A Fast Cryptographic Hash Function Based on Linear Cellular Automata over GF(q)

Miodrag Mihaljević [1], *Yuliang Zheng* [2] *and Hideki Imai* [3]

[1] *Mathematical Institute, Serb. Acad. Sci. & Arts*
Kneza Mihaila 35, Belgrade, Yugoslavia

[2] *School of Comp. & Info. Tech., Monash University*
McMahons Road, Frankston, Melbourne, VIC 3199, Australia

[3] *Institute of Industrial Science, University of Tokyo*
7-22-1 Roppongi, Minato-ku, Tokyo 106, Japan

Abstract

One-way hash functions are an important tool in achieving authentication and data integrity. The aim of this paper is to propose a novel one-way hash function based on linear cellular automata over GF(q). Design and security analysis of the proposed one-way hash function are based on the use of very recently published results on cellular automata and its applications in cryptography. The analysis indicates that the one-way hash function is secure against all known attacks. An important feature of the proposed one-way hash function is that it is especially suitable for compact and fast implementation.

Keywords

Data Integrity, Cryptographic Primitives, One-way Hash Functions, Cellular Automata

1 INTRODUCTION

Cryptographic hash functions play an important role in modern cryptography. The basic idea of cryptographic hash functions is that a hash-value serves as a compact representative image (sometimes called an imprint, digital fingerprint, or message digest) of an input string, and can be used as if it were uniquely identifiable with that string.

Following [1], at the highest level, cryptographic hash functions may be classified into two classes: hash functions, whose specification dictates a single input parameter - a message (unkeyed hash functions); and keyed hash functions, whose specification dictates two distinct inputs - a message and a secret key. This paper is concerned with unkeyed hash functions which are also called one-way hash functions.

A typical usage of one-way hash functions for data integrity is as follows. The hash-value corresponding to a particular message M is computed at time t_1. The integrity of this hash-value (but not the message itself) is protected in some manner. At a subsequent time t_2, the following test is carried out to determine whether the message has been altered, i.e., whether a message M' is the same as the original message. The hash-value of M' is computed and compared to the protected hash-value; if they are identical, one accepts that the inputs are also equal, and thus that the message has not been altered. The problem of preserving the integrity of a potentially large message is thus reduced to that of a small fixed-size hash-value. Since the existence of collisions is guaranteed in many-to-one mappings, the unique association between the inputs and hash-values can, at best, be in a computational sense. A hash-value should be uniquely identifiable with a single input in practice, and collisions should be computationally infeasible to find (essentially never occurring in practice).

In this paper, a novel and fast one-way hash function is proposed and analyzed. The proposed one-way hash function is based on a quite different approach than these employed in other one-way hash functions in that it is based on linear cellular automata over GF(q). Note that a cellular automata is a particular linear finite state machine. The proposed one-way hash function is a development of the previously proposed bit oriented one (see [18]) to the word oriented one-way hashing.

In Sections 2 - 4 relevant background about cellular automata and one-way hash functions is summarized. In Section 5 the novel one-way hash function is proposed. Its security together with efficiency is analyzed in Section 6. Some concluding remarks are made in Section 7.

2 LINEAR CELLULAR AUTOMATA OVER GF(Q)

A linear finite state machine (LFSM) is a realization or an implementation of certain linear operator. Linear feedback shift registers (LFSR) and Linear Cellular Automata (CA) are particular LFSMs. Following [4] this section summarize the main characteristics of the CA over GF(q), assuming q a power of prime (for background see also [2] and [3]).

A null-boundary linear hybrid cellular automata is a LFSM composed of a one-dimensional array of n cells with the following characteristics. Each cell consists of a single memory element capable of storing a member of GF(q), and a next-state computation function. We assume that communication between cells is nearest-neighbor, so that each cell is connected to only its left and right neighbors. The leftmost and rightmost cells behave as though their left and right neighbors, respectively, are in state 0, and this make the CA null-boundary. At each time step t, cell i has a state $s_i^{(t)}$ (that is a member of GF(q)). The next-state function of a cell is its updating rule, or just rule. A linear CA employs the linear next-state functions. If in a CA the same rule is applied to all cells, then the CA is called a uniform CA; otherwise it is called a hybrid CA.

