

Practical (Second) Preimage Attacks on the TCS_SHA-3 Family of Cryptographic Hash Functions

Gautham Sekar* and Soumyadeep Bhattacharya**

Abstract

TCS_SHA-3 is a family of four cryptographic hash functions that are covered by a United States patent (US 2009/0262925). The digest sizes are 224, 256, 384 and 512 bits. The hash functions use bijective functions in place of the standard compression functions. In this paper we describe first and second preimage attacks on the full hash functions. The second preimage attack requires negligible time and the first preimage attack requires $O(2^{36})$ time. In addition to these attacks, we also present a negligible time second preimage attack on a strengthened variant of the TCS_SHA-3. All the attacks have negligible memory requirements. To the best of our knowledge, there is no prior cryptanalysis of any member of the TCS_SHA-3 family in the literature.

Keywords

Cryptanalysis, Hash Function, (Second) Preimage Attack

1. Introduction

A hash function H takes an arbitrary length bit string M as input and outputs a fixed length bit string h (called *hash value* or *digest*). A cryptographic hash function is meant to satisfy certain security properties, the most important of which are listed below.

- **First preimage resistance:** given h , it is computationally infeasible to find an M such that $H(M) = h$.
- **Second preimage resistance:** given an M and $H(M)$, it is computationally infeasible to find an $M' \neq M$ such that $H(M) = H(M')$.
- **Collision resistance:** it is computationally infeasible to find an M and an M' , with $M' \neq M$, such that $H(M) = H(M')$.

The general model for cryptographic hash functions involves what is called a compression function. The function transforms a fixed-length bit string into a shorter, fixed-length bit string. The input message of a hash function, which is of arbitrary length, is partitioned into blocks of a fixed length (called the *block length*). However, before this can be done, it is required that the length of the message is a multiple of the block length. Given this and some security considerations, the message is 'padded' with bits in one of several ways (some padding schemes can be found in [1]). The message blocks are sequentially processed, with the compression function acting on the message blocks until all the blocks are pro-

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Manuscript received October 21, 2013; accepted January 17, 2014; online first December 31, 2014.

Corresponding Author: Gautham Sekar (sgautham@isichennai.res.in)

* Indian Statistical Institute, Chennai Centre, SETS Campus, MGR Knowledge City, CIT Campus, Taramani, Chennai 600113, India (sgautham@isichennai.res.in)

** Institute of Mathematical Sciences, CIT Campus, Taramani, Chennai 600113, India (sbhhta@imsc.res.in)

cessed. The end result is output as the digest. The general model for describing hash functions can be found in greater detail in [1].

A cryptographic hash function family is proposed by Vijayarangan of the Tata Consultancy Services (hereinafter called ‘TCS’) in [2]. The family is comprised of four hash functions, as four digest sizes (224, 256, 384, and 512 bits) are allowed. In [2], the hash functions are all actually called SHA-3, except in one or a few instances (see e.g., Clause 0095 of [2], where a member hash function is called TCS_SHA-3). However, as the name SHA-3 (with ‘SHA’ standing for ‘Secure Hash Algorithm’) has been in use by the National Institute of Standards and Technology (NIST), USA [3], we use the less common ‘TCS_SHA-3’ to denote the SHA-3 of [2]. Further, we denote by TCS_SHA-3- d the member that produces d -bit digests.

The design of TCS_SHA-3 deviates from the general model in that the compression function is replaced by a bijective function. This function uses a linear feedback shift register (LFSR) and a T-function. The design goals, as stated in [2], are to ‘prevent hash collisions’ and to ‘provide a secure hash function.’ This paper establishes that the design goals have not been met.

Motivation behind this work: The TCS, headquartered in India, is one of the largest IT services providers in the world, with annual revenue of more than \$10 billion for 2011–2012 [4]. In May 2012, the company was named the fourth most valuable IT services brand worldwide, based on image, reputation and *intellectual property* assessments [5]. The company’s annual research report for 2007–2008 mentions the following [6]:

“In the current year, major work has been done on cryptographic algorithms and hash functions, which form the basis of all data security today. Past research products [from the E-Security group of the TCS Innovation Labs, Hyderabad, India,]... are in active use around the country (India) by various customers in the banking and financial services industry. Organizations using our technology, directly or indirectly, include the RBI [(Reserve Bank of India)], National Securities Depositories [sic] Limited (NSDL), Ministry of Company [sic] Affairs (MCA), and many public sector banks.”

Since TCS_SHA-3 is a product of the above-mentioned E-Security group of the TCS Innovation Labs [6], there appears to be sufficient motivation to evaluate the security of the hash function family.

