Key Recovery Attack Against 2.5-Round Pi-Cipher

Christina Boura, Avik Chakraborti, Gaëtan Leurent, Goutam Paul, Dhiman Saha, Hadi Soleimany, Valentin Suder

Key Recovery Attack against 2.5-round π-Cipher

Christina Boura[1], Avik Chakraborti[2], Gaëtan Leurent[3], Goutam Paul[2], Dhiman Saha[4], Hadi Soleimany[5,6] and Valentin Suder[7]

[1] University of Versailles, France
christina.boura@uvsq.fr
[2] Indian Statistical Institute, Kolkata, India
avikchkrbrti@gmail.com, goutam.paul@isical.ac.in
[3] Inria, project-team SECRET, Paris, France
gaetan.leurent@inria.fr
[4] Crypto Research Lab, Indian Institute of Technology Kharagpur, India
saha.dhiman@gmail.com
[5] Cyberspace Research Institute, Shahid Beheshti University, Iran
[6] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran
h_soleimany@sbu.ac.ir
[7] University of Waterloo, Canada
valentin@suder.xyz

Abstract. In this paper, we propose a *guess and determine* attack against some variants of the π-Cipher family of authenticated ciphers. This family of ciphers is a second-round candidate of the CAESAR competition. More precisely, we show a key recovery attack with time complexity little higher than $2^{4\omega}$, and low data complexity, against variants of the cipher with ω-bit words, when the internal permutation is reduced to 2.5 rounds. In particular, this gives an attack with time complexity 2^{72} against the variant π16-Cipher096 (using 16-bit words) reduced to 2.5 rounds, while the authors claim 96 bits of security with 3 rounds in their second-round submission. Therefore, the security margin for this variant of π-Cipher is very limited.

The attack can also be applied to lightweight variants that are not included in the CAESAR proposal, and use only two rounds. The lightweight variants π16-Cipher096 and π16-Cipher128 claim 96 bits and 128 bits of security respectively, but our attack can break the full 2 rounds with complexity 2^{72}.

Finally, the attack can be applied to reduced versions of two more variants of π-Cipher that were proposed in the first-round submission with 4 rounds: π16-Cipher128 (using 16-bit words) and π32-Cipher256 (using 32-bit words). The attack on 2.5 rounds has complexity 2^{72} and 2^{137} respectively, while the security claim for 4 rounds are 128 bits and 256 bits of security.

Keywords. Authenticated Encryption, π-Cipher, CAESAR Competition, Guess and Determine, Cryptanalysis.

1 Introduction

Authenticated encryption is a rapidly growing field of cryptography that has wide applications in diverse industries. Even though some efforts over the past few years have been devoted to the design and analysis of authenticated encryption schemes, a well-studied design with the desirable level of security and performance is not yet available. Lack of secure and efficient authenticated ciphers led to devastating attacks in extensive applications like TLS and OpenSSL [4, 1]. To address this challenge, an international contest called CAESAR, funded by the NIST, plans to hold a multi-year effort to identify a promising new portfolio of authenticated ciphers, suitable for widespread applications [3]. The CAESAR competition, launched in 2014, follows the long tradition of contests in secret key cryptography and aims at selecting a portfolio of authenticated ciphers that offer perceptible advantages over AES-GCM and that can be recommended for widespread use. There were 57 proposals accepted for the first round of the competition and recently, 30 ciphers among these proposals were selected to continue in the second round.

The π-Cipher [7] family of authenticated ciphers, designed by Gligoroski et al., is one of the 30 second-round candidates. It is a special case of encrypt-then-MAC designs and makes use, as all such CAESAR candidates, of a nonce and process associated data.

One of the most important design goals of this family of cryptographic functions is the possibility of parallel computations. Other goals, as claimed by the designers, are a better security than AES-GCM in the case of a nonce reuse, and better resistance for producing second-preimage tags. Although the cipher's mode of operation is inspired by the sponge construction [2], and is based on a permutation called the π-function, it has been largely modified by Gligoroski et al. in order to permit parallel computations.

In the initial submission, the authors proposed six different variants of the cipher, where each variant offered a particular level of security and used words of a particular size. More precisely, the level of targeted security, corresponding to the size of the secret key, ranges from 96 to 256 bits, and each variant uses words of 16, 32, or 64 bits. For the second round of the competition, only four variants were kept. Another decision taken by the designers for the second-round version of the cipher, was to decrease the number of rounds of the π-function from 4 to 3. In addition, at NIST's lightweight cryptography workshop, a lightweight version of the π-Cipher [10] was proposed. The lightweight proposal is composed of two variants, both using 16-bit words. Since lightweight ciphers must be as small and power-efficient as possible, the number of rounds in the internal permutation is further reduced to 2 in the lightweight version. An overview of the different variants is given in Table 1.

Our results. In this work, we present a key recovery attack against several variants of the π-Cipher, when the π-function is reduced to 2.5 rounds. This shows that the decision to decrease the number of rounds was precarious. Indeed,

the lightweight version is completely broken, and the affected variant that is still in the second round submission offers only very limited security margin.

More precisely, the time complexity of our attack is 2^{72} for the 16-bit word variants and 2^{137} for the 32-bit word variants, while the data complexity remains very low (a single known plaintext with at least 256 blocks for 16-bit word variants, and 512 blocks for the 32-bit word variants). The attack is faster than exhaustive search of the key for the following variants (reduced to 2.5 rounds):

π16-Cipher096 with 16-bit words and 96-bit key.
This variant was proposed with 4 rounds in version 1, 3 rounds in version 2, and 2 rounds in the lightweight version.

π16-Cipher128 with 16-bit words and 128-bit key.
This variant was proposed with 4 rounds in version 1, and 2 rounds in the lightweight version.

π32-Cipher256 with 32-bit words and 256-bit key.
This variant was proposed with 4 rounds in version 1.

Our cryptanalysis is a *guess and determine* attack exploiting a weakness in the high-level structure of the π-function. Indeed, we show that by knowing two out of the four output chunks of the π-function and by guessing a third one, we can easily recover one of the four input chunks of the permutation. This permits us to recover the internal state and gives us the possibility to recover the secret key by some very simple operations. Note that our attacks work in the case when no secret message number is processed. However, the attacks can be easily extended in cases when a secret message number is used, if one supposes that the secret message number is known together with the plaintext.