For time step $t+1$, each cell i computes its new state $s_i^{(t+1)}$, using its next-state function f_i. In a CA, this function can depend on only the information available to the cell, and in the here considered case, it is the states of cells $i-1$, i, and $i+1$ at the time t. Since we require that f_i be linear,

$$s_i^{(t+1)} = f_i(s_{i-1}^{(t)}, s_i^{(t)}, s_{i+1}^{(t)}) = b_i s_{i-1}^{(t)} + d_i s_i^{(t)} + c_i s_{i+1}^{(t)} \ , \tag{1}$$

and b_i, d_i, and c_i are constants dependent on the particular machine. The multiplication and addition operations are performed in the field GF(q). The number of possible functions f_i is the number of choices for b_i, d_i, and c_i, which is q^3. Hence, the number of rule configurations for an n-cell CA is $(q^3)^n = q^{3n}$.

We define the state of a CA at time t to be the n-tuple formed from the states of the individual cells, $s^{(t)} = [s_1^{(t)}, ..., s_n^{(t)}]$. The next-state function of the CA is computed as $s^{(t+1)} = [f_1(0, s_1^{(t)}, s_2^{(t)}), ..., f_i(s_{i-1}^{(t)}, s_i^{(t)}, s_{i+1}^{(t)}), ...]$. Since each f_i is a linear function, f is also a linear function, mapping n-tuples to n-tuples. Linearity implies that f has an n by n matrix formulation A, so that the previous expression can be rewritten as a matrix-vector product $s^{(t+1)} = f(s^{(t)}) = As^{(t)}$, where A is the transition matrix for the CA, and the product is a matrix-vector multiplication over GF(q). Because the CA communication is restricted to nearest-neighbor, the matrix A is tridiagonal.

A CA has a maximum length cycle if the sequence of states $s^{(0)}$, $s^{(1)}$, $s^{(2)}$, ... , $s^{(0)}$ includes all $q^n - 1$ nonzero states for any nonzero starting state $s^{(0)}$. Let the transition matrix of an LFSM be denoted A_{LFSM}. The characteristic polynomial of the LFSM is defined to be: $|xI - A_{LFSM}|$, where x is an indeterminante, and I is the identity matrix with the same dimension as A_{LFSM}. The characteristic polynomial is primitive if and only if the LFSM has a maximal length cycle.

3 GENERAL MODEL FOR ITERATED HASH FUNCTIONS

Most one-way hash functions $hash(\cdot)$ are designed as iterative processes which hash arbitrary-length inputs by processing successive fixed-size blocks of the input. A hash input M of arbitrary finite length is divided into fixed ℓ-length blocks M_i. This preprocessing typically involves appending extra bits (padding) as necessary to attain an overall length which is a multiple m of the block-length ℓ and often includes (for security reasons, see [6] and [7]) a block indicating the length of the unpadded input. Each block M_i then serves as input to an internal fixed-size function h, the *compression function* of $hash$, which computes a new intermediate result of bit-length n for some fixed n, as a function of the previous n-bit intermediate results and the next input block M_i. Let H_i denote the partial result after Stage i. Then the general process for an iterated one-way hash function with inputs $M = (M_1, M_2, ..., M_m)$ can be modeled as follows:

$$H_0 = IV \ , \quad H_i = h(H_{i-1}, M_i) \ , \ 1 \le i \le m \ , \quad hash(M) = g(H_m) \ . \tag{2}$$

H_{i-1} serves as the n-bit *chaining variable* between Stage $i-1$ and Stage i, and H_0 is a pre-defined starting value or initial value IV. An optional transformation g is used in a final step to map the n-bit chaining variable to an n'-bit result $g(H_m)$; g is often the identity mapping $g(H_m) = H_m$.

