# Smashing MASH-1*

Vladimir Antipkin

**Abstract**

MASH-1 is modular arithmetic based hash function. It is presented in Part 4 of ISO/IEC 10118 standard for one and a half decade. Cryptographic strength of MASH-1 hash function is based on factorization problem of an RSA modulus along with redundancy in the input blocks of compression functions. Despite of this, we are able to introduce two large classes of moduli which allow practical time collision finding algorithm for MASH-1. In one case even multicollisions of arbitrary length can be constructed.

Keywords: hash-functions based on modular arithmetic, collision attack, MASH-1

## 1 Introduction

It becomes a common opinion that nowadays hash functions are working horses of modern cryptography. They are key ingredients in numerous schemes like public-key encryption, digital signatures, message-authentication codes or multiparty functionalities.

For the last years the focus on hash functions has dramatically increased, because of new attacks, mainly on MD hash functions family. All attention of cryptographic community was fixed on so called dedicated hash functions, or hash functions built upon a block cipher. As far as we know, hash-functions based on modular arithmetic didn't raise much interest in researches.

Hash functions based on modular arithmetic are particularly suitable for environments in which implementations of modular arithmetic of sufficient

*This work was supported by Academy of cryptography of the Russian Federation

length are already available. Anyway these hash functions are not widely adopted because of relatively poor performance.

Two constructions, MASH-1 and MASH-2 (for Modular Arithmetic Secure Hash), came as a result of a thorough design process and have been standardized in ISO/IEC 10118-4 [1] in 1998. The two hash functions differ only in the exponent used in the compression function while MASH-1 is obviously the fastest. Up to date there are no published results which would threaten security of any of these hash functions.

In this paper we present two classes of moduli which could be proper RSA moduli and simultaneously allow efficient construction of collisions for MASH-1 hash function. The fact of the existence of such moduli show that malicious third party who provide these moduli is able to forge users of MASH-1 algorithm.

## 2 Description of MASH hash functions

In this section we give a brief description of MASH hash functions. For detailed description we refer to [1]. Let $N$ be an RSA modulus and $L_N$ denote its length in bits. Let $L_\phi$ be a number divisible by 16 and $L_\phi + 1 \leq L_N \leq L_\phi + 16$. Properly padded input data $D$ of length divisible by $\frac{L_\phi}{2}$ is right-appended with $\frac{L_\phi}{2}$ bits of the binary representation of the length of the original (unpadded) data string. The resulting string is divided into a sequence of $q$ half-blocks: $D = D_1||\ldots||D_q$, $D_i \in V_{\frac{L_\phi}{2}}$, $i = 1, \ldots, q$.

Every half-block $D_i$ is than expanded to full block $B_i \in V_{L_\phi}$, $i = 1, \ldots, q$, in the following manner. Each consecutive half-byte of $D_i$ is prepended with a half-byte consisting of four binary ones, $i = 1, \ldots, q$.

The hashing process consists of iteration of compression function and a finalization stage. Compression function $\phi : V_{L_\phi} \times V_{L_\phi} \to V_{L_\phi}$ is defined as follows:

$$\phi(B_i, H_{i-1}) = ((((H_{i-1} \oplus B_i) \vee E)^e \bmod N) \sim L_\phi) \oplus H_{i-1},$$

where $H_0 = IV$ is an initialization vector fixed to all zeroes string, $E$ is a constant block equal to four ones (in the left-most position) followed by

$L_\phi - 4$ zeros, exponent $e$ is equal to 2 for MASH-1 and to 257 for MASH-2 (this is the only difference between these hash functions), $\sim$ is truncate operation to the corresponding number of right-most bits.