Contributions of this paper: This paper makes three contributions. First, we report a second preimage attack that requires negligible time and negligible memory for nearly guaranteed success. Second, we describe a first preimage attack on the TCS_SHA-3- d that requires $O(2^{27} \cdot d)$ time and negligible memory. Third, we present a second preimage attack, which also requires negligible time and negligible memory for nearly guaranteed success, on a strengthened variant of the TCS_SHA-3.

To the best of our knowledge, there is no prior published attack on the (strengthened) TCS_SHA-3.

Organization of this paper: Section 2 describes the TCS_SHA-3 family of hash functions. A second preimage attack and the first preimage attack are respectively described in Sections 3 and 4. In Section 5, we present the second preimage attack on the strengthened TCS_SHA-3. We conclude in Section 6. Appendix A provides the results of our simulations of the first preimage attack.

2. Specifications

We will first list the notation and conventions, which are used in the rest of this paper, in Table 1.

Table 1. Notation and conventions

| Symbol/notation | Meaning |
|-----------------|---|
| $I_i(\omega)$ | i th 32-bit word ($i = 0$ denotes the least significant word) of d -bit ω |
| $ x $ | length, in bits, of x |
| $x_{(i)}$ | i th bit ($i = 0$ denotes the least significant bit) of x |
| $x \parallel y$ | concatenation of two 32-bit words, x and y |
| \oplus | exclusive OR (XOR) |
| $0_2, 1_2$ | bit 0, bit 1 |

The TCS_SHA-3- d takes a message M of arbitrary length as input and returns a digest h (also denoted TCS_SHA-3- $d(M)$) of size d bits after the following sequence of processes, which include six ‘rounds.’

- 1) *Padding*: The input M is partitioned into $k = \lceil |M|/d \rceil$ blocks (denoted M_1, \dots, M_k), where each is d bits in length. Clause 0068 of [2] states that an initialization vector (IV) of length d is added to a message if and only if the *size of the message* is less than d . The designer should have evidently mean the following: if and only if $|M_k| < d$, for any $k \geq 1$, then M_k is added with the IV. Otherwise, the TCS_SHA-3 can only be applied to single block messages or multiple block messages, in which each one satisfies the condition $d \mid |M|$. For the simulations of Appendix A, we made the assignments $IV = 1_2 \parallel \{0_2\}^{d-1}$, $d = 224, 256, 384, 512$. The assignments are motivated by a case provided in [2] where the IV is chosen to be $1_2 \parallel \{0_2\}^{223}$ when the size of the message is less than 224 bits. In our simulations, as well as in the aforesaid example case, the IV is *XORed* with the corresponding message block. In summary, the padding rule is defined as follows: for any $k \geq 1$,

$$\begin{aligned}
 M_k \rightarrow M_k^* &:= M_k \oplus IV \quad \text{if } |M_k| < d, \\
 M_k \rightarrow M_k^* &:= M_k \quad \quad \quad \text{if } |M_k| = d.
 \end{aligned} \tag{1}$$

An implicit assumption in the above discussion is that $|M_k|$ is nonzero. Furthermore, when $|M_k| = d$, we infer that there is no extra ‘padding block’ that is appended to M . This is because, in such a case, there is no message block to which the IV ($= 1_2 \parallel \{0_2\}^{d-1}$) could be ‘added’.

- 2) *Round 1*: The first *round* has k steps; the steps are as follows:
 - (a) *Step 1 when $k > 1$* : An arbitrarily chosen d' -bit (such that $d' \leq d$) constant c is XORed with M_1 . The output, $c \oplus M_1$, is input to a bijective function F (defined later in this section). Thus, a d -bit string, $F(c \oplus M_1)$, is output.
Step 1 when $k = 1$: The arbitrarily chosen constant c is XORed with M_1^* . The output, $c \oplus M_1^*$, is input to F . Thus, a d -bit string, $F(c \oplus M_1^*)$, is output.
 - (b) *Steps 2 to k (i.e., when $k \geq 2$)*: Step i , $2 \leq i \leq k - 1$, is given by the following recursion:

$$z_i^1 = F(z_{i-1}^1 \oplus M_i), \quad z_1^1 := F(c \oplus M_1); \tag{2}$$

z_l^1 denotes the output of step l , $1 \leq l \leq k - 1$, of round 1.

Step k is given by:

$$z_k^1 = F(z_{k-1}^1 \oplus M_k^*), \tag{3}$$

where z_k^1 denotes the output of step k of round 1.

