HASH ALGORITHMS: A DESIGN FOR PARALLEL CALCULATIONS

N.G.Bardis
Research Associate
Hellenic Ministry of the Interior, Public Administration and Decentralization
8, Dragatsaniou str., Klathmonos Sq.
10559, Greece

Alex Polymenopoulos.
Hellenic College of National Defense
Athens, Greece

E.G.Bardis
Senior Security Analyst
Hellenic Ministry of the Interior, Public Administration and Decentralization
Athens, Greece

A.P.Markovskyy
Department of Computer Engineering
National Technical University of Ukraine
37, Peremohy, pr. Kiev 252056, KPI 2003, Ukraine

Nikos E. Mastorakis
Military Institutes of University Education, Hellenic Naval Academy,
Terma Hatzikyriakou, 18539, Piraeus, GREECE

Abstract:- It has been shown that the standardized hash-algorithms SHA-1, RIPEMD-160, MD-5 and others of that type have substantial performance restrictions due to their sequential structure. Modified variants of hash-algorithms are suggested which make it possible to independently calculate partial hash-signatures while maintaining the irreversibility level of the standard algorithms. These variants give new opportunities for wide parallel calculation of hash-signatures. In fact, applying the suggested modified algorithms remove the performance restrictions affiliated with the calculation an information message's hash-signatures. In practice, the hash-algorithms suggested may be used for integrity and authentication of information messages in computer network.

Key words: Cryptographic hash functions, instruction-level parallelism, multiple-issue architectures, parallel calculation arrangement.

1 Introduction

An important component for having modern and effective network technologies is ensuring the integrity and authenticity of the information transmitted. In practice, to solve the problem, special standardized protocols and digital signature procedures are applied, such as DSS (Digital Signature Standard).

In a DSS, the dominant role belongs to algorithms that form hash-signatures, i.e. codes of a small fixed length that accumulate the maximum amount of information contained in the message.

The role of hash-signatures and is not just restricted to systems providing information integrity and authentication in computer networks. Since early 60's they have been widely implemented for associative access in databases and information systems.

During the last decade, the hash-algorithms effective application field extended far at the cost of information security systems. They play an important role in software protection from computer viruses, for the prevention of unauthorized program copying, for designing pseudorandom sequence generators, and for authentication programming products and documents.

In practice, hash-signatures must provide:

- Hash-signature code dependence on each bit of an information message;

- One way transformation, i.e. practical impossibility to find the input message code by the given hash-signature code;
- Possibility to form an information message hash-signature of an arbitrary length;
- Fast calculation of hash-signature using either personal computers or special processors.

To be able to create an information message's hash-signature of any length, modern hash-algorithms, such as MD-5, SHA-1, RIPEMD-160, ГОСТ Р.34.11-94, MDC-4 [4], are developed on a block basis. This principle is based on the procedure of dividing the message text into blocks of fixed length which are then processed with a hash-algorithm sequentially.
Most hash-algorithms have a sequential structure, i.e. each information block that follows is processed taking in account the result of the previous block's processing. This creates major restrictions when trying to achieve high performance hash-signature formation.

The performance requirement of hash-signature formation on personal computers with clock frequency 500 MHz is about 3-5 Mbytes/s [3] and causes significant delays in information flow processing. What is required is development of new methods and means of hash-signature formation in order to increase performance. Because the speed of data communications systems is growing at a faster rate than processor speed itself, the importance placed on the efficiency of the formation of a hash-signature' increases even more.

On the other hand, it should be kept in mind that processor speed growth results in a necessary complication of hash-algorithms in order to ensure their irreversibility. This means an increase in calculation complexity and, correspondingly, computation time. In fact, the earlier hash-algorithms MD-4 and MD-5 are three times faster than the modern SHA-1 and RIPEMD-160 algorithms that have been developed more for irreversibility [3]. Thus, the important problem of the calculation of the information message's hash-signature cannot be solved only by getting higher speed of processing.

The only possible way to drastically increase hash-signature formation speed is to arrange them so that their parallel calculation is possible. This, in its turn, requires development of new approaches to independent formation of partial hash-signatures of several blocks and their merging without decreasing the hash-algorithm's level of irreversibility.