After all expanded data blocks proceed through compression function a finalization stage begins in the following manner. The intermediate block $H_q$ is represented as $H_q = H_{q1}||H_{q2}||H_{q3}||H_{q4},\ H_{q1}, H_{q2}, H_{q3}, H_{q4} \in V_{\frac{L_\phi}{4}}$. Then we define

$$Y_0 = H_{q3},\ Y_1 = H_{q1},\ Y_2 = H_{q4},\ Y_3 = H_{q2},$$

$$Y_i = Y_{i-1} \oplus Y_{i-4},\quad i = 4, \ldots, 15,$$

$$D_{q+i} = Y_{2i-2}||Y_{2i-1},\quad i = 1, \ldots, 8.$$

Data blocks $D_{q+i}$, $i = 1, \ldots, 8$, are processed through expansion process and compression function iteration with $IV$ equal to $H_q$. As a result we obtain $H_{q+8} \in V_{L_\phi}$ and a hash value $H = H_{q+8} \bmod p$, where $p$ is a prime number with bit length at most $\frac{L_\phi}{2}$ and three high order bits equal to ones.

Up to date the best known (2nd) preimage and collision attacks on MASH hash functions are universal attacks, which require about $2^{\frac{L_\phi}{2}}$ and $2^{\frac{L_\phi}{4}}$ operations correspondingly. In spite of the fact that no efficient attacks are known that exploit the factorization of the modulus, it is common opinion that knowledge of the factorization may reduce the security level. Therefore it is strongly recommended that factors of the modulus are kept secret. Our attacks (described below) do not take any advantage of knowledge of factors of the modulus.

## 3 First class of weak modulus of MASH-1

Let us first investigate the problem (which is of independent interest) of finding fixed points of compression function, i.e. strings $H_{j-1}$ such that $\phi(B_j, H_{j-1}) = H_{j-1}$. We are looking for conditions on modulus $N$, strings $H_{j-1}$ and $B_j$, such that

$$\phi(B_j, H_{j-1}) = ((((H_{j-1} \oplus B_j) \vee E)^e \bmod N) \sim L_\phi) \oplus H_{j-1} = H_{j-1}. \quad (1)$$

Equation (1) holds only if

$$(((H_{j-1} \oplus B_j) \vee E)^e \bmod N) \sim L_\phi = 0. \tag{2}$$

Since $E \neq 0$ and $0 < (H_{j-1} \oplus B_j) \vee E < 2^{L_\phi} < N$, then

$$((H_{j-1} \oplus B_j) \vee E)^2 \bmod N \neq 0.$$

Last inequality means that (1) follows

$$2^{L_\phi} \leq ((H_{j-1} \oplus B_j) \vee E)^2 \bmod N < N.$$

The number of possible values of $((H_{j-1} \oplus B_j) \vee E)^2 \bmod N$, for which equation (2) holds, do not exceed $\left\lfloor \frac{N}{2^{L_\phi}} \right\rfloor$. We can find all of them by the exhaustive search. They have the form $2^{L_\phi} \cdot v, 1 \leq v \leq \left\lfloor \frac{N}{2^{L_\phi}} \right\rfloor$, but in general assumptions we do not know the factors of $N$ and so we are not able to find roots by modulus $N$. So we will consider only those numbers $2^{L_\phi} \cdot v$, for which we can find roots easily. These are numbers with integer roots. Since $L_\phi$ is an even number, then $2^{\frac{L_\phi}{2}}$ is a root from $2^{L_\phi}$. Consequently, it is necessary that a quadratic root of $v$ exists in integers. In the latter case, $A = 2^{\frac{L_\phi}{2}}\sqrt{v}$ is a root of $((H_{j-1} \oplus B_j) \vee E)^2 \bmod N$ in integers. But $A < 2^{\frac{L_\phi+16}{2}}$, and if $L_\phi > 18$ then $A < 2^{L_\phi - 1}$. According to the description of MASH-1, the following inequality holds

$$2^{L_\phi - 1} < 2^{L_\phi} - 2^{L_\phi - 4} \leq (H_{j-1} \oplus B_j) \vee E < 2^{L_\phi}.$$

Thus the value $A = 2^{\frac{L_\phi}{2}}\sqrt{v}$ is not suitable.

