V. A. Luzhetsky, Dr. Sc. (Eng), Prof.; Yu. V. Baryshev

METHODS AND TOOLS OF PARALLEL CONTROLLED HASHING

The paper analyses the approaches to the development of driven cryptographic primitives. There had been developed the construction of the driven multi-hash constructions. There had been determined the cryptographic primitives for the reduction functions realization. There had been suggested the approaches to the functions' synthesis and methods of hashing which use these approaches. There had been presented the evaluations of software implementation performance of these methods.

Key words: *hashing, cryptographic primitives, multichannel reduction function.*

Introduction

Appearance of new attacks on hash functions [1] causes the necessity in revising the approaches to both, designing of compression function and recognition of hashing essence. This stipulated for the necessity in the development of the new concept of hashing. One of the directions of hashing improvement is the concept of controlled hashing [2], which stipulates for changing parameters of transformation in compression function from iteration to iteration. Since the famous methods of cryptanalysis as the research object use the compression functions, which parameters do not change from iteration to iteration, the realization of controlled hashing is expected to improve stability.

The insuring of high speed of data processing is of great importance for the cryptographic methods and methods of hashing in particular, which is achieved by the parallelization of computations [3]. The concept of the controlled hash allows to parallelize hashing process, and consequently to decrease its duration [4]. Therefore the researches in the direction of realization of the concept of driven hashing are important.

The objective of this research is the increase in hashing speed by realization of driven hashing conception.

To reach the goal the following tasks are to be solved:

- analysis of the known controlled hashing methods;
- development of controlled hash constructions;
- substantiation of choice of cryptographic primitives and synthesis of reduction functions;
- development of hash methods and their software implementation.

Analysis of the Known Controlled Hashing Methods

In hashing functions Dynamic SHA and Dynamic SHA-2, suggested for the participation in the contest for the new hashing standard SHA-3 [5, 6], there had been realized one of the approaches to the construction of the controlled hashing, based upon the use of controlled shifts. For the creation of set of compression functions, paper[5] suggests to use logical functions analogical, used in the hashing standard SHA-2 [7]. The number of bits to be shifted is suggested to be a controlled parameter. This parameter was suggested to determine depending on some modified input data blocks. This became the reason for the insufficient stability of Dynamic SHA and Dynamic SHA-2. Authors of the work [6] suggested the attack on these hash functions, which uses cryptanalytic ability to influence the drivenness, imposing some data blocks.

The known approach of controlled hash function development based on substitution-permutation networks (SPNs) is suggested in the work [8] and is developed in works [9, 10]. Particularly there are classes of cryptographic primitives and examples of ciphers and hash functions, presented in the work [8]. An example of $F_{2/1}$ class primitives (two input/output data bits, one controlling one) is showed in table 1 in algebraic normal form [8].

Table 1

An example of controlled cryptographic primitives of $F_{2/1}$ class

№	Primitive	№	Primitive	№	Primitive
1	$x_1v \oplus x_2v \oplus x_1$	9	$x_1v \oplus x_2 \oplus v$	17	$x_1v \oplus x_2v \oplus x_1 \oplus v \oplus 1$
2	$x_2v \oplus x_1$	10	$x_1v \oplus x_2 \oplus x_1 \oplus v$	18	$x_2v \oplus x_1 \oplus v \oplus 1$
3	$x_1v \oplus x_2v \oplus x_2$	11	$x_1v \oplus x_2v \oplus x_2 \oplus v$	19	$x_1v \oplus x_2 \oplus x_1 \oplus 1$
4	$x_2v \oplus x_1 \oplus x_2$	12	$x_2v \oplus x_2 \oplus x_1 \oplus v$	20	$x_1v \oplus x_2 \oplus 1$
5	$x_1v \oplus x_2$	13	$x_2v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	21	$x_2v \oplus x_1 \oplus x_2 \oplus 1$
6	$x_1v \oplus x_2 \oplus x_1$	14	$x_1v \oplus x_2v \oplus x_2 \oplus v \oplus 1$	22	$x_1v \oplus x_2v \oplus x_2 \oplus 1$
7	$x_2v \oplus x_1 \oplus v$	15	$x_1v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	23	$x_2v \oplus x_1 \oplus 1$
8	$x_1v \oplus x_2v \oplus x_1 \oplus v$	16	$x_1v \oplus x_2 \oplus v \oplus 1$	24	$x_1v \oplus x_2v \oplus x_1 \oplus 1$

Authors of work [8], combining in pairs these kinds of primitives, determine the best combinations using criteria of balancing of output values and transformation nonlinearity thus receiving SPN of $F_{2/1}$ class. SPNs of $F_{2/2}$ and $F_{3/1}$ classes are received in the same way in the work [8]. Driven functions that have two, three and four driving bits for two input/output bits are presented in the work [9], and reverse transformations for these primitives are presented in the work [10].

