The ISC Int'l Journal of Information Security
ISeCure

January 2020, Volume 12, Number 1 (pp. 13–23)
http://www.isecure-journal.org

Investigation of Some Attacks on GAGE (v1), InGAGE (v1), (v1.03), and CiliPadi (v1) Variants

Majid M. Niknam [1], Sadegh Sadeghi [1], Mohammad Reza Aref [2], and Nasour Bagheri [3,4,*]

[1] *Department of Mathematics, Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran*
[2] *Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran*
[3] *Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran*
[4] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 24 August 2019
Revised: 2 November 2019
Accepted: 31 December 2019
Published Online: 8 January 2020

Keywords:
NIST lightweight cryptography competition, Preimage attack, Integrity, Confidentiality, Key recovery, MILP.

Abstract

In this paper, we present some attacks on GAGE, InGAGE, and CiliPadi, which are candidates of the first round of the NIST-LWC competition. GAGE and InGAGE are lightweight sponge based hash function and Authenticated Encryption with Associated Data (AEAD), respectively, and support different sets of parameters. The length of hash, key, and tag are always 256, 128, and 128 bits, respectively. We show that the security bounds for some variants of its hash and AEAD are less than the designers' claims. For example, the designers' security claim of the preimage attack for a hash function when the rate is 128 bits, and the capacity is 256 bits, is 2^{256}. However, we show that the security of preimage for this parameter set is 2^{128}. Also, the designer claimed security of confidentiality for an AEAD, when the rate is 8 bits, and the capacity is 224 bits, is 2^{116}. However, we show the security of confidentiality for it is 2^{112}. We also investigate the structure of the permutation used in InGAGE and present an attack to recover the key for reduced rounds of a variant of InGAGE. In an instance of AEAD of InGAGE, when the rate is 8 bits and the capacity is 224 bits, we recover the key when the number of the composition of the main permutation with itself, i.e., r_1, is less than 8. We also show that CiliPadi is vulnerable to the length extension attack by presenting concrete examples of forged messages.

1 Introduction

A cryptographic hash function maps any message of arbitrary length to a string of specific length, e.g., n bits, where the output string is known as the message digest or hash value. More formally, we can define a hash function as follows:

$$H : \{0,1\}^* \to \{0,1\}^n$$

Three main criteria for a secure cryptographic hash function are preimage resistant, second-preimage resistant, and collision-resistant. Among them, preimage attack means that given any $h \in \{0,1\}^n$, which is an image of H, the attacker should find a $M \in \{0,1\}^*$

* Corresponding author.
Email addresses: mmniknam@gmail.com, s.sadeghi.khu@gmail.com, aref@sharif.edu, Nbagheri@srttu.edu, na.bagheri@gmail.com
ISSN: 2008-2045

such that we must ensure $H(M) = h$. Also, for an ideal hash function, which is modeled as a random oracle, the expected complexity of finding a preimage for an n-bit hash function is 2^n.

An AEAD scheme is an Authenticated Encryption with Associated Data, which takes a plaintext of arbitrary length upper-bounded to a fixed value, a key, and a nonce and gives a ciphertext and a tag. Its encryption function is as follows where K, N, A, P, C, T are key, nonce, associated data, plaintext, ciphertext, and tag, respectively:

$$E(K, N, A, P) = (C, T).$$

Its decryption function is as follows:

$$D(K, N, A, C, T) = (P, \perp).$$

If the Tag is verified, then decryption function returns the plaintext P, otherwise, it returns the symbol $\perp$. Depending on the security assumption of the scheme, an AEAD mode may allow returning the plaintext even before evaluating the Tag. There are many cryptographic hash functions or AEAD schemes(e.g., SHA1 [1], SHA2 [2], SHA3 [3], PHOTON [4], Quark [5], Trivia [6], Helix [7], and many others), but the majority of them have been designed for desktop or server environments, so most of them are not suitable for constrained devices. Thereby NIST has started a competition for LWC. They published the requirements for AEAD and Hash functions on August 27, 2018 [8]. NIST received 57 schemes to be considered for standardization until the deadline for submission on February 29, 2019. They announced 56 candidates for round 1 on April 18, 2019. The candidates for round 2 announced on September 9, 2019, and they accepted 32 candidates for round 2. We considered some of the 56 schemes, and we found some weaknesses in two of the schemes. We analyze the security of GAGE [9](v1) and InGAGE [10](v1.01) [11](v1.03) and CiliPadi [12] which are two schemes of the 56 candidates for round 1. On the other hand, GAGE and InGAGE are interesting for the size of their s-box, which is very big: 232×232. To the best of our knowledge, our work is the first one to analysis these two schemes. NIST did not accept these two schemes for the second round because of the existing third-party analysis on them that raised security concerns during the first round of the process [8].

1.1 Our Contribution

Our contributions in this work are as follows and are shown in table 1:

(1) We introduce a preimage attack on some variants of GAGE, and we show that the bound of security for them by designers' claim is incorrect.

(2) We introduce an integrity attack on two variants of InGAGE, and we show that the bound of real security for them is less than the designers' claims for them.

(3) We introduce a confidentiality attack on one variant of InGAGE, and we show that its real security is less than the designers' claim.

(4) We introduce an attack for key recovery of one variant of InGAGE when the number of iteration of the composition of the main permutation Q with itself is reduced, and the security bound for it is less than the length of the key.

(5) We present practical forgery attack against CiliPadi, thanks to the flaw in its padding approach.

Table 1. Our contribution.