Anyway, $A = N - 2^{\frac{L_\phi}{2}}\sqrt{v}$ can be suitable: this number is also a root of $((H_{j-1} \oplus B_j) \vee E)^2$ by modulus $N$. We need that $2^{L_\phi} - 2^{L_\phi - 4} \leq A < 2^{L_\phi}$, which leads to

$$2^{L_\phi} < N < 2^{L_\phi} + 2^{\frac{L_\phi}{2}}\sqrt{v}. \tag{3}$$

Hence $L_N = L_\phi + 1$ and according to $2^{L_\phi} \cdot v < N$ we have $v = 1$, i.e. $((H_{j-1} \oplus B_j) \vee E)^2 \bmod N = 2^{L_\phi}$. In addition, the following equality must hold

$$(H_{j-1} \oplus B_j) \vee E = N - 2^{\frac{L_\phi}{2}}. \tag{4}$$

Define $N = \sum\limits_{i=0}^{L_\phi/4} n_i 2^{4i}$, $B_j = \sum\limits_{i=0}^{L_\phi/4-1} b_i^{(j)} 2^{4i}$, $H_{j-1} = \sum\limits_{i=0}^{L_\phi/4-1} h_i^{(j-1)} 2^{4i}$, $N - 2^{\frac{L_\phi}{2}} = \sum\limits_{i=0}^{L_\phi/4-1} w_i 2^{4i}$. Relation (3) implies that

- $n_{L_\phi/4} = 1$,
- $w_i = n_i$ if $i < L_\phi/8$,
- $n_i = 0$ if $L_\phi/8 \leq i < L_\phi/4$,
- $w_i = 0xf$ if $L_\phi/8 \leq i < L_\phi/4$.

Here $0xf$ is a hexadecimal representation of 15. Because of equation (4) and the fact, that $b_i^{(j)} = 0xf$ if $i$ is odd, the following conditions on values of $h_i^{(j-1)}, b_i^{(j)}, n_i$ must hold:

$$\begin{cases} h_i^{(j-1)} \oplus b_i^{(j)} = n_i, & \text{for even } i < L_\phi/8, \\ h_i^{(j-1)} = n_i \oplus 0xf, & \text{for odd } i < L_\phi/8, \\ h_i^{(j-1)} \oplus b_i^{(j)} = 0xf, & \text{for even } L_\phi/8 \leq i < L_\phi/4-1, \\ h_i^{(j-1)} = 0, & \text{for odd } L_\phi/8 < i < L_\phi/4-1, \\ h_i^{(j-1)}, & \text{arbitrary for } i = L_\phi/4-1, \\ n_i = 0, & \text{for } L_\phi/8 < i < L_\phi/4, \\ n_{L_\phi/4} = 1. \end{cases} \tag{5}$$

The fulfilment of these conditions implies (1). Obviously, these conditions are not contradictory.

If $j = 1$ then $H_0 = 0$ and equation (1) holds if the following conditions are satisfied:

$$\begin{cases} n_i, & \text{arbitrary for even } i < L_\phi/8, \\ n_i = 0xf, & \text{for odd } i < L_\phi/8, \\ n_i = 0, & \text{for } L_\phi/8 \leq i < L_\phi/4, \\ b_i^{(j)} = n_i, & \text{for even } i < L_\phi/8, \\ b_i^{(j)} = 0xf, & \text{for even } L_\phi/8 \leq i < L_\phi/4-1, \\ n_{L_\phi/4} = 1. \end{cases} \tag{6}$$

Using the same approach it is possible to find intermediate and data values such that the resulting compression function values differ only in one bit, i.e.

