PRIMATEs v1.02

Submission to the CAESAR Competition

Designers/Submitters:

Elena Andreeva[1], Begül Bilgin[1,2], Andrey Bogdanov[3], Atul Luykx[1], Florian Mendel[4], Bart Mennink[1], Nicky Mouha[1], Qingju Wang[1,5], and Kan Yasuda[6]

[1] KU Leuven, ESAT/COSIC, and iMinds Belgium.
[2] University of Twente, EEMCS/SCS, The Netherlands.
[3] Technical University of Denmark, DTU Compute, Denmark.
[4] Graz University of Technology, IAIK, Austria.
[5] Shanghai Jiao Tong University, Department of Computer Science and Engineering, China.
[6] NTT Secure Platform Laboratories, Japan.

`primates.ae`
`primates@esat.kuleuven.be`

September 8, 2014

Changes From v1.01

1. Clarification of the bounds on collision producing trails (Sect. 4.4.2).

Changes From v1.0

1. Included a ranking of the parameters as required by the CAESAR competition.
2. Removed statements describing APE's nonce as being optional.
3. Improved the descriptions of the algorithms, layout, and wording.
4. Added figures for both fractional and integral, associated data and message cases.

All security analysis performed on the schemes in v1.0 and the reference software implementation provided as v1 also holds for the schemes in v1.01 and v1.02.

Notation

Set $\mathbf{K} := \{0,1\}^k$, $\mathbf{T} := \{0,1\}^\tau$, $\mathbf{N} := \{0,1\}^\nu$, $\mathbf{R} := \{0,1\}^r$, $\mathbf{C} := \{0,1\}^c$, and $\mathbf{C}^{\frac{1}{2}} := \{0,1\}^{c/2}$. Given the state $X \in \mathbf{R} \times \mathbf{C}$, $X_r \in \mathbf{R}$ denotes its rate part and $X_c \in \mathbf{C}$ its capacity part. We write $0^r \in \mathbf{R}$ for a shorthand of $00\cdots0 \in \mathbf{R}$.

The bitwise XOR operation of the bit strings a_1 and a_2 is denoted by $a_1 \oplus a_2$. Both $a_1 \parallel a_2$ and a_1a_2 denote the concatenation of the bit strings a_1 and a_2.

An element of $\mathbf{R}$ is called a block. Let $\mathbf{R}^*$ denote the set of strings whose length is a non-negative multiple of r and at most $2^{c/2}$ blocks. Given a plaintext (message) $M \in \{0,1\}^*$, we divide it into blocks and write $M[1]M[2]\cdots M[w] \leftarrow M$, where each $M[i]$ for $i < w$ is a block and $M[w]$ is a string of length less than or equal to a block. We refer messages with $0 < |M[w]| < r$ as fractional messages (as opposed to integral messages where $|M[w]| = r$). When we write $M \parallel 10^*$, we mean that M is padded with a 1-bit and then zeros until the length of the resulting string is a multiple of r. By definition, an empty input message $M = \varnothing$ is also padded to the one zero-length block $M[1] \leftarrow 10^{r-1}$.

By $\lfloor M \rfloor_n$ we denote the n most significant bits of M (the n leftmost bits) whereas by $\lceil M \rceil_n$ we denote the n least significant bits.

Authenticated encryption with associated data (AEAD).

An authenticated encryption algorithm with associated data consists of a key generation $\mathcal{K}$, an encryption $\mathcal{E}$ and a decryption $\mathcal{D}$ algorithm. The encryption algorithm $\mathcal{E}$ takes as input a key $K \in \mathbf{K}$, associated data $A \in \mathbf{R}^*$, and a message $M \in \mathbf{R}^*$, and returns a ciphertext $C \in \mathbf{R}^*$ and a tag $T \in \mathbf{T}$, as $(C,T) \leftarrow \mathcal{E}_K(A,M)$. The decryption algorithm $\mathcal{D}$ takes as input a key $K \in \mathbf{K}$, associated data $A \in \mathbf{R}^*$, a ciphertext $C \in \mathbf{R}^*$, and a tag $T \in \mathbf{T}$, and returns either a message $M \in \mathbf{R}^*$ or the reject symbol $\perp$, as $M/\perp \leftarrow \mathcal{D}_K(A,C,T)$. The two functionalities $\mathcal{E}$ and $\mathcal{D}$ are sound, in the sense that

$$\mathcal{D}_K(A, \mathcal{E}_K(A,M)) = M,$$

for all K, A and M.

Nonce-based AEAD.

Whenever the AEAD scheme takes an additional nonce $N \in \mathbf{N}$ argument both in encryption and decryption we speak of nonce-based AEAD. The encryption algorithm is then defined as $(C,T) \leftarrow \mathcal{E}_K(N,A,M)$ and the decryption algorithm as $M/\perp \leftarrow \mathcal{D}_K(N,A,C,T)$ with the soundness condition $\mathcal{D}_K(N,A,\mathcal{E}_K(N,A,M)) = M$ holding for all N, K, A, and M.

Nonces.

A nonce $N \in \mathbf{N}$ is an unique non-repeating value, e.g. a counter. The nonces in this work are public values and we alternatively refer to them as public message numbers. We do not use secret message numbers. How the sender and receiver generate and synchronize nonces is left implicit as long as the uniqueness condition is satisfied.

	$s = 80$ bits	$s = 120$ bits
b (state size)	200 bits	280 bits
c (capacity size)	160 bits	240 bits
r (rate size)	40 bits	40 bits
permutations	PRIMATE-80	PRIMATE-120

Table 1: The security levels for the PRIMATEs family.