Specific one-way hash functions proposed in the literature differ from one another in preprocessing, compression function, and output transformation. Certain cellular automata based approaches for constructions of one-way hash functions are reported in [7], [11], [12] and [18].

4 SECURITY OF THE ONE-WAY HASH FUNCTION

Based on [6], [7], [8], [9], in a number of models, it is possible to relate the security of $hash(\cdot)$ to the security of h and g according to the following result:
Theorem 1. (cf. [9]) Let $hash(\cdot)$ be an iterated hash function with MD-strengthening. Then preimage and collision attacks on $hash(\cdot)$ (where an attacker can choose IV freely) have roughly the same complexity as the corresponding attacks on h and g.

Theorem 1 gives a lower bound on the security of $hash(\cdot)$.

According to [9] the iterated hash functions based on the Davies-Meyer compression function given by the following

$$h(M_i, H_{i-1}) = E_{M_i}(H_{i-1}) \oplus H_{i-1} \ , \tag{3}$$

where $E_K(\cdot)$ is a block cipher controlled by the key K, are believed to be as secure as the underlying cipher $E_K(\cdot)$ is.

As a direct extension of the results of security related to cipher block chaining and the assumption 1 from [9] (which is a standard one in cryptography today), we assume the following.
Assumption 1. Let the compression function h be the Davies-Meyer function (2) and the employed cryptographic transformation is a secure one. Then finding collisions for h requires about $2^{n/2}$ operations, and finding a preimage for h requires about 2^n operations, assuming n-bit hash result.

The above discussions imply that the main problem in the design of a secure one-way hash function can be reduced to the design of a secure compression function and a good output function.

5 A NOVEL CELLULAR AUTOMATON BASED HASH FUNCTION

In this section, a novel dedicated one-way hash function is proposed. The proposed function follows the general model for iterated hash functions (see relation (2)), and employs the Davies-Meyer principle, which according to (3) assumes that the compression function h is defined by the following:

$$h(M_i, H_{i-1}) = F_{M_i}(H_{i-1}) \oplus H_{i-1} \ , \tag{4}$$

where $F_{M_i}(H_{i-1})$ is a function which maps H_{i-1} according to M_i, and M_i is the ith part of the whole message M. These would guarantee the approved basis for design and imply secure hash function construction assuming that the compression function and the output function are secure. The novel construction of the compression function h and the output function g is based on cellular automata and recently published results which imply the security of the novel h and g functions. The proposed hash function provides: very fast hashing, and the preimage and collision resistance due to the employed principles and building blocks. The novel compression function h^*, the output function g^*, and the whole hash function $hash^*$ are defined by the next three parts of this section.

5.1 Compression Function $h^*(\cdot)$

We assume the following notations:
• n is number of words in each M_i or H_{i-1}, and ℓ is number of bits in each word, and n is an even integer;
• $M_{i,k}$ is kth word of M_i, and an element of GF(2^ℓ);
• $H_{i-1,k}$ is kth word of H_{i-1}, and an element of GF(2^ℓ);
• $f_k(\cdot)$, $k = 1, 2, ..., K$, are functions each of which nonlinearly maps two elements from GF(2^ℓ) into an element of GF(q), q prime, $2^\ell - 1 < q < 2^{2\ell} - 1$, according to certain Boolean functions, assuming that the criteria from [10] are satisfied;
• CA($\cdot$) is an operator of mapping a current CA state into the next state, assuming n-length CA over GF(q) with primitive characteristic polynomial, and that elements of the matrix (1) satisfy the following:
- $c_i = 1$, $1 \leq i \leq n-1$,
- $b_i = -1$, $2 \leq i \leq n$,
- $d_i \in \{0, 1\}$, $1 \leq i \leq n$,
- the number of d_i, $1 \leq i \leq n$ that are 1 is minimal,
- q is prime;
Note that this yields a very efficient realization of the CA without multiplications over GF(q).
• $\phi_k(\cdot)$, $k = 1, 2, ..., K$, are functions each of which nonlinearly maps an element from GF(q) into an element of GF(q), according to certain Boolean functions, assuming that the criteria from [10] are satisfied.
• $\varphi_k(\cdot)$, $k = 1, 2, ..., K$, are functions each of which nonlinearly maps an element from GF(q) into an element of GF(2^ℓ), according to certain Boolean functions, assuming that the criteria from [10] are satisfied.

