New hash function designs

Igor Semaev
Selmersenteret, University of Bergen, Norway

February 1, 2008

1 Known designs

Let n be a natural number and M denotes the message represented by n-bit blocks m_i, so that

$$M = m_1, m_2, \ldots, m_s.$$

In case the message length is not a multiple of n, it should be padded somehow. We do not specify how at this point.

Most of the known hash functions are based on the Merkle-Damgȧrd construction. First, a compression function $f(h, m)$ is determined, where m is an n-bit input and another input h is of the hash value size r. The initial value h_0 of r-bit length is fixed. Then one computes

$$h_i = f(h_{i-1}, m_i)$$

for $i = 1, 2, \ldots, s$ and puts $H(M) = h_s$, which is the hash value of M.

Two properties of the modern hash functions based on the Merkle-Damgȧrd construction are observed:

1. The compression function is commonly a many round construction of some very simple round functions. Some of them heavily use bit logical operations which are not very compatible with modern 32 and 64-bit computers.

2. The computation of the final hash value $H(m)$ can't be distributed over several processors. That is, in order to compute h_i all $h_1, h_2, \ldots, h_{i-1}$ should be computed before.

2 New design principles

Our new design solves the above two problems. That is, in order to process one n-bit message block, one n-bit modular multiplication and few of n-bit modular additions and n-bit XOR's are calculated. So the hash value computation is especially efficient on 64-bit machines. The whole hash computation of long messages is easily distributed over any number of processors with tree hash

function construction. So that, with $O(s)$ processors the computational cost of $H(M)$ is roughly the processing cost of only $\log_{n/2} s$ blocks, that is essentially $\log_{n/2} s$ of n-bit multiplications. In practice, that makes very few operations. In what follows we introduce line construction, which has restricted ability to be distributed, and tree hash function construction which provides with full distribution.

In what follows n, t, r are generally any natural numbers.

2.1 Line construction

1. The same sequence of t-bit blocks $A = a_1, a_2, \ldots, a_s, \ldots$ is used in order to hash every message $M = m_1, m_2, \ldots, m_s$. In order to process one message block m_i the t-bit block a_i is generated. Blocks a_i are produced with a simple rule like
$$a_i \equiv a_{i-1} + b = a_0 + ib \mod 2^t$$
for fixed initial values a_0 and b. The length of A is bounded by $2^{t/2}$. That means that the longest message to process is $2^{t/2}$ of n-bit blocks. For short M the first few blocks of A may be kept.

2. For each t-bit block a the function
$$x \to f_a(x)$$
is determined. The input to f_a is n-bit and the output is r-bit, which is the hash value size. The function f_a is required to be a one-way permutation or close to that, provided $n = r$. Anyway that should be a one-way collision resistant function for all $a_i \in A$.

3. Let $1 \le u \le \frac{n}{2} - 1$ be a parameter. For $i = 0, 1, \ldots$ one computes
$$\begin{aligned} H_1 &= f_{a_1}(m_1) \oplus f_{a_2}(m_2) \oplus \ldots \oplus f_{a_u}(m_u), \\ H_2 &= f_{a_{u+1}}(m_{u+1}) \boxplus f_{a_{u+2}}(m_{u+2}) \boxplus \ldots \boxplus f_{a_{2u}}(m_{2u}), \\ H_3 &= f_{a_{2u+1}}(m_{2u+1}) \oplus f_{a_{2u+2}}(m_{2u+2}) \oplus \ldots \oplus f_{a_{3u}}(m_{3u}), \\ &\ldots \end{aligned}$$
Generally,
$$H_{i+1} = f_{a_{iu+1}}(m_{iu+1}) + f_{a_{iu+2}}(m_{iu+2}) + \ldots + f_{a_{(i+1)u}}(m_{(i+1)u}).$$
Here $+$ is
$$\begin{cases} \oplus, & i \text{ is even or zero;} \\ \boxplus, & i \text{ is odd,} \end{cases}$$
where $\oplus$ is the r-bit XOR and $\boxplus$ is the integer numbers addition modulo $p = 2^r + e$ with the truncation of the result to r bits. The last value $H_{\lceil \frac{s}{u} \rceil}$ may depend on less than u summands. The hash value $H(M)$ is then computed by the rule

$$H(M) = (\ldots(((H_1 \boxplus H_2) \oplus H_3) \boxplus H_4) \oplus \ldots) + H_{\lceil \frac{s}{u} \rceil}, \quad (1)$$

where $+$ is $\oplus$ or $\boxplus$ depending on $\lceil \frac{s}{u} \rceil$. One sees that the computation of $H(M)$ is fully distributed with at least u processors.