2 Algorithm SHA-1 and the problem of its parallel calculation

Nowadays, the most widespread hash-algorithm is SHA-1 (Secure Hash Algorithm), created at the beginning of 90's as an improvement of the earlier MD-4 and MD-5 [4] hash-algorithms. Later on, its European version RIPEMD-160 was developed. All these algorithms are structurally related, differing only in the length of the input information block and hash-signature as well as in the functions of their hash-transformations. Therefore, further presentation on SHA-1 seems to be justified taking into consideration that all the results obtained may be applied to any of the hash-algorithms mentioned above.

Algorithm SHA-1 provides for preliminary modification of an information message M. The executive message is the concatenation of three parties: the input message M with L-bit length, fragment P=100...0 and a 64-bit code which contained the code L. The length of the P fragment should be such that the total length of the executive message will be a multiple of 512. After that modification, the message is divided into n of 512-bits blocks - $M_1,M_2,\dots,M_n$ and for each jth block M_j, $j=1,\dots,n$ – a hash-transformation $H(\lambda,M_j)$ is performed that results in a 160-bit partial hash-signature h_j. Here λ is the initiating 160-bit code, which is a fixed code α at the processing of the first block and is formed as an XOR of all the previous partial hash-signatures while all the next blocks are being processed.

The total hash signature HS(M), of information message M, is formed as the XOR of all the partial hash-signatures.

$$h_1 = H(a, M_1)$$
$$h_j = H(\bigoplus_{k=1}^{j-1} h_k, M_j), j = 2,...,n \qquad (1)$$
$$HS(M) = \bigoplus_{j=1}^{n} h_j$$

The structure described provides that each information message bit has equal influence on the hash-signature HS(M) code. At the same time, the impossibility to find a collision message $M_k \neq M$: $HS(M_k)=HS(M)$ by block manipulations (by permutation of the input message blocks), is assured by the fact that the XOR of the partial hash-signatures of all previous blocks is used as the initiating code of the next block. The most significant part of the hash-algorithm is the hash transformation structure H(λ,M). It gives the hash-algorithm's requirements as stated above and consists of 80 cycles of the recursive calculation of the 160-bit code λ presented as the 5 of 32-bit variables A, B, C, D, E with the participation of the information block M. The 512 bit information block is expanded from the 16 of 32-bit sub-blocks $W_0,\ldots,W_{15}$ to 80 of 32-bit sub-blocks $W_0,\ldots,W_{79}$ according to the following formula:

$$W_j = ROL_1(W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16}),$$
$$\forall j = 16,...,79 \qquad (2)$$

where ROL_1 is the operation of cyclic shift to the left of the 32-bit code by one bit. The 80 cycles that compose the process of hash-transformation are divided into 4 groups by 20 cycles, each one of them uses the corresponding Boolean function $f_1,\ldots,f_4$ that is based on variables B, C and D:

$$f_1(B,C,D) = B \cdot C \vee \overline{B} \cdot D = B \cdot C \oplus B \cdot D \oplus D$$
$$f_2(B,C,D) = B \oplus C \oplus D$$
$$f_3(B,C,D) = B \cdot C \vee B \cdot D \vee C \cdot D = B \cdot C \oplus B \cdot D \oplus C \cdot D \qquad (3)$$
$$f_4(B,C,D) = B \oplus C \oplus D$$

In each of the 80 cycles, algorithm SHA-1 calculates a 32-bit variable T, as the sum modulo 2^{32} of:

- the code A shifted circularly by 5 bits,
- the value result of the calculation of function f,
- the corresponding to a number cycle - 32-bit sub-block W,
- the variable E,

$$T = ROL_5(A_i) + f_q(B,C,D) + K_q + W_i + E_i,$$
$$i = 1,...,80, q = (i+1) \bmod 20 \qquad (4)$$

Execution of an i^th cycle (i=2,...,80) is finished with recursive re-determination of variables A_i, B_i, C_i, D_i and E_i:

$$E_i = D_{i-1}, D_i = C_{i-1}$$
$$C_i = ROL_{30}(B_{i-1}) \qquad (5)$$
$$B_i = A_{i-1}$$
$$A_i = T$$

The partial hash-signature $h(M_j)$ of a j^th block of the information message consists of the 5 variables A_{80}, B_{80}, C_{80}, D_{80}, E_{80} after the execution of all the 80 cycles. The hash-transformation information message M is the 160-bit hash-signature HS(M) code that has the property of the maximum total and differential entropy regarding to any of the information message bits.