1 Parameters

The authenticated encryption family PRIMATEs is defined by the following two parameters:

1. The mode of operation $Scheme \in \{\mathsf{APE}, \mathsf{HANUMAN}, \mathsf{GIBBON}\}$.
2. The security level $s \in \{80, 120\}$ bits;

The security level will regularly be expressed in terms of bits (80 or 120 bits). The security level determines: the state size b, where the state consists of a rate part with r and a capacity part with c bits; and the permutation family PRIMATE-s. The PRIMATE-s : $\{0,1\}^b \rightarrow \{0,1\}^b$ family consists of four permutations p_1, p_2, p_3, and p_4. On the other hand, each mode of operation determines the key length k, the tag length τ, the nonce length ν, and the subset of permutations from PRIMATE-s.

For the purpose of the CAESAR competition, we rank the PRIMATEs as follows:

1. APE-120
2. HANUMAN-120
3. GIBBON-120
4. APE-80
5. HANUMAN-80
6. GIBBON-80

We note, however, that the different PRIMATEs serve different security goals, as we will clarify in this document.

1.1 Recommended Parameters

We recommend a security level s of either 80 or 120 bits for PRIMATEs family, as shown in Table 1.

In Table 2, we provide the respective key, tag and nonce values for the three modes where *Scheme*-s indicates the mode under $s = 80$ and $s = 120$ bits respectively. For HANUMAN and GIBBON we have identical values as compared to APE.

	APE-s	HANUMAN-s	GIBBON-s
k (key size)	$2s$	s	s
τ (tag size)	$2s$	s	s
ν (nonce size)	s	s	s
PRIMATE	p_1	p_1, p_4	p_1, p_2, p_3

Table 2: Key,tag and nonce values for the three modes of the PRIMATEs family.

When any of the parameters of the PRIMATEs are modified, a new key must be chosen uniformly at random. The length of plaintext and associated data processing is discussed in Sect. 2. Our recommendation for lightweight authenticated encryption is HANUMAN. For lightweight applications where speed is critical we recommend GIBBON and for lightweight environments where additional security requirements are needed or security is critical we recommend APE. The primary recommended security level is $s = 120$, whereas we recommend $s = 80$ for extremely lightweight applications. The primary recommended mode on the other hand is APE followed by HANUMAN and GIBBON. Hence, in this document, we prioritize security. APE is robust even when the nonce is misused whereas HANUMAN is secure as long as the nonce is unique and non-repeating. GIBBON is the most efficient at the cost of weaker security guarantees. As such, for different applications HANUMAN or GIBBON may be preferred over APE. We also note that a possible weakness observed in one of the modes, does not necessarily apply to the other modes.

2 Specification of PRIMATEs

We now provide the specifications of APE, HANUMAN and GIBBON. In this document, we specify that the message and associated data can consist of a non-integer number of bytes. However, due to restrictions on the API imposed by the CAESAR competition, all implementations that will be provided during the competition only support an integer number of bytes.

2.1 APE

The APE algorithm is described in Fig. 1. In APE, as opposed to HANUMAN and GIBBON, we treat the nonce the same way as the associated data whenever present. APE supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data and the plaintexts to be of size at most $2^{c/2}$ bits. For APE-80 this is approximately 2^{77} bytes and for APE-120 this is 2^{117} bytes. The APE algorithm uses the permutation p_1 together with its inverse p_1^{-1} for decryption. The key is used twice for: (1) part of the capacity of the initial state; and (2) after the tag generation. The fractional message cases are dealt with differently as compared to the integral data as elaborated below:

Consider a message M and denote its last block by $M[w]$, where $|M[w]| = |M| \bmod r$. We distinguish among three cases:

Algorithm 1: $\mathcal{E}_K(N, A, M)$

```
Input: $K \in \mathbf{C}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}$
$V \leftarrow 0^r \parallel K$
$N[1]N[2] \cdots N[y] \leftarrow N$
for $i = 1$ to $y$ do
    $V \leftarrow p_1\big(N[i] \oplus V_r \parallel V_c\big)$
end
if $A \neq \varnothing$ then
    $A[1]A[2] \cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \parallel 10^*$
    for $i = 1$ to $u$ do
        $V \leftarrow p_1\big(A[i] \oplus V_r \parallel V_c\big)$
    end
end
$V \leftarrow V \oplus (0^{b-1} \parallel 1)$
$M[1]M[2] \cdots M[w] \leftarrow M$
$l \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \parallel 10^*$
for $i = 1$ to $w$ do
    $V \leftarrow p_1\big(M[i] \oplus V_r \parallel V_c\big)$
    $C[i] \leftarrow V_r$
end
$C \leftarrow C[1]C[2] \cdots C[w-2]$
$C \leftarrow C \parallel \lfloor C[w-1] \rfloor_l$
$C \leftarrow C \parallel C[w]$
$T \leftarrow V_c \oplus K$
return $(C, T)$
```

Algorithm 2: $\mathcal{D}_K(N, A, C, T)$

```
Input: $K \in \mathbf{C}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $C \in \{0,1\}^*$, $T \in \mathbf{C}$
Output: $M \in \{0,1\}^*$ or $\perp$
$IV \leftarrow 0^r \parallel K$
$N[1]N[2] \cdots N[y] \leftarrow N$
for $i = 1$ to $y$ do
    $IV \leftarrow p_1\big(N[i] \oplus IV_r \parallel IV_c\big)$
end
if $A = \varnothing$ then
    $A[1]A[2] \cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \parallel 10^*$
    for $i = 1$ to $u$ do
        $IV \leftarrow p_1\big(A[i] \oplus IV_r \parallel IV_c\big)$
    end
end
$C[1]C[2] \cdots C[w] \leftarrow C$
$l \leftarrow |C[w]|$
$C[w] \leftarrow \lceil C[w-1] \rceil_{r-l} \parallel C[w]$
$C[w-1] \leftarrow \lfloor C[w-1] \rfloor_l$
$C[0] \leftarrow IV_r$
$V \leftarrow p_1^{-1}\big(C[w] \parallel K \oplus T\big)$
$M[w] \leftarrow \lfloor V_r \rfloor_l \oplus C[w-1]$
$V \leftarrow V \oplus M[w]10^* \parallel 0^c$
for $i = w-1$ to $1$ do
    $V \leftarrow p_1^{-1}(V)$
    $M[i] \leftarrow C[i-1] \oplus V_r$
    $V \leftarrow C[i-1] \parallel V_c$
end
$M \leftarrow M[1]M[2] \cdots M[w]$
if $IV_c = V_c \oplus (0^{c-1} \parallel 1)$ then
    return $M$
else
    return $\perp$
end
```

Figure 1: The APE encryption $\mathcal{E}_K(A, M)$ and decryption $\mathcal{D}_K(A, C, T)$ algorithms for fractional messages with $w \geq 2$.