Cryptographic algorithms should be designed with enough security margin to thwart classical attacks but also to resist to new and unknown vulnerabilities. Surplus security cannot be obtained for free, since it has impacts on the performance of the ciphers. In particular, due to a number of important limitations in the resources of pervasive devices, it is of utmost importance to analyze lightweight cryptographic designs that allow reduction of superfluous margins. Our attack shows that the security margin offered by these three members of the π-Cipher family is too small and that these variants are much less secure than expected. This kind of analysis is very important for the progress of the CAESAR competition, as the final portfolio of the selected authenticated ciphers should offer a high level of security. Thus, evaluating the security of the remaining candidates, leads to a more clear overview of which candidates are robust and which should be eliminated.

Outline. The rest of the paper is organised as follows. In Section 2 we briefly provide the specifications of π-Cipher. Then, we present our attack on 2.5 round π-Cipher in Section 3 and we discuss how to mount a full-round attack on the lightweight version of π-Cipher in Section 4. Finally, we perform a complexity analysis of our attacks in Section 5 and conclude.

2 π-Cipher Specifications

There exist different variants of π-Cipher, depending on the bit-length of the words used and the expected level of security expressed in bits. Therefore, πω-Cipher$n$ represents a variant defined with ω-bit words and offering n-bit security. The six variants of π-Cipher submitted to the first round of the competition, together with the corresponding parameters, are summarized in Table 1. The first four rows in the table represent the only four variants conserved for the second round. Furthermore, the two variants of the recently presented lightweight π-Cipher proposal [10], are described in the last two rows of Table 1.

Table 1. π-Cipher variants. The first four rows represent the four variants kept for the second round of the CAESAR competition. The last two rows describe the two lightweight variants proposed in [10]. PMN and SMN are the two parts of the nonce and stand for *Public Message Number* and *Secret Message Number* respectively. All the parameters are given in bits.
For variants both in version 1 and 2, there are 4 rounds in v1 and 3 rounds in v2.

Version	Variant	Word size ω	PMN	SMN	Rate r	Tag size t	Key length	Rounds
v1 & v2	π16-Cipher096	16	32	0 or 128	128	128	96	3
	π32-Cipher128	32	128	0 or 256	256	256	128	3
	π64-Cipher128	64	128	0 or 512	512	512	128	3
	π64-Cipher256	64	128	0 or 512	512	512	256	3
v1	π16-Cipher128	16	32	0 or 128	128	128	128	4
	π32-Cipher256	32	128	0 or 256	256	256	256	4
Lightweight	π16-Cipher096	16	32	0 or 128	128	128	96	2
	π16-Cipher128	16	32	0 or 128	128	128	128	2

2.1 Authenticated Encryption

The encryption/authentication function accepts as input a triplet (K, AD, M), where K is a secret key, AD is a string of associated data of a blocks, and M is a message composed of m blocks of size r bits each. The main building block of the authenticated encryption procedure is a construction that the authors call the *e-triplex* component and which is depicted in Figure 1. The encryption procedure starts by initializing the internal state with the string $K||PMN||10^*$, where the number of 0's appended should be such that the length of the concatenated string equals the size of the state of the π-function. This internal state is then updated by applying the π-function. The result is called the *Common Internal State* (CIS) and is used as the initial state for the first parallel computations:

$$CIS \leftarrow \pi(K||PMN||10^*).$$

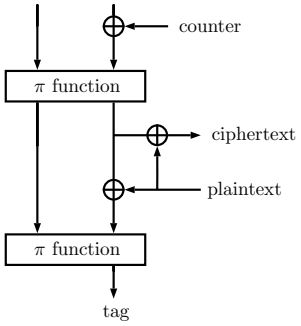

Fig. 1. The e-triplex component of π-Cipher.

By following the same notation as in the sponge construction, we can see each internal state, say IS, as the concatenation of a *rate* part and a *capacity* part: $IS = IS_{capacity}||IS_{rate}$. In particular, each internal state IS of the procedure is the concatenation of four 4ω-bit chunks, that we will denote as $IS = IS_1||IS_2||IS_3||IS_4$. From the specification of π-Cipher, the capacity part of the state is $IS_{capacity} = IS_2||IS_4$, and the rate part of the state is $IS_{rate} = IS_1||IS_3$. The counter, denoted by ctr, is then initialized by extracting the first 64 bits of $CIS_{capacity}$. This procedure is depicted at the top left part of Figure 2.

The next step in the authenticated encryption procedure is the process of the associated data. The associated data AD is cut into equal-sized blocks: $AD = AD_1||\dots||AD_a$. All blocks are treated in parallel by the e-triplex component. The input to the e-triplex component for the block i is CIS, $ctr + i$ and AD_i, and the output is an intermediate tag t'_i. The way that each block of associated data is processed can be observed in Figure 2. At the end of this procedure a tag for the associated data T' is computed as

$$T' = t'_1 \boxplus_d \cdots \boxplus_d t'_a,$$

where $\boxplus_d$ is a component-wise addition of vectors of dimension d, where d is the number of ω-bit words in the *rate* part ($d = 8$ for all proposed variants of π-Cipher). Finally, the internal state is updated in the following way to create a new internal state that we will denote by CIS':

$$CIS' \leftarrow \pi(CIS_{capacity}||CIS_{rate} \oplus T').$$

After this first phase, the secret message number SMN, if any, is processed. This procedure is depicted in Figure 2 and described by the following expressions:

$$\begin{aligned} IS &\leftarrow \pi(CIS'_{capacity}||CIS'_{rate} \oplus (ctr + a + 1)), \\ CIS'' &\leftarrow \pi(IS_{capacity}||IS_{rate} \oplus SMN). \end{aligned}$$