3) *Round 2*: Like round 1, round 2 also proceeds iteratively. The number of steps, s , is such that k “is not always the same as” s (see [2]). The input to round 2 is z_k^1 , the final output of round 1. The steps are as follows:

- (a) *Step 1*: The d' -bit constant c is XORed with z_k^1 . The output, $c \oplus z_k^1$, is input to the bijective function F . Thereby, a d -bit string, $F(c \oplus z_k^1)$, is generated as output.
- (b) *Steps 2 to s (i.e., when $s \geq 2$)*: Step i , $2 \leq i \leq s$, is given by the recursion:

$$z_i^2 = F(z_{i-1}^2 \oplus z_k^1), \quad z_1^2 := F(c \oplus z_k^1); \tag{4}$$

z_l^2 denotes the output of step l , $1 \leq l \leq s$, of round 2.

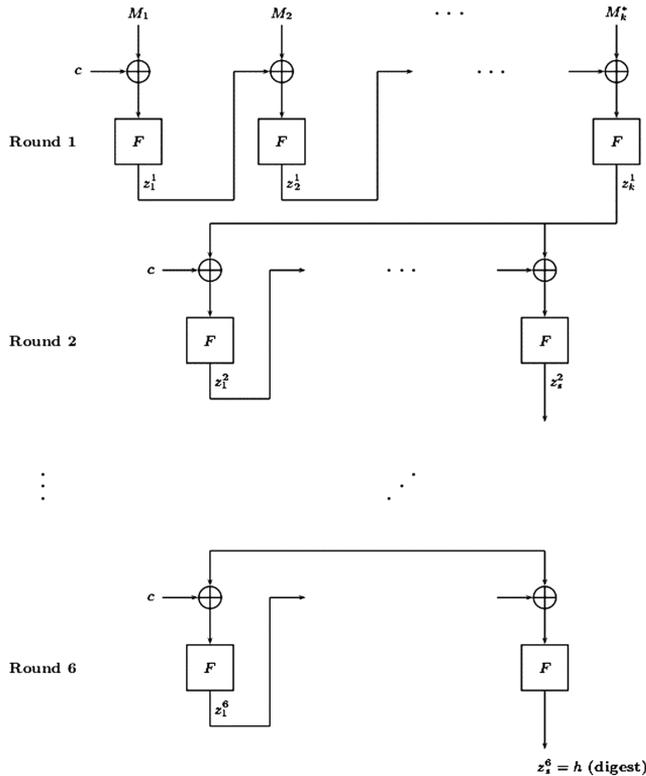


Fig. 1. The working of TCS_SHA-3 ($k \geq 2$) works.

4) *Rounds 3 to 6*: These rounds are similar to round 2. The number of steps in each of these rounds is, again, s . The input to round j , $3 \leq j \leq 6$, is z_s^{j-1} (i.e., the final output round $j - 1$ when z_l^{j-1} denotes the output of step l , $1 \leq l \leq s$, of round $j - 1$). The steps are as follows:

- (a) Step 1: Constant c is XORed with z_s^{j-1} . The output, $c \oplus z_s^{j-1}$, is input to the bijective function F . Thereby, a d -bit string, $F(c \oplus z_s^{j-1})$, is generated as output.
- (b) Steps 2 to s (i.e., when $s \geq 2$):¹ Step i , $2 \leq i \leq s$, is given by the recursion:

$$z_i^j = F(z_{i-1}^j \oplus z_s^{j-1}), \quad z_1^j = F(c \oplus z_s^{j-1}); \quad (5)$$

z_i^j denotes the output of step i , $1 \leq i \leq s$, of round j .

The d -bit digest h is simply the final output, z_s^6 . Fig. 1 shows the working of TCS_SHA-3 works. As stated earlier, [2] only says that k is not always the same as s . Therefore, s may be greater than k . We make this clarification because Fig. 1 may misleadingly suggest that s is always less than k .² Algorithm 1 describes the function $F: \{0_2, 1_2\}^d \rightarrow \{0_2, 1_2\}^d$.

Algorithm 1. The bijective function $F: \{0_2, 1_2\}^d \rightarrow \{0_2, 1_2\}^d$

Require: d -bit input α

Ensure: d -bit output λ

1. *Partition:* $\alpha \rightarrow \alpha_1 \parallel \alpha_2 \parallel \dots \parallel \alpha_{d/32}$ such that $|\alpha_i| = 32$ for all $1 \leq i \leq d/32$;
 2. *Shuffle:* $\alpha_i \rightarrow \beta_i$, for all $1 \leq i \leq d/32$, such that $\beta_{i(j)} = \alpha_{i(j/2)}$ if $2 \mid j$ and $\beta_{i(j)} = \alpha_{i(16+(j-1)/2)}$ otherwise;
 3. *Apply T-function:* $\beta_i \rightarrow \gamma_i := 2\beta_i^2 + \beta_i \bmod 2^{32}$, for all $1 \leq i \leq d/32$;
 4. *Apply LFSR:* $\gamma_i \rightarrow \lambda_i$, for all $1 \leq i \leq d/32$, such that $|\lambda_i| = 32$;³
 5. *Concatenate:* $\lambda := \lambda_1 \parallel \lambda_2 \parallel \dots \parallel \lambda_{d/32}$;
-

Note: The TCS_SHA-3 may be strengthened by introducing cipher block chaining in Algorithm 1. This point is further explained in Section 5.