- $|M[w]| \leq r-1$ and $w = 1$ (Figs. 6 and 7). Note that the corresponding ciphertext will be of r bits. This is required for decryption to be possible;
- $|M[w]| \leq r-1$ and $w \geq 2$ (Figs. 8 and 9). Note that the ciphertext $C[w-1]$ is of size equal to $M[w]$. The reason we opt for this design property is the following: despite $M[w]$ being smaller than r bits, we require its corresponding ciphertext to be r bits for decryption to be possible. As a consequence ciphertext $C[w-1]$ is of size equal to $M[w]$;
- $|M[w]| = r$ (Figs. 10 and 11). In this special case where M is an integral message, we employ a form of '10*'-padding. Instead of occupying an extra message block for this, the usual '10*'-padding spills over into the capacity. This can be seen as an XOR of $10 \cdots 00$ into the capacity part of the state.

The adjustments have no influence on the decryption algorithm $\mathcal{D}$, except if $|M| \leq r$ for which a slightly more elaborate function is needed. Note that the spilling of the padding in case $|M[w]| = r$ causes security to degrade by half a bit: intuitively, APE is left with a capacity of $c' = c - 1$ bits. We have opted for this degradation over an efficiency loss due to an additional round.

A similar spilling of the padding is also applied to the fractional associated data as indicated in Figs. 7, 9 and 11.

2.2 HANUMAN

The HANUMAN algorithm is described in Fig. 2. HANUMAN supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data and the plaintexts to be of size at most $2^{c/2}$ bits. For HANUMAN-80 this is approximately 2^{77} bytes and for HANUMAN-120 this is 2^{117} bytes. The algorithm uses two independent permutations, p_1 and p_4. The key is used twice for: (1) a part of the capacity of the initial state; and (2) after the tag truncation. A fractional input message (resp. associated data) is padded as usual by applying 10^* padding to the message (resp. associated data). In the case when $|M[w]| = r$ with M is an integral message, instead of occupying an extra message block for this, we employ the '10^*' spill over into the capacity, which also can be seen as an XOR of $10\cdots00$ into the capacity part of the state. The encryption procedure of HANUMAN for all choices of A and M are illustrated in Figs. 12, 13, 14 and 15.

```
Algorithm 3: $\mathcal{E}_K(N, A, M)$
Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
$V \leftarrow p_1(0^r \parallel K \parallel N)$
if $A \neq \varnothing$ then
    $A[1]A[2]\cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \parallel 10^*$
    for $i = 1$ to $u - 1$ do
        $V \leftarrow p_4(A[i] \oplus V_r \parallel V_c)$
    end
    $V \leftarrow p_1(A[u] \oplus V_r \parallel V_c)$
end
$M[1]M[2]\cdots M[w] \leftarrow M$
$\ell \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \parallel 10^*$
for $i = 1$ to $w$ do
    $C[i] \leftarrow M[i] \oplus V_r$
    $V \leftarrow p_1(C[i] \parallel V_c)$
end
$C \leftarrow C[1]C[2]\cdots C[w-1]\lfloor C[w] \rfloor_\ell$
$T \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $(C, T)$
```

```
Algorithm 4: $\mathcal{D}_K(N, A, C, T)$
Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
Output: $M \in \{0,1\}^*$ or $\perp$
$V \leftarrow p_1(0^r \parallel K \parallel N)$
if $A \neq \varnothing$ then
    $A[1]A[2]\cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \parallel 10^*$
    for $i = 1$ to $u - 1$ do
        $V \leftarrow p_4(A[i] \oplus V_r \parallel V_c)$
    end
    $V \leftarrow p_1(A[u] \oplus V_r \parallel V_c)$
end
$C[1]C[2]\cdots C[w] \leftarrow C$
$\ell \leftarrow |C[w]|$
for $i = 1$ to $w - 1$ do
    $M[i] \leftarrow C[i] \oplus V_r$
    $V \leftarrow p_1(C[i] \parallel V_c)$
end
$M[w] \leftarrow \lfloor V_r \rfloor_\ell \oplus C[w]$
$V \leftarrow p_1((M[w] \parallel 10^* \oplus V_r) \parallel V_c)$
$M \leftarrow M[1]M[2]\cdots M[w-1]M[w]$
$T' \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $T = T'$ ? $M : \perp$
```

Figure 2: The HANUMAN encryption $\mathcal{E}_K(N, A, M)$ and decryption $\mathcal{D}_K(N, A, C, T)$ algorithms for fractional messages.

2.3 GIBBON

The GIBBON algorithm is described in Fig. 3. GIBBON supports variable length associated data and plaintexts. As discussed in Sect. 3 we recommend the associated data and the plaintexts to be of size at most $2^{c/2}$ bits. For GIBBON-80 this is approximately 2^{77} bytes and for GIBBON-120 this is 2^{117} bytes. The algorithm uses three independent permutations, p_1, p_2 and p_3. The key K is used for: (1) a part of the capacity of the initial state; (2) after the initialization (first p_1 iteration); (3) before the finalization (last p_1 iteration); and (4) after the tag truncation. A fractional input message (resp. associated data) is padded as usual by applying 10^* padding to the message (resp. associated data). In the case when $|M[w]| = r$ with M an integral message, instead of occupying an extra message block for this, we employ the '10^*' spill over into the capacity, which also can be seen as an XOR of $10\cdots00$ into the capacity part of the state. The encryption procedure of GIBBON for all choices of A and M are illustrated in Figs. 16, 17, 18 and 19.

