2.B Algorithm Specifications and Supporting Documentations

# 2.B.4 STATEMENT of the expected strength of the NaSHA hash algorithm

## 1 Resistance to preimage and $2^{nd}$ preimage attacks

The quasigroup used for NaSHA-$(256, 2, 6)$ is of order $2^n = 2^{64}$ and $\mathcal{MT}$ is performed on $t = 16$ 64-bit words, so by Proposition 5, given in Part 2.B.1, one can find a second preimage or collision after around $2^{68}$ checks, but *under condition that the attacker knows the quasigroup operations and the values of the leaders.* The quasigroup operations

$$*_{a_1,b_1,c_1,a_2,b_2,c_2,a_3,b_3,c_3,\alpha_1,\beta_1,\gamma_1,A_1,B_1,C_1}$$

and

$$*_{a_1,b_1,c_1,a_2,b_2,c_2,a_3,b_3,c_3,\alpha_2,\beta_2,\gamma_2,A_2,B_2,C_2}$$

of NaSHA-$(256, 2, 6)$ hash function and the leaders $l_1$, $l_2$ depend on the input values of $\mathcal{MT}$.

Let $\mathcal{MT}(x_1||x_2||x_3||\dots||x_{16}) = (d_1, d_2, \dots, d_{16})$, where $x_i$ are 64-bit unknowns and $d_i$ are given 64-bit words. Let $\mathcal{A}_{l_2}(x_1||x_2||x_3||\dots||x_{16}) = z_1||z_2||z_3||\dots||z_{16}$ and put $y_i = \rho(z_i, 32)$ for $i = 1, 2, \dots, 16$. Then we obtain the following system of functional equations with unknowns $x_i$ and $y_i$ (i.e., $z_i$), unknown quasigroup operations $\bullet = *_{a_1,b_1,c_1,a_2,b_2,c_2,a_3,b_3,c_3,\alpha_1,\beta_1,\gamma_1,A_1,B_1,C_1}$

and $\star = *_{a_1,b_1,c_1,a_2,b_2,c_2,a_3,b_3,c_3,\alpha_2,\beta_2,\gamma_2,A_2,B_2,C_2}$, and unknown leaders $l_1,\ l_2$:

$$\begin{cases} (l_2 + x_1) \bullet x_1 = y_1 \\ (y_1 + x_2) \bullet x_2 = y_2 \\ \dots \\ (y_{15} + x_{16}) \bullet x_{16} = y_{16} \\ \hline y_{16} \star (y_{16} + l_1) = d_{16} \\ y_{15} \star (y_{15} + d_{16}) = d_{15} \\ \dots \\ y_1 \star\ (y_1 + d_2) = d_1. \end{cases} \tag{1}$$

For solving the system (1) we need at first to define the quasigroup operation $\bullet$ and $\star$ and the leaders $l_1$ and $l_2$. So, we have to choose 8 bytes $a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3$ (note that $c_3 = a_1$), 6 16-bit words $\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2$, 6 32-bit words $A_1, B_1, C_1, A_2, B_2, C_2$ and 2 64-bit words $l_1, l_2$, and that can be done in $2^{480}$ ways. Fix a choice of all of the constant words and then, after around $2^{68}$ checks, a solution $x_1, x_2, x_3, \dots, x_{16}$ of (1) can be found. Now, we have to see if the obtained solution $(x_1, x_2 \dots, x_16)$ satisfies the equalities

$$x_1 \oplus x_2 = l_1, \tag{2}$$

$$x_3 \oplus x_4 = l_2, \tag{3}$$

$$x_5 \oplus x_6 = a_1||b_1||c_1||a_2||b_2||c_2||a_3||b_3, \tag{4}$$

$$x_7 + x_8 = \alpha_1||\beta_1||\gamma_1||\alpha_2, \tag{5}$$

$$(x_9 + x_{10})(\texttt{mod}\ 2^{32}) = \beta_2||\gamma_2, \tag{6}$$

$$x_{11} + x_{12} = A_1||B_1, \tag{7}$$

$$x_{13} + x_{14} = C_1||A_2, \tag{8}$$

$$x_{15} + x_{16} = B_2||C_2. \tag{9}$$

For each of the equalities (2)–(5), (7)–(9), we have that the probability to be true is $2^{-64}$, so these seven equalities will be true with probability $2^{-448}$. The equality (6) will be true with a probability $2^{32}$. So, all the equalities (2)–(9) will be true with probability $2^{-480}$. (Namely, there are $(2^{64})^2$ pairs $(x_1, x_2)$, and there are $2^{64}$ different solutions of (2) when (2) is considered as an equation with 2 unknowns $x_5, x_6$. So, the equality (2) is true with a probability $2^{-64}$. The same discussion holds for the others equalities as well.)