3. Second Preimage Attack on TCS_SHA-3

Let $M = M_1 \parallel M_2 \parallel M_3 \parallel \dots \parallel M_k^*$, $k \geq 2$, (when $k = 2$, $M = M_1 \parallel M_2^*$; in general, when $k = l > 2$, $M = M_1 \parallel M_2 \parallel \dots \parallel M_l^*$), denote the given message and h its hash value. Let y_i denote the input to the i th invocation of the function F (see Fig. 2). Let $M' = M'_1 \parallel M'_2 \parallel M_3 \parallel \dots \parallel M_k^*$, with $M'_1 \neq M_1$ and $M'_2 \neq M_2$, denote another message, h' its hash value, and y'_i the corresponding input to the i th invocation of F . It immediately follows from Fig. 2 that if $y'_2 = y_2$, and then the outputs of round 1 are identical. This, in turn, implies that $h' = h$ (see Fig. 1) and we have a second preimage. The condition $y'_2 = y_2$ implies that:

$$F(M_1 \oplus c) \oplus M_2 = F(M'_1 \oplus c) \oplus M'_2. \quad (6)$$

¹ Perhaps the only criterion that s must satisfy is $s > 1$; otherwise, the TCS_SHA-3- d will have only one round.

² The (second) preimage attacks that we report in this paper are independent of the value of s , which is due to reasons that would be understood from Sections 3 and 4.

³ We omit the full description of the LFSR as it is elaborate and not relevant to our analysis (to be understood from Sections 3 and 4). In [2], the LFSR is described in Clauses 0078-0083.

It is straightforward to see that the conditions:

$$M'_2 = F(M'_1 \oplus c) \oplus M'_1, \tag{7}$$

$$M'_1 = M_2 \oplus F(M_1 \oplus c), \tag{8}$$

satisfy (6) and when $M'_1 \neq M_1$, we have a second preimage.⁴ Under reasonable assumptions of uniformity, the event $M'_1 = M_1$ occurs with negligible probability.

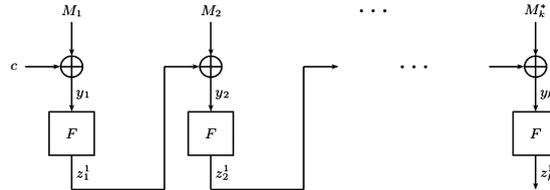


Fig. 2. Round 1 of TCS_SHA-3.

4. First Preimage Attack on TCS_SHA-3

Fig. 3 illustrates Algorithm 1 for the (sample) case when $d = 256$.

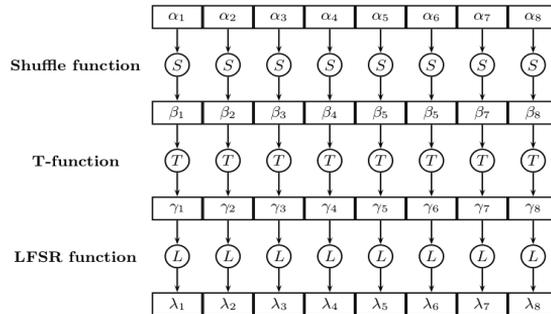


Fig. 3. The bijective function F of TCS_SHA-3-256; S , T and L are 32-bit to 32-bit functions.

From Fig. 3, we see that the TCS_SHA-3-256 (and the TCS_SHA-3 per se) has poor diffusion properties. A difference in α_i , for any $i \in \{1, 2, \dots, d/32\}$, affects λ_i alone. A single-bit difference in α_i , for any $i \in \{1, 2, \dots, d/32\}$, is ideally expected to affect 16 bits of λ_i .

Let us consider the case when $k = 1$. Then, given an input difference $\Delta I_i(M_1^*)$ for some $i \in \{0, 1, \dots, d/32 - 1\}$, the differential characteristic is shown in Fig. 4. Algorithm 2 exploits this differential characteristic to recover M_1 from its corresponding digest value. In step 3 of this algorithm, in place of $\{0_2\}^{32i}$ and $\{0_2\}^{32(d/32-i-1)}$, one can respectively have any 32i-bit and 32(d/32 - i - 1)-bit value. We have used of $\{0_2\}^{32i}$ and $\{0_2\}^{32(d/32-i-1)}$ for ease of understanding how the attack works.