Algorithm 5: $\mathcal{E}_K(N, A, M)$

```
Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
$V \leftarrow p_1(0^r \| K \| N)$
$V \leftarrow V_r \| (K \| 0^{\frac{c}{2}}) \oplus V_c$
if $A \neq \varnothing$ then
    $V \leftarrow p_2(V)$
    $A[1]A[2]\cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \| 10^*$
    for $i = 1$ to $u-1$ do
        $V \leftarrow p_2(A[i] \oplus V_r \| V_c)$
    end
    $V \leftarrow A[u] \oplus V_r \| V_c$
end
$M[1]M[2]\cdots M[w] \leftarrow M$
$\ell \leftarrow |M[w]|$
$M[w] \leftarrow M[w] \| 10^*$
$V \leftarrow p_3(V)$
for $i = 1$ to $w$ do
    $C[i] \leftarrow M[i] \oplus V_r$
    $V \leftarrow p_3(C[i] \| V_c)$
end
$V \leftarrow p_1(V_r \| (K \| 0^{\frac{c}{2}}) \oplus V_c)$
$C \leftarrow C[1]C[2]\cdots C[w-1]\lfloor C[w]\rfloor_\ell$
$T \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $(C, T)$
```

Algorithm 6: $\mathcal{D}_K(N, A, C, T)$

```
Input: $K \in \mathbf{C}^{\frac{1}{2}}$, $N \in \mathbf{C}^{\frac{1}{2}}$, $A \in \{0,1\}^*$,
       $C \in \{0,1\}^*$, $T \in \mathbf{C}^{\frac{1}{2}}$
Output: $M \in \{0,1\}^*$ or $\perp$
$V \leftarrow p_1(0^r \| K \| N)$
$V \leftarrow V_r \| (K \| 0^{\frac{c}{2}}) \oplus V_c$
if $A \neq \varnothing$ then
    $V \leftarrow p_2(V)$
    $A[1]A[2]\cdots A[u] \leftarrow A$
    $A[u] \leftarrow A[u] \| 10^*$
    for $i = 1$ to $u-1$ do
        $V \leftarrow p_2(A[i] \oplus V_r \| V_c)$
    end
    $V \leftarrow A[u] \oplus V_r \| V_c$
end
$C[1]C[2]\cdots C[w] \leftarrow C$
$\ell \leftarrow |C[w]|$
$V \leftarrow p_3(V)$
for $i = 1$ to $w-1$ do
    $M[i] \leftarrow C[i] \oplus V_r$
    $V \leftarrow p_3(C[i] \| V_c)$
end
$M[w] \leftarrow \lfloor V_r \rfloor_\ell \oplus C[w]$
$V \leftarrow p_3((M[w] \| 10^* \oplus V_r) \| V_c)$
$M \leftarrow M[1]M[2]\cdots M[w-1]M[w]$
$V \leftarrow p_1(V_r \| (K \| 0^{\frac{c}{2}}) \oplus V_c)$
$T' \leftarrow \lfloor V_c \rfloor_{\frac{c}{2}} \oplus K$
return $T = T'$ ? $M$ : $\perp$
```

Figure 3: The GIBBON encryption $\mathcal{E}_K(N, A, M)$ and decryption $\mathcal{D}_K(N, A, C, T)$ algorithms for fractional messages.

2.4 PRIMATE Permutation

The underlying permutation of PRIMATEs which is called PRIMATE is inspired by [8]. It has two different sizes (we write PRIMATE-80 for a 200-bit permutation and PRIMATE-120 for a 280-bit one) as well as 4 variants of each size (referred to as p_1, p_2, p_3 and p_4). PRIMATE is designed according to the wide trail strategy [11] and its structure resembles the data transform part of the Rijndael block cipher [12]. PRIMATE-80 and PRIMATE-120 operate on a 5×8 and a 7×8 state of 5-bit elements, respectively.

The first row of the state (5 bytes) is the rate of the state whereas the rest of the state is the capacity for both sizes. The state and each individual element possess big-endian encoding. PRIMATE update the internal state by means of the sequence of transformations

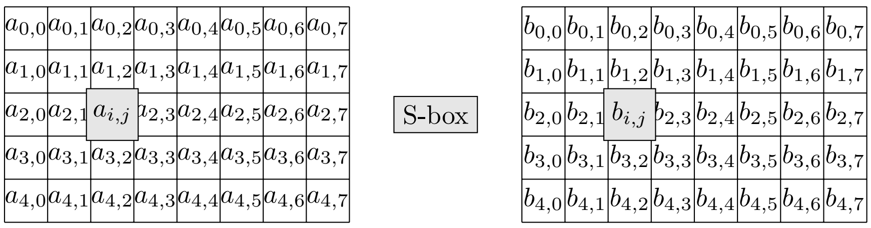
$$\mathsf{CA} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SE}\ .$$

The four permutations p_1, p_2, p_3 and p_4 of PRIMATE are defined by means of different round constants, which are generated by a 5-bit LFSR, and different number of rounds as shown in the following table.

	p_1	p_2	p_3	p_4
Number of rounds	12	6	6	12
Initial value of the LFSR	1	24	30	24

2.4.1 SubElements (SE).

The SubElements step is the only non-linear transformation of PRIMATE. It is a permutation consisting of a 5-bit S-box applied to each element of the state (shown below for PRIMATE-80).



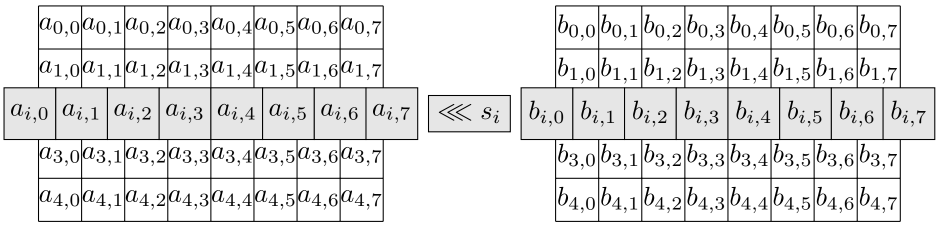
The S-box is an almost bent (AB) permutation as defined in Table 3. The maximum differential and linear probability for this S-box is 2^{-4}, which is best attainable [10] and, thus, optimal in this sense. This particular S-box has been chosen from the AB permutation set such that the area of both plain and shared implementation provide a good tradeoff, cf. [8].

