π-Cipher: Authenticated Encryption for Big Data

Danilo Gligoroski[1] and Hristina Mihajloska[2] and Simona Samardjiska[1,2] and Håkon Jacobsen[1] and Rune Erlend Jensen[3] and Mohamed El-Hadedy[1]

[1] ITEM, NTNU, Trondheim, Norway, danilog@item.ntnu.no, simonas@item.ntnu.no, hakoja@item.ntnu.no, hadedy@alumni.ntnu.no
[2] FCSE, "Ss Cyril and Methodius" University, Skopje, Republic of Macedonia, hristina.mihajloska@finki.ukim.mk
[3] IDI, NTNU, Trondheim, Norway, runeerle@stud.ntnu.no

Abstract. In today's world of big data and rapidly increasing telecommunications, using secure cryptographic primitives that are parallelizable and incremental is becoming ever more important design goal. π-Cipher is parallel, incremental, nonce based authenticated encryption cipher with associated data. It is designed with the special purpose of providing confidentiality and integrity for big data in transit or at rest. It has, as an option, a secret part of the nonce which provides nonce-misuse resistance. The design involves operations of several solid cryptographic concepts such as the Encrypt-then-MAC principle, the XOR MAC scheme and the two-pass sponge construction. It contains parameters that can provide the functionality of tweakable block ciphers for authenticated encryption of data at rest. The security of the cipher relies on the core permutation function based on ARX (Addition, Rotation and XOR) operations. π-Cipher offers several security levels ranging from 96 to 256 bits.

Keywords: Authenticated encryption, AEAD, parallelizability, incrementality, nonce-misuse resistance, sponge construction

1 Introduction

The possibility to have both encryption and authentication in one mode of operation of a block cipher, was first mentioned in Jutla's paper "Encryption Modes with Almost Free Message Integrity" [14] submitted to NIST [21] for their call for proposals for new modes of operation for AES. Soon after that, Jutla proposed a highly parallelizable authenticated encryption mode in [15]. Then, a whole series of proposals appeared in the open literature such as OCB [22], CCM [6], EAX [17] and GCM [18].

The interest for authenticated encryption with associated data was recently intensified with the announced new competition "CAESAR" for authenticated ciphers [2]. The scope of this competition is not just to seek for authenticated modes of operations for AES, but also for proposals of new ciphers that offer advantages over AES-GCM and are suitable for widespread adoption.

π-Cipher is a proposal for an authenticated cipher with associated data for the ongoing "CAESAR" crypto competition. The recent developments with the introduction of AES-NI instructions in latest Intel CPUs [12] made AES-GCM mode really efficient. Thus, in order to design a cipher that will offer advantages over AES-GCM we anticipated the following trends:

1. The exponential increase of the data in rest continues. Our modern civilization has entered the era where the total size of the digital universe has surpassed the zettabyte size and is entering the zettabyte communication era with a fast pace. For example "The EMC Digital Universe study - with research and analysis by IDC" [7] reports that in 2013 the size of our digital

universe was 4.4 zettabytes, and it projects that *"by 2020 the digital universe - the data we create and copy annually - will reach 44 zettabytes, or 44 trillion gigabytes."*
2. The trend of the exponential drop (in US$/MB) of the cost of computer memory and storage continues (for example see [8]).
3. The exponential increase of the global telecommunication traffic continues. In the report prepared by Cisco, the Cisco Visual Networking Index [5] predicts that *"Annual global IP traffic will pass the zettabyte threshold by the end of 2015, and will reach 1.4 zettabytes per year by 2017."*
4. The trend of increasing the number of cores in modern CPUs continues. From dual-core or quad-core processors nowadays we have processors with tens or even hundreds of cores.
5. The trend of increasing the SIMD (Single Instriction Multiple Data) computing power continues (from SSE registers with 128 bits, up to 256 bits in AVX2, and even 512 bits in the next AVX-512).

Taking everything previously mentioned in consideration, we designed π-Cipher with the following characteristics:

- The parallelism in π-Cipher is simpler than in AES-GCM.
- It is more suitable than AES-GCM for encrypting data in rest with tunable parameters that fit the sizes of modern disk sectors.
- Achieving incrementality with π-Cipher will be much simpler than achieving it with AES-GCM.
- It has, as an option, a secret part of the nonce which provides much more robust nonce-misuse resistance than AES-GCM.
- By tuning the parameters in π-Cipher, a certain level of tag preimage resistance can be achieved that is non-existing in AES-GCM.

Organization of the Paper. The rest of the paper is organized as follows. In Section 2 we provide a general description of the π-Cipher. The details of how it is designed are given in the following design rationale section. Section 4 presents some of the security goals and analysis of the π-Cipher, and Section 5 is dedicated to the software performances of it. Finally we conclude our paper in Section 6.

2 π-Cipher: General Description

π-Cipher is parallel, incremental, nonce based authenticated encryption cipher with associated data. Its design involves several solid cryptographic concepts such as:

1. Encrypt-then-MAC principle.
2. Its parallel and incremental design is similar to the design of the randomized XOR MAC scheme of Bellare et al. [1], but there are also some intrinsic differences.
   First of all, both schemes rely on the randomize-then-combine paradigm, by processing the message blocks using a keyed pseudo-random function, and then combining the results via a group operation. However, the π-cipher, in order to achieve light tag-second-preimage, when the key is known, instead of the XOR operation for the intermediate tag components (as in XOR MAC), uses componentwise addition of two d-dimensional vectors of ω-bit words in $(\mathbb{Z}_{2^\omega})^d$.
   In both cases, the independent processing of the message blocks provides straightforward parallelism and incrementality, and in both cases, two important mechanisms protect against forgeries, the inclusion of the ordinal number of the message blocks, and a publicly known nonce. The first protects against forgeries using rearrangements of the message blocks. The second protects against forgeries obtained by combining valid tags, and further ensures that

we always get a different tag even when encrypting the same message. But, there is also an important difference. In XOR MAC the publicly known nonce is processed independently of the message, but this approach is not desirable in an AEAD scheme, since we want the different nonce to influence all of the processed blocks. Thus, the nonce in π-cipher is included in the internal state and is an input to each of the calls of the internal permutation. Furthermore, the π-cipher also provides an option of using a secret nonce in addition to the public one.