⁴ A similar *correcting block attack* on the hash function Khichidi-1 [7] has been reported by Mouha [8].

Algorithm 2 requires that the attacker knows whether $|M_1| = d$ or $|M_1| < d$. However, even without this information the attacker can, at the very least, recover all but the most significant bit of M_1 by simply computing $M_1 = M_1^* \oplus IV$. This is explained as follows: suppose that $|M_1| < d$. Then, by computing $M_1 = M_1^* \oplus IV$ the attacker correctly recovers M_1 . Now suppose that $|M_1| = d$. This time, the attacker is supposed to compute $M_1 = M^*$. Nevertheless, by computing $M_1 = M_1^* \oplus IV$, the attacker is still able to correctly recover $d - 1$ least significant bits of M_1 because the IV is simply $1_2 \parallel \{0_2\}^{d-1}$.

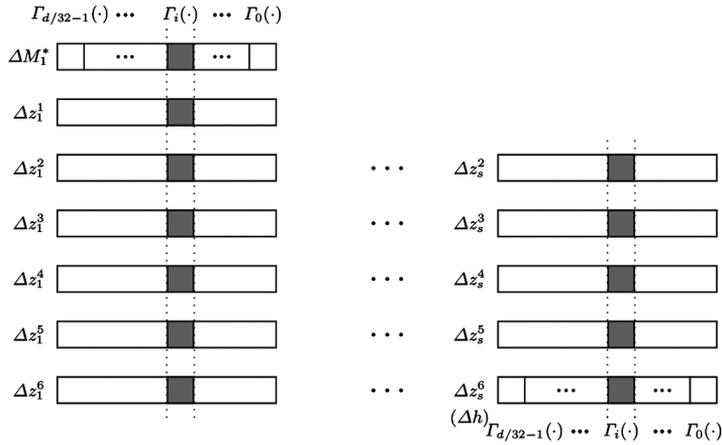


Fig. 4. Differential characteristic for TCS_SHA-3- d when $k = 1$; non-zero differences are confined to the grey boxes.

Algorithm 2. Recovering M_1 from h when $k = 1$

Require: Whether $|M_1| = d$ or $|M_1| < d$

Ensure: d -bit output M_1

1. **for** $i = 0 \rightarrow d/32 - 1$ **do**
 2. **for** $j = \{0_2\}^{32} \rightarrow \{1_2\}^{32}$ **do**
 3. $l \leftarrow \{0_2\}^{32(d/32-i-1)} \parallel j \parallel \{0_2\}^{32i}$;
 4. Compute $\tilde{h} := \text{TCS_SHA-3-}d(l)$;
 5. **if** $\Gamma_i(\tilde{h}) = \Gamma_i(h)$ **then**
 6. $\Gamma_i(M_1^*) \leftarrow j$;
 7. **break**;
 8. **else**
 9. $j \leftarrow j + 1$;
 10. $i \leftarrow i + 1$;
 11. Compute $M_1^* = \Gamma_{d/32-1}(M_1^*) \parallel \Gamma_{d/32-2}(M_1^*) \parallel \dots \parallel \Gamma_0(M_1^*)$;
 12. **if** $|M_1| < d$ **then**
 13. Output $M_1 = M_1^* \oplus IV$;
 14. **else**
 15. Output $M_1 = M^*$;
-

As we can see, Algorithm 2 has $d/32 \cdot 2^{32} = d \cdot 2^{27}$ iterations. Since $d \leq 512$, single block messages can be recovered from their respective hash values in $O(512 \cdot 2^{27}) = O(2^{36})$ time. Therefore, it may be extremely risky to use the TCS_SHA-3 for something like password hashing (a well-known application of cryptographic hash functions, see [9]).

For the case when $k > 1$, if the message blocks M_1, \dots, M_{k-1} are available to the attacker, then M_k may be recovered. The attack procedure is now given by Algorithm 2 with M_1 replaced by M_k and M_1^* by M_k^* . Once again, the attacker is able to recover, at the very least, all but the most significant bit of M_k .

5. Cipher Block Chaining in Algorithm 1: Impact on Security

In Algorithm 1, the 32-bit words $\alpha_1, \alpha_2, \dots, \alpha_{d/32}$ are processed independently of one another (see Fig. 3). This is an inference that we draw from [2], which does not explicitly mention there being any dependence between the processing of α_i and the processing of α_{i+1} , $1 \leq i \leq d/32 - 1$. Furthermore, [2] provides several implementation results, but the corresponding implementation is missing. We were therefore unable to verify the correctness of the implementation results of [2]. If the implementation results are correct, then the processing of α_i and the processing of α_{i+1} , $1 \leq i \leq d/32 - 1$, may not be independent (see e.g., [2]). However, even in such a case, the existence of dependence must have been clearly mentioned in [2] in any of the clauses preceding Clause 0087.