2.2 Tree constructions

In contrast with the previous line construction, the tree structure allows the hash function computation to be distributed over any number of processors. The price to pay is using up to $\lceil \log_u |M| \rceil$ additional memory locations for r-bit blocks during the computation. Here $s = |M|$ denotes the length of M in n-bit blocks. In Fig.1 hash computation tree is depicted for $u = 3$. The hash value of $M = m_1, m_2, \ldots, m_{27}$ is $m_{3,1}$. We suggest two ways to compute

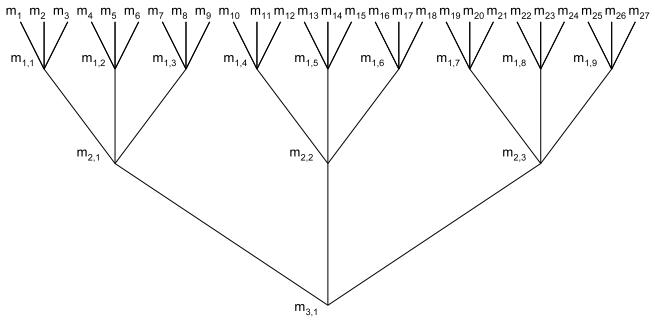

Figure 1: Hash computation tree

r-bit blocks $m_{k,i}$. In both the constructions, for every block of the message $M = m_1, m_2, \ldots, m_s$ a t-bit block of the sequence $A = a_1, a_2, \ldots, a_s, \ldots$ is generated, such that a_i is generated when m_i is being processed.

1. In the first variant of the tree construction we compute

$$m_{1,i+1} = f_{a_{iu+1}}(m_{iu+1}) \oplus f_{a_{iu+2}}(m_{iu+2}) \oplus \ldots \oplus f_{a_{(i+1)u}}(m_{(i+1)u})$$

for $i = 0, 1, \ldots$ The last block $m_{1,j}$ doesn't necessarily depend on u summands. Then

$$m_{k+1,i+1} = m_{k,iu+1} + m_{k,iu+2} + \ldots + m_{k,(i+1)u},$$

where $k = 1, 2, \ldots$ and $i = 0, 1, \ldots$ Here $+$ is

$$\begin{cases} \oplus, & k \text{ is even;} \\ \boxplus, & k \text{ is odd.} \end{cases}$$

The last computed block $m_{k,i}$ is the hash value $H(M)$.

2. In the second variant in addition to A, for every block of $m_{k,1}, \ldots, m_{k,i}, \ldots$ a t-bit block of the sequence $A_k = a_{k,1}, \ldots, a_{k,i}, \ldots$ is generated, such that $a_{k,i}$ is generated when $m_{k,i}$ is being processed. One uses a simple rule as $a_{k,i} \equiv a_{k,i-1} + b_k \equiv a_{k,0} + ib_k \mod 2^t$ for fixed initial values $a_{k,0}$ and b_k. Then

$$m_{1,i+1} = f_{a_{iu+1}}(m_{iu+1}) \oplus f_{a_{iu+2}}(m_{iu+2}) \oplus \ldots \oplus f_{a_{(i+1)u}}(m_{(i+1)u})$$

for $i = 0, 1, \ldots$ and

$$m_{k+1,i+1} = f_{a_{k,iu+1}}(m_{k,iu+1}) \oplus f_{a_{k,iu+2}}(m_{k,iu+2}) \oplus \ldots \oplus f_{a_{k,(i+1)u}}(m_{k,(i+1)u})$$

for $k = 1, 2, \ldots$ and $i = 0, 1, \ldots$ The last computed block $m_{k,i}$ is the hash value $H(M)$.

In both cases, given $O(s)$ processors the computational cost of $H(M)$ is $\lceil \log_u s \rceil$ computations of $f_a(x)$ values for some different a.