The property of the total entropy maximum consists of the equal chance for all the hash-signature bits to be equal either to zero or to one, at any codes of the message.

The property of the differential entropy maximum (the property of the avalanche effect) is that, by altering the bits of any message, there is a 50% chance that each hash-signature bit will change its value. In fact, SHA-transformation is the system of 160 nonlinear balanced Boolean functions, whose set of variables is created by all the bits of the input message. The differential of these functions with respect to any subset of variables is also a balanced Boolean function.

The properties of SHA–transformation stated above, give it a practical irreversibility when attempting to find a collision message by means of linear and differential analysis. Because the length of the input block equals 512, while the length of a partial hash-signature equals only to 160, there exist $2^{512-160}=2^{342}$ input block variations that have a given code of the hash-signature. However, to find at least one of them, 2^{160} input block variations are to be searched. This exceeds the resources of contemporary computers many times. Unfortunately, the SHA-1 weak point is the sequential calculation of partial hash-signatures. Although the SHA-1 structure is intended to employ Intel 32-bit multipurpose microprocessors, the complexity of the algorithm for this processor is rather high because, to calculate a partial hash-signature, about 1000 commands need to be executed.

Paralleling of hash-signature calculation may be implemented both as a collateral (parallel) calculation of a partial hash-signature and as collateral (parallel) formation of several partial hash-signatures. It should be noticed that the former does not imply an alteration of the algorithm structure while the latter requires modification of the algorithms themselves.

The problem of collateral execution of a single hash-signature calculation was studied in [1]. It was partly shown that the critical path's length for the SHA-1 algorithm is made by 160 operations and the maximum collating sequence level is 7. Employing a 7-processor architecture, calculation of a partial hash-signature may be theoretically performed in the time it takes to execute 160 commands. This makes it possible to increase hash-signature formation speed by about 6.25 times. In this case the number of the registers required is 26, and the processor efficiency is 0.85.

Thus, according to the results of investigation [1], by collating the operations and employing a special-purpose multiprocessing computer the speed of one partial hash-signature calculation cannot be increased more than by 6.25 times. This amount is not enough for practical problems. Hence, in practice the only possible way to greatly increase the hash-signature formation speed is to arrange a parallel calculation of partial hash-signatures. This requires modification of SHA-like algorithms while maintaining maximum security level against manipulation of the blocks of the input message.

3 Requirements to arrange the partial hash-signatures parallel formation

The main reason for arranging hash-signature sequel formation is that it makes it impossible to find collision messages through block manipulations, i.e. by permutation of the initial information message blocks.

On the other hand, sequential formation of partial hash-signatures does not allow multiple searching for a collision message. That means searching input block codes by way of calculation of the corresponding partial hash-signature till the latter coincides with one of *n-1* partial hash-signatures of the original information message is impossible.

It is obvious that at in a single attempt to search a collision block, the success probability is 2^{-160}. Let *t* attempts to find a collision block for a concrete information block be carried out. The probability of success in this case is $1-(1-2^{-160})^t$. If the same number of attempts is carried out taking into account the possibility of coincidence between the formed partial hash-signature and any of *(n-1)* original blocks, the probability of success is much higher at $1-(1-2^{-160})^{t\cdot(n-1)}$.

Thus, the problem consists of arranging the calculation of partial hash-signatures and the total hash-signature in such a way so that, on one hand, allows for independent calculation of partial hash-signatures and on the other hand, makes it possible to

- maintain the property of the total and differential entropy maximum
- prevent the finding of a collision information message by means of block manipulations and multiple searching.

Also, complex calculations should not be used because they require considerable computational resources.

The problem may be solved by modifying the function that creates the total hash-signature from the partial hash-signatures, or by modifying the function that creates the partial hash-signatures.

The modification of the function that creates the partial hash–signature is necessary when the total hash signature is formed as an XOR of partial hash-signatures:

$$HS(M) = \bigoplus_{j=1}^{n} h_j = \bigoplus_{j=1}^{n} H(M_j) \qquad (6)$$

Such a function retains the properties of the total and differential entropy maximum but it does not, by itself, assure the impossibility of block manipulations.