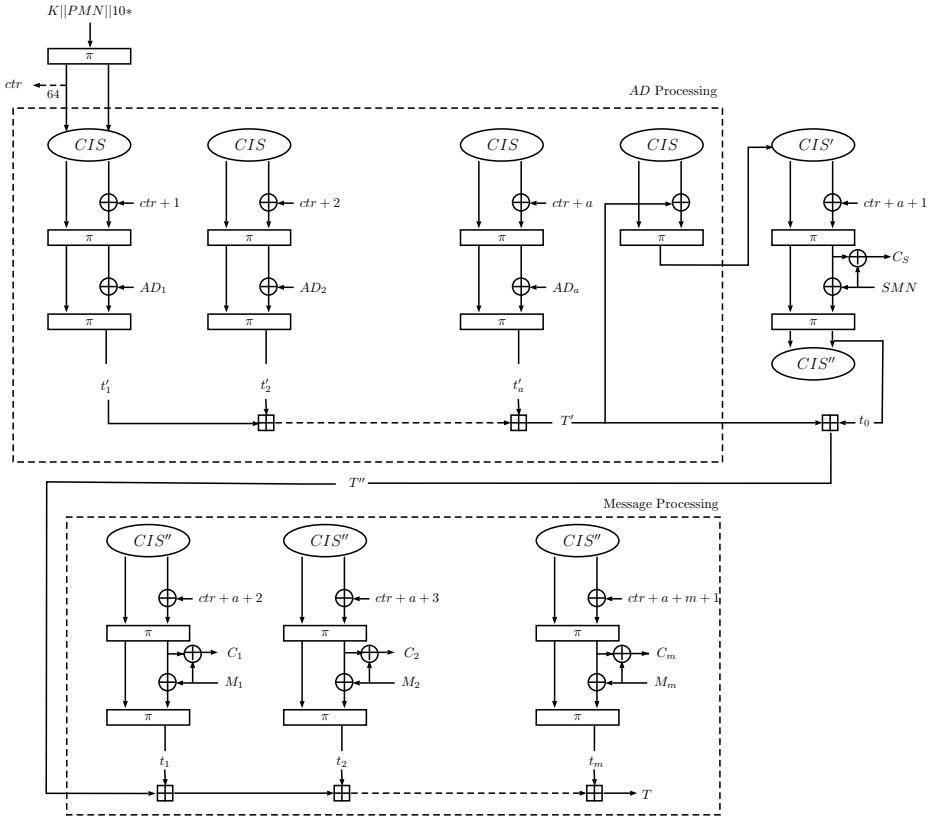

Fig. 2. π-*Cipher* encryption structure.

The new state CIS'' will be used as the common state for the parallel process of the message blocks. The tag produced during this phase is

$$T'' = T' \boxplus_d t_0,$$

where t_0 is the output tag of the last call to the e-triplex component after absorbing the SMN. If no secret message number is used, then the above steps are ignored. The authenticated encryption procedure without SMN is depicted in Figure 4.

In the last phase, the message blocks are treated. As for the associated data, the message M is cut into blocks $M = M_1||\ldots||M_m$ and each block is processed in parallel by the e-triplex construction. Note that the length of each message block, as well as of each ciphertext block is equal to the bitrate, i.e. r bits (e.g. $r = 128$ in the case of π16-Cipher096). A unique block counter is associated with each message block. The counter for the message block M_j is computed as $ctr + a + j$ if the secret message number is empty, and as $ctr + a + 1 + j$ otherwise.

During encryption, each e-triplex component takes as input the common state CIS'', the counter ctr and a message block M_j and outputs a pair (C_j, t_j), where C_j is a ciphertext block and t_j is a partial tag. The final tag T is computed as

$$T = T'' \boxplus_d t_1 \cdots \boxplus_d t_m.$$

2.2 The π-function

The core of π-Cipher is an ARX-based permutation called the π-function. This permutation somehow uses similar operations as the hash function Edon-R [8]. We denote the size of the permutation in bits by b and the number of rounds by R. For the first version of the cipher, R was fixed to 4, however the authors decided to reduce this number to 3 for the second round of the competition. The internal state (IS) of the π-function can be seen as a concatenation of four chunks of four words, so that $b = 4 \times 4 \times \omega$ bits. The π-function is mainly based on an operation that will be denoted by $\circledast$. However, as our attack does not take advantage of the internal structure of $\circledast$ we omit here its description. The only important thing to know about this operation in order to understand the attack is that it is a 2-input 1-output operation (in Figure 3, the two outputs of a $\circledast$ operation are equal) that is invertible with respect to each of its inputs. Its full specifications can be found in [7]. A round of the π-function is depicted in Figure 3, where S_1 and S_2 are constants.

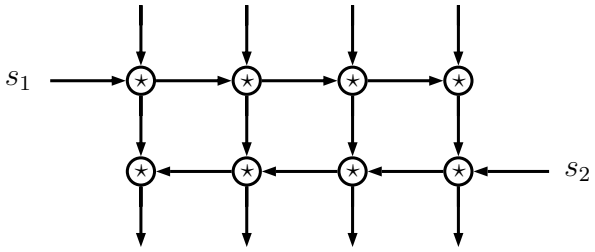

Fig. 3. One round of the π-function.

2.3 Previous Cryptanalysis Results

In [6], Fuhr and Leurent showed that forgeries can be computed for the first round variants of π-Cipher due to a weakness in the padding algorithm. More precisely, they noticed that the padding used for both the associated data and the plaintext was not injective. This observation permitted to mount a forgery attack by producing valid tags and forced the designers to modify the padding rule for the second round of the competition.

One of the advertised features of π-Cipher is tag second-preimage resistance, meaning that it should be hard to generate a message with a given tag, even for the legitimate key holder. However, Leurent demonstrated in [9] that practical tag second-preimage attacks could be mounted against π-Cipher by using Wagner's generalized birthday attack. More specifically, Leurent showed that tag second-preimages can be computed with optimal complexities ranging from 2^{22} to 2^{45} depending on the word size ω.

The best attack mentioned by the designers [7, Section 3.3] is a distinguisher on reduced versions with 1 round, using a guess and determine technique. Their attack has complexity about $2^{4\omega}$ (time and memory); in particular, it is applicable to the same variants as our attack. Our attack actually uses similar ideas, but reaches 2.5 rounds, and a full key recovery.

3 Key Recovery Attack against 2.5-round π-Cipher

We describe in this section our key recovery attack against reduced-round variants of π-Cipher when no secret message number (SMN) is used. The authenticated-encryption procedure for this case is described in Figure 4. Note that if no SMN is used then the intermediate tags T' and T'' are equal and that the state CIS'' of Figure 2 is equal to the state CIS'. In order to be consistent with the notation of Section 2, we will keep denoting the common state for processing the message blocks as CIS'' even if this is exactly the same as CIS' in the empty SMN case.

We consider an m-block message $M = M_1||\cdots||M_m$ and an a-block string of associated data, with the corresponding ciphertext $C = C_1||\cdots||C_m$. The message should have at least 16ω blocks, *i.e.* 256 blocks when $\omega = 16$, and 512 blocks when $\omega = 32$.

We denote the input and output states of the first π-function for processing the message block M_i by $I^i = I^i_1||I^i_2||I^i_3||I^i_4$ and $O^i = O^i_1||O^i_2||O^i_3||O^i_4$ respectively, where each chunk I^i_j, O^i_j, for $1 \leq j \leq 4$, is of size 4ω bits.