Table 3: 5-bit S-box (decimal).

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	1	0	25	26	17	29	21	27	20	5	4	23	14	18	2	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S(x)	15	8	6	3	13	7	24	16	30	9	31	10	22	12	11	19

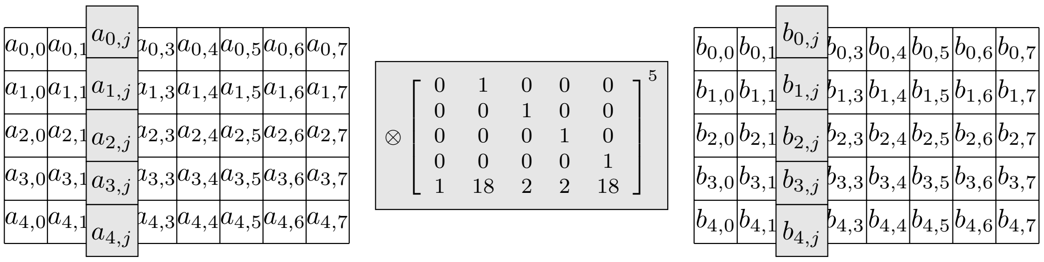
2.4.2 ShiftRows (SR).

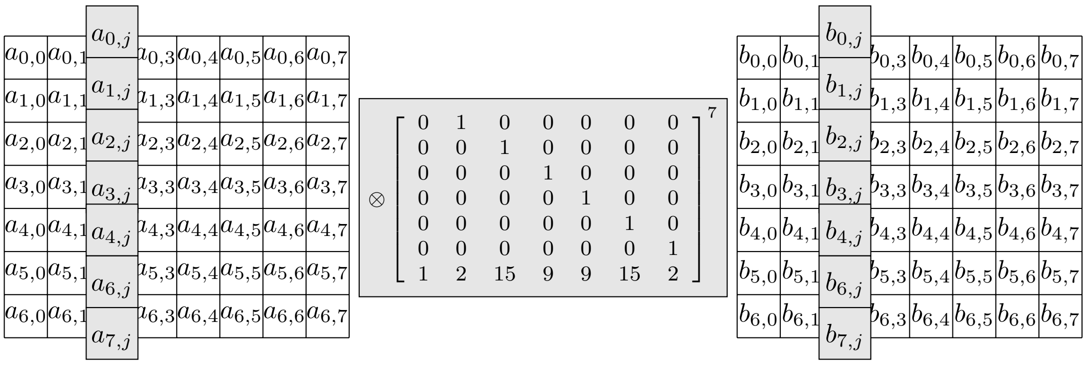
The ShiftRows step is an element transposition that cyclically shifts the rows of the state over different offsets. Row i is shifted left by $s_i = \{0, 1, 2, 4, 7\}$ positions for PRIMATE-80 (shown below) and by $s_i = \{0, 1, 2, 3, 4, 5, 7\}$ positions for PRIMATE-120. Since ShiftRows is only wiring in hardware, its overall cost is negligible.



2.4.3 MixColumns (MC).

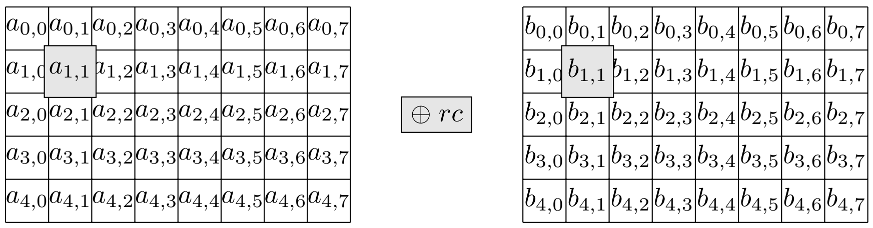
The MixColumns step is operating on the state column by column. It is a left-multiplication by a 5×5 (resp. 7×7) matrix over $\mathbb{F}_{2^5} \cong \mathbb{F}_2[x]/(x^5 + x^2 + 1)$. The main design goal of the MixColumns transformation is to follow the wide trail strategy and that it can be implemented efficiently. Therefore, we use a recursive approach [3, 14, 15, 22] to generate an MDS matrix that has a maximum (6 and 8 respectively) *branch number* (the smallest nonzero sum of active inputs and outputs of each column).


$\otimes \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 18 & 2 & 2 & 18 \end{bmatrix}^5$


$\otimes \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 15 & 9 & 9 & 15 & 2 \end{bmatrix}^7$

2.4.4 ConstantAddition (CA).

In this transformation the state is modified by combining the second element of the second row with a predefined constant by a bitwise XOR operation. The purpose of adding round constants is to make each round different and to break the symmetry of the other transformations. Furthermore, it provides a natural opportunity to make the parts for processing associated data and message different from each other if needed. A 5-bit Fibonacci LFSR with taps in the first (i.e. the most significant) bit and the fourth bit is used to generate the round constants rc. Therefore, the hardware implementation of ConstantAddition is in fact very cheap.



3 Security Claims

The designers claim the following levels of security, expressed in bits:

	Scheme-s	*Scheme-80*	*Scheme-120*
confidentiality of M	$c/2$	80	120
integrity of M	$c/2$	80	120
integrity of A	$c/2$	80	120
integrity of N	$c/2$	80	120

The claimed security levels correspond to the birthday bound security on the capacity of PRIMATEs-80 and PRIMATEs-120, respectively (see also Sect. 4). The security of GIBBON and HANUMAN relies on the uniqueness of the nonce, whereas APE is robust against nonce reuse. Technically, this implies that all security results of APE only hold up to common prefix: under the same associated data and nonce, two messages with the same prefix (in r-bit blocks) have the same corresponding ciphertext blocks. We refer to [1] for the technicalities.