The compression function maps the input variables M_i and H_{i-1} into the output according to the following steps 1 - 5.

1. nonlinear combining with compression of the M_i and H_{i-1} words: the mappings $\{0, 1, ..., 2^\ell - 1\}^2 \rightarrow \{0, 1, ..., q-1\}$
   Generate an n-dimensional vector X_i with elements $X_{i,k}$, $k = 1, 2, ..., n$, from GF(q) according to the following:

$$X_{i,k} = f_{((M_{i,k}+H_{i-1,k}) mod \ell) mod K}(M_{i,k}, H_{i-1,k}) \ , \quad k = 1, 2, ..., n \ . \tag{5}$$

2. first CA processing
   Generate an n-dimensional vector Y_i with elements $Y_{i,k}$ from GF(q):

$$Y_i = CA(X_i) \ . \tag{6}$$

3. nonlinear mapping and permutation

Generate an n-dimensional vector Y'_i with elements $Y'_{i,k}$ from GF(q) according to the following:

$$Y'_{i,(k+k_0)modn} = \phi_{kmodK}((\ Y_k\ +\ Y_{n+1-k}\ )modq)\ , \quad k = 1, 2, ..., \frac{n}{2}\ , \tag{7}$$

$$Y'_{i,(k+k_0)modn} = \phi_{kmodK}(\ Y_k\ +\ Y_{k-\frac{n}{2}}\ )modq)\ , \quad k = \frac{n}{2}+1, \frac{n}{2}+2, ..., n\ , \tag{8}$$

where k_0 is certain constant $k_0 < n/2$.

4. second CA processing
   Generate an n-dimensional vector Z_i with elements $Z_{i,k}$ from GF(q):

$$Z_i = CA(Y'_i)\ \ . \tag{9}$$

5. nonlinear transformation and compression
   Generate an n-dimensional vector H'_i with elements $H'_{i,k}$ from GF(2^ℓ) according to the following:

$$H'_{i,k} = \varphi_{kmodK}(Z_{i,k})\ \ . \tag{10}$$

Accordingly, the compression function $h^*(\cdot)$ is then defined by the following

$$h^*(M_i, H_{i-1}) = H'_i \oplus H_{i-1} = H_i\ , \tag{11}$$

where $\oplus$ denotes bit-by-bit $mod2$ addition.

5.2 Output Function $g^*(\cdot)$

The output function maps H_m into an n-dimensional binary vector. The output function $g^*(\cdot)$ is a variant of the cellular automaton based keystream generator proposed and analyzed in [17]. The input argument for the generator is a transformation of H_m into an n-dimensional binary vector which serves as the "secret key", according to the following: ith bit of the binary vector is the mod2 sum of the bits in ith word of H_m. Using this "secret key" $g^*(\cdot)$ generates n output bits.

The main parts of the key stream generator which realizes the output function g^* are the following: an n-cell CA over GF(2), a ROM which contains the configuration rules for the CA, an n-length binary buffer, and an n-dimensional varying permutation.