Attack	Scheme	Refer to
Preimage	Some variants of GAGE	Sec. 2.2,Table 2
Integrity	Two variants of InGAGE	Sec. 3.2, Table 4
Confidentiality		Sec. 3.3, Table 5
Key recovery	A variant of InGAGE	Sec. 3.4, Table 6
Length extension	Cilipadi	Sec. 4.2, Table 7

The rest of the paper is organized as follows: in Section 2, we present a brief description of GAGE and CiliPadi. In Section 3, we present our finding against the security of GAGE and InGAGE. Section 4, presents a forgery attack on CiliPadi. Finally, the paper is concluded in Section 5

2 Preliminaries

In this section, we give some notations that are used in the rest of the paper and then give a brief description of GAGE and InGAGE [9](v1) and [11](v1.03). We will give a brief description of CiliPadi(v1) [12] in Section 4.1.

2.1 Notations

In this paper, the logical operation XOR and the concatenation of x and y are referred to as $\oplus$, and $x||y$, respectively. The logical operation "and" of two string with the same length is denoted by $\wedge$. Also, all `0xi` symbols are hexadecimal of `i`, and we may omit the `0x` symbol.

2.2 A Brief Description of GAGE an INGAGE

GAGE [9](v.1) uses sponge [13] based construction to produce a 256-bit hash value for any given message M. The input message is padded by a string $\{80\|00^*\}$; at first, however, it has no impact on the proposed preimage attacks in this work. GAGE supports different parameter sets that provide different levels of security. A variant of this scheme has the rate $r = 128$ bits, the capacity $c = 256$ bits, the state $b = r + c = 384$ bits and produces outputs of length $n = 256$ bits. For this variant, the security claim against preimage attack is 2^{256}. Given the message M is padded as $M_{pad} = M_0\|M_1\|\ldots\|M_{l-2}\|M_{l-1}$ and the permutation $Q : \{0,1\}^b \rightarrow \{0,1\}^b$, where Q^{32} denotes the composition of Q with itself 32 times, a brief representation of this scheme is depicted in Figure 1 and works as follows, where $\perp$ denotes an empty string:

(1) $M_{pad} \rightarrow M_0\|M_1\|\ldots\|M_{l-2}\|M_{l-1}$
(2) $(S = S_r\|S_c) \leftarrow 0$
(3) $H(M) \leftarrow \perp$:
(4) **Absorbing Phase:** for $0 \leq i \leq l-1$ do:
  (a) $(S = S_r\|S_c) \leftarrow (S_r \oplus M_i)\|S_c$
  (b) $(S = S_r\|S_c) \leftarrow Q^{32}(S)$
(5) **Squeezing Phase:** for $0 \leq i \leq \frac{n}{r} - 1$ do:
  (a) $(S = S_r\|S_c) \leftarrow Q^{32}(S)$
  (b) $H(M) \leftarrow H(M)\|S_r$
(6) return $H(M)$

Given that for the target parameter set $n = 2 \times r$, to produce the hash value we need to call the permutation function 2 times in the squeezing phase.

In Table 2, the security claim for different parameter sets and our bound for them are presented. In the next section, we describe a preimage attack against a variant of GAGE, when $r = 128$ and $c = 256$, i.e., the parameter set #8 in Table 2.

Table 2. The claimed preimage security of all instances of GAGE [9], where for all of them $|Hash| = n = 256$ and the maximum message length is expected to be less than 2^{64} and our bounds for the security of each variant (details will be presented in Section 2).

#	b	c	r	Preimage security	Ref.
1	232	224	8	223	[9, Sec. 1.2]
				112	Sec. 2
2	240	224	16	223	[9, Sec. 1.2]
				112	Sec. 2
3	256	224	32	223	[9, Sec. 1.2]
				112	Sec. 2
4	288	224	64	223	[9, Sec. 1.2]
				192	Sec. 2
5	272	256	16	256	[9, Sec. 1.2]
				240	Sec. 2
6	288	256	32	256	[9, Sec. 1.2]
				224	Sec. 2
7	320	256	64	256	[9, Sec. 1.2]
				192	Sec. 2
8	384	256	128	256	[9, Sec. 1.2]
				128	Sec. 2
9	544	512	32	256	[9, Sec. 1.2]
				256	Sec. 2
10	576	512	64	256	[9, Sec. 1.2]
				256	Sec. 2

InGAGE [9](version 1) is an AEAD built on sponge-based construction. It has some different instances, and the sets of their parameters and security claims are given in InGAGE [9, Subsec. 2.2], as depicted in Table 3. The plain-text and associated data can be empty, this means it is possible that $|P| = 0$ or $|A| = 0$. Given plain-text P, associated data A, key K, and nonce N, first, the associated data and plain-text are padded. Figure 3 shows a brief representation of how encryption and decryption of this scheme work, where for the permutation $Q : \{0,1\}^b \rightarrow \{0,1\}^b$ we have $P^{r_1} = Q^{32}, P^{r_2} = Q^{16}$.

3 Security Analysis of GAGE and InGAGE (v1,v1.03)

In this section, we present the attacks against GAGE and InGAGE variants.