In our attack, we deploy a *guess and determine* technique for recovering the secret key for three variants of the π-Cipher family, where the π-function is reduced to 2.5 rounds. Our attack targets the first π-function of the message processing phase, for 16ω consecutive blocks of plaintext. We provide now the main observations that the attack takes advantage of.

3.1 Observations on the π-Cipher Structure

The first observation concerns the nature of the inner operation $\circledast$, that takes two chunks of size 4ω bits as input and outputs a single chunk of the same size. This operation is the core of the π-function. It has the property, that when fixing one of the two input chunks to a constant and letting the other chunk take all possible values, then the output chunk equally takes all possible values (it defines a quasi-group).

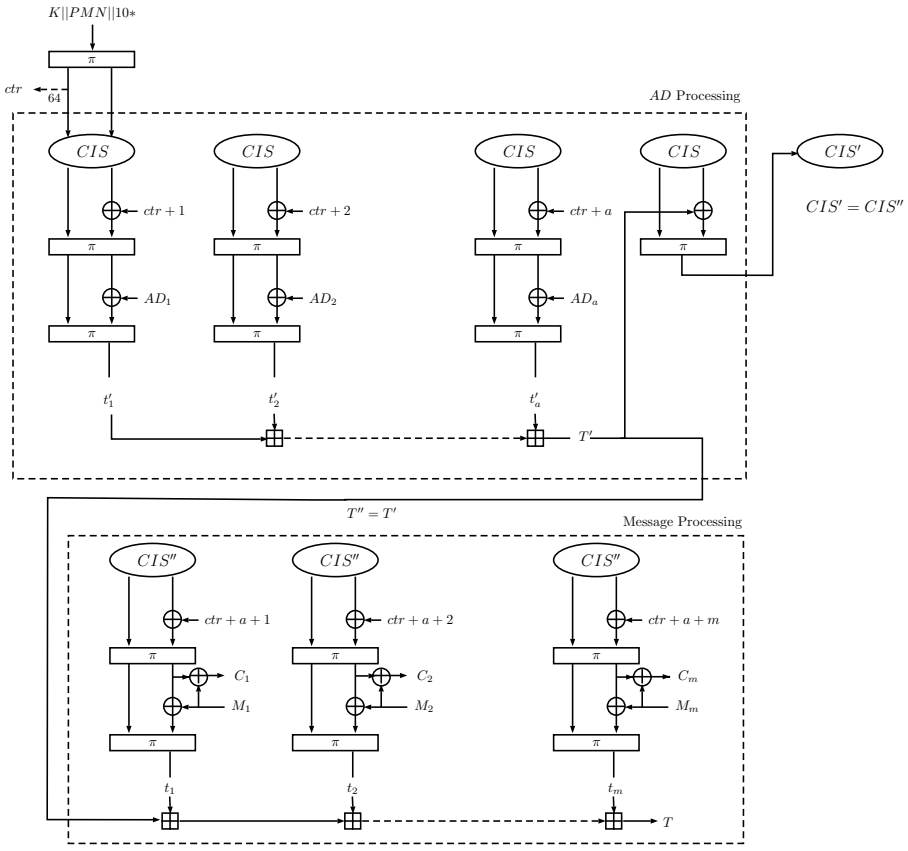

Fig. 4. π-cipher encryption procedure when no secret message number is used.

Observation 1 *Both* $\circledast(a,.)$ *and* $\circledast(.,b)$ *are invertible for all* $a, b \in \mathbb{F}_2^{4\omega}$ *and if* $\circledast(a,b) = c$*, then the knowledge of any two chunks among* a, b *and* c *can determine the third one.*

The next observation is in the core of the guess and determine technique and exploits a weakness in the high-level structure of the π-function. It shows, that when the function is reduced to 2.5 rounds, the knowledge of 3 output chunks of 4 words each, can completely determine an input chunk. This observation demonstrates that the inverse π-function has a limited diffusion when the number of rounds is reduced to 2.5, as we can see that in this case an input word does not depend on all the output words.

Observation 2 *Let,* $I = I_1||I_2||I_3||I_4$ *and* $O = O_1||O_2||O_3||O_4$ *be the input and the output state respectively of the* π*-function reduced to* 2.5 *rounds. Then the knowledge of* O_1*,* O_3 *and a guess of* O_2 *can determine* I_1*.*

Proof. This claim can be proven by the following guess and determine steps described below. The pictorial description of the steps is given in Figure 5. In the figure the green boxes denote the determined chunks D_i, $1 \leq i \leq 9$, the orange boxes denote the guessed chunk i.e. O_2 and the chunks denoted by K_1, K_2 corresponding to O_1 and O_3 respectively are known. At the end of this procedure, one computes D_9 which corresponds exactly to I_1. Note that each step of the below procedure makes use of Observation 1.

1. Use K_1, S_1 and G to determine D_1 and D_2.
2. Use K_2 and G to determine D_3.
3. Use D_1 and D_2 to determine D_4.
4. Use D_2 and D_3 to determine D_5 and D_4, S_1 to determine D_6.
5. Use D_4 and D_5 to determine D_7.
6. Use D_6 and D_7 to determine D_8.
7. Use D_8 and S_1 to determine D_9. □

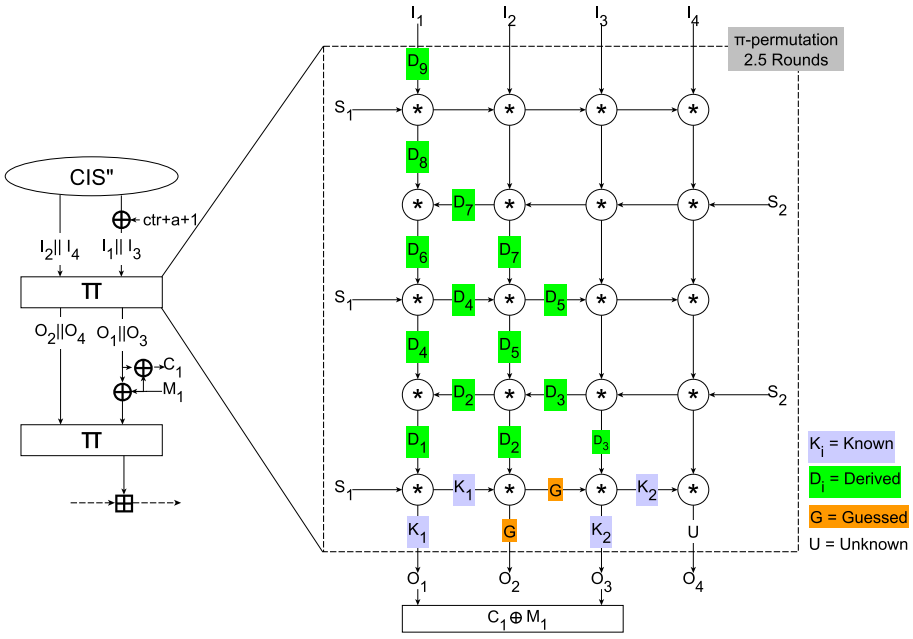

Fig. 5. Guess and determine steps for the first π-function.

