MCSSHA

Secure hash algorithm's family

based on the regular Shift Registry and operations with bytes only

Principal Submitter: Mikhail Maslennikov
Revision: January 19, 2009

INTRODUCTION

- Any MCSSHA family algorithm is iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits). MCSSHA algorithm generate message digest with length 224, 256, 384 or 512 bits.
- In this presentation will be described algorithm MCSSHA-4 from MCSSHA family.

MCSSHA stages

- Any MCSSHA algorithm can be described in three stages: preprocessing, pre-hash computation and final hash computation. Each stage change Shift Registry state (SR-state) and final SR-state is message digest.

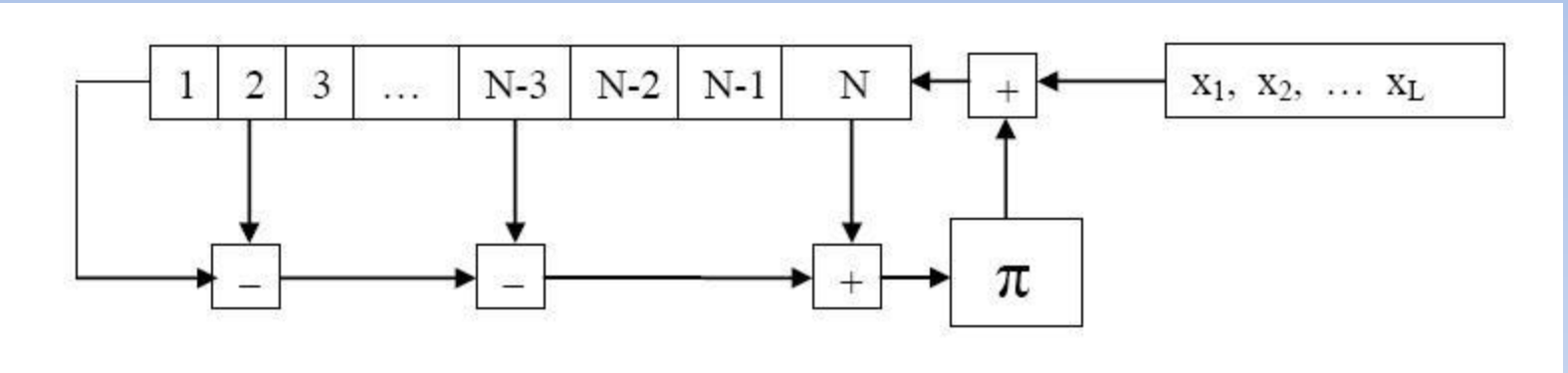
  Preprocessing setting initial SR-state to be used in the hash computation. Initial SR-state not depended from message and padding not used in MCSSHA algorithms.

  The pre-hash computation generates pre-final SR-state from the message.

  The final hash computation generates message digest – final SR-state - from pre-final SR-state.

CRYPTOGRAPHY SCHEME

- Shift Registry cryptography scheme.



where N – SR length (in bytes);

π - substitution from symmetric group S_{256};

$x_1, x_2, \ldots x_L$ – input sequence in bytes, i.e from Z/256.

Shift-registry elements are bytes, i.e. elements from Z/256,

shift-registry state (SR state) is vector $(y_i, y_{i+1}, \ldots, y_{i+N-1})$, where any value $y_i, y_{i+1}, \ldots, y_{i+N-1}$ is byte, i.e. element from Z/256.

If $(y_i, y_{i+1}, \ldots, y_{i+N-1})$ - SR state before i-step of work, than after i-step of work it will be $(y_{i+1}, y_{i+2}, \ldots, y_{i+N})$, where

$$y_{i+N} = \pi(y_i - y_{i+1} - y_{i+N-4} + y_{i+N-1}) + x_i$$

Shift Registry parameters

- Let's N – Shift Registry (SR) length in bytes. SR parameters are:
  - **SR state** - N bytes: $(y_0, y_1, \ldots, y_{N-1})$;
  - **SR points** – 4 values from 0 to N-1: (p_1, p_2, p_3, p_4) ;
  - **SR substitution** – group of 256 bytes where all values are different. This is 1 – 1 transformation π for bytes values 0,1,…,255: $\pi(y)$ - replacement byte y on substitution π.

Shift Registry Step

- SR step is transformation of the SR state and SR points during one step.
  Let's:

  $Y=(y_0,y_1, \ldots,y_{N-1})$ – SR state before step;
  $P=(p_1,p_2,p_3,p_4)$ – SR points before step;
  p - changeable position: $p= (p_4+1)(\text{mod } N)$;
  x – input byte for step.