3.1 Preimage Attack

Almost similar to the analysis already used to prove the security of a SPONGE based hash function by Guo *et al.* in [14, 15], given a valid $h = H_0\|H_1$, to find a preimage in GAGE, when $r = 128$ and $c = 256$, where $(Q^{32})^{-1}$ denotes the inverse of the permutation Q^{32} and $\{0\}^t$ denotes a t-bit zero string , an adversary may do as follows: choose a random string of $\{0,1\}^{128}$ for S_c and compute $S_r^{-1}\|S_c^{-1} = (Q^{32})^{-1}(S_r\|S_c)$ until $S_r^{-1} = H_0$. We expected there exists such string for S_c, because the length of S_c is equal to the length of S_r^{-1}.The attack procedure is also represented in Figure 2. After that go backward two steps to compute $S_r^{-2}\|S_c^{-2} = (Q^{32})^{-1}(S_r^{-1}\|S_c^{-1})$ and $S_r^{-3}\|S_c^{-3} = (Q^{32})^{-1}(S_r^{-2}\|S_c^{-2})$. Then chose a random string $m \in \{0,1\}^{120}$ and put $M_3 = m\|80$. Compute $S_r^{-4}\|S_c^{-4} = (Q^{32})^{-1}((S_r^{-3} \oplus M_3)\|S_c^{-3})$ and for all value $i \in \{0,1\}^{128}$ compute $S_r^i\|S_c^i = (Q^{32})^{-1}((S_r^{-4} \oplus i)\|S_c^{-4})$ and save the pair $(S_r^i\|S_c^i, i)$ in a table T_{rev}. Then for all value $j \in \{0,1\}^{128}$ compute $S_r^j\|S_c^i = Q^{32}((\{0\}^{128} \oplus j)\|\{0\}^{256})$ and save the pair $(S_r^i\|S_c^j, j)$ in a table T_{dir}. Now find an i and a j such that $S_c^i = S_c^j$ where $(S^i = S_r^i\|S_c^i, i) \in T_{rev}$ and $(S^j = S_r^j\|S_c^j, j) \in T_{dir}$. By using these conditions for the message $M = j\|(S_r^i \oplus S_r^j)\|i\|M_3$ the hash of M is h. The pseudo code of the attack is as follows:

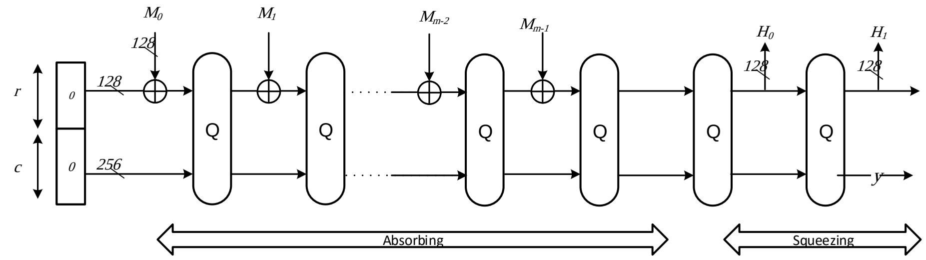

Figure 1. The hash mode of GAGE when the rate is 128 bits and the capacity is 256 bits; here Q denotes Q^{32}[9].

Table 3. All instances of InGAGE [9] or [11].

#	$\|K\|$	$\|N\|$	$\|T\|$	b	c	r	Confidentiality	Integrity of P	Integrity of A	Nonce reuse	Message size limit (Power of 2 bytes)
1.	128	96	128	232	224	8	116	128	128	No	64
2.	128	96	128	240	224	16	120	128	128	No	64
3.	128	96	128	256	224	32	128	128	128	No	64
4.	128	128	128	320	256	64	128	128	128	No	64
5.	128	96	128	512	448	64	256	128	128	No	64
6.	256	128	128	512	448	64	256	128	128	No	64

(1) $S_r \leftarrow H_1$
(2) $S_r^{-1} \leftarrow \perp$
(3) while $(S_r^{-1} \neq H_0)$ *and* $((S_r^{-1}) \wedge 0x \notin \{\})$:
   (a) $S_c \xleftarrow{\$} \{0,1\}^{256}$
   (b) $S_r^{-1} \| S_c^{-1} \leftarrow (Q^{32})^{-1}(S_r \| S_c)$
(4) $S_r^{-2} \| S_c^{-2} \leftarrow (Q^{32})^{-1}(S_r^{-1} \| S_c^{-1})$
(5) $S_r^{-3} \| S_c^{-3} \leftarrow (Q^{32})^{-1}(S_r^{-2} \| S_c^{-2})$
(6) $M_3 \xleftarrow{\$} \{0,1\}^{120} \| 80$
(7) $S_r^{-4} \| S_c^{-4} \leftarrow (Q^{32})^{-1}((S_r^{-3} \oplus M_3) \| S_c^{-3})$
(8) for $0 \leq i \leq 2^{128} - 1$ do:
   (a) $(S) \leftarrow (S_r^{-4} \oplus i) \| S_c^{-4}$
   (b) $T_{rev} \xleftarrow{Stored\ in\ a\ Table} (S^i = (Q^{32})^{-1}(S), i)$
(9) for $0 \leq j \leq 2^{128} - 1$ do:
   (a) $(S) \leftarrow (\{0\}^{128} \oplus j) \| \{0\}^{256}$
   (b) $T_{dir} \xleftarrow{Stored\ in\ a\ Table} (S^j = Q^{32}(S), j)$
(10) find a record $(S^i = S_r^i \| S_c^i, i) \in T_{rev}$ and a record $(S^j = S_r^j \| S_c^j, j) \in T_{dir}$ such that $S_c^i = S_c^j$.
(11) return $M = j \| (S_r^i \oplus S_r^j) \| i \| M_3$.

Given that the tables T_{rev} and T_{dir}, each has the size 2^{128} and $|S_c| = 256$, we expect to find a matching in Step 10. Finding such matching, the rest of the attack will be straight forward. The attack complexity is dominated by Steps 3, 9, and 8, each having the complexity of 2^{128} calls to the underlying permutation Q^{32} or its reverse $(Q^{32})^{-1}$. On the other hand, given any $M \neq \perp$, calculating the hash value costs at least three calls to Q^{32}, for the target parameter set. Hence the total complexity is of the order 2^{128} calculations of the hash value of a message. Following this attack, the designers have changed these security bounds and announced it on page 4 of GAGE and InGAGE document [10].