$$\phi(B_j, H_{j-1}) = ((((H_{j-1} \oplus B_j) \vee E)^2 \bmod N) \sim\ L_\phi) \oplus H_{j-1} = H_{j-1} \oplus 2^k, \tag{7}$$

for certain $k$, $0 \leq k < L_\phi$, or equivalently

$$(((H_{j-1} \oplus B_j) \vee E)^2 \bmod N) \sim\ L_\phi = 2^k. \tag{8}$$

Using similar computations we obtain that $k$ must be even, and the following inequality should hold

$$2^{L_\phi} < N < 2^{L_\phi} + 2^{\frac{k}{2}}. \tag{9}$$

At the same time

$$(H_{j-1} \oplus B_j) \vee E = N - 2^{\frac{k}{2}}. \tag{10}$$

From (9) we obtain that bits of $N$, starting from bit number $k/2$, are all equal to zero (with the exception of bit number $L_\phi + 1$), i.e.

- $n_i = 0$ if $\lceil \frac{k}{8} \rceil \leq i < L_\phi/4$;
- left-most $4 - t$ bits of $n_i$ are equal to zero if $i = \lfloor \frac{k}{8} \rfloor$ and $t \equiv \left(\frac{k}{2}\right) \pmod 4$, $t \neq 0$.

Denote $N - 2^{\frac{k}{2}} = \sum_{i=0}^{L_\phi/4-1} w_i 2^{4i}$. Bits of $N - 2^{\frac{k}{2}}$, starting from bit number $k/2$, are all equal to 1, i.e.

- if $i \geq \lceil \frac{k}{8} \rceil$, then $w_i = 0xf$;
- if $i = \lfloor \frac{k}{8} \rfloor$, then left-most $4 - t$ bits of $w_i$ are all equal to 1, where $t \equiv \left(\frac{k}{2}\right) (\mathrm{mod} 4), t \neq 0$;
- other bits of $w_i$ coincide with the corresponding bits of $n_i$.

Equation (10) holds if the following conditions on $h_i^{(j-1)}$, $b_i^{(j)}$, $n_i$ fulfil.

$$\begin{cases} h_i^{(j-1)} \oplus b_i^{(j)} = n_i, & \text{for even } i < \lfloor k/8 \rfloor, \\ h_i^{(j-1)} = n_i \oplus 0xf, & \text{for odd } i < \lfloor k/8 \rfloor, \\ h_i^{(j-1)} \oplus b_i^{(j)} = 0xf, & \text{for even } \lceil k/8 \rceil \le i < L_\phi/4 - 1, \\ h_i^{(j-1)} = 0, & \text{for odd } \lceil k/8 \rceil \le i < L_\phi/4 - 1, \\ h_i^{(j-1)}, & \text{arbitrary for } i = L_\phi/4 - 1, \\ h_i^{(j-1)} = w_i \oplus 0xf, & \text{for odd } i = \lfloor k/8 \rfloor, \\ h_i^{(j-1)} \oplus b_i^{(j)} = w_i, & \text{for even } i = \lfloor k/8 \rfloor, \\ \text{left-most bits of } N & \text{starting from } \frac{k}{2} \text{ are all equal to zero}, \\ n_{L_\phi/4} = 1. \end{cases} \tag{11}$$

For $j = 1$ and $H_0 = 0$ fulfilment of the following conditions is enough for equation (7) to hold.

$$\begin{cases} b_i^{(1)} = n_i, & \text{for even } i < \lfloor k/8 \rfloor, \\ n_i = 0xf, & \text{for odd } i < \lfloor k/8 \rfloor, \\ b_i^{(1)} = 0xf, & \text{for even } \lceil k/8 \rceil \le i < L_\phi/4 - 1, \\ w_i = 0xf, & \text{for odd } i = \lfloor k/8 \rfloor, \\ b_i^{(1)} = w_i, & \text{for even } i = \lfloor k/8 \rfloor, \\ \text{left-most bits of } N & \text{starting from } \frac{k}{2} \text{ are all equal to zero}, \\ n_{L_\phi/4} = 1. \end{cases} \tag{12}$$