The disadvantage of SPNs approach is their primary orientation on cipher development, and that is why cryptographic primitives fail to consider hashing peculiarities. For example, hashing does not necessarily requires the availability of cryptographic primitive which conducts the reverse transformation, unlike ciphering, where this requirement is obligatory. Moreover, functions suggested in works [8 – 10] are primarily oriented to the hardware implementation, where permutation of two neighboring bits of data block is implemented by native conductor divaricating. At the same time the software implementation of these primitives is complicated by universal processors, which are not adopted to such kind of transformations. That's why they should be realized, using the reserve copying of the variable, its shifting, masking, which decrease the speed of calculation of logic functions.

Thus, the known compression functions for the methods of controlled hashing suffer from drawbacks related to the insufficient stability or speed of software realization. Therefore, it is necessary to develop the new compression functions, which require, first of all, the development of the models of hashing iterations which will consider the parallelization of calculations.

Driven Hash Constructions Development

Since the hashing process has to ensure the arbitrary length for data hashing, it has to be iterative [8]. Iterativeness stipulates for data processing by blocks and their coupling with the intermediate hash-value, received at previous iteration. For the parallelization it is suggested to use several hashing channels (computing cores, that realize the compression function). To ensure the firm channel coupling with each other it is necessary that the calculations in each channel on each iteration would consider the intermediate hash-values, received in the other channels [4]. The controlled hashing however allows to tie channels with each other by using channel intermediate hash-values as argument-like as well as using control vector:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, m_i\right) \\ v_i^{(1)} = g\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(q)}\right) \\ v_i^{(2)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(q)}\right) \\ \ldots \\ v_i^{(q)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q-1)}\right) \end{cases}, \quad (1)$$

where $h_i^{(j)}$ – intermediate hash received by the j^{th} pipe ($j = \overline{1,q}$) on the i^{th} iteration ($i = \overline{1,l}$); m_i – the i^{th} data block; $f_{v_i^{(j)}}(\cdot)$ – compression function, which ensures the constant length of the output value; $v_i^{(j)}$ – control vector, which determines compression function parameters $f_{v_i^{(j)}}(\cdot)$ in the j^{th} channel on the i^{th} iteration; $g(\cdot)$ – function of driving vector forming.

Let us generalize the construction (1) for k intermediate hashe-values, which are used as compression function $f_{v_i^{(j)}}(\cdot)$ argument, and ϕ intermediate hashe-values – as function $g(\cdot)$ argument:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(k)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(k+1)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, \ldots, h_{i-1}^{(k-1)}, m_i\right) \\ v_i^{(1)} = g\left(h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+1)}\right) \\ v_i^{(2)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+2)}\right) \\ \ldots \\ v_i^{(q)} = g\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q-2)}, \ldots, h_{i-1}^{(q-\phi)}\right) \end{cases}, \quad (2)$$

where $k < (q - \phi)$.

For the realization of the multichannel controlled hash construction (2), we determine the cryptographic primitives, which are used for synthesis of many compression functions.

Compression Functions Set for Driven Hashing

In order that the controlled hashing methods would allow for quick realization of this process, it is necessary that the operations, used as cryptographic primitives, would be "convenient" for the universal processors. Those operation are: adding modulo 2^n (+); exclusive or ($\oplus$); inverting (~); logic multiplication ($\wedge$); logic adding ($\vee$);cyclic shift by *u* bits rightwards (>>> *u*); simple shift by *u* bits rightwards/leftwards (>> *u*/ << *u*). At the same time the cryptographic operations should ensure the equal influence of each bit of the input data on the result of transformation, and operation of simple shift (>> *u*/ << *u*) doest allow to execute this condition. That's why the operations >> *u* and << *u* couldn't be used for driven hashing compression functions constructing. It is suggested to use the number of bits *u* , being cyclic shifted rightwards, as the driven parameter, because its

change is easy and quick for the realization by software. Moreover each driven parameter is suggested to be used as a part of driving vector.

It is suggested to use the logic functions, used in standard SHA-2 [7] as a base for compression function and to modify them for considering the exactly driven hashing. Thus, each channel would realize the following type of transformations ($k = 2$):

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}, \tag{3}$$

where $u_i^{(j)(x\kappa)}$ – the number of bits, for which the variable x shifts to the κ^{th} position in the compression function in the i^{th} iteration in the j^{th} channel of hashing.

Three controlled parameters are used in compression function (3), consequently the control vector is the concatenation of these parameters $v_i^{(j)} = u_i^{(j)(m1)} \parallel u_i^{(j)(h1)} \parallel u_i^{(j)(h2)}$.