3 Choice of $f_a(x)$

Although it is not necessary by the previous Section, we put here $r = t = n$ and $n = 224, 256, 384, 512$ as in the NIST requirements or so. Let $p = 2^n + e$ be a prime number for a small natural number e. By $x \star y$ we denote the product of n-bit numbers x and y modulo p with the truncation of the result to n bits. Then we define the function

$$f_a(x) = ((x \oplus a) \star x^{(1)} \oplus c) \boxplus x, \tag{2}$$

where c is a constant related to the constant a, for instance, $c \equiv a + b' \mod p$, for some constant b'. The n-bit block $x^{(1)}$ is the swap of $n/2 + 1$ of the least significant bits and $n/2 - 1$ of the most significant bits of x, that is

$$x = x_0, \ldots, x_{n/2}, x_{n/2+1}, \ldots, x_{n-1}$$
$$x^{(1)} = x_{n/2+1}, \ldots, x_{n-1}, x_0, \ldots, x_{n/2}.$$

The choice of $x^{(1)}$ is justified by the following proposition.

Lemma 1 *If $n \equiv 0 \mod 4$, then $x = x^{(1)}$ if and only if $x = 0$ or $2^n - 1$.*

Proof. If $x = 0$ or $2^n - 1$, then obviously $x = x^{(1)}$. Let now $x = x^{(1)}$. Then

$$x_0 = x_{\frac{n}{2}+1} = x_1 = \ldots = x_{\frac{n}{2}-1} = x_{n-1} = \ldots = x_{n-2} = x_{\frac{n}{2}-1}.$$

As $n \equiv 0 \mod 4$ all x_i occur in the chain and they are all equal. Therefore, $x = 0$ or $2^n - 1$.

4 Notes on cryptanalysis

The task of the hash function cryptanalysis is to produce two different messages $M_1 = m_1, m_2, \ldots, m_s$ and $M_2 = m'_1, m'_2, \ldots, m'_t$ such that $H(M_1) = H(M_2)$. Therefore, the described functions are considered to be broken if e.g. for at least one $a \in A$(or A_k) the function $f_a(x)$ is not collision resistant or one-way.

Let $f_{a_i}(x)$ be not a collision resistant function and x_1, x_2 be a collision. Then two i-block messages

$$M_1 = m_1, \ldots, m_{i-1}, x_1 \quad \text{and} \quad M_2 = m_1, \ldots, m_{i-1}, x_2$$

collide in both the line and tree constructions.

Assume that some of $f_{a_i}(x)$ are not one-way. Let for instance $f_{a_i}(x)$ and $f_{a_j}(x)$ be not one-way and $1 \le i < j \le u$. For any t-bit blocks b, c one finds n-bit blocks x_1, x'_1 and x_2, x'_2 such that

$$\begin{aligned} f_{a_i}(x_1) &= b, \quad f_{a_i}(x'_1) = b, \\ f_{a_j}(x_2) &= c, \quad f_{a_j}(x'_2) = c. \end{aligned}$$

Then two j-block messages

$$\begin{aligned} M_1 &= m_1, \ldots, m_{i-1}, x_1, m_{i+1} \ldots, m_{j-1}, x'_1 \\ M_2 &= m_1, \ldots, m_{i-1}, x_2, m_{i+1} \ldots, m_{j-1}, x'_2 \end{aligned}$$

collide in both the line and tree constructions.

4.1 Bound on u

We will now explain the bound on the parameter $u \le \frac{n}{2} - 1$. Assume there is not any restriction for u. In this case the hash value may be computed by

$$H(M) = f_{a_1}(m_1) \oplus f_{a_2}(m_2) \oplus \ldots \oplus f_{a_u}(m_u)$$

for any M of any length u in n-bit blocks. In order to construct a collision, one produces $2u$ random n-bit blocks: $m_1, \ldots, m_u, m'_1, \ldots, m'_u$. Then one computes u n-bit binary vectors:

$$f_{a_1}(m_1) \oplus f_{a_1}(m'_1), \ldots, f_{a_u}(m_u) \oplus f_{a_u}(m'_u).$$

If u is close to n these vectors are linearly dependent with the probability close to 1. Let this dependence be

$$f_{a_{i_1}}(m_{i_1}) \oplus f_{a_{i_1}}(m'_{i_1}) \oplus \ldots \oplus f_{a_{i_v}}(m_{i_v}) \oplus f_{a_{i_v}}(m'_{i_v}) = 0.$$

Then the two messages

$$M_1 = m \ldots m_{i_1} \ldots m \ldots m_{i_v} \quad \text{and} \quad M'_1 = m \ldots m'_{i_1} \ldots m \ldots m'_{i_v}$$

have the same hash value, where the same n-bit block m appears at the places $j \notin \{i_1, \ldots, i_v\}$ in both the messages. This gives the collision.