The last observation aims at showing that the knowledge of the input state of the π-function for several message blocks can be used to determine the common state CIS''.

Observation 3 *The message processing phase uses the same common internal state,* $CIS'' = CIS''_1||CIS''_2||CIS''_3||CIS''_4$*, to process each of the message blocks* M_i*,* $1 \leq i \leq m$*. Then, the input to the first* π*-function is* $I^i = I^i_1||I^i_2||I^i_3||I^i_4 = CIS''_1 \oplus (ctr + a + i)||CIS''_2||CIS''_3||CIS''_4$ *for each block.*

3.2 High Level Description of the Attack

This section provides a high level description of our attack. As already mentioned, the attack requires a single known plaintext message, with at least 16ω blocks. The attack can be seen as the succession of the five main steps that we describe below:

1. *Guess and determine* step. In this first part of the attack, we target the first computation of the π-function in the message processing part. Two of the output chunks are known to the attacker as they only depend on the plaintext and ciphertext blocks (i.e. $O^i_1||O^i_3 = M_i \oplus C_i$). Then by guessing a third output chunk, namely O^i_2, we are able to determine one input chunk, I^i_1. We repeat this procedure for all message blocks. This step is described in more details in Subsection 3.3. At the end of this part we are left with a collection of lists of candidates for one input chunk. We recover the right value by treating the lists in the way described in the next step.
2. *Computation of the intersection of the created lists.* During this phase, detailed in Subsection 3.4, we show how to treat the created lists in order to recover the right value of the common part for the first input chunk of the π-function, or more precisely, of the value $CIS''_1 \oplus (ctr + a)$ from Observation 3.
3. *Recovery of the intermediate* I^i *state.* This step shows the procedure to recover a list of candidates for the state I^i and is described by the *Recover-IS* Algorithm in Subsection 3.5.
4. *Recovery of the common internal state* CIS. We show here how one can compute the state CIS, once the intermediate state I^1 has been completely identified. This phase is described by the *Recover-CIS Algorithm* in Subsection 3.6.
5. *Computation of the secret key.* This phase is pretty straightforward once we have recovered CIS, since, as already mentioned in Section 2.1, $CIS = \pi(K||PMN||10^*)$ and π-function is a known permutation.

The high level description of the attack is furnished in Algorithm 1.

3.3 Guess and determine

This section describes the guess and determine phase, which recovers the input chunk I^i_1 of the first π-function for the i^{th} block for the plaintext-ciphertext pair $(M = M_1||\cdots M_i \cdots||M_m, C = C_1||\cdots C_i \ldots||C_m)$. Note that we can compute $O^i_1||O^i_3 = M_i \oplus C_i$. Then by making a guess on the value of O^i_2, we can compute I^i_1 independently of O^i_4, following Observation 2. In particular, we can compute it as $I_1 = \pi^{-1}(O_1\|O_2\|O_3\|\langle 0\rangle)$

Algorithm 1 Overview of the attack.

Input: 1 Known Plaintext-Ciphertext Pair ($M = M_1||\cdots||M_{16\omega}, C = C_1||\cdots||C_{16\omega}$)
Output: Master Key K

for all $1 \leq i \leq 16\omega$ **do**
  $\mathcal{L}_i \leftarrow$ Guess-Determine(M_i, C_i) ▷ Subsection 3.3
for all $1 \leq j \leq 8\omega$ **do**
  $\mathcal{S} \leftarrow \bigcap_{0 \leq k < 8\omega} \mathcal{L}_{j+k} \oplus k$ ▷ Subsection 3.4
  if $\mathcal{S} \neq \varnothing$ **then**
    $\mathcal{L}'_0 \leftarrow$ Recover-IS$(M_j, C_j, 0, \mathcal{S})$ ▷ Subsection 3.5
    $\mathcal{L}'_1 \leftarrow$ Recover-IS$(M_{j+1}, C_{j+1}, 1, \mathcal{S})$
    $I^j, I^{j+1} \leftarrow \{I, J \in \mathcal{L}'_0 \times \mathcal{L}'_1 \mid I_2\|I_3\|I_4 = J_2\|J_3\|J_4\}$ ▷ Single value expected
    for all ctr, s.t. $ctr + a + j \equiv 0 \bmod 8\omega$ **do** ▷ Subsection 3.6
      $CIS'' \leftarrow I^j \oplus (ctr + a + j)$
      $CIS \leftarrow$ Recover-CIS(CIS'')
      if $ctr =$ first 64 bits of $CIS_{capacity}$ **then**
        $K||PMN||10^* \leftarrow \pi^{-1}(CIS)$
        return K

We compute all candidates for I_1^i corresponding to the $2^{4\omega}$ choices of O_2^i, and store them in a list $\mathcal{L}_i$. The guess and determine phase is described in Algorithm 2.

Note that there will be less than $2^{4\omega}$ different values of I_1^i in a list $\mathcal{L}_i$ as the π-function is a permutation of the four chunks and not a permutation from one chunk (O_2^i) to one chunk (I_1^i). In the following, we assume that the function from O_2^i to I_1^i behaves as a random function, so that the expected size of $\mathcal{L}_i$ is $(1-e^{-1}) \times 2^{4\omega}$ (see [5, Theorem 2]). In the next part, we describe how to compute the intersection and filter out the correct value of I_1^i for some $1 \leq i \leq 16\omega$.

Algorithm 2 Build the list of candidates for the first input chunk of the first π-function.