When there is dependence in the form of chaining, given the structural similarities between TCS_SHA-3 and Khichidi-1 (see [7]), it appears reasonable to expect the chaining mechanisms in the two cases to be identical. From [7] then, we see that Fig. 2 changes to Fig. 5 when $k = 2$.^{5,6} Clearly this would also mean that FIGURE 3 of [2] is incorrect.

Note: If one goes by [7], then in Fig. 5, $\Gamma_0(z_1^1)$ is not XORed with $\alpha_{2,1}$, instead $\Gamma_{d/32-1}(c)$ is assigned the value of $\Gamma_0(z_1^1)$ once M_1 is processed. This may be inferred, for example, from the statement, ' $H_0^{(i)} = H_7^{(i)}$ ', in [7] (which, in fact, happens to be the only statement in the Khichidi-1 algorithms of [7] to describe the chaining process in 224-bit Khichidi-1). If, on the other hand, the statement had read ' $H_0^{(i+1)} = H_7^{(i)}$ ', then the chaining process would have appeared meaningful. Therefore, we presume that the chaining statements in the Khichidi-1 algorithms of [7] are typographically flawed in the manner described above.

However, the case of the aforementioned independent processing of the 32-bit blocks complies with FIGURE 3 of [2] and thus enhances our belief in the correctness of the above-mentioned inference of independent processing of the α_i 's ($1 \leq i \leq d/32$). Yet we shall now examine the impact of cipher block chaining in Algorithm 1.

⁵ The bijective function $g: \{0_2, 1_2\}^{32} \rightarrow \{0_2, 1_2\}^{32}$ of Fig. 5 is given by Algorithm 3.

⁶ See Fig. 5. Clause 0069 of [2] states that $|c| \leq d$. If $|c| = e < d$, then $c \leftarrow \{0_2\}^{d-e} \parallel c$. We find no mention in [2] that it is the most significant word of c that is XORed with $\alpha_{1,1}$; we simply make such an assumption without the loss of generality.

Algorithm 3. The bijective function $g: \{0_2, 1_2\}^{32} \rightarrow \{0_2, 1_2\}^{32}$

Require: 32-bit input α

Ensure: 32-bit output λ

1. *Shuffle:* $\alpha \rightarrow \beta$ such that $\beta_{(j)} = \alpha_{(j/2)}$ if $2 \mid j$ and $\beta_{(j)} = \alpha_{(16+(j-1)/2)}$ otherwise;
2. *Apply T-function:* $\beta \rightarrow \gamma := 2\beta^2 + \beta \pmod{2^{32}}$;
3. *Apply LFSR:* $\gamma \rightarrow \lambda$ such that $|\lambda| = 32$;

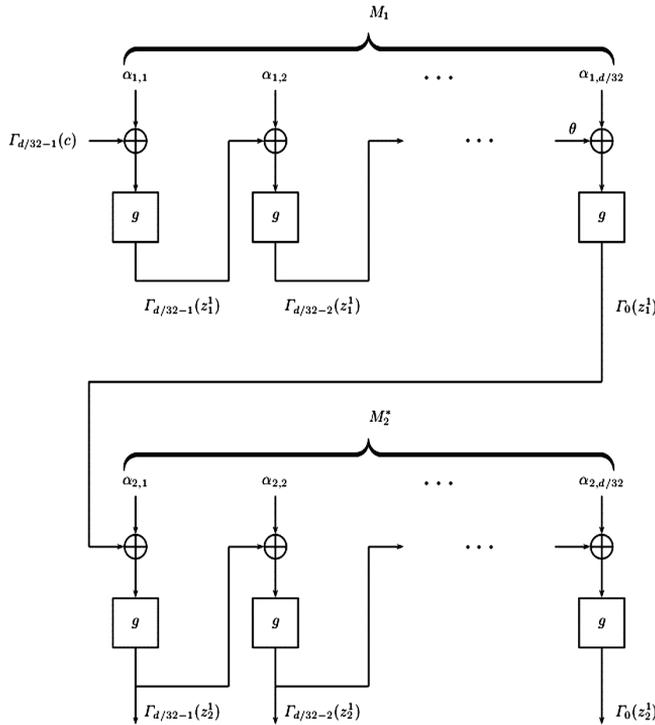


Fig. 5. Round 1 of TCS_SHA-3 ($k = 2$) with 32-bit cipher block chaining; $\alpha_{1,j} = \Gamma_{d/32-j}(M_1)$ and $\alpha_{2,j} = \Gamma_{d/32-j}(M_2^*)$, for all $j \in \{1, 2, \dots, d/32\}$.