The modification of the function that forms the partial hash-signatures is to be carried out in such a way that at permutation of a jth block M_j, the corresponding partial hash-signature $h_j=H(M_j)$ code should change. The idea may be implemented in the following ways:

- Altering the information block code at the hash-transform input, i.e. feeding its input with the modified code $M^m_j=\varphi(M_j,m_j)$, instead of code M_j, where m_j is the modifying code for the jth block;

- Using the modifying code m_j as the initiating code λ at processing the j-th block;
- Transforming the partial hash-signatures codes after their obtaining $h^m_j=\phi(h_j,m_j)$.

The presented ways of modifying the function that forms the partial hash-signatures may be used either independently or simultaneously. The modifying code m_j is determined by the number of the jth block in the message and may be formed with use of some additional information.

4 Arrangement of independent calculation of partial hash-signatures by modifying their formation procedure

The simplest form for the arrangement of the independent calculation of partial hash-signatures is modifying the input block $M^m_j=\varphi(M_j,m_j)$ directly before calculating its partial hash-signature $M^m_j=\varphi(M_j,m_j)$. In this case it should be kept in mind that, on one hand, the functional transformation $\varphi(M_j,m_j)$ has to ensure the change of one bit $M^m j$ at the change of any single bit M_j, and on the other hand, that the transformations $\varphi(M_j,m_j)$ as such are to eliminate the finding $M'_j \neq M_j$: $\varphi(M'_j,m_j)= \varphi(M_j,m_j)$. This may be achieved if the function $\varphi(M_j,m_j)$ is either single-valued or a one–way function (i.e. computation resource expenditure on finding the collision block $M'_j\neq M_j$: $\varphi(M'_j,m_j) =\varphi(M_j,m_j)$ should be beyond the computer implementation). Application of one–way functions requires significant computation resources. Therefore, utilization of modified information blocks of single-valued functions as function $\varphi(M_j,m_j)$ has better prospects. For such functions the condition $L(M^m_j)=L(M_j)+L(m_j)$ is held. Here $L(M_j)$ is the information block length, $L(m_j)$ is the modifying code length and $L(M^m_j)$ is the hash-transformation input code length. The simplest function that satisfies the stated demands is concatenation of the information block and its ordinal number in the message ($m_j=j$):

$$M^m_j = \varphi(M_j, m_j) = M_j \parallel j \qquad (7)$$

Decreasing the length of an information (block assigned to) a partial hash-signature by value $L(m_j)$ of the modifying code m_j results in the growth of the hash-signature formation speed by a value of $L(M^m_j)/L(M_j)= L(M^m_j)-L(m_j))$ on the same processor. In particular, for SHA-1 and MD-5 the $L(M^m_j)=512$, correspondingly the speed of hash-signature formation decreases by $512/(512-L(m_j))$ times. If the lower bound of the modifying code length equals $\mathrm{Log}_2 n$, the lower bound of hash-signature calculation speed decreases by $512/(512-\mathrm{Log}_2(n))$ times. Specifically, if $L(m_j)=32$, it is possible to form hash-signatures of messages with a length of up to 240 Gbyte. (In this case), the block length decreases from 512 to 480 bits and the time for the total hash-signature formation increases, upon exploiting k processors, by k/1.076 times.

Further growth of operation speed may be achieved by modifying code m_j as the initial code λ at processing the jth block, which, in this case, is calculated as the XOR of the constant α and the block ordinal number:

$$s_j = m_j = j \oplus a \qquad (8)$$

The advantage of being able to modify partial hash-signatures after their formation and before the total hash-signature calculation is that falsification of partial hash-signatures after their calculation before the total hash-signature calculation is practically impossible at the input of modifying procedures. On the other hand, special requirements to keep statistic and differential characteristics of the total hash-signature are put forward to the procedures. Modification of the partial hash-signature codes h_j of each of the jth of the n blocks M_j of the information message after their formation as the result of hash-transformation $h_j=H(M_j)$ is performed by means of modifying function $h^m_j=\phi(h_j,m_j)$. If the set of non-modified partial hash-signatures of all the n blocks of the information message is designated as $\vartheta=\{h_1,h_2,\ldots,h_n\}$, then, to exclude the possibility of finding a colliding message by block permutation (resistance to attacks), it is necessary that the modifying function fulfills the following condition:

$$\phi(h_j, m_k) \neq \phi(h_k, m_j),$$
$$\forall h_j, h_k \in \vartheta, h_j \neq h_k \qquad (9)$$