3. In [4, Sec. 3.3] the authors mention that it is possible to construct an authenticated encryption using single pass sponge construction [3] that is proven to be secure as long as the underlying sponge permutation has no structural distinguishers. In the same paper [4] the duplex sponge construction is introduced. Although these constructions have the property to be tag second-preimage resistant, neither of them is incremental and parallel. An incremental and parallel two pass scheme is proposed in [19]. However, the design goal for that scheme was not to be nonce-misuse resistance. Combining all these ideas, in our design we use a two pass counter based sponge component that we call *triplex component*. Our design is fully parallelizable, incremental and lightly tag second-preimage resistant. The used π permutation is based on ARX (Addition, Rotation and XOR) operations.

2.1 Authenticated encryption

The encryption/authentication procedure of π-Cipher accepts key K with fixed-length of $klen$ bytes, message M with $mlen$ bytes and associated data AD with $adlen$ bytes. The cipher uses a fixed-length public message number PMN and secret message number SMN. The output of the encryption/authentication procedure is a ciphertext C with $clen$ bytes and a tag T with fixed-length of $tlen$ bytes. The length $clen$ of the ciphertext C is a sum of the byte length of the message, the authentication tag and the encrypted secret message number. The decryption/verification procedure accepts key K, associated data AD, ciphertext C, public message number PMN, secret message number SMN and tag T, and returns the decrypted message M if the tag has been verified or $\perp$ otherwise.

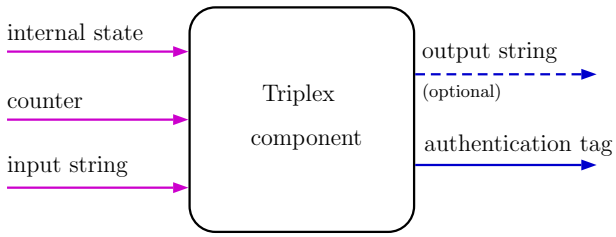
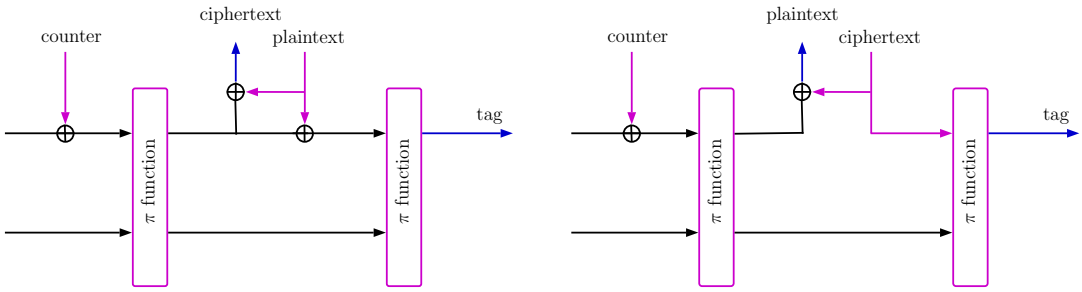


Fig. 1: A general scheme of the triplex component

The main building element in the operations of encryption/authentication and decryption/verification is our new construction related to the duplex sponge, called triplex component. It uses the permutation function π twice, it injects a counter into the internal state and digests an input string. The triplex component always outputs a tag. Optionally after the first call of the permutation function it can output a string (that can be a ciphertext block or a message block). The general scheme of the triplex component is presented in Figure 1.

Because of the differences in the encryption/authentication and decryption/verification procedures, there are two different variants of the triplex component. We call them *e-triplex* (for the phase of encryption) and *d-triplex* (for the phase of decryption). The only difference in these two

components is how the input string is treated after the first call of the permutation function. In the first one, the input string (plaintext) is XORed with the current internal state and the result proceeds to the second invocation of the permutation function π. In the d-triplex component, the input string (ciphertext) is directly injected as part of the internal state before the second invocation of the permutation function π. The graphical representation of the e-triplex and d-triplex components is given in Figure 2.



(a) e-triplex (b) d-triplex

Fig. 2: The Triplex component

The encryption/authentication operation of π-Cipher can be described in four phases:

1. **Initialization**. In this phase we first initialize the internal state to $K||PMN||10^*$, where 10^* denotes a single "1", and the smallest number of 0's (both in hexadecimal representation) such that the total length is less than or equal to the length of the internal state IS of the permutation function π. Then the internal state is updated by applying the permutation function π. Because, π-Cipher works in parallel mode, it has an initial value for the state for all of the parallel parts. We call it *Common Internal State (CIS)*. The CIS is initialized as:

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

The next part of this phase is initializing the counter ctr.
Since $CIS = CIS_{bitrate}||||CIS_{capacity}$ we initialize the ctr as the first 64 bits (little endian representation) of the $CIS_{capacity}$. $||||$ is an operator of interleaved concatenation in order to correctly denote a concatenation of the *bitrate* and *capaity* parts that restore the internal state. In this case, $CIS_{bitrate} = I_1||I_3||\ldots||I_{N-1}$ and $CIS_{capacity} = I_2||I_4||\ldots||I_N$, where I_i is a chunk of the internal state, and N is the number of chunks. The Initialization phase is depicted in Figure 3.

2. **Processing the associated data**.
The associated data $AD = AD_1||\ldots||AD_i||\ldots||AD_a$ is processed block by block in parallel using e–triplex components. The padding rule for the last block is the following:

$$AD_a \leftarrow \begin{cases} AD_a & \text{if } |AD_a| = bitrate, \\ AD_a||10^* & \text{if } |AD_a| < bitrate. \end{cases}$$

To every block AD_i we associate a unique counter calculated as a sum of the initial counter ctr and the ordinal number of the processed block i. The input to every e–triplex component is CIS, $ctr + i$ and AD_i, and the output is a block tag t'_i. The tag T' for the associated data is computed as

$$T' = \boxplus_{i=1}^{a}{}_d\, t'_i = t'_1 \boxplus_d t'_2 \boxplus_d \ldots \boxplus_d t'_a$$

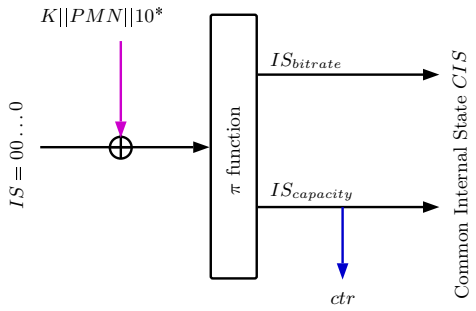


Fig. 3: Initialization step

where $\boxplus_d$ denotes component-wise addition, $t'_i = (t'_{i1}, t'_{i2}, \ldots, t'_{id}) \in (\mathbb{Z}_{2^\omega})^d$, and d is the number of ω-bit words in the *bitrate*.
The final part of this phase is to update the value of the *Common Internal State* CIS as

$$CIS \leftarrow \pi(CIS_{bitrate} \bigoplus T' \ \ |||| \ \ CIS_{capacity})$$

This step is described graphically in Figure 4.