Assume that $\eta < n$ maximal length CA's are chosen out of all possible maximal length CA's. These rules are noted as $\{R_0, R_1, ..., R_\eta\}$. The rule configuration control word corresponding to a rule R_i is stored in a ROM word. The output function operates as following:

- Initially the CA is configured with the rule $R_{0+\Delta_0}$, where Δ_0 is $mod\ \eta$ value of the secret key.
  With this configuration the CA runs one clock cycle. Then it is reconfigured with next rule (i.e., R_i) and runs another cycle. The rule configuration of CA changes after every

run, i.e., in the next run, a rule is $R_{(i+1+\Delta) mod\ \eta}$, where Δ is a numerical equivalent of the previous CA state.
- After each clock cycle, the content of a middle cell of the CA is taken as an output and stored in the n-length binary buffer.
- After n clock cycles, the buffer content is permuted according to a varying permutation controlled by the current CA state.

5.3 Hash Function $hash^*(\cdot)$

1. INPUT. The message M, and the n-words initial value IV.
2. PREPROCESSING. MD-strengthening and padding using the approach proposed in [10].
   Splitting the processed message into m blocks of n-words each: $M = (M_1, M_2, ..., M_m)$.
3. ITERATIVE PROCESSING. Assuming that $H_0 = IV$, for each $i = 1, 2, ..., m$, do the following:

   - calculate the compression function $h^*(\cdot)$ value:
     $H_i = h^*(M_i, H_{i-1})$,

   where $h^*(\cdot)$ is defined in the Section 5.1.
4. If H_m is the all zero vector recalculate H_m according to the following: $H_m = h^*(M_m, H_0)$, and proceed to the next step.
5. OUTPUT FUNCTION. Calculate $g^*(H_m)$, where $g^*(\cdot)$ is defined in the Section 5.2.
6. OUTPUT. n-bits message digest: $hash^*(M) = g^*(H_m)$.

6 ANALYSIS OF THE PROPOSED HASH FUNCTION

6.1 Security Analysis

Note that according to the Theorem 1, a lower bound on security of the proposed hash function is determined by the characteristics of its compression and output functions. Accordingly, the security will be considered through the security of the proposed functions $g^*(\cdot)$ and $h^*(\cdot)$. Security of both the functions will be examined on the preimage / 2nd preimage and collisions attacks.

Security of Compression Function $h^*(\cdot)$

Processing of each message block M_i, $i = 1, 2, ..., m$, by the compression function $h^*(M_i, H_{i-1})$ consists of the following:
- nonlinear mapping of M_i and H_{i-1} into an n-dimensional vector with elements from GF(q): the CA current state Y_i;
- CA mapping of its current state into the next one - an n-dimensional vector $CA(Y_i)$ with elements from GF(q);
- nonlinear mapping of $CA(Y_i)$ into the n-dimensional vector Y_i' with elements from GF(q).

- CA mapping of its current state equal to the the vector Y_i' into the next one - an n-dimensional vector Z_i with elements from GF(q);
- nonlinear mapping of the vector Z_i into an n-dimensional vector H_i' with elements from GF(2^ℓ);
- bit-by-bit $mod2$ addition of the elements of n-dimensional vectors H_i' and H_{i-1} yielding the new intermediate result H_i.

Accordingly, the following facts imply the security of the compression function:
(a) The CA has primitive characteristic polynomial so that any nonzero state is mapped into a nonzero state which belongs to the sequence of all possible different $2^n - 1$ nonzero n-dimensional vectors.
(b) High nonlinearity of the compression function due to the employed Boolean functions and CA.
(c) So far published algorithms for reconstruction of a CA state employing certain CA outputs, are the following:
algorithm from [13] based on noiseless sequence of bits generated by certain CA cell assuming, in general, a nonlinear configuration rule; algorithm from [15] based on error-free next CA state assuming a nonlinear configuration rule; algorithm from [16] based on the sequence of noisy CA (PCA) states assuming an additive configuration rule; algorithm from [17] based on the noisy sequence of bits sampled from CA (PCA) states assuming an additive configuration rule.
It can be directly shown that all these methods for reconstruction of certain CA state can not work in the case of $h^*(\cdot)$.
(d) The compression function is a cryptographic transformation.