The designers claim that APE offers certain additional security benefits. Most importantly, it is secure under the release of unverified plaintext (RUP). This means that APE is still secure if the decryption algorithm is implemented so as to output the decrypted plaintext before successful verification. This scenario arises for example when devices have insufficient memory to store the entire plaintext [13], or when the decrypted plaintext needs to be processed early due to real-time requirements [9, 21]. We refer to [2] for more information on the security under the release of unverified plaintext.

	APE-s	APE-80	APE-120
confidentiality under RUP	$c/2$	80	120
integrity under RUP	$c/2$	80	120

4 Security Analysis

PRIMATEs are indistinguishable from an ideal authenticated encryption scheme up to about $2^{c/2}$ primitive calls; implying that PRIMATEs achieve a security level of $c/2$ bits. This result is proven in the ideal model, where the underlying primitive permutations PRIMATE are assumed to be perfectly random permutations.

$a_{0,0}$ $a_{0,1}$ … $a_{4,7}$
$\oplus\, rc$
$b_{0,0}$ $b_{0,1}$ … $b_{4,7}$

4.1 APE

The security results for APE can be found in [1]. APE is the first and currently the only misuse resistant permutation based authenticated encryption. The security results for APE apply *both* in the cases when nonces are unique values (full security) and also when nonces are reused (full security up to common prefix, the maximum attainable for single pass schemes). As a mode of operation for a permutation, APE is secure in the ideal model. Considering a distinguisher whose queries are of total length at most m blocks, APE is proven secure in the ideal model up to a bound of $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^c}$ (integral messages) and $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^{c-1}}$ (for fractional messages) [1].

We can also look at APE as a mode of operation for a block cipher where we replace the operation $(0^r \parallel K) \oplus p_1 \oplus (0^r \parallel K)$ with that of a block cipher (see [1] for a more detailed explanation). This version of APE is secure in the standard model, meaning if the underlying block cipher is a secure strong pseudorandom permutation (SPRP), then APE with a block cipher is secure as well. The bounds from the ideal model also hold in the standard model, up to twice the SPRP security of E_K. We interpret this result to mean that if $(0^r \parallel K) \oplus p_1 \oplus (0^r \parallel K)$ with p_1 instantiated by a PRIMATE is a secure SPRP, then APE with a PRIMATE is secure as well.

In the same vein, a formal security proof for APE in the case unverified plaintext is given in [2]. In more detail, the authors introduce a security model for the analysis of authenticated encryption schemes in the case when unverified plaintext is released upon decryption. In this model APE is proven to be secure upon release of unverified plaintext with no security loss (compared to the above-mentioned bounds).

Taking $c = 160$, $r = 40$ for APE-80 or $c = 240$, $r = 40$ for APE-120, the security levels approach the ones claimed in Sect. 3, but not exactly. For instance, for APE-80 we claim 80-bit security, while the proven security bound (fractional case) satisfies $\frac{m^2}{2^{r+c}} + \frac{2m(m+1)}{2^c} = \frac{1}{2}$ for $m \approx 2^{79.5}$. Similarly, for the fractional case the security bound equals $\frac{1}{2}$ for $m \approx 2^{79}$. The difference is due to the security model and proof techniques applied.

4.2 HANUMAN

HANUMAN follows a design similar to that of SpongeWrap [7]. The scheme constructs a keystream which depends on the nonce (public message number) and message, with which the message is then XORed to produce the ciphertext. As long as the nonce remains unique for each encryption, confidentiality will be achieved since the keystream will be close to uniformly random, assuming the PRIMATE permutations are close to ideal. Note that if the nonce is repeated, then the XOR of the first message blocks can be determined from the XOR of the ciphertexts. Associated data is processed via an independent permutation in order to prevent forgery attacks in which a message is first encrypted as associated data, and then again as plaintext. Attacks can be found if a collision occurs in the capacity, yet this is expected to happen only after roughly $2^{c/2}$ total queries to the underlying permutations. It is also assumed that if the verification step of the algorithm reveals that the ciphertext has been tampered with, then the algorithm returns no information beyond the verification failure.

4.3 GIBBON

The structure of GIBBON is similar to the MonkeyWrap [5] construction. The scheme generates a stream of a ciphertext and a tag depending on the nonce (public message number) and the message. Security is achieved as long as the nonce is used only once with the same key. It is also assumed that if the verification step of the algorithm reveals that the ciphertext has been tampered with, then the algorithm returns no information beyond the verification failure. In particular, no plaintext blocks are returned. A state recovery for GIBBON does not lead to trivial key recovery and also does not lead to trivial universal forgery attacks due to the key additions.

4.4 PRIMATE

This section shows some known properties of the non-linear permutation PRIMATE.

4.4.1 Differential and Linear Trails

PRIMATE has diffusion properties according to the wide trail design strategy and hence provides good bounds against differential an linear cryptanalysis. We use the technique in [19] to calculate the differential and linear hull probabilities of PRIMATE. Since the 5-bit S-box of PRIMATE is an almost bent (AB) permutation, the maximum differential and linear probability for this S-box is 2^{-4}, which provides optimal security against linear and differential cryptanalysis [10].

For PRIMATE-80, as the branch number of the linear diffusion is 6, the differential/linear probability over any two rounds does not exceed $16 \cdot (2^{-4})^6 = 2^{-20}$ and the differential/linear probability over any four rounds is upper-bounded by $(2^{-20})^5 = 2^{-100}$. For PRIMATE-120, the branch number of the linear diffusion is 8 and, therefore, the differential/linear probability over any four rounds of PRIMATE-120 is upper-bounded by $(16 \cdot (2^{-4})^8)^7 = 2^{-196}$.