Function $\phi(h_j,m_j)$ is to conserve the statistic properties of hash-signatures, i.e. by altering any bit of unmodified hash-signature h_j, the same number of bits of modified hash-signature h^m_j should change at any value of modifying code m_j:

$$HW(h_j \oplus (h_j \oplus \alpha)) =$$
$$= HW(\phi(h_j, m_j) \oplus \phi(h_j \oplus \alpha, m_j)) \qquad (10)$$

where HW(X) is Hamming weight of binary code X, equal to the number of ones in the code X, and α is an arbitrary binary vector whose length coincides with h_j.

The last requirement put forward is satisfied by the functions of initial partial hash-signature bit permutation. In this case, the permutation functions should be simple and effective in their implementation as well able to satisfy condition (9) at the given maximal number of information blocks in the message.

Specifically, (it should be) a permutation function that satisfies the stated demands and may be applied in practice to modify the output partial hash-signatures is function P(X,r) of the fragmental circular shift.

The operation of fragmental circular shift of the q-bit code X supposes its division into m of k-bit fragments, where k=m/q: $X=D_1||D_2||\ldots||D_m$. In turn, each fragment D_j, j=1,…,m comprises k of binary bits $D_j=d_{j1},d_{j2},\ldots,d_jk$, $d\in\{0,1\}$. Each jth of m fragments D_j of code X is put into correspondents with analogous binary bit of m-bit code $U=\{u_1,u_2,\ldots,u_m\}$, $u_j\in\{0,1\}$. The bit permutation P(X,r) supposes sequential execution of r circular shifts of fragments X, in so doing, in each tth shift (t=1,…,r) only those fragments X take part to which single values of code U corresponds. The numerical value of the code equals $U(t)= (t-1) \bmod (2^m-1)+1$. Thus, the basic operation for the functional transformation P(X,r) is the logical shift by one bit to the left of k-bit fragment D_j for which $u_j=1$ with filling the deleted bit by the bit of c_j transfer into jth fragment : $D_j<<c_j$. Denoting Θ as the set of fragment numbers that participate in the shift, i.e. $\Theta=\cup j:u_j=1$, the process of single bit permutation P(h,1) may be presented formally as:

$$\forall j \in \Theta: D_j = D_j << c_j,$$
$$\forall t \in \Theta,\ \ t \neq \min\Theta: c_t = d_{l1},\ \ l = \min_{l>t}\Theta,$$
$$c_q = \max\Theta,\ \ q = \min\Theta \qquad (11)$$
$$U < 2^m - 1: U = U + 1, U = 2^m - 1: U = 1$$

From a processing aspect, the Forre method does not correspond to the requirements imposed above for the design of balanced SAC-functions.

For example, let X=1100 1010 0110 1001 i.e. q=16; code h is divided into 4 fragments by 4 bits, correspondingly m=4 and k=4. The values of the fragments are equal to: D_1=0100, D_2=1010, D_3=0110, D_4=1001. Let U=6=(1010) in the current shift, correspondingly, set $\Theta=\{1,3\}$. This implies that the first and third fragments are shifted. In so doing, the transfer is carried out to the first fragment from the output of overflow at the third fragment shift $c_1=d_{31}=0$, and the transfer into the third fragment is carried out from the output of overflow of the first fragment $c_3=d_{11}=1$. Code h has the form of X=1000 1010 1101 1001 after shifts of D_1 and D_3. The value of U=6 is less than 2^4-1=15, correspondingly the next value of U is equal to U+1 = 7=(0111).

If the modified jth partial hash-signature of the fragmental shift function $P(h_j,m_j)$ described above is taken as function $\phi(h_j,m_j)$, it is important to assume a high speed of partial hash-signature modification. Apparently, realization of r fragmental shifts P(X,r) requires time in proportion to r. However, because the function of the fragmental shift permutation has the property: $P(h\oplus t,y)=P(h,y)\oplus P(t,y)$, the modification of partial hash-signatures may be executed simultaneously. Specifically, at each step, modification of all the partial hash-signatures calculated earlier may be executed collaterally with use of difference n-j as the modifying code m_j for the j-th partial hash-signature h_j, i.e. at m_j=n-j and, correspondingly, $\phi(h_j,m_j)=P(h_j,n-j)$. At each step, the modification is determined by the time of one fragmental shift. In this case the total hash-signature code is being formed recursively according to the following expression:

$$S_j = P(S_{j-1}, 1) \oplus h_j, S_0 = h_0 \qquad (12)$$

Concerning the 160-bit code of the SHA-1 hash-signature (q=160), it may be divided into 5(five) 32-bit fragments, 10(ten) 16-bit fragments and 20(twenty) 8-bit fragments. In principle, other divisions may also be considered but the ones mentioned above provide highest processing simplicity in implementing the bit permutation function. Considering that the processor efficiency for calculation of a partial hash-signature in SHA-1 takes about 1000 computer commands, modification of partial hash-signatures, even with use of the 20-fragment division, doesn't require more than 20 shift commands, i.e. slowing down the total hash-signature calculation makes not more than 2%.

Periods of permutation iteration at the fragmental circular shift for dividing a 160-bit hash-signature into 5, 10 and 20 fragments make correspondingly $2^{31}\cdot 120=3720$, $2^{1023}\cdot 60=61380$ and $2^{1048575}\cdot 495\approx 5\cdot 10^{8}$. In such a way, upon division into 5 fragments, the application of the permutation function $P(h_j,m_k)$ as partial hash-signatures modification function $\phi(h_j,m_k)$ ensures condition (9) to be held at the message length up to 232.5 Kbytes. With division into 10 and 20 fragments, condition (9) is held at the message length equal correspondingly to 3.74 Mbytes and 31.68 Gbytes.

5 Arrangement of partial hash-signatures independent calculation at by modifying the total hash-signature formation function

Applying asymmetric functions $\psi(h_1,h_2,\dots,h_n)$ to form the total hash-signature from the codes of partial hash-signatures is another way to prevent breaks of SHA-1 hash-signatures through block manipulations and arrangement of independent calculation of partial hash-signatures.

$$HS(M) = \psi(h_1,h_2,...,h_n) \qquad (13)$$

The asymmetry condition is that for any pair of partial hash-signatures, $h_k \neq h_t$, $t,k \in \{1,\dots,n\}$ the condition $\psi(h_1,\dots,h_{k-1},h_k,h_{k+1},\dots,h_{t-1},h_t,h_{t+1},\dots,h_n) \neq \psi(h_1,\dots,h_{k-1}, h_t, h_{k+1},\dots, h_{t-1}, h_k,h_{t+1},\dots,h_n)$ is held, i.e. function ψ alternates with a change of components succession on which it is determined. Besides, function ψ is to conserve the statistic properties of the whole hash-signature. Namely, upon alteration of a half of the partial hash-signature bits a half bit of the total hash-signature should change. This property may be replaced with a simpler one: under the condition that the other partial hash-signatures are left without change, upon alternating an arbitrary number of bits of a partial hash-signature the corresponding number of the total hash-signature bits should alternate.

Because n the number of block changes which cannot be known in advance, function ψ should be simply implemented through the recursive procedure.

One of the possible variations in implementing partial the hash-signatures collation function is the utilization of SHA-transformation. Existence of the 512-bit input makes it possible to collate 3 partial hash-signatures. The ordinal number of the jth triplet of the information message blocks being collated may feed, as the modifying code, the 32 inputs left unused. Correspondingly, the total hash-signature is calculated according to the formula:

$$HS(M) = \bigoplus_{j=1}^{n/3} H(h_{3\cdot j-2} \,\|\, h_{3\cdot j-1} \,\|\, h_{3\cdot j} \,\|\, j) =$$
$$= \bigoplus_{j=1}^{n/3} H(H(M_{3\cdot j-2}) \,\|\, H(M_{3\cdot j-1}) \,\|\, H(M_{3\cdot j}) \,\|\, j) \quad (14)$$

Applying the SHA-transform for partial hash-signatures collation provides for asymmetry and conservation of the total hash-signature statistic properties. It reliably protects the total hash-signature from breaks by means of block manipulations. On the other hand, applying a rather time-consuming SHA–algorithm for partial hash-signatures collation distinctly slows down the total hash-signature formation by 30%. In practice, it may be justified using some specialized high efficiency hardware for SHA–transform.