Facts (a)-(d) imply that $h^*(\cdot)$ can be considered as a cryptographically secure one-way function, so that according to the Assumption 1 the following hold:
- finding preimage for given $h^*(\cdot)$ output requires about 2^n operations (i.e. testing of 2^n hypothesis);
- finding collision for $h^*(\cdot)$ requires about $2^{n/2}$ operations (testing of $2^{n/2}$ hypothesis).

Security of Output Function $g^*(\cdot)$

Recall that the output function $g^*(\cdot)$ is realized by a variant of the keystream generator proposed and analyzed in [17]. Cryptographic security examination of this generator shows that it is resistant on all attacks known so far, assuming that the length of employed PCA is greater than 120, [17]. Accordingly, we can accept that the output function $g^*(\cdot)$ is the secure one, and that finding the input argument of $g^*(\cdot)$ (preimage or 2nd preimage), i.e., the value H_m for given hash value $hash^*(M)$ has complexity 2^n assuming that $n > 120$. Due to the same reasons, i.e., because $g^*(\cdot)$ is realized by the cryptographically secure keystream generator, we can accept that no better attack than the Yuval's birthday attack, [5], can be expected for finding the collisions for the output function. The previous implies that finding a collision for $g^*(\cdot)$ requires testing about $2^{n/2}$ hypothesis, i.e. employing about $2^{n/2}$ operations.

6.2 Complexity Analysis

As the first, note that the set of functions (see the Section 5.1) can be efficiently realized by the truth tables in ROM, assuming moderate value of ℓ. Obviously, for realization of this approach certain "space-cost" should be paid. Based on the structure of the compression function $h^*(\cdot)$ it can be directly shown that processing of each n-words message block employs no more than $3n+n+2n+3n+n=10n$ additions, and approximately no more than $3n$ reading from ROM. Similarly, it can be directly shown that the processing cost in the output function $g^*(\cdot)$ (for its n-bits input) is approximately equal to $3n^2\ mod2$ additions + $n\ mod\ \eta$ additions + realization of the permutation. Accordingly, the overall complexity of processing (hashing) a message consisting of m blocks and each n-words long, can be estimated as approximately equal to performing $m(10n)+3n^2\ modq$ additions (including the ROM reading costs, $mod\ \eta$ additions, and realization of the permutation). So, the proposed hash function employs the number of operations approximately equal to $10+\frac{3n}{m}$ additions over GF(q) for hashing each message word.

7 CONCLUSIONS

This paper addresses the problem of designing a fast one-way hash function for word oriented applications, and it points out a new application of linear cellular automata over GF(q). The aim paper was to extend the applications of cellular automata based building blocks to the word oriented hash functions instead of the recently proposed bit oriented hashing. A theoretical basis for this goal were the recently published results related to the one-dimensional linear hybrid cellular automata over GF(q) [4].

A novel hash function is proposed and its security and complexity are analyzed. The proposed hash function employs the approved model of iterative hash function with novel compression and output functions.

The proposed compression function is one of the Davies-Meyer type based on cryptographic transformation employing cellular automata, and the output function is a keystream generator, also based on cellular automata. The employment of cellular automata ensures the efficiency of the proposed hash function.

The security of the proposed hash function was analyzed through the security of the compression and output functions. The analysis, based on the so far published results, implies that the proposed hash function has ideal security, i.e., given a hash n-bits output, producing each of a preimage or 2nd preimage requires testing of approximately 2^n hypothesis, and producing of a collision requires testing of approximately $2^{n/2}$ hypothesis, assuming $n > 120$.

Assuming a message of m blocks, each with n words, and each word of ℓ bits, the proposed hash function employs number of operations approximately equal to $10+\frac{3n}{m}$ additions over GF(q), for hashing each message word, or equivalently $\frac{10+3n/m}{log_2\ell}\ modq$ additions for hassing each message bit.