  Then SR state F1(Y,P,x) and SR points F2(P) after step will be:

$$F1(Y,P,x) = (y_0,y_1, \ldots,y_{p-1},z,y_{p+1},\ldots,y_{N-1}) \quad 0 < p < N-1$$
$$F1(Y,P,x) = (z,y_1,y_2, \ldots,y_{N-1}) \quad p = 0$$
$$F1(Y,P,x) = (y_0,y_1, \ldots,y_{N-2},z) \quad p = N-1$$

  where

$$z = \pi(y_{p1} - y_{p2} - y_{p3} + y_{p4}) + x$$

$$F2(P) = ((p_1+1)(\text{mod } N), (p_2+1)(\text{mod } N), (p_3+1)(\text{mod } N), (p_4+1)(\text{mod } N))$$

Logarithmic substitution π

- For creation of necessary cryptographic reliability of transformation, substitution should provide “avalanche effect” duplication of distinctions of the text of the message. Such "avalanche effect " is defined by a matrix of transitive probabilities nonzero bigram P(π), i. e. a matrix of the size 255x255 in which on crossing of i-th row and j-th column there is p_{ij} - number of solutions of the system:

$$x - y = i$$
$$\pi(x) - \pi(y) = j$$

in Z/256 for each i,j ≠0. The "avalanche effect " that will be better, than less zero the matrix P(π) will contain. It is easy to prove, that the number of zero in this matrix cannot be less, than 253.

The example of the substitutions possessing minimally possible number of zero in a matrix P(π) is known – this is logarithmic substitutions, i.e.

$$\pi(x) = \log_\theta(\theta^{x+r} \oplus \rho), \quad \text{for } \theta^{x+r} \oplus \rho \neq 0,$$
$$\pi(x) = \log_\theta \rho, \quad \text{for } \theta^{x+r} \oplus \rho = 0$$

Here:

θ – primitive element of the GF(257) field,

ρ – non-zero element of the GF(257) field,

r – any element of the Z/256 ring.

Here two types of operations:

“+” operations in Z/256,

“⊕“ operations in GF(257).

- Substitution π in MCSSHA algorithms is logarithmic substitution with θ = 3, ρ = 1, r=0.

Shift Registry Length

- In MCSSHA-4 algorithm there are different SR length for preprocessing - pre-hash computation, and final hash computation.

Hash length (bits)	SR length (bytes) for preprocessing and pre-hash computation	SR length (bytes) for final hash computation
224	64	28
256	64	32
384	128	48
512	128	64

Initial Shift Registry Parameters

- If N – Shift Registry length, then initial SR parameters will be:

Initial SR state are

0 1 2 … N-1

Initial SR points are:

$$P_1 = 0$$
$$P_2 = 1$$
$$P_3 = N-4$$
$$P_4 = N-1$$

PRE-HASH COMPUTATION

- Pre-hash computation prepare SR state that depended from all message's bits except remain bits. For each byte m_i from message M pre-hash computation perform two types of steps:

  Type 1: SR step with input byte m_i.
  Type 2: SR step with input byte 0 (delay).

  Any MCSSHA algorithms during pre-hash computation perform one step type 1 (with input message's byte) and some steps type 2 (delay). In MCSSHA-4 value of delay (empty steps after each step with message's byte) can be set as parameter.

  For calculating message digest value of delay must be 2 or above!

  Default value for delay in MCSSHA-4 is 2.

  Starting parameters for pre-hash computation are initial SR parameters.

FINAL HASH COMPUTATION

- In MCSSHA-4 starting parameters for final hash computation are initial SR parameters and input sequence is double SR state after pre-hash computation. In final hash computation delay not used. SR state after final hash computation is message digest.

ESTIMATED COMPUTATIONAL EFFICIENCY AND MEMORY REQUIREMENTS

- During work of algorithm all operations are carried out extremely with bytes. ***Any operations with words - group of either 32 bits (4 bytes) or 64 bits (8 bytes) – not used.*** So algorithm can be realized on any kind of processors: 8-bits, 16-bits, 32-bits and 64-bits. Efficiency depended from processor's architecture.

Memory Requirement

- Memory requirement for MCSSHA-4 algorithm.

256 bytes for SR substitution (global variable);

Hash Structure – 143 bytes;

```
typedef struct MCSSHA4state_st
    {
    DataLength hashbitlen;    // (8 bytes)
    BitSequence x[6];         // (6 bytes; SR-points, remain byte and its length)
    BitSequence data[128];    // (128 bytes; SR-state for pre-hash computation)
    BitSequence delay;        // ( 1 byte; delay)
    } MCSSHA4_CTX,hashState;
```

For final hash computation necessary additional 64 bytes for SR state and 256 bytes for input sequence (for optimize speed).