The other variant (of the function) adequate for practical application is employing nonlinear functional transformers with memory. The function collating partial hash-signatures remains asymmetric and conserves the statistic properties of SHA-1 hash-signatures. The memory of the transformers is being utilized through q 160-bit vectors $W_1,\dots,W_q$, which are initially set to zero. Collation of the partial hash-signatures is performed by alteration of vectors $W_1,\dots,W_q$ accordingly to the formula:

$$v = W_1 \oplus h_j \oplus \delta(W_2,\ldots,W_q),$$
$$\forall j = 1,\ldots,q-1: W_j = W_{j+1}, W_q = v \qquad (15)$$

where $\delta(W_2,\ldots,W_q)$ is a nonlinear balanced Boolean SAC-function of q–1 variables.

The total hash-signature is formed after implementing a collating operation over the last partial hash-signature h_n in the following way:

$$HS(M) = \delta(W_2,\ldots,W_q) \qquad (16)$$

The asymmetry property is provided for by nonlinear transformations and conservation of SHA hash-signature statistic property is determined by application of SAC-function δ. After collating the jth partial hash-signature h_j each bit of vector W_q alternates with the change of the analogous bit h_j. Because function $\delta(W_1,\ldots,W_q)$ is a SAC one, and W_q is completely correlated with h_j, and the probability for any bit W_q to alternate with the corresponding bit h_j is 0.5 when the next (j+1)th hash-signature h_{j+1} is being collated. Similar reasoning may reveal that after collating the (h+q)th partial hash-signature, any bit of all the q vectors $W_1,\ldots,W_q$ changes with a probability of 0.5 when alternating the corresponding bit h_j. Because the total hash-signature HS(M) is formed as a SAC-function of $W_2,\ldots,W_q$, half of the total hash-signature bits change upon alteration of any number of bits of any of the partial hash-signatures.

The studies have shown that at q=8, less than 100 processor commands are spent on the operations of total hash-signature formation at exploiting the nonlinear function transformers with memory. Hence, increasing the time expenditures on formation of the total SHA-1 hash-signature resistant to breaks by block manipulations at independent calculation of partial hash-signatures makes not more than 1% for single processor.

6 Annotation

Research on widespread hash-algorithm SHA-1, RIPEMD-160, MD-5 and others of the type whose resources are modified with the intent to arrange their parallel implementation on multiprocessor complexes or high efficiency specialized hardware systems.

7 Conclusions

Through investigation, it has been shown that standardized algorithms SHA-1, RIPEMD-160, MD-5 and others of the type have substantial restrictions on performance due to their sequential structure. The sequential pattern of hash-signature formation according to known schemes is determined by the fact that, in this case, finding a collision message by block manipulation and multiple searches becomes impossible. Variants of modified hash-algorithms proposed are resistant to block manipulations and multiple search. At the same time they enable the calculation of independent partial hash-signatures. This gives the opportunity for wide parallel calculation of hash-signatures and, in practice, solves the problem of performance restrictions on calculation of an information message's hash-signature. This is of great importance for high-speed telecommunication systems, especially for mobile communication systems and image transmission.

Two groups of methods are suggested. One is connected with the modification of the procedure that obtains a partial hash-signatures and the other relates to modification of the function that obtains the total hash-signature. The main advantage of the former method compared to the latter is the impossibility to reproduce a collision message both by means of block manipulation and by multiple searches. The methods developed may be applied in combination or independently.

References

1. Bosslaers A., Govaerts R., Vandewalle J. SHA: A Design for Parallel Architectures. EUROCRYPTO'97. LNCS 1233, pp.348-363. 1997.
2. Chabaud F., Joux A. Differential Collision in SHA-0. CRYPTO'98. LNCS 1462, pp.56-71. 1998.
3. Dobbertin H., Bosselaer A., Preneel B. "RIPEMD-160: A strengthened Version of RIPEMD". Fast Software Encryption, LNCS-1039, pp.71-82, 1996.
4. Schneier B. "Applied Cryptography. Protocols. Algorithms and Source codes in C. Ed.John Wiley, 1996 - 758 pp.

5. Secure Hash Standard. Federal Information Processing Standard Publication #180,U.S. Department of Commerce, National Institute of Standard and Technology,1993.

6. Secure Hash Standard. Federal Information Processing Standard Publication #180-1, U.S. Department of Commerce, National Institute of Standard and Technology, 1995.