Proceeding from derived results, we will built a collision for MASH-1 with weak modulus. Let $k = k_1, 0 < k_1 < L_\phi, 16|k_1$. Suppose modulus $N$ is chosen in such a way that 2th, 6th and 7th conditions of (12) are fulfilled (since $k/8$ is even number, we do not consider 4th condition). We choose the first data block $B_1$ of the first message according to the remaining conditions of (12). After its transformation we get $H_0 = IV = 0, H_1 = 2^{k_1}$. The second data block $B_2$ of the first message is chosen according to conditions of (11) when $k = k_1$. It is easy to notice that $B_2 = B_1 \oplus 2^{k_1}$. After the transformation of the second block we get $H_2 = H_1 \oplus 2^{k_1} = 2^{k_1} \oplus 2^{k_1} = 0$.

Let us choose $k = k_1 + 2$. Since $\lfloor k_1/8 \rfloor = \lfloor (k_1 + 2)/8 \rfloor$ we get that 2th, 6th and 7th of (12) are fulfilled for previously chosen $N$. That's why we can choose the first block $B_1$ of the second message according to the remaining conditions of (12) when $k = k_1 + 2$ (note that values $w_i, i = k_1/8$ are different for $k = k_1$ and $k = k_1 + 2$). As a result of the transformation of the first block we get $H_0 = IV = 0, H_1 = 2^{k_1+2}$. The second block $B_2$ of the second message is chosen according to (11) when $k = k_1 + 2$. In this case $B_2 = B_1 \oplus 2^{k_1+2}$, and as a result we get $H_2 = H_1 \oplus 2^{k_1+2} = 2^{k_1+2} \oplus 2^{k_1+2} = 0$. For these messages intermediate hash-values after the first round are different, but intermediate hash-values after the second round are the same. Since lengths of the messages are the same, we get the desired collision.

It is obvious, that the described construction can be easily transformed to obtain multi-collisions.

## 4 Second class of weak modulus of MASH-1

Now let us consider another approach for collision finding. We will look for so called single block collision or collision for compression function, i.e. we will try to find such values $H_{j-1}$, $B_j^{(1)}$ and $B_j^{(2)}$, that

$$\phi(B_j^{(1)}, H_{j-1}) = \phi(B_j^{(2)}, H_{j-1}). \tag{13}$$

From (13) it follows that

$$(((H_{j-1} \oplus B_j^{(1)}) \vee E)^2 \bmod N) \sim L_\phi = (((H_{j-1} \oplus B_j^{(2)}) \vee E)^2 \bmod N) \sim L_\phi,$$

which deliberately holds if

$$((H_{j-1} \oplus B_j^{(1)}) \vee E)^2 \equiv ((H_{j-1} \oplus B_j^{(2)}) \vee E)^2 \bmod N.$$

Since we do not know the factors of $N$, we can only examine the case

$$((H_{j-1} \oplus B_j^{(1)}) \vee E) = -((H_{j-1} \oplus B_j^{(2)}) \vee E)(\bmod N),$$

i.e.

$$((H_{j-1} \oplus B_j^{(1)}) \vee E) + ((H_{j-1} \oplus B_j^{(2)}) \vee E) = N. \tag{14}$$

Define $A_1 = (H_{j-1} \oplus B_j^{(1)}) \vee E$ and $A_2 = (H_{j-1} \oplus B_j^{(2)}) \vee E$. From (14) it follows that $L_N = L_\phi + 1$ and at least 4 left-most bits of $N$ are all equal to 1. Half-bytes of $A_1$ and $A_2$ with odd numbers coincide and are equal to $h_i^{(j-1)} \oplus 0xf$, $i = 1, 3, \ldots$, with the exception of left-most half-byte, which is equal to $0xf$.