Total memory requirement: 256+143+64+256 = **719** bytes.

This memory requirements same for 8-bits, 16-bits, 32-bits and 64-bits processors.

Comparisons speed with SHA-2 from OpenSSL

- Speed (32-bits OS) MS Windows Vista OS, 32-bits, Genuine Intel Core 2 CPU T2500 @ 2.00 GHz 2.00 GHz, time in sec.

Algorithm	Number of tests	Text length	224	256	384	512
SHA	1000000	1	1,743	1,727	6,301	6,422
MCSSHA-4 (delay = 2)	1000000	1	1,765	1,534	3,266	2,917
MCSSHA-4 (delay = 3)	1000000	1	1,762	1,479	3,298	2,894
SHA	1000	100000	2,685	2,639	4,804	4,726
MCSSHA-4 (delay = 2)	1000	100000	2,254	2,26	2,239	2,227
MCSSHA-4 (delay = 3)	1000	100000	3,032	3,014	2,986	3,002

- Speed (64-bits OS) MS Windows Server 2008, 64-bits, AMD Athlon 64 Processor 3200+ 2.00 GHz, time in sec.

Algorithm	Number of tests	Text length	224	256	384	512
SHA	1000000	1	1,422	1,422	4,75	4,766
MCSSHA-4 (delay = 2)	1000000	1	2,156	1,954	4,109	3,797
MCSSHA-4 (delay = 3)	1000000	1	2,188	1,969	4,172	3,812
SHA	1000	100000	2,422	2,407	3,563	3,562
MCSSHA-4 (delay = 2)	1000	100000	3,344	3,344	3,344	3,359
MCSSHA-4 (delay = 3)	1000	100000	4,766	4,766	4,766	4,781

Speed vs Security

- Second-preimage attack – attempt to find same SR states during pre-hash computation.
- Let's assume that we try to find two different messages M1 and M2, that some SR states are same, i.e.

  $$(y_t, y_{t+1}, \ldots, y_{t+N-1}) = (z_t, z_{t+1}, \ldots, z_{t+N-1})$$

  for some integer t.
- There are 3 zones for intermediate values:
  - Red Zone – values from t to t+N-1 (final SR states);
  - Yellow Zone – values from t-N to t-1 (SR state for N steps before final);
  - Green Zone – values less, than t-N (all SR states for N+1 and above steps before final).
- Byte from message M will be from some zone, if this byte present in some intermediate value from this zone as linear variable.
- Bytes from red zone provide concurrence of bytes in final SR states with probability 1.
- Bytes from yellow zone provide some relations, that allow to find same bytes in final SR states with high probability under some conditions
- Bytes from green zone provide concurrence of bytes in final SR states with probability approximately 2^{-8}.
- Values of bytes from message M in each zone depended from MCSSHA-4 delay.
- Safety will be that above, than less bytes will be in red and yellow zones.
- Ideally, in red and yellow zones must be as maximum N-H bytes, where H – hash length in bytes.

Speed vs Security

- Second-preimage attack, using red zone only

Hash length	MCSSHA-3			NIST requirements			MCSSHA-4 (delay=2)			MCSSHA-4 (delay=3)		
	Second preimage	Collision	Bytes	Second preimage	Collision	Bytes	Second preimage	Collision	Bytes	Second preimage	Collision	Bytes
224	2^{168}	2^{84}	21	2^{224}	2^{112}	28	2^{336}	2^{168}	42	2^{384}	2^{192}	48
256	2^{192}	2^{96}	24	2^{256}	2^{128}	32	2^{336}	2^{168}	42	2^{384}	2^{192}	48
384	2^{288}	2^{144}	36	2^{384}	2^{192}	48	2^{680}	2^{340}	85	2^{768}	2^{384}	96
512	2^{384}	2^{192}	48	2^{512}	2^{256}	64	2^{680}	2^{340}	85	2^{768}	2^{384}	96

Speed vs Security

- Number of bytes in red and yellow zones

SR length	Delay	Red Zone	Yellow Zone	Second preimage Q
64	0	64	64	$Q=1$
64	1	32	32	$1<Q<2^{256}$
64	2	22	21	$2^{168}<Q<2^{336}$
64	3	16	16	$2^{256}<Q<2^{384}$
128	0	128	128	$Q=1$
128	1	64	64	$1<Q<2^{512}$
128	2	43	42	$2^{344}<Q<2^{680}$
128	3	32	32	$2^{512}<Q<2^{768}$

Speed vs Security

- **Final conclusion**
- For calculating message digest in MCSSHA-4 algorithm values of delay must be 2 or above.
- For guarantee security level values of delay must be 3 or above.