3. **Processing the secret message number**. This phase is omitted if the length of the secret message number SMN is 0 (it is the empty string). If SMN is not the empty string, then the first step in this phase is a call to the e-triplex component. The input is the following triplet: $(CIS, ctr + a + 1, SMN)$, and the output is the following pair: (C_0, t_0). The second step of this phase is updating the CIS (for free) which becomes the value of the current internal state after the processing of SMN. Formally, the updating can be described by the following two expressions:

$$IS \leftarrow \pi(CIS_{bitrate} \bigoplus (ctr + a + 1) \ \ |||| \ \ CIS_{capacity}),$$
$$CIS \leftarrow \pi(IS_{bitrate} \bigoplus SMN \ \ |||| \ \ IS_{capacity})$$

The tag produced from this phase is $T'' = T' \boxplus_d t_0$. This phase is depicted in Figure 5.

4. **Processing the message**. The message $M = M_1||\ldots||M_j||\ldots||M_m$ is processed block by block in parallel by e–triplex components. The padding rule for the last block is the following:

$$M_m \leftarrow \begin{cases} M_m & \text{if } |M_m| = bitrate, \\ M_m||10^* & \text{if } |M_m| < bitrate. \end{cases}$$

To every block M_j we associate a unique block counter as:

$$ctr \leftarrow \begin{cases} ctr + a + j & \text{if } |SMN| = 0, \\ ctr + a + 1 + j & \text{if } |SMN| = bitrate, \end{cases}$$

where j is the ordinal number of the processed block in the message, and $0 < j \leqslant m$. The input to every e–triplex component is the CIS, block ctr and M_j, and the output is a pair (C_j, t_j). By definition we put that the length of the final ciphertext block C_m is the same as the length of the un-padded last plaintext block M_m i.e., $|C_m| = |M_m|$. The final tag T is obtained as:

$$T = T'' \boxplus_d t_1 \boxplus_d \ldots \boxplus_d t_j \boxplus_d \ldots \boxplus_d t_m,$$

where $t_j = (t_{j1}, t_{j2}, \ldots, t_{jd})$ is a d-dimensional vector of ω-bit words.
This phase is described graphically in Figure 6.