Remark 1. It is possible to extend the proposed attack against other variants of GAGE also. However, the complexity will be more than 2^{128}, although it could be less than the claimed security by the designer, as it has been reported in Table 2. For instance, when $r = 64$ and $c = 320$, i.e., parameters set number 7, it is possible to adapt the present attack and find preimage with the complexity of 2^{192}. In general, the preimage complexity of any variant is upper-bounded by $min[c, n, max(\frac{c}{2}, (\frac{n}{r} - 1) \times r)]$, also pointed out in an independent work by Guo *et al.* [15].

3.2 Integrity Attack

From Table 3, we can see in variants number 1. and 2. of GAGE [9] we have $b - |T| < |T|$. In this situation, for an arbitrary plaintext P and associated data A (which their lengths are multiply of 8 in bits) when $|A| \geq (b - |T| - 16)$, an adversary can obtain a valid (C, T). We denote the state after initializing and the states after it by $S_0, \cdots, S_n$ where S_0 is the state after XORing the key to the state after initializing state and S_n is the final state (the state that the tag is extracted from it) Figure 4. The attack works as follows:

(1) Assume $P' = \perp$ and produce a valid cipher and

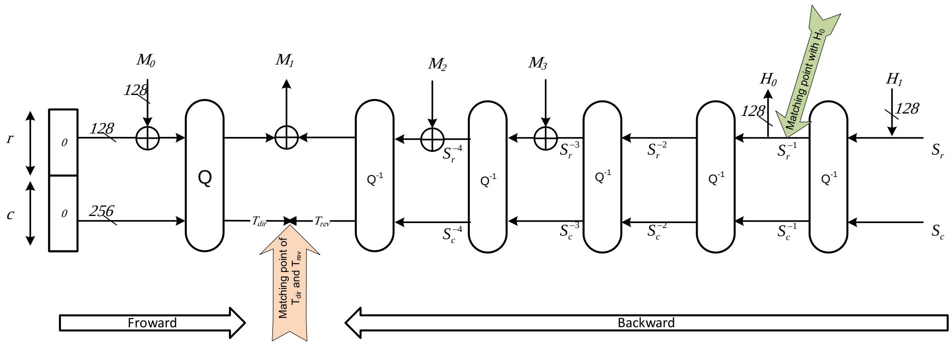

Figure 2. Illustration of the proposed preimage attack on GAGE when the rate is 128 bits and the capacity is 256 bits

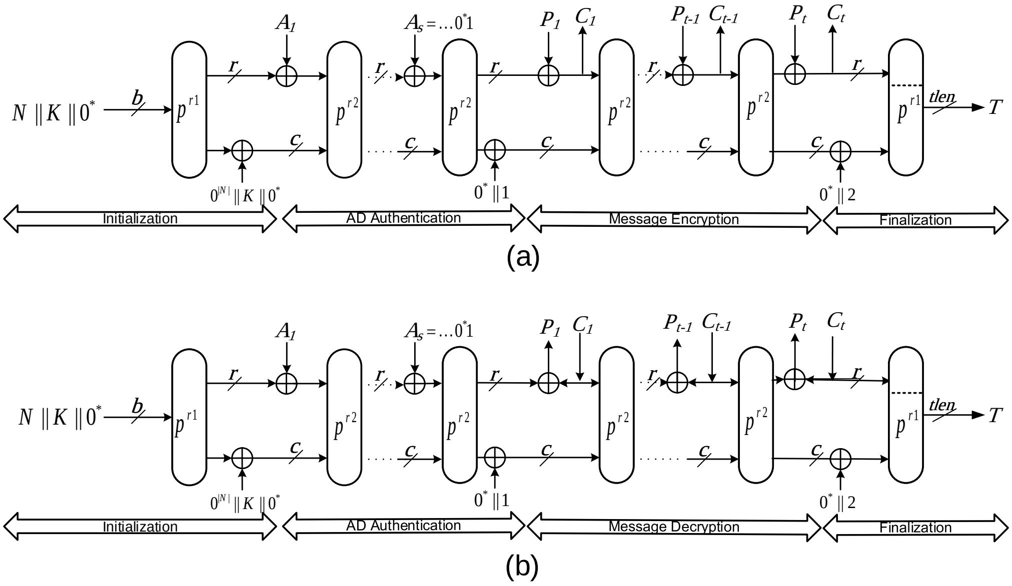

Figure 3. (a): Encryption and (b): Decryption of InGAGE [9] or [11].

its tag (C', T') for (K, N, A, P').

(2) By the padding method we have $|A_{pad}| \geq |A|+8$ and $|P'_{pad}| = 8$. Notice that $|A_{pad}|+|P'_{pad}| \geq |A_{pad}|+8+8 \geq b-|T|$.

(3) Guess the remaining $(b-|T|)$ bits of the final state S_n and with padded plaintext P'_{pad}, associated data A_{pad}, calling $(p^{r_1})^{-1}$, and $(p^{r_2})^{-1}$ go backward and omit wrong guesses to recover the final state S_n.

(4) Calculate $(p^{r_1})^{-1}(S_n)$ to recover the state S_{n-1} and then calculate $(p^{r_2})^{-1}(S_{n-1})$ to recover the state S_{n-2}.