This means that the probability of any twelve-round differential (and linear approximation) in PRIMATE-80 (respectively, PRIMATE-120), assuming independent rounds, does not exceed 2^{-100} (respectively, 2^{-196}). Thus, there is only a very small chance that the standard differential or linear approach would lead to a successful attack here.

4.4.2 Collision Producing Trails

Assume we have a certain difference for the message that may result in a zero difference in the state with a high probability after the difference has been injected. We call the trails corresponding to this behaviour collision producing trails. They can be used in a forgery attack on PRIMATE. Note that a linear trail of a similar shape might be used for a distinguishing attack on the keystream of PRIMATE.

The simple design of PRIMATE allows to prove good bounds against this kind of differential and linear attacks. To obtain such bounds, we adopt the mixed-integer linear programming (MILP) technique proposed in [18] to find the minimum number of differentially and linearly active S-boxes of the target ciphers. Using this technique and the optimizer CPLEX [17], we obtained the results provided in Table 4 for PRIMATE-80 and PRIMATE-120.

Table 4: Bounds for the minimum number of active S-boxes for collision producing trails of PRIMATE-80 and PRIMATE-120.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12
PRIMATE-80	44	48	52	56	61	84	105	113	117	122	145	162
PRIMATE-120	63	64	72	78	113	128	143	152	177	193	209	224

For 6 rounds of GIBBON-80 (and GIBBON-120), we find that trails with at least 84 (respectively, 128) active S-boxes can produce a collision. This results in an upper bound on the differential and linear trail probability of 2^{-336} (respectively, 2^{-512}).

For 12 rounds of HANUMAN and APE, we find that a collison requires at least 162 active S-boxes of HANUMAN-80/APE-80 (224 of HANUMAN-120/APE-120 respectively). The upper bounds of the differential and linear trail probability are therefore 2^{-648} and 2^{-896} respectively.

4.4.3 Impossible Differential Cryptanalysis

Now we discuss the application of impossible differential cryptanalysis to PRIMATE. Since the branch number of PRIMATE-80 and PRIMATE-120 is 6 and 8, respectively, the number of nonzero element differences in each column before and after the MixColumns operation can never be smaller than these values. Based on this property, we construct impossible differentials for 6 and 5 rounds of PRIMATE-80 and PRIMATE-120, respectively, which are depicted in Figures 4 and 5.

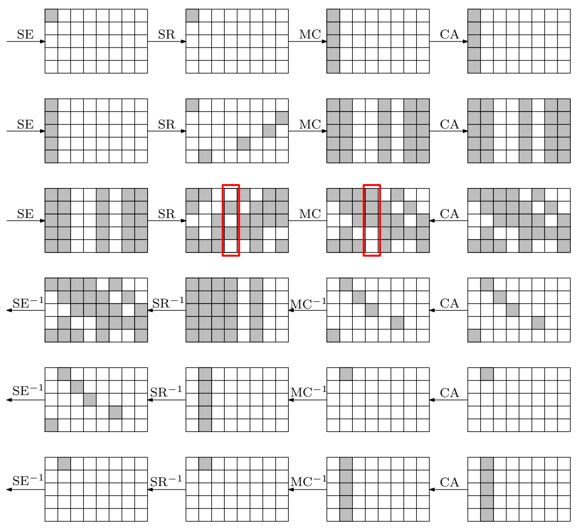


Figure 4: Impossible differential for 6 rounds of PRIMATE-80.

Taking PRIMATE-80 as an example, assume we start from the first round, if there is a nonzero difference at position (0,0) of the state, then after 2.5 rounds of PRIMATE encryption, the vector in column 3 before the MixColumns operation is $(0, *, 0, *, 0)^T$, where "*" denotes a nonzero difference. At the same time, if there is a nonzero difference in column 1 after the MixColumns of round 6 at the bottom of the distinguisher, there is a single nonzero difference at position (0,1) before MixColumns in round 5, which leads to the output vector in column 3 after MixColumns operation in round 3

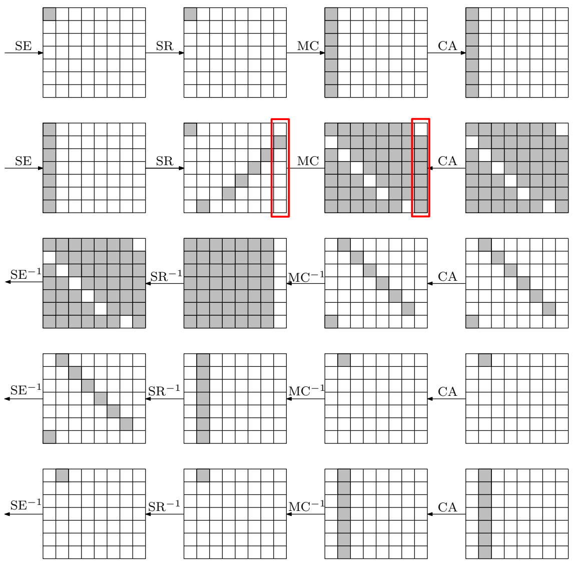


Figure 5: Impossible differential for 5 rounds of PRIMATE-120.

to be $(*,*,*,0,0)^T$. These columns are highlighted in red in Figure 4. This means that $M(0,*,0,*,0)^T = (*,*,*,0,0)^T$, that is, that the number of nonzero element differences before and after MixColumns is 5 which contradicts to the branch number being 6. Therefore, a 6-round impossible differential has been constructed for PRIMATE-80. Similarly, we can obtain the 5-round impossible differential of Figure 5 for PRIMATE-120.

5 Features of PRIMATEs

Permutation-based AE for lightweight applications.

The PRIMATEs authenticated encryption family is designed for *lightweight* cryptographic applications. The domain of lightweight cryptography focuses on cryptographic algorithms for extremely constrained hardware devices, where the goal is to implement an efficient cryptographic algorithm using only a very limited number of gates.