REFERENCES

A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography.* Boca Roton: CRC Press, 1997.

S. Wolfram, *Cellular Automata and Complexity.* Reading MA: Addison-Wesley, 1994.

P.P. Chaudhuri, D.R. Chaudhuri, S. Nandi and S. Chattopadhyay, *Additive Cellular Au-*

tomata: Theory and Applications. New York: IEEE Press, 1997.

K. Cattell and J.C. Muzio, "Analysis of one-dimensional linear hybrid cellular automata over GF(q)", *IEEE Trans. Comput.*, vol. 45, pp. 782-792, 1996.

G. Yuval, "How to swindle Rabin", *Cryptologia* vol. 3, pp. 187-190, 1979.

R. Merkle, "One way hash functions and DES", Advances in cryptology - CRYPTO 89, *Lecture Notes in Computer Science*, vol. 435, pp. 428-446, 1990.

I.B. Damgard, "A design principle for hash functions", Advances in Cryptology - CRYPTO 89, *Lecture Notes in Computer Science*, vol. 435, pp. 416-427, 1990.

Y. Zheng, T. Matsumoto and H. Imai, "Structural properties of one-way hash functions", Advances in cryptology - CRYPTO 90, *Lecture Notes in Computer Science*, vol. 537, pp. 303-313, 1991.

L. Knudsen and B. Preneel, "Fast and secure hashing based on codes", Advances in cryptology - CRYPTO 97, *Lecture Notes in Computer Science*, vol. 1294, pp. 485-498, 1997.

Y. Zheng, J. Pieprzyk and J. Sebery, "HAVAL - a one-way hashing algorithm with variable length of output", Advances in cryptology - AUSCRYPT 92, *Lecture Notes in Computer Science*, vol. 718, pp. 83-104, 1993.

J. Daemen, R. Govaerts and J. Vandewalle, "A framework for the design of one-way hash functions including cryptanalysis of Damgard's one-way function based on cellular automaton", Advances in cryptology - ASIACRYPT '91, *Lecture Notes in Computer Science*, vol. 739, 1993.

S. Hirose and S. Yoshida, "A one-way hash function based on a two-dimensional cellular automaton", *The 20th Symposium on Information Theory and Its Applications (SITA97)*, Matsuyama, Japan, Dec. 1997, Proc. vol. 1, pp. 213-216.

W. Meier and O. Staffelbach, "Analysis of pseudo random sequences generated by cellular automata", Advances in Cryptology - EUROCRYPT 91, *Lecture Notes in Computer Science*, vol. 547, pp. 186-189, 1992.

S.R. Blackburn, S. Murphy and K.G. Peterson, "Comments on "Theory and Applications of Cellular Automata in Cryptography"", *IEEE Trans. Comput.* vo. 46, pp. 637-638, May 1997.

C.K. Koc and A.M. Apohan, "Inversion of cellular automata iterations", *IEE Proc. - Comput. Digit. Tech.*, vol. 144, pp. 279-284, 1997.

M. Mihaljević, "Security examination of a cellular automata based pseudorandom generator using an algebraic replica approach", Applied Algebra, Algorithms and Error Correcting Codes - AAECC 12, *Lecture Notes in Computer Science*, vol. 1255, pp. 250-262, 1997.

M. Mihaljević, "An improved key stream generator based on the programmable cellular automata", Information and Communication Security - ICICS '97, *Lecture Notes in Computer Science*, vol. 1334, pp. 181-191, 1997.

M. Mihaljević, Y. Zheng, and H. Imai, "A cellular automaton based fast one-way hash function suitable for hardware implementation", *1998 International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, Japan, Yokohama, Feb. 1998, Pre-Proceedings, pp. 187-200 (also to appear in *Lecture Notes in Computer Science*).