(5) Start with S_{n-2} and after padding an arbitrary plaintext P go forward to produce a valid cipher and its tag (C, T) for (K, N, A, P).

(6) The complexity of the attack is $2^{b-|T|}$ which for the variant number 1 of InGAGE, the complexity is 2^{104} and for the variant number 2 of InGAGE it will be 2^{112}.

Thereby, the security bounds of integrity for these two instances are less than what the designers claimed. They have represented them in Table 3. The results are shown in Table 4. It should be noted that, following this attack, the designers have changed this security bound and announced it on page 4 of GAGE and InGAGE document [10].

Remark 2. Notice that in the above attack, we do not query from encryption oracle with a fixed nonce more than once, so we followed the nonce respecting assumption by the designers.

Remark 3. We can use this attack to produce a ciphertext and its tag for two arbitrary associated data and plaintext A_1, P_1. For this purpose, we start from known S_n in the encryption of (N, K, A, P) and by calling $(p^{r_1})^{-1}$ and $(p^{r_2})^{-1}$ go backward to recover S_0 then by stating from S_0 and going forward we can produce the ciphertext and its tag (C_1, T_1) for

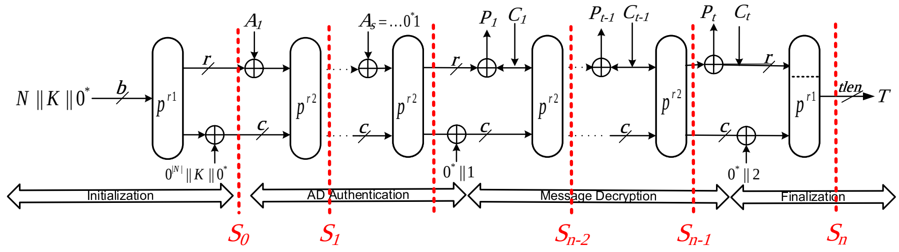

Figure 4. The position of states $S_0, \cdots, S_n$ in InGAGE.

Table 4. Our bounds for integrity in variants number 1. and 2.

#	$\|K\|$	$\|N\|$	$\|T\|$	b	c	r	Our bound	Designers' bounds
							P - A	P - A
1.	128	96	128	232	224	8	104 - 104	128 - 128
2.	128	96	128	240	224	16	112 - 112	128 - 128

Table 5. Our bounds for confidentiality in variants number 1. and 2.

#	$\|K\|$	$\|N\|$	$\|T\|$	b	c	r	Our bound of P	Designers' bound
1.	128	96	128	232	224	8	112	116

(N, K, A_1, P_1).

3.3 Confidentiality Attack

In this section, we introduce an attack to recover the plaintext in instance number 1 of InGAGE version (v1.03) [11], whose parameters are presented in Table 3. The attack works as follows:

(1) Assume $A, |A| \geq (b - |T| - 24)$, is an associated data and P is a plaintext with only one byte, that is $|P| = 8$. For a nonce N and a key K, suppose $E(K, N, A, P) = (C, T)$.
(2) BY the padding method we have $P_{pad} = P \| 80$. Therefore $|P_{pad}| = 16, |A_{pad}| \geq |A| + 8$ so $|A_{pad}| + |P_{pad}| \geq |A_{pad}| + 8 + 16 \geq b - |T|$.
(3) We have the tag T. Guess the remaining bits of the final state S_n Figure 4 and the eight bits of the plaintext P. By calling $(p^{r_1})^{-1}, (p^{r_2})^{-1}$ and padded plaintext P_{pad} and associated data A_{pad} in the backward side, omit wrong guesses to recover the plaintext P.
(4) The complexity of the attack is $2^{(232-128)+8} = 2^{112}$.

Therefore, the designer's claimed bound in InGAGE [11] for the confidentiality of this variant of InGAGE must be modified. The result is shown in Table 5.

Remark 4. The above attack can be developed for the instance 1 of InGAGE version (v1.03). Suppose $A, |A| \geq (b - |T| - 24)$, is an associated data and $P = P_0 \| P_1$ is a plaintext such that the P_1 is only one byte and both of A, P_0 are some bytes. By using a method like the above attack we can find the suffix P_1 of plaintext with complexity 2^{112}. Like the above attack we do:

(1) $P_{pad} = P_0 \| P_1 \| 80$ so $|P_1 \| 80| = 16, |A_{pad}| \geq |A| + 8$ and then
$|A_{pad}| + |P_1 \| 80| \geq |A_{pad}| + 8 + 16 \geq b - |T|$.
(2) We have the tag T. Guess the remaining bits of the final state S_n and the eight bits of the plaintext P_1. By calling $(p^{r_1})^{-1}, (p^{r_2})^{-1}$ and the end part of plaintext $P_1 \| 80$ and associated data A_{pad} in the backward side, omit wrong guesses to recover the plaintext P_1. Notice in the backward side, when we reach a state whit a byte of P_0 XOR with it, we ignore that state and by calling $(p^{r_2})^{-1}$ we go to the state before it. We continue this way until we reach the states which the bytes of A_{pad} XOR with them, and after that we use these states to omit the wrong guesses.
(3) The complexity of this attack is $2^{(232-128)+8} = 2^{112}$.