Further we will investigate whether modulus $N$ can be represented as a sum of $A_1$ and $A_2$. Define $A_j = \sum\limits_{i=0}^{L_\phi/4-1} a_i^{(j)} 2^{4i}$, $j = 1, 2$. We have

$$N = A_1 + A_2 = \sum_{i=0}^{L_\phi/8-1} (a_{2i}^{(1)} + a_{2i}^{(2)}) 2^{8i} + \sum_{i=0}^{L_\phi/8-1} 2a_{2i+1}^{(1)} 2^{8i+4}. \qquad (15)$$

Hence three left-most bits of each half-byte with odd numbers of $N$ are equal to three right-most bits of corresponding half-byte with odd numbers of $A_1$ and $A_2$.

Let us consider the five left-most bits of $N$. It follows from (15), that they should be equal to the binary representation of $a_0^{(1)} + a_0^{(2)}$. On the other hand $a_0^{(1)} + a_0^{(2)} \leq 15 + 15 < 31$. Hence if all five left-most bits of $N$ are equal to 1, i.e. $N \equiv 31(\text{mod}32)$, then $N$ can not be represented as a sum $A_1 + A_2$. If at least one of five left-most bits is equal to zero, then we can obtain $a_0^{(1)} \neq a_0^{(2)}$ in such a way, that five left-most bits of $N$ are equal to $a_0^{(1)} + a_0^{(2)}$.

Let us consider the bits of $N$ with numbers starting from $8i$ to $8i + 4$. These bits are equal to the binary representation if $a_{2i}^{(1)} + a_{2i}^{(2)} + \varepsilon$, where $\varepsilon$ is a bit of $A_1$ with number $8i - 1$. Obviously, for every values of these bits, it is always possible to obtain such $a_{2i}^{(1)}, a_{2i}^{(2)}, \varepsilon$, that their sum is equal to the necessary value.

Hence, if $L_N = L_\phi + 1$, 4 left-most bits of $N$ are equal to 1 and $N \neq 31(\bmod 32)$, then we can always obtain such numbers $A_1$ and $A_2$ (and $A_1 \neq A_2$), that $N = A_1 + A_2$ and half-bytes of $A_1$ and $A_2$ with even numbers coincide.

Let $N$ be represented in the mentioned above manner. We chose for $H_{j-1}$ an arbitrary number, that is less then $2^{L_\phi}$, and whose half-bytes with odd numbers are inverse of half-bytes with odd numbers of $A_1$ (with exception of left-most half-byte). Than we can obtain such $B_j^{(1)}$ and $B_j^{(2)}$, that $A_i = (H_{j-1} \oplus B_j^{(i)}) \vee E, i = 1, 2$. In this case (13) holds and we get a collision for compression function (with chosen value of $H_{j-1}$).

Let us consider the case $j = 1$. This lead us to the following additional condition: half-bytes of $A_1$ and $A_2$ with odd numbers are equal to $0xf$. This implies that bits of $N$ with numbers $8t_1 + t_2$, $t_1 = 0, 1, \ldots$, $t_2 \in \{5, 6, 7\}$, must be equal to 1, and each set of five bits with numbers $8t_1, 8t_1 + 1, \ldots, 8t_1 + 4$, $t_1 = 1, 2, \ldots$ must contain at least one bit equal to 1. In this case it is possible to represent $N$ as a sum $A_1 + A_2$. Now we are left to take $A_1$ and $A_2$ as the values for $B_1^{(1)}$ and $B_1^{(2)}$, and get a collision.

## 5 Conclusion

We have demonstrated the existence of weak moduli which lead to the total break of collision resistance of MASH-1 hash function. Our attacks do not use any knowledge of secret factors of the modulus. Users of MASH-1 hash function can be enforced to use weak modulus by malicious third party who is capable of their generation.

## References

[1] ISO/IEC 10118, Information technology — Security techniques — Hash-functions, Part 1: General, 2000, Part 2: Hash-functions using an n-bit block cipher algorithm, 2000. Part 3: Dedicated hash-functions, 2003. Part 4: Hash-functions using modular arithmetic, 1998.