Let $M = M_1 \parallel M_2 \parallel M_3 \parallel \dots \parallel M_k^*$, $k \geq 2$ (when $k = 2$, $M = M_1 \parallel M_2^*$; in general, when $k = l > 2$, $M = M_1 \parallel M_2 \parallel \dots \parallel M_l^*$), denote the message and h its hash value. Let $M' = M'_1 \parallel M'_2 \parallel M_3 \parallel \dots \parallel M_k^*$, with $M'_1 \neq M_1$ and $M'_2 \neq M_2$, denote another message and h' its hash value. We now define the following for all $j \in \{1, 2, \dots, d/32\}$:

$$\begin{aligned} \alpha_{i,j} &:= \Gamma_{d/32-j}(M_i) \quad \text{when } i < k, \\ \alpha_{i,j} &:= \Gamma_{d/32-j}(M_k^*) \quad \text{when } i = k. \end{aligned} \tag{9}$$

Suppose that the conditions $M'_1 \neq M_1$ and $M'_2 \neq M_2$ are such that $\alpha_{1,i} = \alpha'_{1,i}$ for all $i \in \{1, 2, \dots, d/32 - 1\}$ and $\alpha_{2,i} = \alpha'_{2,i}$ and for all $i \in \{2, \dots, d/32\}$. Then, from Fig. 5, it follows that $h' = h$ when the following condition is satisfied:

$$g(\alpha_{1,d/32} \oplus \theta) \oplus \alpha_{2,1} = g(\alpha'_{1,d/32} \oplus \theta) \oplus \alpha'_{2,1}. \quad (10)$$

It is straightforward to see that the conditions:

$$\alpha'_{2,1} = g(\alpha'_{1,d/32} \oplus \theta) \oplus \alpha'_{1,d/32}, \quad (11)$$

$$\alpha'_{1,d/32} = \alpha_{2,1} \oplus g(\alpha_{1,d/32} \oplus \theta), \quad (12)$$

satisfy (10) and when $\alpha'_{1,d/32} \neq \alpha_{1,d/32}$, we have a second preimage. Under reasonable assumptions of uniformity, the event $\alpha'_{1,d/32} = \alpha_{1,d/32}$ occurs with negligible probability.

6. Conclusions and Open Problems

In this paper, we have presented what we see as being the *first* practical (second) preimage attacks on the TCS_SHA-3 family of patented cryptographic hash functions. The second preimage attack requires negligible time and negligible memory for nearly guaranteed success. The attack works when the number of message blocks is at least two. The first preimage attack requires $O(2^{36})$ time and negligible memory. This attack is most efficient (going by data requirements) on single block messages -- negligible data is required in such cases. We have also reported a negligible time/memory second preimage attack on the TCS_SHA-3 that is strengthened with 32-bit cipher block chaining. This attack also works only when the number of message blocks is at least two.

Our findings establish, amongst other things, that the TCS_SHA-3 may be particularly unsuitable for password hashing (unless, for example, it is strengthened with 32-bit cipher block chaining).

It may be an interesting exercise to find countermeasures to our attacks.

References

- [1] B. Preneel, "Analysis and design of cryptographic hash functions," PhD dissertation, Katholieke Universiteit Leuven, Belgium, 1993.
- [2] N. Vijayarangan, "Method for designing a secure hash function and a system thereof," U.S. Patent 20090262925, Oct 22, 2009.
- [3] National Institute of Standards and Technology, "Cryptographic hash algorithm competition," <http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>.
- [4] Tata Consultancy Services, "TCS annual report 2011-2012," http://www.tcs.com/investors/Documents/Annual%20Reports/TCS_Annual_Report_2011-2012.pdf.
- [5] Tata Consultancy Services, "TCS recognized as Big Four IT Services brand," http://www.tcs.com/news_events/press_releases/Pages/TCS_recognized_Big_Four_IT_Services_brand.aspx.
- [6] Tata Consultancy Services, "TCS annual report 2007-2008," http://www.tcs.com/investors/Documents/Annual%20Reports/TCS_Annual_Report_2007_2008.PDF.
- [7] N. Vijayarangan, "A new hash algorithm: Khichidi-1," 2008; <http://ehash.iaik.tugraz.at/uploads/d/d4/Khichidi-1.pdf>.