3.4 Key Recovery Attack for Reduced Rounds of Iteration

From table 3 in InGAGE [11] for variants number 1. and 2., it can be seen that we have $b - |T| < |K|$. We introduce an attack to recover the key in these two variants of InGAGE when the number of iteration r_1 is less than 9 instead of its real value which is 32. It works as follows:

(1) Suppose A is an associated data which $|A| \geq b - |T| - 16, P = \perp$, therefore, $P_{pad} = 80$. So by the method of padding $|A_{pad}| \geq |A| + 8, |P_{pad}| = 8$ and then $|A_{pad}| + |P_{pad}| \geq b - |T|$.
(2) Guess the remaining $b - |T|$ bits of the final state S_n Figure 4 and by using padded associated data and plaintext and calling $(p^{r_1})^{-1}, (p^{r_2})^{-1}$, in backwards side omit the wrong guesses to find the final state S_n. It is possible because of the

length of padded associated data and plaintext is greater than or equal to the number of guessing bits. Then again by calling $(p^{r_1})^{-1}, (p^{r_2})^{-1}$, go backward to find the state S_0.

(3) By knowing S_0 solve next equation

$$p^{r_1}(N\|K\|0^8) \oplus (0^{|N|+r}\|K) = S_0 \qquad (1)$$

to find the key K. The number of iteration for composition of the permutation Q with itself is $r_1 = 32$ in InGAGE [11]. We cannot solve this equation for real value of $r_1 = 32$ yet. Therefore we solve it for the reduced number of $r_1 < 8$.

(4) We obtained an MILP (Mixed Integer Linear Programing) model [16], for solving equation 1 by using Gurobi software [17], (some details of our MILP model has come in Appendix). To check our program we choose a random nonce N and key K, then calculate the state S_0 for these N, K and a $r_1 \leq 9$, and after that we solve equation 1 with this S_0 to find the key K again. In our experiments all the time, we got the key K only and we didn't find any other keys which satisfy equation 1 with a fixed S_0. We used a personal computer (Intel Core (TM)i-7, 8 Gig RAM, Windows 10, x64) and the results are shown in Table 6.

(5) The complexity of this attack is related to guessing 104 bits of state S_n and omitting the wrong guesses. To omit every wrong guess we call $(p^{r_1})^{-1}$ one time and $(p^{r_2})^{-1}$ several times. The cost of these calling is less than the cost of encryption of the plaintext with its associated data. Therefore, the total cost of the attack is of the order 2^{104} calculation of the encryption of a message with its associated data. 2^{104} is less than the length of the key in both variants 1. and 2. of InGAGE.

4 Length Extension Attack on CiliPadi

4.1 A Brief Description of CiliPadi

In this section, we describe a family of lightweight authenticated encryption with associated data called CiliPadi (v.1) [12].

The CiliPadi$[n, r, a, b]$ mode of operation is based on the MonkeyDuplex construction and it consists of four phases: initialization, associated data authentication, message encryption/decryption, and finalization that is shown in Figure 5.

The key K and nonce N construct an $n-$bit value which is used to initialize the mode of operation. The bit-rate of this scheme is r bits and the capacity is $c = n - r$ bits. The permutation for the initialization and finalization phases has a rounds while the permutation for the associated data and message encryption and decryption phases has b rounds, where $b < a$.

The CiliPadi$[n, r, a, b]$ has four versions as CiliPadi-Mild, CiliPadi-Medium, CiliPadi-Hot, and CiliPadi-ExtraHot which are listed in Table 7, based on the increasing level of security.

The components of CiliPadi are given in [12].

4.2 The Attack on CiliPadi

Note that, the designers of [12] for padding the associated data and message blocks wrote that *"Both the associated data and message blocks are individually padded only if its length is not a multiple of r bits. Padding is performed by adding a bit 1, and then as many zero bits as necessary until the padded data is in multiple of r bits. If the length of the last block is* $(r-1)$ *bits, then only bit 1 is added."*

Based on this padding approach, the CiliPadi is vulnerable against length extension attack, e.g., $E(M, K) = E(M||\texttt{80})$, when $M \in \{0,1\}^{r-8}$. Table 8 shows an example of such a collision/forgery with empty plaintext for the "Mild" version, based on their reference source code.

Also, Table 9 shows a forgery example with non-emphy plaintext. Note that the proposed attack works against all variants of CiliPadi, i.e., Mild, Medium, Hot, and ExtraHot. It should be noted, following this attack, the designers have tweaked their proposal [12]

5 Conclusion

GAGE and InGAGE are candidates of the first round of the NIST competition for lightweight cryptography. In this work, we presented a preimage attack against some variants of hash function GAGE (version 1) and an integrity attack against two variants of InGAGE (version 1) and a confidentiality attack against a variant of InGAGE (version 1.03) and an attack for the recovery of the key against the reduced composition of permutation for two variants of InGAGE (version 1.03). The proposed attacks are some structural attack, which can be summarized as follows:

(1) The exact security for preimage of the variant of GAGE for which the rate is 128 bits and the capacity is 256 bits is upper-bounded by 2^{128}, much below the designers' claim, which is 2^{256}. This attack decreases the security bound of some other variants of GAGE.

(2) The exact security for the integrity of the variant of InGAGE, for which the rate is 16 bits and the capacity is 240 bits, is upper-bounded by 2^{112}, and the exact security for the integrity of plaintext and the associated data of the vari-

Table 6. Our time for running the MILP model to solve the equation 1 for finding the key, t is the time (second) and r_1 is the number of composition of permutation Q with itself.