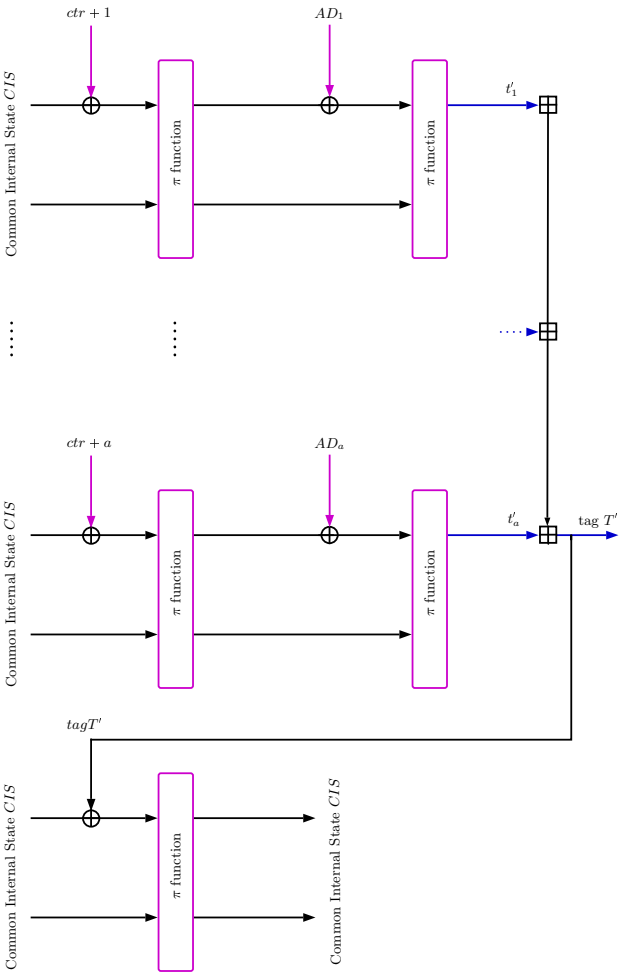


Fig. 4: Processing the associated data AD with a blocks in parallel

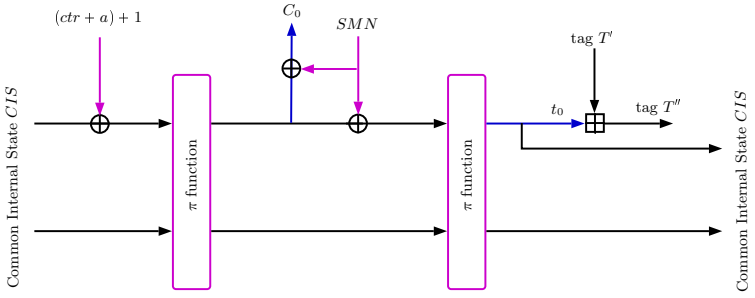


Fig. 5: Processing the secret message number SMN

The output of the encryption/authentication procedure is the ciphertext

$$C = C_0||C_1||\dots||C_m||T.$$

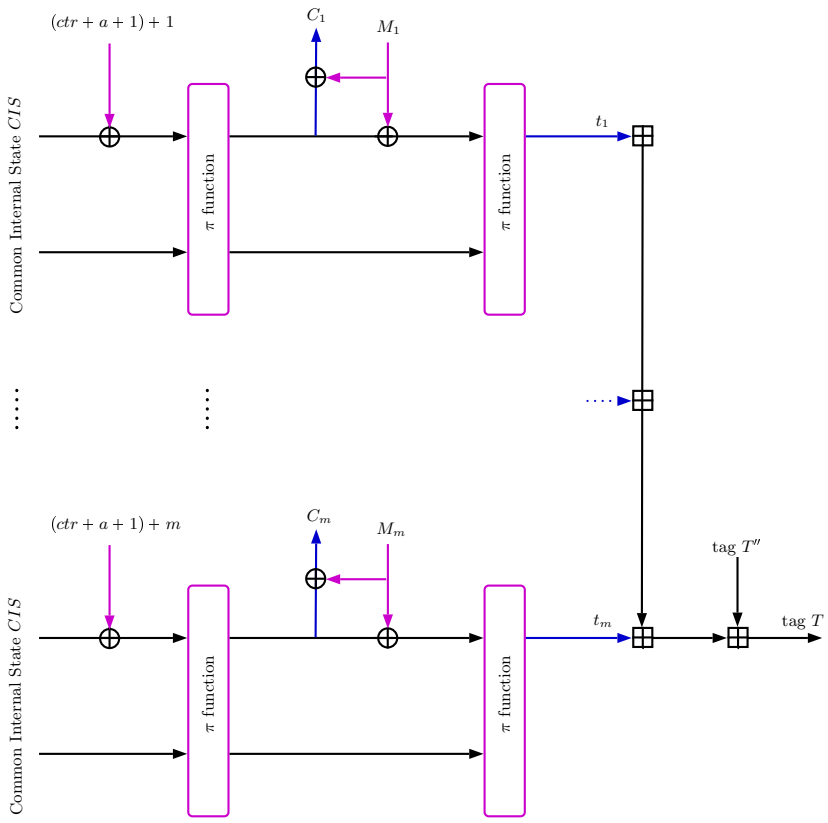


Fig. 6: Processing the message M with m blocks in parallel

2.2 Decryption and verification

The decryption/verification procedure is defined correspondingly. There are four phases and the only difference is in the last two (so the Initialization phase and Processing the associated data phase are completely the same as in the encryption/authentication procedure).

The decryption of the SMN is performed in the phase of Processing the secret message number. Thus, instead of using an e-triplex component, we use a d-triplex component. The input parameters are: CIS, incremented counter $ctr + a + 1$ and the ciphertext block C_0. The output is a pair (SMN, t_0). The tag is processed in the same way as in the encryption/authentication procedure.

For the decryption of the rest of the ciphertext we continue to use a d-triplex component (instead of e-triplex). The output is now a decrypted message block and a tag value.

At the end, the supplied tag value T is compared to the one computed by the algorithm. Only if the tag is correct, the decrypted message is returned.

2.3 The π-function

The core part of every sponge construction is the permutation function, and the whole security of the primitive relies on it. The design goal for our sponge construction was to obtain a strong permutation, which for different values of the parameter ω (the bit size of the words) provides different features, i.e. to be very efficient when $\omega = 64$ and lightweight when $\omega = 16$.

π-Cipher has an ARX based permutation function which we denote as π function. It uses similar operations as the ones used in the hash function Edon-R [11] but instead of using 8–tuples here we use 4–tuples. The permutation operates on a b bits state and updates the internal state through a sequence of R successive rounds. The state IS can be represented as a list of N 4-tuples, each of length ω-bits, where $b = N \times 4 \times \omega$, i.e.,

$$IS = (\underbrace{(IS_{11}, IS_{12}, IS_{13}, IS_{14})}_{I_1}, \underbrace{(IS_{21}, IS_{22}, IS_{23}, IS_{24})}_{I_2}, \ldots, \underbrace{(IS_{N1}, IS_{N2}, IS_{N3}, IS_{N4})}_{I_N}). \quad (1)$$

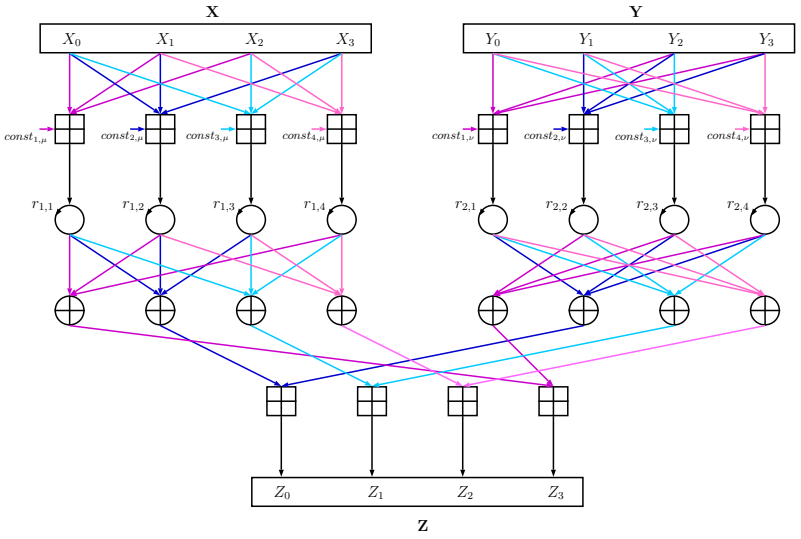


Fig. 7: Graphical representation of the ARX operation $*$.

The general permutation function π consists of three main transformations $\mu, \nu, \sigma : \mathbb{Z}_{2^\omega}^4 \rightarrow \mathbb{Z}_{2^\omega}^4$, where $\mathbb{Z}_{2^\omega}$ is the set of all integers between 0 and $2^\omega - 1$. These transformations perform diffusion and nonlinear mixing of the input. It uses the following operations:

- Addition + modulo 2^ω;
- Left rotation (circular left shift) $ROTL^r(X)$, where X is an ω–bit word and r is an integer, $0 \leqslant r < \omega$;
- Bitwise XOR operation $\oplus$ on ω–bit words.

Let $\mathbf{X} = (X_0, X_1, X_2, X_3)$, $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$ and $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ be three 4-tuples of ω–bit words. Further, denote by $*$ the following operation:

$$\mathbf{Z} = \mathbf{X} * \mathbf{Y} \equiv \sigma(\mu(\mathbf{X}) \boxplus_4 \nu(\mathbf{Y})) \quad (2)$$

where $\boxplus_4$ is the component-wise addition of two 4-dimensional vectors in $\left(\mathbb{Z}_{2^\omega}\right)^4$.

Table 1: An algorithmic description of the ARX operation $*$ for ω–bit words.