[8] N. Mouha, “Automated techniques for hash function and block cipher cryptanalysis,” Ph.D. dissertation, Katholieke Universiteit Leuven, Belgium, 2012.

[9] B. Preneel, “The state of cryptographic hash functions,” in *Lectures on Data Security*. Heidelberg: Springer, 1999, pp. 158-182.

[10] National Institute of Standards and Technology, “Khichidi-1 package,” <http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Khichidi-1.zip>.

Appendix A. Experiments

We took a few sample outputs of TCS_SHA-3 and attempted to verify their first preimages. Simulations were performed on a NVIDIA GeForce GT 540M graphics processing unit (GPU) having 96 CUDA cores (2 multiprocessors × 48 CUDA cores / multiprocessor) and a clock rate of 1.34 GHz. The CUDA C compiler nvcc 4.2 was used. Below is a list of assignments made for the simulations and their justifications (see also footnotes 1–3 and Section 4).

Table 2. First preimage samples

| Digest length (bits) | Output of hash function (in hex) | Preimage | Time taken (second) |
|----------------------|--|---|---------------------|
| 224 | F18DE455 827C1EE6 00000000 00000000 00000000 00000000 00000000 | ‘password’ | 28.66 |
| 256 | F18DE455 827C1EE6 00000000 00000000 00000000 00000000 00000000 00000000 | ‘password’ | 28.67 |
| 384 | F18DE455 827C1EE6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 | ‘password’ | 28.74 |
| 512 | F18DE455 827C1EE6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 | ‘password’ | 28.80 |
| 224 | 2C2B32B7 305056B5 709FF3A9 7A99955B 3D271585 78E21E7E 2848DD92 | ‘The quick brown fox jumps’ | 91.28 |
| 256 | 2C2B32B7 305056B5 709FF3A9 7A99955B 3D271585 78E21E7E CC1779EB A3ED2089 | ‘The quick brown fox jumps over’ | 103.80 |
| 384 | 2C2B32B7 305056B5 709FF3A9 7A99955B 3D271585 78E21E7E CC1779EB 37407107 89D12C8B EA6630B4 3929A26F 00000000 | ‘The quick brown fox jumps over the lazy dog’ | 141.08 |
| 512 | 2C2B32B7 305056B5 709FF3A9 7A99955B 3D271585 78E21E7E CC1779EB 37407107 89D12C8B EA6630B4 3929A26F 00000000 00000000 00000000 00000000 00000000 | ‘The quick brown fox jumps over the lazy dog’ | 141.14 |

- See footnote 6. As the value of c has no bearing on the analysis or the experiments, we assign the value $\{0_2\}^d$ to c .
- The IV is $1_2 \parallel \{0_2\}^{d-1}$ (see Section 2).
- The value of s is taken to be the value of s in Khichidi-1 (see [7] and *khichidi.c* in the URL provided in [10]), and is therefore equal to 2.
- The value of η_i is 1 for all $i \in \{1, 2, \dots, 6\}$; the number of LFSR shifts in round i of Khichidi-1 is also 1 for all $i \in \{1, 2, \dots, 6\}$ (see [7] and *khichidi.c* in the URL provided in [10]).

The system time taken to find the first preimage for each output, given the above set of assignments, is provided in Table 2. These system times are expected to remain unaltered when a different IV or c is used.

As shown in Sections 2 and 4, it follows that the time/memory complexity of our first preimage attack does not change when one or more of the above-listed assignments are altered.



Gautham Sekar

He received his bachelor's degree in Electronics and Instrumentation Engineering from the Birla Institute of Technology and Science, Pilani, in June 2006. He simultaneously graduated with a master's degree in Physics from the same institute. In November 2006, he joined the COSIC (Computer Security and Industrial Cryptography) group of the Katholieke Universiteit Leuven, Belgium, as a pre-doctoral student and from October 2007 to March 2011 he worked there as a doctoral student under the supervision of Prof. Dr. Bart Preneel. Upon obtaining his Ph.D., he worked at the National University of Singapore as a Research Scientist until March 2012. Since then, he is a Visiting Assistant Professor at the Indian Statistical Institute, Chennai Centre. In May 2007, he received the Dr. Ranjit Singh Chauhan Undergraduate Research Award from the Birla Institute of Technology and Science, Pilani. His research interests include cryptography, cryptanalysis, and information security in general.



Soumyadeep Bhattacharya <http://orcid.org/0000-0002-9386-7513>

He received his bachelor's degree in Electrical and Electronics Engineering and master's degree in Physics from the Birla Institute of Technology and Science, Pilani, in June 2010. He is currently a Senior Research Fellow in Theoretical Physics at The Institute of Mathematical Sciences, Chennai. His research interests are statistical physics and parallel computing.