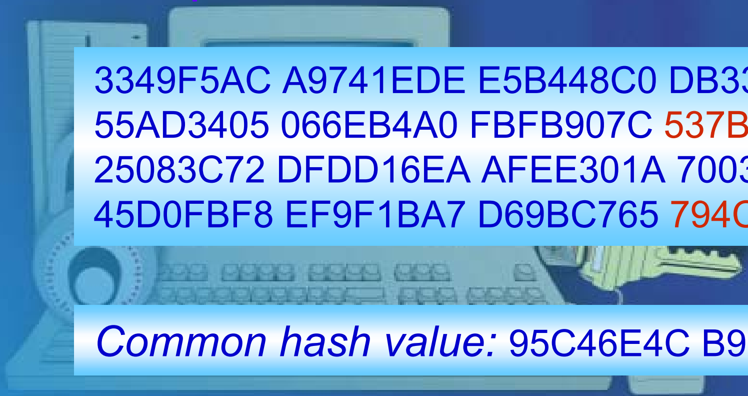
Klima’s method is about 3-6 faster then Wang & Yu techniques (due insufficient conditions).

Input values which generates collisions using Klima's algorithm

Message of 1024 bits length composed from blocks M and N of 512 bits length each:

3349F5AC A9741EDE E5B448C0 DB33CDD6 16E9A926 5C958038 DE975790 BF8D3F87
55AD3405 066EB4A0 FBFB907C 53FB0154 1FADB99D 25D9FA81 A35F5AEC 4AEAC35A
25083C72 DFDD16EA AFEE301A 7003F1E2 D6937735 0CF70D3D 1ADD33E7 4F9BDDDC
45D0FBF8 EF9F1BA7 D69BC765 79CC569D BC8877BB 7A134473 6F1D71F4 4DE89889

Message of 1024 bits length composed from blocks M and N* of 512 bits length each:*



3349F5AC A9741EDE E5B448C0 DB33CDD6 16E9A9A6 5C958038 DE975790 BF8D3F87
55AD3405 066EB4A0 FBFB907C 537B0254 1FADB99D 25D9FA81 A35F5A6C 4AEAC35A
25083C72 DFDD16EA AFEE301A 7003F1E2 D69377B5 0CF70D3D 1ADD33E7 4F9BDDDC
45D0FBF8 EF9F1BA7 D69BC765 794C569D BC8877BB 7A134473 6F1D7174 4DE89889

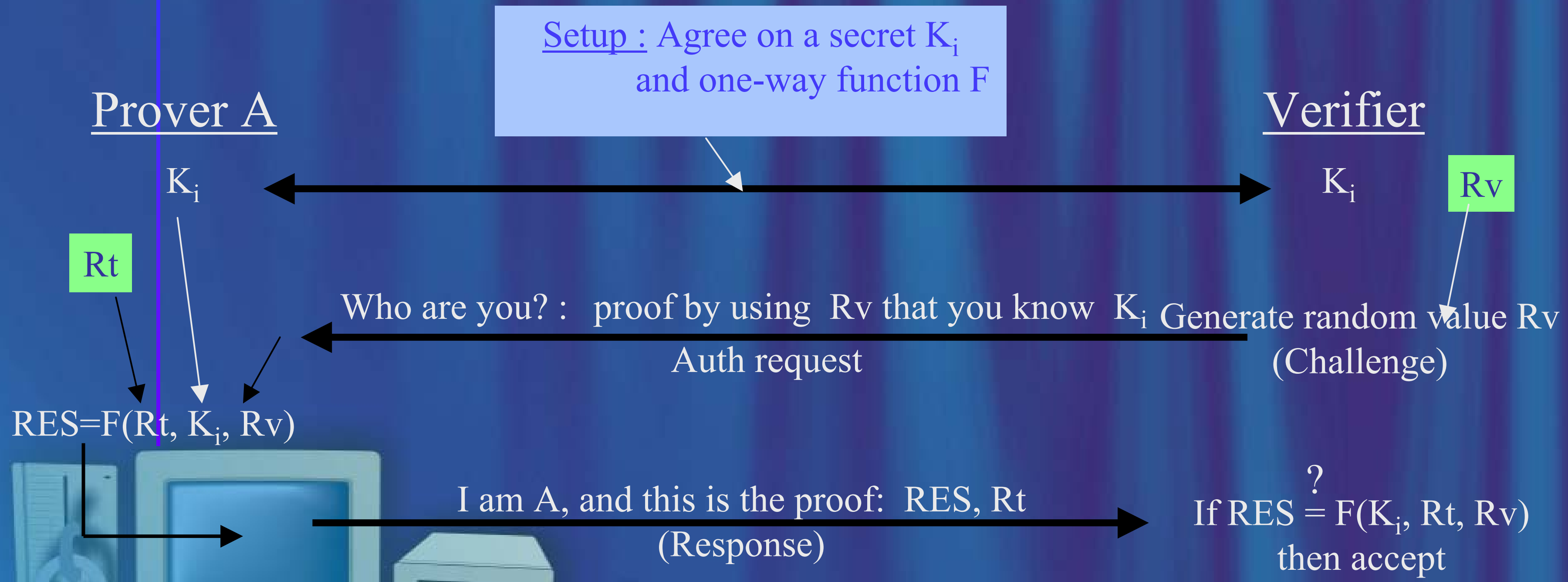
Common hash value: 95C46E4C B928A99A 3A4515AC 33A95B5C

Others applications

1) Hash functions MD4, MD5, SHA-0 si SHA-1, RIPEMD;
   - characteristics, evaluation criteria of hash functions;
   - preimage attack;
   - birthday attack;
   - generation collisions/pseudocollisions;
   - meet in the middle attack.

2) Area of applicability:
   - cryptographic protocols identification/authentication key agreement etc;

Improved Challenge-Response Identification Mechanism


Setup : Agree on a secret K_i
and one-way function F
Prover A
Verifier
K_i
K_i
Rv
Rt
Who are you? : proof by using Rv that you know K_i
Auth request
Generate random value Rv
(Challenge)
RES=F(Rt, K_i, Rv)
I am A, and this is the proof: RES, Rt
(Response)
If RES $\overset{?}{=}$ F(K_i, Rt, Rv)
then accept

References

1. Menenzes A.J., et. al., Handbook of Applied Cryptography, CRC Press, 1997.
2. Schneier B., Applied Cryptography, Adison-Wesley, 1998.
3. Simion E., Preda V., Popescu A., Criptanaliza. Rezultate si Tehnici Matematice, Ed. Universitatii Bucuresti, 2004.
4. Stallings W., Cryptography and Network Security: Principles and Practice, Prentice Hall, Second Edition, 1999.
5. Tilborg, Henk C.A. van, Fundamentals of Cryptology, Kluwer Academic Publisher, Second Edition, 2001.

Questions and Answers