$*$ operation for ω–bit words

Input: $\mathbf{X} = (X_0, X_1, X_2, X_3)$ and $\mathbf{Y} = (Y_0, Y_1, Y_2, Y_3)$ where X_i and Y_i are ω–bit variables.
Output: $\mathbf{Z} = (Z_0, Z_1, Z_2, Z_3)$ where Z_i are *omega*–bit variables.
Temporary ω–bit variables: $T_0, \ldots, T_{11}$.

μ–transformation for X:

1. $T_0 \leftarrow ROTL^{r_{1,\omega,1}}(const_{1,\mu\omega} + X_0 + X_1 + X_2)$;
   $T_1 \leftarrow ROTL^{r_{1,\omega,2}}(const_{2,\mu\omega} + X_0 + X_1 + X_3)$;
   $T_2 \leftarrow ROTL^{r_{1,\omega,3}}(const_{3,\mu\omega} + X_0 + X_2 + X_3)$;
   $T_3 \leftarrow ROTL^{r_{1,\omega,4}}(const_{4,\mu\omega} + X_1 + X_2 + X_3)$;
2. $T_4 \leftarrow T_0 \oplus T_1 \oplus T_3$;
   $T_5 \leftarrow T_0 \oplus T_1 \oplus T_2$;
   $T_6 \leftarrow T_1 \oplus T_2 \oplus T_3$;
   $T_7 \leftarrow T_0 \oplus T_2 \oplus T_3$;

ν–transformation for Y:

1. $T_0 \leftarrow ROTL^{r_{2,\omega,1}}(const_{1,\nu\omega} + Y_0 + Y_2 + Y_3)$;
   $T_1 \leftarrow ROTL^{r_{2,\omega,2}}(const_{2,\nu\omega} + Y_1 + Y_2 + Y_3)$;
   $T_2 \leftarrow ROTL^{r_{2,\omega,3}}(const_{3,\nu\omega} + Y_0 + Y_1 + Y_2)$;
   $T_3 \leftarrow ROTL^{r_{2,\omega,4}}(const_{4,\nu\omega} + Y_0 + Y_1 + Y_3)$;
2. $T_8 \leftarrow T_1 \oplus T_2 \oplus T_3$;
   $T_9 \leftarrow T_0 \oplus T_2 \oplus T_3$;
   $T_{10} \leftarrow T_0 \oplus T_1 \oplus T_3$;
   $T_{11} \leftarrow T_0 \oplus T_1 \oplus T_2$;

σ–transformation for both $\mu(X)$ and $\nu(Y)$:

1. $Z_3 \leftarrow T_4 + T_8$;
   $Z_0 \leftarrow T_5 + T_9$;
   $Z_1 \leftarrow T_6 + T_{10}$;
   $Z_2 \leftarrow T_7 + T_{11}$;

An algorithmic definition of the $*$ operation over two 4–dimensional vectors $\mathbf{X}$ and $\mathbf{Y}$ for ω-bit words is given in Table 1. The values of the rotation vectors $r_{1,\omega}$ and $r_{2,\omega}$ and of the constants $const_{i,\mu\omega}$, $const_{i,\nu\omega}$, $i = 1, 2, 3, 4$ used in the μ and ν transformations are given in the official documentation of the π-Cipher [10]. A graphical representation of the $*$ operation is given in Figure 7.

Let us recall equation (1) where the internal state is presented as $IS = (I_1, I_2, \ldots, I_N)$. One round of the π function consists of two consecutive transformations E_1 and E_2 defined as follows.

Definition 1. *The function $E_1 : (\mathbb{Z}_{2^\omega}^4)^{N+1} \rightarrow (\mathbb{Z}_{2^\omega}^4)^N$ used in the π function is defined as:*

$$E_1(C, I_1, \ldots, I_N) = (J_1, \ldots, J_N),$$

where $J_1 = C * I_1, J_i = J_{i-1} * I_i,\ i = 2, \ldots, N$ and C is a 4-tuple of ω-bit constant values.

Definition 2. *The function $E_2 : (\mathbb{Z}_{2^\omega}^4)^{N+1} \rightarrow (\mathbb{Z}_{2^\omega}^4)^N$ used in the π function is defined as:*

$$E_2(C, I_1, \ldots, I_N) = (J_1, \ldots, J_N),$$

where $J_N = I_N * C, J_{N-i} = I_{N-i} * J_{N-i+1},\ i = 1, \ldots, N-1$ and$C$ is a 4-tuple of ω-bit constant values.

Finally, one round of the π function is defined as:

$$\pi(I_1, \ldots, I_N) = E_2(C2, E_1(C1, I_1, \ldots, I_N)) \tag{3}$$

One round of the cipher is graphically described in Figure 8. In the figure, the diagonal arrows can be interpreted as $*$ operations between the source and destination, and the vertical or horizontal arrows as equality signs " = ".

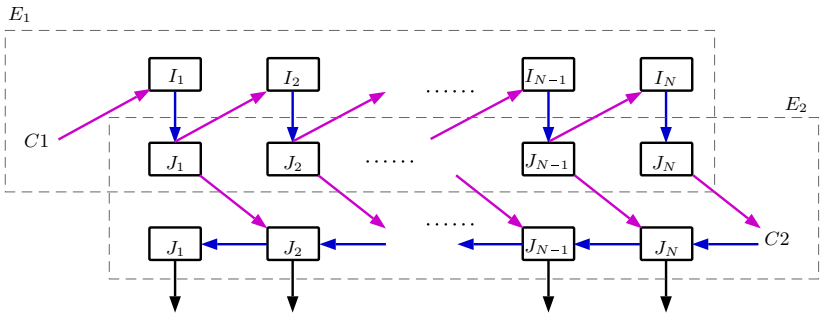


Fig. 8: One round of π-Cipher

The number of rounds R is a tweakable parameter. We recommend $R = 4$. The complete formula for the π function with $R = 4$ is the following:

$$\pi(I_1, \ldots, I_N) = E_2(C8, E_1(C7, E_2(C6, E_1(C5, E_2(C4, E_1(C3, E_2(C2, E_1(C1, I_1, \ldots, I_N)))))))))$$

The constants $C1, C2, \ldots, C8$ are generated in the same way as the constants of the $*$ operation and their values are given in the official documentation of the π-Cipher [10].