So, after having around $2^{68}$ checks, we can find a solution of (1) with a probability $2^{-480}$. The space of all possible values of $(a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, \alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2, A_1, B_1, C_1, A_2, B_2, C_2, l_1, l_2)$ consists of $2^{480}$ elements. Then, after making $2^{68} \cdot 2^{480} = 2^{548}$ checks, a solution of (1) can be found with probability $1 - (1 - 2^{-480})^{2^{480}} \approx 0.53$.

We conclude that NaSHA-$(256, 2, 6)$ is $2^{nd}$ preimage resistant under this analysis. Consequently, it is preimage resistant with much higher complexity, since in this cases only $d_4, d_8, d_{12}$ and $d_{16}$ are known (the hash value of NaSHA hash is $d_4||d_8||d_{12}||d_{16}$). To discover the original image one has to choose $d_1, d_2, d_3, d_5, d_6, d_7, d_9, d_{10}, d_{11}, d_{13}, d_{14}, d_{15}$ in such a way the true values of $y_1, \ldots, y_{16}$ of (1) to be find, and that can be done with probability around $(2^{-64})^{12}$.

The analysis given above for NaSHA-$(256, 2, 6)$ holds true for NaSHA-$(224, 2, 6)$ too. So we give the following statement.

**Statement 1** *The best known preimage and second preimage attacks on NaSHA-*$(224, 2, 6)$ *and NaSHA-*$(256, 2, 6)$ *are the generic attacks.*

The same analysis holds true for NaSHA-$(384, 2, 6)$ and NaSHA-$(512, 2, 6)$. (In this case, a slightly better results are obtained since the value of $t$ is 32.) So we give the following statement.

**Statement 2** *The best known preimage and second preimage attacks on NaSHA-*$(384, 2, 6)$ *and NaSHA-*$(512, 2, 6)$ *are the generic attacks.*

## 2 Collision resistance

For attacking the collision resistance we have to find $(x_1, \ldots, x_{16}) \neq (x'_1, \ldots, x'_{16})$ such that $\mathcal{MT}(x_1, x_2, \ldots, x_{16}) = \mathcal{MT}(x'_1, x'_2, \ldots, x'_{16})$. We infer equations of kind

$$\begin{cases} (l_2 + x_1) \bullet x_1 = y_1 \\ (y_1 + x_2) \bullet x_2 = y_2 \\ \ldots \\ (y_{15} + x_{16}) \bullet x_{16} = y_{16} \end{cases} \tag{10}$$

$$\begin{cases} (l'_2 + x'_1) \bullet' x'_1 = y'_1 \\ (y'_1 + x'_2) \bullet' x'_2 = y'_2 \\ \ldots \\ (y'_{15} + x'_{16}) \bullet' x'_{16} = y'_{16} \end{cases} \tag{11}$$

$$\begin{cases} y_{16} \bullet (y_{16} + l_1) = y'_{16} \bullet' (y'_{16} + l'_1) \\ y_{15} \bullet (y_{15} + (y_{16} \bullet (y_{16} + l_1))) = y'_{15} \bullet' (y'_{15} + (y'_{16} \bullet' (y'_{16} + l'_1))) \\ \dots \\ y_1 \bullet (y_1 + (y_2 \bullet \dots) \dots) = y'_1 \bullet' (y'_1 + (y'_2 \bullet' \dots) \dots). \end{cases} \tag{12}$$

Now, besides the equalities (2)–(9), we will have eight more

$$x'_1 \oplus x'_2 = l'_1, \tag{13}$$

$$x'_3 \oplus x'_4 = l'_2, \tag{14}$$

$$x'_5 \oplus x'_6 = a'_1||b'_1||c'_1||a'_2||b'_2||c'_2||a'_3||b'_3, \tag{15}$$

$$x'_7 + x'_8 = \alpha'_1||\beta'_1||\gamma'_1||\alpha'_2, \tag{16}$$

$$(x'_9 + x'_{10})(\texttt{mod } 2^{32}) = \beta'_2||\gamma'_2, \tag{17}$$

$$x'_{11} + x'_{12} = A'_1||B'_1, \tag{18}$$

$$x'_{13} + x'_{14} = C'_1||A'_2, \tag{19}$$

$$x'_{15} + x'_{16} = B'_2||C'_2. \tag{20}$$

Then, even we assume that we have a solution of the system of equations (12), after $2^{1028}$ checks we can find a solution of (10) and (11) with probability $\approx 0.5$. So we have the following statement.