Input: Plaintext-ciphertext block M, C
Output: List $\mathcal{L}$ of possible candidates for I_1

function Guess-Determine(M, C)
  $\mathcal{L} \leftarrow \varnothing$
  $O_1||O_3 \leftarrow M \oplus C$
  for all O_2 **do**
    $I_1 \leftarrow \pi^{-1}(O_1||O_2||O_3||\langle 0 \rangle)$ ▷ Following Observation 2
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{I_1\}$
  return $\mathcal{L}$

3.4 Intersecting the lists

In this phase, we compare the list of candidates for I_1^i for each message block, using the fact that they are all derived from a common state CIS''. More precisely,

the first input chunk to the first π-function of each block is computed as:

$$I_1^i = CIS_1'' \oplus (ctr + a + i), \qquad \text{for } 1 \leq i \leq 16\omega.$$

By construction of the lists $\mathcal{L}_i$, we have that:

$$CIS_1'' \oplus (ctr + a + i) \in \mathcal{L}_i, \qquad \text{for } 1 \leq i \leq 16\omega.$$

Let $j \in \{1, \ldots, 8\omega\}$ be such that $ctr + a + j \equiv 0 \bmod 8\omega$ (*i.e.* $j \equiv -(ctr + a) \bmod 8\omega$). In other words, with $\omega = 16$, j is the first message block such that the 7 least significant bits of $ctr + a + j$ are equal to zero (and similarly, 8 bits when $\omega = 32$). This implies:

$$\begin{aligned}
(ctr + a + j) + k &= (ctr + a + j) \oplus k && \text{for } 0 \leq k < 8\omega \\
CIS_1'' \oplus (ctr + a + j) \oplus k &\in \mathcal{L}_{j+k} && \text{for } 0 \leq k < 8\omega \\
CIS_1'' \oplus (ctr + a + j) &\in \mathcal{L}_{j+k} \oplus k && \text{for } 0 \leq k < 8\omega
\end{aligned}$$

Thus,

$$CIS_1'' \oplus (ctr + a + j) \in \bigcap_{k=0}^{8\omega-1} (\mathcal{L}_{j+k} \oplus k).$$

We will compute this intersection for all guesses of $j \in \{1, \ldots, 8\omega\}$. We are interested now in determining the size of the intersection of the 8ω lists. Each list has about $(1 - e^{-1})2^{4\omega}$ elements. If the guess of j is wrong, we assume that the lists are independent; an element is a part of all the 8ω lists with probability $(1 - e^{-1})^{8\omega}$. As there is a total of $2^{4\omega}$ elements, the probability that there is no element in the intersection is $(1 - (1 - e^{-1})^{8\omega})^{2^{4\omega}}$. This probability is very close to one:

$$\begin{aligned}
\left(1 - \left(1 - e^{-1}\right)^{8\omega}\right)^{2^{4\omega}} &= \exp\left(2^{4\omega} \ln(1 - \left(1 - e^{-1}\right)^{8\omega})\right) \\
&\geq 1 + 2^{4\omega} \ln\left(1 - \left(1 - e^{-1}\right)^{8\omega}\right) \\
&\approx 1 - 2^{4\omega} \left(1 - e^{-1}\right)^{8\omega} \\
&\approx 1 - 0.9^{8\omega}
\end{aligned}$$

In particular, it is about $1 - 2^{-20}$ for $\omega = 16$.

On the contrary, if the guess is right, the intersection contains 1 element. With high probability, the test at line 5 of Algorithm 1 will succeed only for the correct value of j, and the corresponding set $\mathcal{S}$ will contain a single value.

3.5 Recovering the intermediate state

So far, we have recovered the value $CIS_1'' \oplus ctr + a + j$, that is to say the first chunk I_1^j of the input of the first π-function. In addition, the least significant bits of $ctr + a + j$ are known to be zero, so that we can compute $I_1^{j+k} = I_1^j \oplus k$ for $0 \leq k < 8\omega$ (adjusting the effect of the counter).

From this, we can build a small list of candidates for any O_2^{j+k}. We just have to try all $2^{4\omega}$ values O_2^{j+k}, recompute I_1^{j+k}, and compare the result to the known value. We know that there will be at least one remaining value, and there can be a few false positives.

Now we make a guess of O_4^{j+k} and use the invertibility of the π-function to built a list $\mathcal{L}'_k$ of all potential values of the full input I^{j+k} of the permutation. This second phase of guess and determine through the π-function is demonstrated in Figure 6. The list $\mathcal{L}'_k$ contains about $2^{4\omega}$ values. This step is described in Algorithm 3.

In order to identify the correct value in the list, we build the lists $\mathcal{L}'_0$ and $\mathcal{L}'_1$, and we use the way I^j and I^{j+1} are derived from CIS''. In particular, we have $I_2^j \| I_3^j \| I_4^j = I_2^{j+1} \| I_3^{j+1} \| I_4^{j+1}$. This allows us to recover the correct value I^j and I^{j+1}.

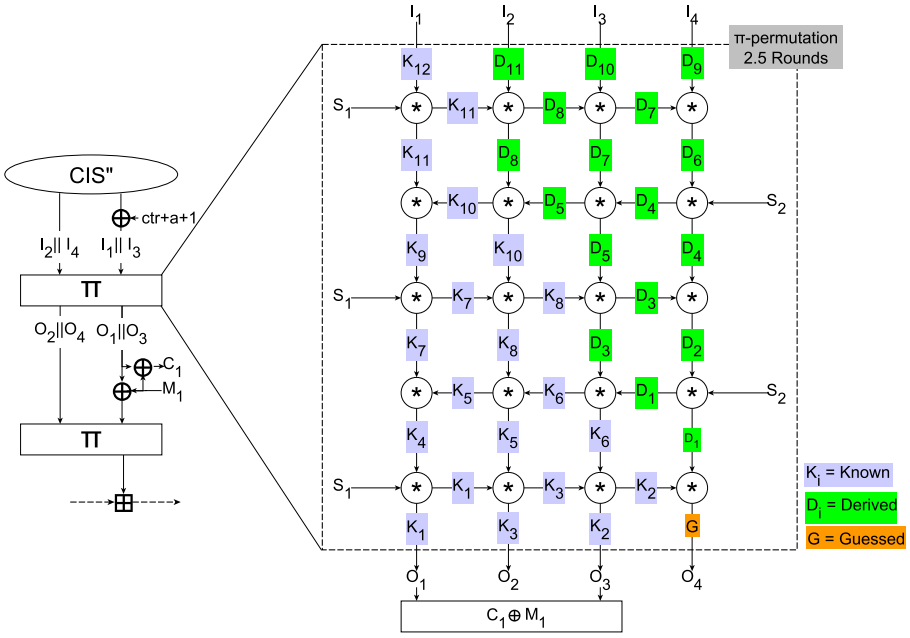

Fig. 6. Guessing O_4 after I_1 has been determined

3.6 Recovering the Common Internal State CIS

In this section we show how to recover the common internal states CIS'' and CIS. We remind once again, that the state CIS' is equal to CIS''. From the

Algorithm 3 Build the list of candidates for the full input of the first π-function, knowing the first input chunk.