3 Design Rationale

π-Cipher is designed to be tweakable for different word sizes, different block sizes and different security levels. The recommended variants are presented in Table 2. The goal of π-Cipher is to achieve both high performance and strong security. To achieve the goal of having high performance we designed π-Cipher to be trivially parallel and to have a tweakable parameter that will enable it to work with wide sizes of the blocks (thus to be suitable to process multiple disk sectors in the same time). For example, the computer industry is moving to a 4096 (4K) byte sector size for hard disk drives called Advanced Format [13]. The new AF HDD storage devices read data from or write data to 4K-byte (4,096 bytes) physical sectors on the HDD media. However, using the encryption algorithm that can encrypt one sector as one block of message will also be an advantage. We can very efficiently use π64-Cipher128 or π64-Cipher256 with N=32 (instead of the recommended value N=4) for encrypting the whole disk, where every sector will be processed independently and in parallel.

Another design rationale about π-Cipher is its feature of being tag second-preimage resistant. While AES-GCM can be parallelized and can be used to perform incremental updates of the ciphertext and the tag, it is not tag second-preimage resistant. Still, since *Robustness* is a major goal for an AEAD cipher, we believe tag second-preimage resistance should be one of the features that it should posses. In other words, MACs should retain some hash properties when the key is known. There exist realistic scenarios (for ex. "Secure audit logs" and Multi-cast authentication) when the lack of tag second-preimage resistance in authenticated encryption can be exploited [9].

Table 2: Basic characteristics of all variants of the π-Cipher

	Word size ω (in bits)	$klen$ (in bits)	PMN (in bits)	SMN (in bits)	b (in bits)	N	$bitrate$ (in bits)	Tag t (in bits)	R
π16-Cipher096	16	96	32	0 or 128	256	4	128	128	4
π16-Cipher128	16	128	32	0 or 128	256	4	128	128	4
π32-Cipher128	32	128	128	0 or 256	512	4	256	256	4
π32-Cipher256	32	256	128	0 or 256	512	4	256	256	4
π64-Cipher128	64	128	128	0 or 512	1024	4	512	512	4
π64-Cipher256	64	256	128	0 or 512	1024	4	512	512	4

While the sponge constructions of authenticated encryption offer tag second-preimage resistance, they lack some of the properties that are also useful and desirable (such as parallelizability and incrementality). As a result of this line of reasoning, we designed a cipher for authenticated encryption that is parallel, incremental and offers *some level* of tag second-preimage resistance. We say some level, since the computational efforts for finding a second-preimage for a given pair *(message, tag)* are not the same as for finding second-preimages for hash functions. And if the π-Cipher is used for authenticated encryption of messages with arbitrary length (in the framework of the maximally allowed size of the messages which is up to 2^{64} bytes), as it was shown by Leurent in [16] the tag second-preimages can be computed with complexities from 2^{22} using messages that are long 2^{11} blocks up to complexity 2^{45} using messages that are long 2^{22} blocks. Laurent's analysis was based on Wagner's generalized birthday attack [23] which complexity can be described as follows. The complexity of finding a preimage message $M = M_1||M_2||\ldots||M_m$ where m is a power of 2 and $m \leqslant N_{max}$ (the largest number of blocks in any message that we plan to authenticate) is:

$$\min_{m \leqslant N_{max}} O(m \cdot 2^{\frac{tlen}{1+\lg\lceil m \rceil}}). \tag{4}$$

If the length of the messages is not restricted, then the minimum in equation (4) is achieved for messages of $m = 2^{\sqrt{tlen}-1}$ blocks.

However, there are situations when we need to encrypt relatively short messages. For example, if we use the π-Cipher in IPSec or TLS where the most common IP packet size is 1500 bytes, then for π64-Cipher128 or π64-Cipher256 the encrypted and authenticated message (each IP packet of a communication session) will have just 24 blocks ($m = 24$). In this case the Wagners generalized birthday attack in order to find a second-preimage for a given tag of one IP packet will need $2^{106.4}$ e-triplex invocations and a space of 2^{113} bytes.

We point out another property of the π-Cipher to address the issue of producing second-preimage resistant tags: the whole IP packet can be processed as a one block, by choosing the parameter $N = 48$ (instead of the recommended value $N = 4$). In this case, the tag second-preimage resistance is 2^{512} because of the fact that $m = 1$.

4 Security of π-Cipher

The following requirements should be satisfied in order to use π-Cipher securely:

1. The key should be secret and generated uniformly at random;
2. A nonce in the π-Cipher can be only PMN, or (PMN, SMN) pair;
   If the legitimate key holder uses the same nonce to encrypt two different pairs of (plaintext, associated data) (M_1, AD) and (M_2, AD) with the same secret key K then the confidentiality and the integrity of the plaintexts are not preserved in π–Cipher. This can be achieved under the assumption that PMN is always different for any two pairs of messages with the same key.

Additionally, π-Cipher offers an intermediate level of robustness when a legitimate key holder uses the same secret key K, the same associated data AD, the same public message number PMN but different secret message numbers SMN_1 and SMN_2 for encrypting two different plaintexts M_1 and M_2. In that case confidentiality and integrity of the plaintexts are preserved. However, in that case the confidentiality of SMN_1 and SMN_2 is not preserved.

3. If verification fails, the decrypted message and the wrong tag should not be given as an output;

Cipher-structure (Encrypt than MAC). First we want to point out that it is relatively straightforward to show that the π-Cipher is an Encrypt-then-MAC authenticated cipher. Let us recall the definition for the Encrypt-then-MAC authenticated cipher: We say that the authenticated cipher is Encrypt-then-MAC if a message M is encrypted under a secret key K_1 and then the tag T is calculated with another secret key K_2 as $MAC(K_2, C)$. The pair (C, T) is the output of the authenticated encryption procedure.

If we describe the e-triplex component used in π-Cipher in a mathematical form we have the following. First the message M is encrypted producing the ciphertext C as

$$IS \leftarrow \pi(CIS_{bitrate} \bigoplus counter \;\; |||| \;\; CIS_{capacity}),$$
$$C \leftarrow M \bigoplus IS_{bitrate}.$$

Then, the tag T is calculated as

$$t \leftarrow \pi(C \;\; |||| \;\; IS_{capacity})_{bitrate}.$$

Here, the value of $CIS_{bitrate} \bigoplus counter \;\; |||| \;\; CIS_{capacity}$ has the role of K_1 in the definition of Encrypt-then-MAC, and the value of $C \;\; |||| \;\; IS_{capacity}$ has the role of the pair (K_2, C) in the $MAC(K_2, C)$ part of the definition of Encrypt-then-MAC.