**Statement 3** *The best known collision attack on NaSHA-*$(224, 2, 6)$*, NaSHA-*$(256, 2, 6)$*, NaSHA-*$(384, 2, 6)$ *and NaSHA-*$(256, 2, 6)$ *is the birthday attack.*

We give the next statement too:

**Statement 4** *The best known collision attacks on m-bit hash function specified by taking a fixed subset of the candidate function's output bits of NaSHA-*$(224, 2, 6)$*, NaSHA-*$(256, 2, 6)$*, NaSHA-*$(384, 2, 6)$ *and NaSHA-*$(512, 2, 6)$ *is the generic attack, by replacing the length of the message digest by* $m$*.*

## 3 Resistance to length extension attacks

Generic length extension attack is an attack known upon Merkle-Damgård construction of hash functions. Let $M$ be a message that consists of $k$ message blocks $M = M_1||M_2||\dots||M_k$, with hash result $H(M)$. Two cases are examined:

1. The last message block does not include message length and it must be padded by 0's. Then the hash result $H(M)$ is an intermediate result. Let suppose that the attacker knows the intermediate result $H(M)$ and the message length $|M|$, so he can process at least one more block $M_{k+1}$. If a hash function is using an MD-straitening, the attacker encodes the message length by a new block $M_{k+2}$, finding a new hash result of the message $M||M_{k+1}$ without knowing the message $M$.

2. The last message block does include message length, so hash result $H(M)$ is the final hash result. Let suppose that the attacker knows the hash result $H(M)$ and the message length $|M|$, so he can process at least one more block $M_{k+1}$. Similarly as 1., the attacker encodes the message length in new block $M_{k+2}$, finding a new hash result of the message $M||M_{k+1}$ without knowing the message $M$.

NaSHA-$(m, k, r)$ has natural resistance against these generic attacks due to the fact that use wide-pipe design of Lucks [5, 6] and Coron's [3] suggestions. In every iterative step of the compression function, we use $2n$-bit message blocks and $2n$-bit chaining variable, so the strings of length $4n$ bits are mapped to strings of length $4n$ bits and then only $2n$ bits are kept for the next iterative step. More important is that *the length of chaining variable is at least two times wider that final digest value*, and this is enough for resistance of length extension attacks.

# 4 Support of HMAC, PRFs and Randomized Hashing

HMAC, invented by Bellare and all [1, 2], is a widely used message authentication code and a pseudo random function generator based on cryptographic hash functions. It takes a message of an arbitrary length and hashes it with a secret key. For the same length of the message it calls the compression function of the underlying hash function additionally three more times than the MD construction. Also, HMAC is proved to be a pseudo random function under the assumption that the compression function of the underlying hash function is a pseudo random function. It is defined as

$$HMAC(K, M) = H((K \oplus opad)||H(K \oplus ipad||M))$$

where $H$ is hash function, $K$ is the secret key, $M$ is message, *opad* and *ipad* are predefined constants.

Using the birthday paradox, a general distinguishing attack can be induced on HMAC (from a random function), that require about $2^{n/2}$ messages and easily can be converted into a general forgery attack on HMAC. There are two so called differential and rectangle distinguishers (Kim and all [7]) of the general structure of HMAC, which can lead to distinguishing attack, if HMAC is instantiated with hash function with a low difference propagation. Rectangle distinguisher on HMAC cannot be use as better attack then previous general attack, because its required data complexity is always larger then $2^{n/2}$ messages. In our knowledge, NaSHA hash algorithm has good difference propagation. We give the following statement.

**Statement 5** *NaSHA hash algorithm can be used as base of HMAC construction.*

Randomized hashing suggested by Halevi and Krawczyk [4], is a mode of operation for cryptographic hash functions for use with standard digital signatures and without necessitating of any changes in the internals of the underlying hash function or in the signature algorithms. This mode provide a safety net in case the (current or future) hash functions in use turn out to be less resilient to collision search than initially thought. Let $M$ be the message consisting of $k$ message blocks $M = m_1 m_2 \dots m_k$. Randomized hashing with hash function $H$ use random salt $r$ for hashing message $M$ as

$$H_r(M) = (r||m_1 \oplus r||m_2 \oplus r|| \dots ||m_k \oplus r)$$

Security of randomized hashing scheme is related to second-preimage resistance properties of the compression function. We give the following statement.

**Statement 6** *NaSHA hash algorithm can be used in randomized hashing scheme.*

# References


[1] M. Bellare, R. Canetti and Hugo Krawczyk, *Keying Hash Functions for Message Authentication*, Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, 1996, Proceedings, volume 1109 of LNCS, pages 1–15

[2] M. Bellare, R. Canetti and Hugo Krawczyk, *Message Authentication using Hash Functions - The HMAC Construction*, RSA Laboratories' CryptoBytes, Vol. **2**, No. 1, Spring 1996

[3] J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya, *Merkle-Damgård revisited: How to construct a hash function*, CRYPTO 2005, LNCS **3621**

[4] S. Halevi and H. Krawczyk, *Strengthening Digital Signatures via Randomized Hashing*, (2007)

[5] S. Lucks, *Design Principles for Iterated Hash Functions*, Cryptology ePrint Archive, Report 2004/253

[6] S. Lucks, *A Failure-Friendly Design Principle for Hash Functions*, ASIACRYPT 2005, LNCS **3788** (2005), pp. 474–494

[7] J. Kim, A. Biryukov, B. Preneel and S. Hong, *On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1*,