It is suggested to modify one more function, being used in the standard [7] for ensuring the greater influence of the compression function arguments:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}. \tag{4}$$

The control vector would not change for the function (4) in comparison with function (3). It is suggested to use both, the operation $\wedge$, as well as the operation $\vee$ during the increase in number of arguments in compression function. For example, for the case of $k = 4$ the compression function (3) would change as:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{5}$$

The compression function (4) in case of $k = 4$ would change as:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{6}$$

During the increase in parameter *k*, it is suggested to use an approach which is analogues to the used one during the construction of the compression function (5) and (6) from functions (3) and (4) correspondingly.

Compression functions (3-6) were used for software realization of the controlled hashing on programming language C and were compiled by programming environment Visual Studio 2005. The received programs stipulated for the use of the similar procedures of driving vector formation and initialization. Programs testing was done by computer with processor Intel Pentium 4 with frequency 3 GHz, operating memory 2 Gb. For the evaluation of hashing speed there was used the standard library "time.h". The evaluation of data hashing duration by these methods of controlled

hashing for 16 channels, processing 16-bit data (output hash-value is 256 bits) is represented in the table 2.

Table 2

Estimations of Hashing Duration

Reduction function	Hashing duration, cycles			
	10 Kb	100 Kb	1 Mb	10 Mb
(3)	78	781	8219	82328
(4)	141	1516	15297	151813
(5)	94	812	8547	84718
(6)	171	1719	17406	171719

The developed programs stipulate for the processing of data of any length and formation of output hash-value for them with length, which is multiplied to channel bitwise length.

Conclusions

The necessity in achieving higher speed of hashing and the need in ensuring stability in this process to the attacks cause the necessity in searching for new concepts of hashing, such as controlled hashing conception. Development of controlled hashing methods by cryptographic primitives, the execution of which is natural for universal processors allowed to achieve high speed increase linearly along with the increase in the volume of hashing data, whereas the known methods achieve the best speed during hashing of larger input data sets. Therefore the most appropriate use of hashing methods, suggested in this paper is hashing of small data amount, in particular, in protocols of authenticity.

REFERENCES

1. Second Preimage Attacks on Dithered Hash Functions [Електронний ресурс] / E. Andreeva, C. Bouillaguet, P.-A. Fouque and others. – 19 с. – Режим доступу: http://homes.esat.kuleuven.be/~eandreev/eurocrypt08.pdf.

2. Баришев Ю. В. Підхід до хешування, що стійке до аналізу зловмисника / Ю. В. Баришев // Системи обробки інформації. – № 3. – 2010. – С. 99 – 100.

3. Корченко А. Г. Построение систем защиты информации на нечетких множествах. Теория и практические решения / А. Г. Корченко. – К.: "МК-Пресс", 2006. – 320 с.

4. Баришев Ю. В. Методи побудови швидкого хешування / Ю. В. Баришев // Методи та засоби кодування, захисту й ущільнення інформації. Тези другої Міжнародної науково-практичної конференції, м. Вінниця, 22 – 24 квітня 2009 року. – Вінниця: ВНТУ, 2009. – С. 138 – 139.

5. Zijie Xu. Dynamic SHA [Електронний ресурс] / Zijie Xu // Cryptology ePrint Archive. – 2007. – 34 с. – Режим доступу: http://eprint.iacr.org/2007/476.pdf.

6. Aumasson J.-P. Cryptanalysis of Dynamic SHA(2) [Електронний ресурс] / J.-P. Aumasson, O. Dunkelman, S. Indesteege and B. Preneel // COSIC publications. – 2009. – 18 с. – Режим доступу: https://www.cosic.esat.kuleuven.be/publications/article-1277.pdf.

7. Secure Hash Standard: Federal Information Processing Publication Standard Publication 180-3 [Електронний ресурс] – Gaithersburg, 2008. – 27 с. – Режим доступу: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

8. Молдовян Н. А. Криптография: от примитивов к синтезу алгоритмов. / Н. А. Молдовян, А. А. Молдовян, М. А. Еремеев. – СПб.: БХВ-Петербург, 2004. – 448 с.

9. Рудницький В. М. Модель уніфікованого пристрою криптографічного перетворення інформації / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 3. – 2009. – С. 91 – 95.

10. Рудницький В. М. Синтез математичних моделей пристроїв декодування інформації для криптографічних систем / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 2. – 2009. – С. 124 – 128.

Volodymyr Luzhetsky – Dr. Sc. (Eng), Professor, Head of the Department for Information Protection.

Yuriy Baryshev – Post Graduate student with the Department for Information Protection.
Vinnytsia National Technical University