Associated Data and NONCE reuse. If we encrypt two different plaintexts M_1 and M_2 with the same secret key K, associated data AD and nonce $NONCE = (PMN, SMN)$, then neither the confidentiality nor the integrity of the plaintexts are preserved in the π–Cipher. However, as one measure to reduce the risks of a complete reuse of the $NONCE$ we have adopted the strategy of a composite $NONCE = (PMN, SMN)$. If either PMN or SMN are different, then both the confidentiality and integrity of plaintexts are preserved.

Plaintext corruption, associated-data corruption, message-number corruption, ciphertext corruption. We posit that the π-Cipher can straightforwardly be proven INT-CTXT secure under the assumption that the permutation π is an ideal random permutation without any structural distinguishers, by adapting the proof of the XOR-MAC scheme [1]. This is due to the close resemblance of the tag-generation part of the π-Cipher with the XOR-MAC.

Ciphertext prediction. The best distinguishing attack that we know for the π-Cipher is for the versions π16-Cipher096 and π16-Cipher128 with just one round and is described in [10]. The complexity of the attack is 2^{65} computations of the operation $*$, and the space is $2^{65} \times 16 = 2^{69}$ bytes.

Replay and reordering. For the π-Cipher, the standard defense against both replay and reordering is for the sender to use strictly increasing public message numbers PMNs, and for the receiver to refuse any message whose message number is no larger than the largest number of any verified message. This requires both the sender and receiver to keep state.

Sabotage. The π-Cipher puts the encryption of the SMN value as the first block of the ciphertext C. Thus, in protocols that use the π-Cipher, the receiver can make an early reject of invalid messages by decrypting the first block (containing the SMN) and comparing it to its expected value. Only if this check passes the receiver continues with the rest of the decryption and tag computation. Note however, that this requires the protocol to not return error messages to the sender, in order to avoid timing attacks. AES-GCM does not have this property.

Plaintext espionage. Since the attacker's goal here is to figure out the user's secret message, the only feasible attack can happen when the size of the secret message is small by building a table of encrypted secret messages. To defend against this attack the π-Cipher requires the nonce pair $NONCE = (PMN, SMN)$ to have a unique value for every encryption.

Message-number espionage. In the π-Cipher there is a dedicated phase for encrypting the secret message number SMN, and figuring out the value of SMN is equivalent to breaking the whole cipher which is infeasible under the assumptions that the permutation $\pi()$ is random.

General input scheduling. The π-Cipher can offer two ways for reducing the latency: **(1)** If the key K and the public message number PMN are known in advance and used repeatedly, then it is possible to precompute phase 1. and store the resulting Common Internal State (CIS) for subsequent applications of the cipher. **(2)** If the key K, the public message number PMN and the associated data AD are known in advance and used repeatedly, then it is possible to precompute both phase 1. *and* phase 2. for subsequent uses. In both cases, in order to preserve the confidentiality and the integrity of the plaintext, for every encryption the secret message numbers SMN's must be unique.

4.1 Security Analysis

We give a bit diffusion analysis for one round of the π function of π16-Cipher, π32-Cipher and π64-Cipher.

In our analysis, we examine the propagation of a one bit difference in a 1000 randomly generated Internal states IS for one round of the π function.

We performed several experiments for different word sizes ($\omega = 16, 32, 64$). We measure the Hamming distance between the outputs ($\pi(IS)$ and $\pi(IS')$), where one bit is changed in the IS ($HammingDistance(IS, IS') = 1$). The results for $\omega = 16, 32, 64$ are shown in Figure 9, Figure 10 and Figure 11.

5 Performances

Measurement bias has been shown to be commonplace and significant in real applications [20]. In order to minimize bias we have disabled any frequency scaling, disabled Address Space Layout Randomization and verified that there are no environment bias. Performance has been obtained on an Intel® Core™ i7-4771 Haswell CPU.

5.1 Software performance

For efficient software implementations, we propose to use the π64-Cipher. On modern Intel CPUs (Sandy Bridge and Haswell) the initial and slightly optimized non-SIMD implementation achieves between 6.5 cycles per byte (cpb) and up to 12 cpb depending on round count. On Haswell the non-SIMD version with 4 rounds have a performance around 11.8 cpb for longer messages. A preliminary 128-bit AVX version has a performance of around 9.4 cpb for longer messages, while it

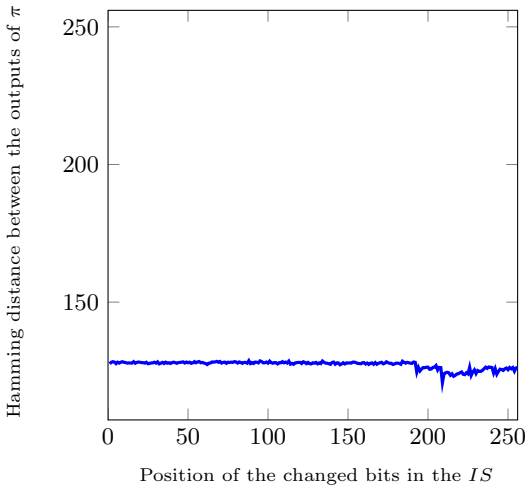


Fig. 9: Avalanche effect of one round of the π function where $\omega = 16$ ($Min = 120.732$, **Avg = 127.255**, $Max = 128.731$)

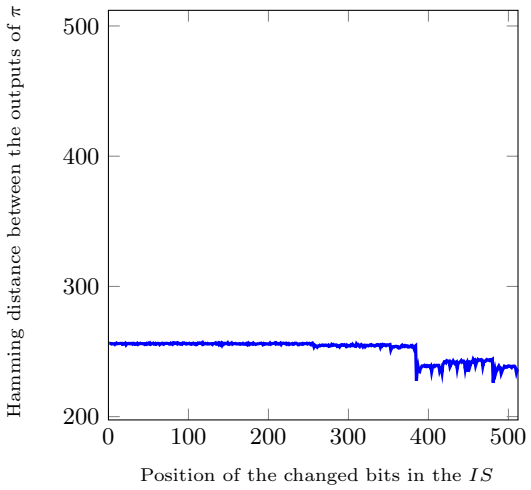


Fig. 10: Avalanche effect of one round of the π function where $\omega = 32$ ($Min = 226.063$, **Avg = 256.765**, $Max = 251.472$)