We realize that if $u \leq \frac{n}{2}$, then the probability of the dependence doesn't exceed $\frac{1}{2^{n/2}}$. Then the running time of the attack overcomes the generic bound on the hash function security that is $2^{n/2}$ operations. The similar observation is valid if we change the operation $\oplus$ by $\boxplus$.

4.2 Broken $f_a(x)$

In this Section we show the functions, similar in appearance to that described in Section 3 or based on some other combination of the arithmetic operations, which are already broken. In other words, it was shown them not being one-way or collision resistant.

1. The function
$$f_a(x) = ((x \oplus a) \star x^{(1)} \oplus c) \oplus x, \tag{3}$$
differs from (2) in that the operation $\boxplus$ at the last occurrence of x was changed to $\oplus$. The function admits the simple explicit collision:
$$x_1 = a, \quad x_2 = 1^{(-1)} \quad \text{such that} \quad f_a(x_1) = f_a(x_2),$$
where $x \to x^{(-1)}$ denotes the inversion of $x \to x^{(1)}$. This observation is due to Seyed Mehdi Mohammad Hassanzadeh.

2. The function
$$f_a(x) = a \star x \oplus x, \tag{4}$$
is not one-way. We show that if b is any n-bit number of binary weight $|b| = s$ and there is x such that $f_a(x) = b$, then x is found in at most 2^{s+1} trials. Let
$$b = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_s}.$$
We observe that $x \oplus b = x + b_1$, where
$$b_1 \in \{\pm 2^{i_1} \pm 2^{i_2} \pm \ldots \pm 2^{i_s}\}$$
and $+$ is ordinary addition of integer numbers. So the equation $f_a(x) = b$ is represented as $ax - d \equiv x + b_1 \mod p$, where $d = 0$ if the product ax modulo p is within $0, \ldots, 2^n - 1$ and $d = 2^n$ if it is bigger than $2^n - 1$. In other words, $a \star x \equiv ax - d \mod p$. On the whole, there are 2^{s+1} possibilities for d, b_1. For each of them we find x satisfying
$$x \equiv \frac{b_1 + d}{a - 1} \mod p$$
and check whether $f_a(x) = b$. The correct x should be among them.

3. The function
$$f_a(x) = ((x \oplus a) \star x \boxplus c) \oplus x, \tag{5}$$
differs from (2) that the operations at c and x are swept and $x^{(1)}$ is changed to the plain x. We show that if for some $b \subseteq a$ there exists a collision $x, x \oplus b$

for $f_a(x)$, then it can be computed in at most 2^{3s} trials on the average, where s is the binary weight of b. The notation $b \subseteq a$ means that the $1's$ in the binary representation of b are among $1's$ in the binary representation of a.

Let $b = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_s}$ and $s \leq n/2$. By the previous observation we have for the sought x that

$$\begin{aligned} x \oplus b &= x + b_1, \\ x \oplus a \oplus b &= (x \oplus a) - b_1, \end{aligned} \tag{6}$$

where $-b_1$ appears in the second formula due to $b \subseteq a$, and

$$((x \oplus a \oplus b) \star (x \oplus b) \boxplus c) \oplus b = ((x \oplus a) \star x \boxplus c) + b_3, \tag{7}$$

where $b_1, b_3 \in \{\pm 2^{i_1} \pm 2^{i_2} \pm \ldots \pm 2^{i_s}\}$. Let $a = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_r}$, then

$$x \oplus a = x + a_1,$$

where $a_1 \in \{\pm 2^{j_1} \pm 2^{j_2} \pm \ldots \pm 2^{j_r}\}$. Finally,

$$\begin{aligned} (x \oplus a) \star x \boxplus c &= (x \oplus a)x + c - d_1 - d_1' \mod p, \\ (x \oplus a \oplus b) \star (x \oplus b) \boxplus c &= (x \oplus a \oplus b)(x \oplus b) + c - d_2 - d_2' \mod p, \end{aligned} \tag{8}$$

where d_1, d_1', d_2, d_2' are 0 or 2^n. Taking all this into account, we get $f_a(x) = f_a(x \oplus b)$ is equivalent to

$$-d_1 - d_1' = b_1 a_1 - b_1^2 + b_3 - d_2 - d_2', \tag{9}$$

We realize that the last condition doesn't depend on x. We formulate the algorithm to compute x:

- for any $b_1, b_3 \in \{\pm 2^{i_1} \pm 2^{i_2} \pm \ldots \pm 2^{i_s}\}$ and $d_1, d_1', d_2, d_2' \in \{0, 2^n\}$ compute a_1 from the equation (9). If $a_1 \in \{\pm 2^{j_1} \pm 2^{j_2} \pm \ldots \pm 2^{j_r}\}$, then proceed further, otherwise take another $b_1, b_3, d_1, d_1', d_2, d_2'$.
- Take any x that satisfies two equations in (6). The bits of x are determined at the positions, where bits of b equal 1. All other bits are free.
- Check the equations (7) and (8). If they are satisfied, then the collision is found, otherwise take another x or another $b_1, b_3, d_1, d_1', d_2, d_2'$.

It is obvious that the number of trials of $b_1, b_3, d_1, d_1', d_2, d_2'$ is at most 2^{2s+4}. We estimate the probability of the success. The number of choices for x is 2^{n-s} the probability that (7) is fulfilled is 2^{-s}. The probability that (8) is fulfilled is almost 1 if $d_1, d_1', d_2, d_2' = 0$. We can assume that for the true x. Therefore, the number of the trials for x, when the true $b_1, b_3, 0, 0, 0, 0$ is being processed, is 2^s on the average. That shows that the complexity of finding at least one x satisfying $f_a(x) = f_a(x \oplus b)$ is on the average 2^{3s} trials.

4. The function

$$f_a(x) = ((x \oplus a) \star x \oplus c) \boxplus x, \tag{10}$$

differs from (2) in that $x^{(1)}$ is changed to the plain x. Let $|a| = r$ and $|c| = s$ denote the weight of a, c. Let α be a parameter and assume that $(r \le \alpha)$ or $(r \ge n - \alpha)$ and $(s \le \alpha)$ or $(s \ge n - \alpha)$. Under this assumption we show that for a random b the equation $f_a(x) = b$ is solved with the probability at least P and in at most T trials. The values P and T are determined by r and s and shown in the following table.

r and s	T	P
$(r \ge n - \alpha)$ and $(s \ge n - \alpha)$	4	$2^{-2\alpha}$
$(r \le \alpha)$ and $(s \ge n - \alpha)$ or $(r \ge n - \alpha)$ and $(s \le \alpha)$	$2^{\alpha+2}$	$2^{-\alpha}$
$(r \le \alpha)$ and $(s \le \alpha)$	$2^{2\alpha+2}$	1

This data demonstrates that the function (10) should not be considered one-way if the weight of a and c much differ from the average value $\frac{n}{2}$. We only consider the case $r \le \alpha$ and $s \ge n - \alpha$, as other possibilities are similarly treated.

Let x satisfy $f_a(x) = b$, where $a = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_r}$. Then $x \oplus a = x + a_1$, where $a_1 \in \{\pm 2^{j_1} \pm 2^{j_2} \pm \ldots \pm 2^{j_r}\}$. So there are $2^r \le 2^\alpha$ possibilities for a_1. With the probability

$$\frac{2^s}{2^n} = 2^{-(n-s)} \ge 2^{-\alpha}$$

we get that

$$(x \oplus a) \star x \subseteq c$$

and so $(x \oplus a) \star x \oplus c = c - (x \oplus a) \star x$ in this case. Summing up all these observations, we conclude that

$$f_a(x) = (c - (x \oplus a) \star x) \boxplus x = \\ c - (x + a_1)x + d_1 - d_1' \mod p,$$

where d_1, d_1' are 0 or 2^n. One now solves $f_a(x) = b$ which is equivalent to the quadratic equation

$$-x^2 - (a_1 - 1)x + c + d_1 - d_1' \equiv b \mod p. \tag{11}$$

We find its root x if there are any roots and check whether $f_a(x) = b$. If not, then take another root or another tuple a_1, d_1, d_1' or another b. So the number of trials is at most $2^{\alpha+2}$ for each b.

We observe that we also find a collision for $f_a(x)$ with the same complexity parameters P and T as the quadratic equation (11) should generally admit two solutions if it admits one.