Input: Plaintext-ciphertext block M, C; index k; list of I_1 candidates $\mathcal{S}$
Output: List $\mathcal{L}$ of candidates for $I_2||I_3||I_4$

function RECOVER-IS($M, C, k, \mathcal{S}$)
  $\mathcal{L} \leftarrow \varnothing$
  $O_1||O_3 \leftarrow M \oplus C$
  for all O_2 **do**
    $I \leftarrow \pi^{-1}(O_1||O_2||O_3||\langle 0 \rangle)$
    if $I_1 \oplus k \in \mathcal{S}$ **then** ▷ Only one candidate expected
      for all O_4 **do**
        $I \leftarrow \pi^{-1}(O_1||O_2||O_3||O_4)$
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{I_2||I_3||I_4\}$
  return $\mathcal{L}$

previous sections, the input state of the first π-function for message block j, I^j has been recovered. Note that

$$I^j = I_1^j||I_2^j||I_3^j||I_4^j = CIS_1'' \oplus (ctr + a + j)||CIS_2''||CIS_3''||CIS_4''.$$

By making a guess for the value of the counter ctr, we can compute the value of CIS'' which equals CIS'.

The next step is to retrieve the tag T'' and therefore T' (since both tags are equal) by computing $T'' = T \boxminus_d t_1 \boxminus_d \cdots \boxminus_d t_{16\omega}$, where each tag t_i, $1 \leq i \leq 16\omega$ can be recovered from the knowledge of CIS'', ctr and the message blocks.

Once this step is done, the recovery of the common internal state CIS is immediate, as one can compute it as $CIS = \pi^{-1}(CIS') \oplus T'$. Note that, at this point, we can easily verify if the guess of ctr was correct, since ctr corresponds to 64 bits extracted directly from the initial state CIS (as described in Section 2.1). The above procedure is described by Algorithm 4.

Algorithm 4 Recover the initial state CIS.

Input: Common Internal State CIS'', corresponding message M
Output: Common Internal State CIS

function RECOVER-CIS(CIS'', M)
  for $1 \leq i \leq 16\omega$ **do**
    Compute t_i from CIS'' and M_i
  $T' = T \boxminus_d t_1 \boxminus_d \cdots \boxminus_d t_{16\omega}$
  $CIS \leftarrow \pi^{-1}(CIS')_{capacity}||\pi^{-1}(CIS')_{rate} \oplus T'$
  return CIS

3.7 Key recovery

Once the internal state CIS has been successfully recovered, one can retrieve the master key K by simply inverting the π-function, as described by Line 13 of Algorithm 1.

3.8 About the use of SMN

The above described analysis supposes that no secret message number is used. This is a legitimate assumption, as $|SMN| = 0$ is a valid scenario mentioned in the cipher's proposal. Our attack can be easily extended to the case when an SMN is used if one supposes that this number is known to the attacker together with the plaintext. In the case that the knowledge of SMN is not available to the attacker, our analysis fails. However, it is still possible to mount a forgery attack in this case.

More precisely (see Figure 2), if one is given an m-block message M with associated data AD and the corresponding tag T, one can easily construct a forgery as follows. Suppose that the new message M^{forged} has $(m+1)$ blocks where the first m blocks are identical to the first m blocks of M (i.e., M is a prefix of M^{forged}) and the last block of M^{forged} is any fixed value. We follow the steps of Algorithm 1 with message M up to Step 8. At this point we intend to recover ctr. However, we cannot follow the same strategy as the one followed in Algorithm 1 since CIS cannot be recovered without the knowledge of SMN. But we can use the value of C_s which is the output of the SMN processing branch (see Figure 2). So basically we guess ctr to determine CIS'' as before. Subsequently, we ascertain the value ctr by exploiting the relation $(\pi^{-1}(CIS''))_{rate} = C_s$. Since at this point, ctr is known, we can easily compute t_{m+1} and thus, the new tag T^{forged} will be given by $T \boxplus t_{m+1}$.

4 Key Recovery Attack against Full Round Lightweight Version of π-Cipher

We argue here that the previously presented attack against various versions of the π-Cipher CAESAR candidate, completely breaks the lightweight version [10] of the same cipher, where the number of rounds is reduced to 2.

The only difference with the previous attack is that, as the number of rounds is reduced, the guess and determine part of the attack is slightly modified to fit this reduction. This part, depicted at the left part of Figure 7 is described by the following steps:

1. Use K_1 and G to determine D_1.
2. Use K_2 and G to determine D_2.
3. Use D_1 and S_1 to determine D_3.
4. Use D_1 and D_2 to determine D_4.
5. Use D_3 and D_4 to determine D_5.

6. Use D_5 and S_1 to determine D_6.

After the chunk I_1 has been determined, the other chunks I_2, I_3 and I_4 can be derived by further guessing the value of O_4, as shown at the right part of Figure 7. The other steps of the attack remain unchanged, thus we ignore their full description.

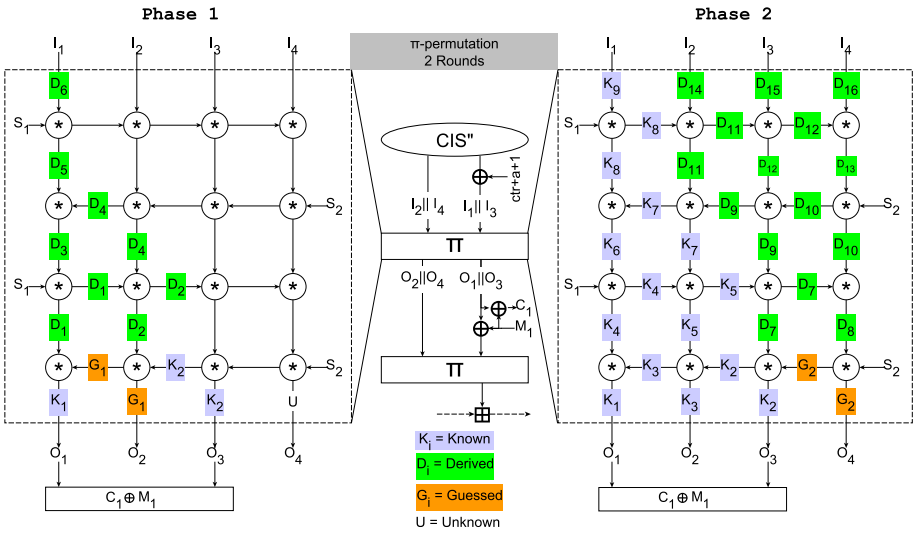

Fig. 7. Guess and determine phases for the attack on lightweight π-Cipher variants.