#	*nonce* N	*key* K	r_1	t seconds
1.	FE1FADD3BF068066306F5BDB	5B2A479228606D12D56844BBA862987E	1	.01
2.	FE1FADD3BF068066306F5BDB	5B2A479228606D12D56844BBA862987E	2	.01
3.	FE1FADD3BF068066306F5BDB	5B2A479228606D12D56844BBA862987E	3	.04
4.	FE1FADD3BF068066306F5BDB	5B2A479228606D12D56844BBA862987E	4	141.88
5.	FE1FADD3BF068066306F5BDB	5B2A479228606D12D56844BBA862987E	5	5.12
6.	1E1FADD3BF068066306F5BDB	DB2A479228606D12D56844BBA862987E	5	18.35
7.	1E1FADD3BF068066306F5BDB	DB2A479228606D12D56844BBA8629875	5	4.80
8.	1E1FADD3BF068066306F5BDB	DB2A479228606D12D56844BBA8629875	6	110.44
9.	1E1FADD3BF068066306F5BDB	DB2A479228606D12D56844BBA8629875	7	215.05

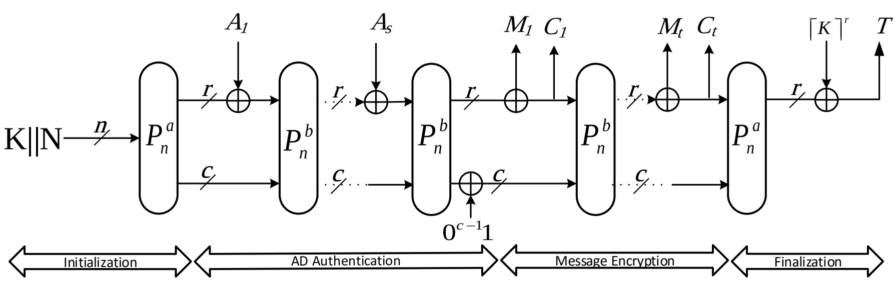

Figure 5. CiliPadi mode of operation [12].

Table 7. All instances of Cilipadi, # r' denotes number of rounds.

CiliPadi-	$[n, r, a, b]$	$\|K\|$	$\|N\|$	$\|T\|$	$\|Block\|$	# r' of P_n^a	# r' of P_n^b
Mild	[256,64,18,16]	128	128	64	64	18	16
Medium	[256,96,20,18]	128	128	96	96	20	18
Hot	[384,96,18,16]	256	128	96	96	18	16
ExtraHot	[384,128,20,18]	256	128	128	128	20	18

Table 8. An example for a collision/forgery with empty plaintext for the "Mild" version of CiliPadi.

Key	0001020304050680080 90A0B0C0D0E80	000102030405068008090A0B0C0D0E80
Nonce	000102030405068008090A0B0C0D0E80	000102030405068008090A0B0C0D0E80
Plaintext	-	-
AD	00010203040506	0001020304050680
Ciphertext	-	-
Tag	158244EEA881F6C9	158244EEA881F6C9

Table 9. An example for a collision/forgery with non-empty plaintext for the "Mild" version of CiliPadi.

Count	529	496
Key	000102030405060708090A0B0C0D0E80	000102030405060708090A0B0C0D0E80
Nonce	000102030405060708090A0B0C0D0E80	000102030405060708090A0B0C0D0E80
Plaintext	000102030405060708090A0B0C0D0E80	000102030405060708090A0B0C0D0E
AD	-	-
Ciphertext	4A1EAAD2F68E41B3891A5632EC092000	4A1EAAD2F68E41B3891A5632EC0920
Tag	CA7773AC3434B7	CECA7773AC3434B7

ant of InGAGE, for which the rate is 8 bits and the capacity is 232 bits, is upper-bounded by 2^{104}, below the designers' claim which is 2^{128} for both variants.

(3) The exact security for the confidentiality of plaintext of the variant of InGAGE, for which the rate is 8 bits and the capacity is 232 bits, is upper bounded by 2^{112}, a little below the designers claim which is 2^{116}.

(4) The exact security for the recovery of the key of two variants of InGAGE, for which the rate is 8(*or* 16) bits and the capacity is 232(*or* 240) bits with reduced number of compositions $r1 < 8$ are $2^{b-|T|}$, less than the length of the key 128.

(5) Also, In this paper, we applied a length extension attack on CiliPadi variants.

Acknowledgment

The authors gratefully thank the use of School of Computer Science, Institute for Research in Fundamental Science (IPM) as the computations were partially done there. We would like also to thank all the anonymous reviewers for their valuable comments which helped us to improve the presentation of the work significantly. This work was supported in part by the Iran National Science Foundation (INSF) under contract No. 98010674. Majid Mahmoudzadeh Niknam and Mohammad Reza Aref have been supported by the INSF under contract No. 96/53979.

References

[1] Secure Hash Standard. National institute of standards and technology, fips 180-1.(apr. 1995).

[2] Secure Hash Standard. National institute of standards and technology, fips 180-2.(aug. 2002).

[3] Secure Hash Standard. National institute of standards and technology, fips 202. sha-3 standard: Permutation-based hash and extendable-output functions, (aug. 2015).

[4] Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. In *Annual Cryptology Conference*, pages 222–239. Springer, 2011.

[5] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. *J. Cryptology*, 26(2):313–339, 2013.

[6] Avik Chakraborti, Anupam Chattopadhyay, Muhammad Hassan, and Mridul Nandi. Trivia: a fast and secure authenticated encryption scheme. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 330–353. Springer, 2015.

[7] Niels Ferguson, Doug Whiting, Bruce Schneier, John Kelsey, Stefan Lucks, and Tadayoshi Kohno. Helix: Fast encryption and authentication in a single cryptographic primitive. In *International workshop on fast software encryption*, pages 330–346. Springer, 2003.