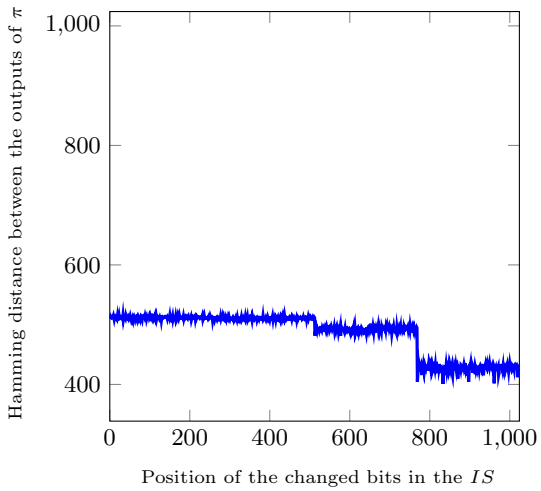


Fig. 11: Avalanche effect of one round of the π function where $\omega = 64$ ($Min = 400.88$, **Avg = 485.646**, $Max = 515.76$)

is around 5.1 cpb for a similar 256-bit AVX2 version. Both encoding and decoding have the similar performance.

For π-Cipher making a SIMD or GPU version is relatively simple as there are no crossing data dependencies. Namely, chunks of 64 bytes can be processed in parallel, and only a single SIMD wide horizontal addition is required at the very end, independent of how many chunks have been processed.

The main reason the performance scaling is not linear for the AVX version is that Haswell has more scalar pipelines than AVX and the lack of a AVX rotate instruction. Rotate must therefore be emulated with two shifts and one *or* instructions. For SSE up to 2 additional move instructions are needed as well. Intel's new 512-bit SIMD extention *AVX-512F* (AVX512 Foundation) has direct

support for 64 bit rotations. This will both cut around 1/3 of the required instructions and double the throughput.

The initial recommended number of rounds in π-Cipher is 4, and we presented here the measurements with that number of rounds. However, we expect the ongoing cryptanalysis to show that less number of rounds are also sufficiently secure and does not jeopardize the overall security of the cipher. That will make π-Cipher even faster than AES-GCM.

5.2 Lightweight hardware performance

For a lightweight hardware implementation we propose to use the π16-Cipher. Our initial and slightly optimized implementation of the basic operation $*$ on FPGA Xilinx Virtex6-XC6VLX240T needs 41 slices and two RAM blocks.

6 Conclusion

We have presented the design of π-Cipher. It is parallel, incremental, nonce based authenticated encryption cipher with associated data. It is designed with the special purpose of providing confidentiality and integrity for big data in transit or at rest and it offers many advantages compared with AES-GCM. Additionally, it has as an option, a secret part of the nonce which provides nonce-misuse resistance.

The design is based on several solid cryptographic concepts such as the Encrypt-then-MAC principle, the XOR MAC scheme and the two-pass sponge construction. It contains parameters that can provide the functionality of tweakable block ciphers for authenticated encryption of data at rest.

The security of the cipher relies on the core permutation function based on ARX (Addition, Rotation and XOR) operations. π-Cipher offers several security levels ranging from 96 to 256 bits.

References

1. Mihir Bellare, Roch Guérin, and Phillip Rogaway. Xor macs: New methods for message authentication using finite pseudorandom functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 1995.
2. D. J. Bernstein. Caesar: Competition for authenticated encryption: Security, applicability, and robustness. CAESAR web page, 2013. `http://competitions.cr.yp.to/index.html`.
3. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Proceedings of the 18th International Conference on Selected Areas in Cryptography*, SAC'11, pages 320–337, 2012.
5. Cisco. Cisco visual networking index: Forecast and methodology, 2012-2017. *White Paper*, May 2013. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf`.
6. N. Ferguson D. Whiting, R. Housley. Counter with cbc-mac (ccm). *IETF Request for Comments: 3610*, September 2003. `http://tools.ietf.org/html/rfc3610`.
7. EMC. The emc digital universe study with research and analysis by idc. *Open Report*, April 2014. `http://www.emc.com/leadership/digital-universe/index.htm?pid=home-dig-uni-090414`.
8. Electronics Freedom and Tech. Historical cost of computer memory and storage. *hblok.net*, February 2013. `http://hblok.net/blog/storage/`.
9. Danilo Gligoroski, Hristina Mihajloska, and Håkon Jacobsen. Should MAC's retain hash properties when the key is known in the next AEAD? Presentation at DIAC 2013, 2013. `http://2013.diac.cr.yp.to/slides/gligoroski.pdf`.

10. Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hakon Jacobsen, Mohamed El-Hadedy, and Rune Erlend Jensen. π-cipher v1. Cryptographic competitions: CAESAR, 2014. http://competitions.cr.yp.to/caesar-submissions.htmls.
11. Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, and Vlastimil Klima. Cryptographic hash function EDON-$\mathcal{R}'$. In *1st International Workshop on Security and Communication Networks*, pages 85–95, Trondheim, Norway, May 2009. IEEE.
12. Shay Gueron. Intel's new aes instructions for enhanced performance and security. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2009.
13. IDEMA. The advent of advanced format. *idema.org*, 2013. http://www.idema.org/?page_id=2369.
14. Charanjit S. Jutla. Encryption modes with almost free message integrity. Cryptology ePrint Archive, Report 2000/039, 2000. http://eprint.iacr.org/.
15. Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.
16. Gaëtan Leurent. Tag Second-preimage Attack against π-cipher. March 2014.
17. D. Wagner M. Bellare, P. Rogaway. A conventional authenticated-encryption mode. *NIST Modes Operation Symmetric Key Block Ciphers*, 2003. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf.
18. D. A. McGrew and J. Viega. The galois/counter mode of operation (gcm). *NIST Modes Operation Symmetric Key Block Ciphers*, 2005. http://www.csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcmrevised-spec.pdf.
19. Pawel Morawiecki and Josef Pieprzyk. Parallel authenticated encryption with the duplex construction. Cryptology ePrint Archive, Report 2013/658, 2013. http://eprint.iacr.org/.
20. Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. Producing wrong data without doing anything obviously wrong! In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIV, pages 265–276, New York, NY, USA, 2009. ACM.
21. National Institute of Standards and Technology (NIST). Modes development. Computer Security Resourc Center, 2000. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html.
22. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. Ocb: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM Conference on Computer and Communications Security*, pages 196–205. ACM, 2001.
23. David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.