5 Complexity Analysis

Time complexity. The two steps of the attack with the highest time complexity are the guess and determine step, and the intersection of lists. The guess and determine step involves 16ω lists and we evaluate the π-function $2^{4\omega}$ times for each list. This gives a time complexity of $16\omega \times 2^{4\omega}$ evaluations of the π-function.

Each list will be stored as a bit-field: we use an array of $2^{4\omega}$ bits, where a bit b is set to one if and only if the value b is in the list. This allows to compute the intersection of two lists efficiently, with only $2^{4\omega}$ bit-operations. We have to compute $64\omega^2$ list intersections at Line 4 of Algorithm 1. This amounts to a total complexity of $64\omega^2 \times 2^{4\omega}$ bit-operations.

Since a computation of the π-function obviously requires more than 4ω bit-operations, we will neglect the time complexity of lists intersection, and the total complexity is $16\omega \times 2^{4\omega}$ evaluations of the π-function. This leads to a time complexity of 2^{72} when $\omega = 16$ and 2^{137} when $\omega = 32$.

Memory complexity. The memory complexity of the attack comes from the storage of lists. As explained above, each list $\mathcal{L}_i$ takes only $2^{4\omega}$ bits, for a total storage of $16\omega \times 2^{4\omega}$ bits. On the other hand, lists $\mathcal{L}'_0$ and $\mathcal{L}'_1$ contain $2^{4\omega}$ values of 16ω bits, so we must store the full values. We can store a single list, and compute the intersections with the second list on the fly, so that this step also requires $16\omega \times 2^{4\omega}$ bits of storage.

For $\omega = 16$ this leads to a memory complexity of 2^{69} bytes, while for $\omega = 32$, we need to store 2^{134} bytes.

Table 2 presents a summary of our attacks on different variants of π-Cipher. The last three columns of this table contain the time, data and memory complexities of the attacks.

Table 2. Summary of our attacks against different variants of π-Cipher. The data complexity is counted as the number of known plaintexts. The minimal number of blocks of each plaintext is denoted in the parenthesis.

Version	Variant	Word size ω	Security Claim	# Rounds Attacked	Time	Data (# KP)	Memory (bytes)
v1 & v2	π16-Cipher096	16	96	2.5/3	2^{72}	1 (256 B)	2^{69}
v1	π16-Cipher128	16	128	2.5/4	2^{72}	1 (256 B)	2^{69}
	π32-Cipher256	32	256	2.5/4	2^{137}	1 (512 B)	2^{134}
Lightweight	π16-Cipher096	16	96	2/2	2^{72}	1 (256 B)	2^{69}
	π16-Cipher128	16	128	2/2	2^{72}	1 (256 B)	2^{69}

6 Conclusion

In this work we provided an analysis of the security level offered by the π-Cipher family of authenticated ciphers. The designers of π-Cipher decided to decrease the number of rounds of the π-function from 4 to 3 for the second round of the CAESAR competition and to consider only 2 rounds for the recently proposed lightweight version. However, when reducing the number of rounds, special care must be taken, as this can lead to a dangerous reduction of the security margin offered by the new variants.

Our results indicate that π-Cipher, whose round function is reduced to 2.5 rounds, is vulnerable against guess and determine attacks. More precisely, we manage to recover the secret key in three reduced-round versions of the π-Cipher as well as in the two lightweight variants of the cipher. Taken together, these results suggest that the decision taken by the designers to reduce the number of rounds for the candidates of the second round of the CAESAR competition as well as for the lightweight version was risky.

In this work, we focused on the application of deterministic guess and determine properties. As a possible direction for future research, one can explore

other guess and determine methods for breaking the full version of the cipher. Alternatively, it would be also challenging to see if the analysis of the properties of the $\circledast$ operation could lead to the extension of our attack to an extra half round. Furthermore, a question that naturally arises after this analysis is whether increasing the number of rounds of the cipher is the only remedy to resist to our attack, or whether there is another tweak that could be applied to render the cipher immune against such type of cryptanalysis.

Acknowledgments. This work was initiated during the group sessions of the 5th Asian Workshop on Symmetric Cryptography (ASK 2015) held in Singapore. Christina Boura and Gaëtan Leurent are partially supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015. Avik Chakraborti and Goutam Paul are thankful to the Centre of Excellence in Cryptology (Project CoEC) and R. C. Bose Centre for Cryptology and Security of Indian Statistical Institute for partial support towards their work. Finally, the work of Hadi Soleimany is partly supported by grants from IPM and Shahid Beheshti University.

References

1. AlFardan, N.J., Paterson, K.G.: Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In: Society, I.C. (ed.) IEEE Symposium on Security and Privacy 2013 (2013)
2. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the Indifferentiability of the Sponge Construction. In: EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer (2008)
3. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness (2014), http://competitions.cr.yp.to/caesar.html/
4. Duong, T., Rizzo, J.: Here Come The XOR Ninjas
5. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. https://hal.inria.fr/inria-00075445 (2006)
6. Fuhr, T., Leurent, G.: Observation on π-Cipher. CAESAR's competition mailing list (November 2014)
7. Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., El-Hadedy, M., Jensen, R., Otte, D.: π-Cipher v2.0. Submission to the CAESAR competition (2014), http://competitions.cr.yp.to/caesar-submissions.html/
8. Gligoroski, D., Ødegård, R.S., Mihova, M., Knapskog, S.J., Kocarev, L., Drápal, A., Klima, V.: Cryptographic hash function EDON-$\mathcal{R}'$. In: 1st International Workshop on Security and Communication Networks. pp. 85–95. IEEE (2009)
9. Leurent, G.: Tag Second-preimage Attack against π-cipher. https://hal.inria.fr/hal-00966794 (March 2014)
10. Mihajloska, H., El-Hadedy, M., Gligoroski, D., Skadron, K.: Lightweight version of π-cipher. In: NIST Lightweight Cryptography Workshop 2015 (July 2015)