[8] NIST. Submission requirements and evaluation criteria for the lightweight cryptography standardization process. csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf, 2018. `https://csrc.nist.`

[9] Danilo Gligoroski, Hristina Mihajloska, and Daniel Otte. GAGE and InGAGE. NIST, Information Technology Laboratory COMPUTER SECURITY RESOURCE CENTER, 2019. `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography /documents/round-1/spec-doc/GAGEandInGAGE-spec.pdf.`

[10] Danilo Gligoroski, Hristina Mihajloska, and Daniel Otte. GAGE and InGAGE V1.01. http://gageingage.org, 07.05.2019.

[11] Danilo Gligoroski, Hristina Mihajloska, and Daniel Otte. GAGE and InGAGE V1.03. http://gageingage.org, 01 Aug 2019.

[12] Muhammad Reza Z'aba, Norziana Jamil, Mohd Saufy Rohmad, Hazlin Abdul Rani, and Solahuddin Shamsuddin. The cilipadi family of lightweight authenticated encryption. 2019.

[13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd*

Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, 2013.

[14] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

[15] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. *IACR Cryptology ePrint Archive*, 2011:609, 2011.

[16] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.

[17] LLC Gurobi Optimization. Gurobi optimizer reference manual.

[18] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M Youssef. Milp modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Transactions on Symmetric Cryptology*, pages 99–129, 2017.

Majid Mahmoudzadeh Niknam is now a candidate of Ph.D. in mathematical cryptography in Kharazmi University under the supervision of Prof. Mohammad Reza Aref and the advisor of Dr. Nasour Bagheri. His current research interests include symmetric cryptography and cryptanalysis.

Sadegh Sadeghi received his Ph.D. in mathematical cryptography from Kharazmi University in 2019. His Ph.D. dissertation focused on automated cryptanalysis of light weight symmetric ciphers under the supervision of Prof. Nasour Bagheri. His research interest includes security analysis of symmetric primitives, e.g., block ciphers and authenticated encryption schemes.

Mohammad Reza Aref received the B.S. degree in 1975 from the University of Tehran, Iran, and the M.Sc. and Ph.D. degrees in 1976 and 1980, respectively, from Stanford University, Stanford, CA, USA, all in electrical engineering. He returned to Iran in 1980 and was actively engaged in academic affairs. He was a faculty member of Isfahan University of Technology from 1982 to 1995. He has been a professor of electrical engineering at Sharif University of Technology, Tehran, since 1995, and has published more than 290 technical papers in communication and information theory and cryptography in international journals and conferences proceedings. His current research interests include areas of communication theory, information theory, and cryptography.

Nasour Bagheri received the M.S. and Ph.D. degrees in Electrical Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2002 and 2010 respectively. He is currently an associate professor at the electrical engineering department, Shahid Rajaee Teacher Training University, Tehran, Iran. He is the author of over 100 articles in information security and computational neuroscience. His research interests include cryptology, more precisely, designing and analysis of symmetric schemes such as lightweight ciphers, e.g., block ciphers, hash functions and authenticated encryption schemes, cryptographic protocols for constrained environment, such as RFID tags and IoT edge devices and hardware security, e.g., the security of symmetric schemes against side-channel attacks such as fault injection and power analysis. In the field of neuroscience, he is more interested in the computational modeling of human vision system. A record of his publication is available at google scholar: https://scholar.google.com/citations?hl=en&user=32llx44AAAAJ&view_op=list_works&sortby=pubdate. His web-page is: https://sites.google.com/view/nasour-bagheri.

6 Appendix: MILP model

In this section we describe some details of our model for finding the key K by given S_n. The main nonlinear substitution s-box in InGAGE is a 4×2 boolean function Q whose algebraic normal form (ANF) is $Q(x_1, x_2, x_3, x_4) = (x_1 \oplus x_3 \oplus x_2x_3 \oplus x_2x_4, 1 \oplus x_1 \oplus x_2 \oplus x_2x_3 \oplus x_4 \oplus x_2x_4)$, page 7 of InGAGE (version v1.03) [11]. We did like the method sugeted in Abdelkhalek et.al.'s paper [18]. We define a 6×1 boolean function $f(x_1, x_2, x_3, x_4, y_1, y_2) = 1$ if and only if $Q(x_1, x_2, x_3, x_4) = (y_1, y_2)$ and by using free program "*Logic Friday*" we took the product of sum of f and by using them, we obtained 16 linear inequalities for four inputs and two outputs of this s-box as follows:

$$x1+x2+x3-y1 >= 0 \ , \ -x1-x2-x4-y1 >= -3$$

$$x1-x2+x4-y1 >= -1 \ , \ x1+x2-x3+y1 >= 0$$

$$x1-x2-x4+y1 >= -1 \ , \ -x1-x2+x4+y1 >= -1$$

$$-x1-x2-x3-y2 >= -3 \ , \ x1-x2+x3-y2 >= -1$$

$$-x1+x2+x4-y2 >= -1 \ , \ x1-x2-x3+y2 >= -1$$

$$-x1-x2+x3+y2 >= -1 \ , \ -x1+x2-x4+y2 >= -1$$

$$x2-x3-x4-y1-y2 >= -3 \ , \ x2+x3-x4+y1-y2 >= -1$$

$$x2-x3+x4-y1+y2 >= -1 \ , \ x2+x3+x4+y1+y2 >= 1$$

Then in a similar way, we obtained four inequalities for every XOR $x \oplus y = z$ as follows:

$$-x+y+z >= 0 \ , \ x-y+z >= 0$$

$$x+y-z >= 0 \ , \ x+y+z <= 2$$

We defined for every round states, b unknown input and b unknown output variables. There are some relations between these variables in the InGAGE algorithm; by using the above inequalities and these relation, we obtained all linear inequalities of our MILP model.