At the the core of the PRIMATEs family are the PRIMATE permutations. Since the introduction of the Sponge functions methodology [6], permutation-based cryptographic algorithms have rapidly been gaining acceptance due to their efficient implementation properties. Very recently, the sponge-based hash function Keccak [4] has been selected as the winner of the NIST SHA-3 competition.

The PRIMATE permutation is a substitution-permutation network using a 5-bit S-box with optimal linear and differential properties, and a recursive MDS matrix, which leads to a very small and efficient implementation in hardware.

Resistance against hardware side-channel attacks.

To meet the requirements of resistance against hardware side-channel attacks, the underlying permutation has been designed to offer an efficient threshold implementation to counter first-order DPA attacks, based on glitch-free secret-sharing-based masking, cf. [8].

Online.

All PRIMATEs offer online encryption, thereby allowing the algorithm to output ciphertext blocks without the knowledge of plaintext lengths or the next plaintext blocks. PRIMATEs are inherently sequential. For lightweight applications, this is not an issue: the design goal is to use a very small number of gates, therefore parallelism would not be of any benefit.

Comparison to AES-GCM.

- GCM-AES is a block cipher based design. In comparison, PRIMATEs are smaller than similar AEAD algorithms based on a block cipher (such as AES), as our implementation does not contain a key schedule, uses smaller S-boxes (5 bits instead of 8 bits), and uses a more compact, recursive MDS matrix implementation.
- Unlike in AES-GCM, PRIMATEs handle all nonce lengths in the same way, thereby reducing the complexity of the implementation and simplifying the security analysis. We must stress, however, that when any of the parameters of the PRIMATEs (such as the nonce length) are modified, a new key must be chosen uniformly at random.
- The PRIMATEs modes avoid all the attacks that are inherent to AEAD modes based on a polynomial hash [16, 20], such as AES-GCM.

Key and Nonce Agility.

Changing the key or nonce has very little overhead for all modes in the PRIMATEs family, requiring only one permutation function call and one key XOR for GIBBON, and requires only one permutation function call for HANUMAN. In the case of APE, changing the key also requires one permutation function call. Changing the nonce requires ν/r permutation function calls.

Besides the aforementioned features that hold in general for the PRIMATEs algorithm family, several features make specific modes stand out.

Features Specific to APE, HANUMAN and GIBBON.

- APE should be used in applications where additional security is required. Like HANUMAN, APE is provably secure, based on the security of the underlying permutation. Additionally, APE provides resistance against nonce reuse [1], as well as resistance against adversaries that can observe the unverified plaintext during decryption [2]. The price to pay for this additional security is that decryption is performed backwards using the inverse of the permutation.
- HANUMAN is based on the SpongeWrap [6] design strategy. More concretely, it is the hermetic Sponge design strategy, which means that its underlying permutation should be free of any structural distinguishers.

- GIBBON is intended for lightweight applications where speed is critical and a formal security proof (based on the security of the underlying permutation) is not required. To achieve high throughput, GIBBON employs reduced-round permutations p_2 and p_3 to process the associated data and message respectively, next to the full-round permutation p_1 used for initialization and finalization.

6 Design Rationale

The PRIMATEs have been designed with lightweight hardware requirements as present in constrained devices in mind. For the mode of operation, they follow the principles of the sponge methodology, more specifically, some of the principles of SpongeWrap and MonkeyDuplex. The modes of operation are generic and free of weaknesses as justified by the formal security proofs. For the underlying permutations, the PRIMATEs follow the well-established SPN approach of Rijndael (and its wide-trail design strategy), based on almost bent S-boxes (attaining best possible differential and linear properties) as well as MDS diffusion matrices (achieving best possible differential and linear local diffusion). To favor lightweight implementations of the PRIMATEs, the MDS diffusion matrices are chosen to be recursive and the S-boxes to be 5-bit.

The PRIMATEs family includes three modes: GIBBON and HANUMAN are nonce-based, while APE has been designed to maintain security under both nonce reuse and release of unverified plaintext – scenarios that are likely to persist in highly constrained embedded systems. GIBBON does not follow the hermetic sponge-based design approach, while both HANUMAN and APE do. This allows GIBBON to be considerably faster and more energy-efficient. A state recovery for GIBBON does not lead to trivial key recovery and also does not lead to trivial universal forgery attacks due to the key additions.

The designers have not hidden any weaknesses in these ciphers.

7 Intellectual Property

The submitters are not aware of any patent involved in PRIMATEs family. Furthermore, PRIMATEs will not be patented. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

8 Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analysis that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analysis then they are expected to promptly and publicly respond to those analysis, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Acknowledgments. We would like to thank Miroslav Knežević for his various implementations and comments. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and by the Research Fund KU Leuven, OT/13/071. In addtion, the work was supported in part by the Austrian Government through the research program COMET (Project SeCoS, Project Number 836628). Elena Andreeva and Nicky Mouha are supported by Postdoctoral Fellowships from the Flemish Research Foundation (FWO-Vlaanderen). Atul Luykx and Bart Mennink are supported by Ph.D. Fellowships from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Begül Bilgin was partially supported by the FWO project G0B4213N. Qingju Wang is also funded by the Major State Basic Research Development Program of China (973 Plan) (No.2013CB338004), National Natural Science Foundation of China (No. 61073150), and Chinese Major Program of National Cryptography Development Foundation.

References

[1] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE 2014. Lecture Notes in Computer Science, Springer (2014), to appear

[2] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. Cryptology ePrint Archive, Report 2014/144 (2014)

[3] Augot, D., Finiasz, M.: Direct Construction of Recursive MDS Diffusion Layers using Shortened BCH Codes. In: FSE 2014. Lecture Notes in Computer Science, Springer (2014), to appear

[4] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak SHA-3 submission. Submission to the NIST SHA-3 Competition (Round 3) (2011)

[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-Based Encryption, Authentication and Authenticated Encryption. Directions in Authenticated Ciphers (July 2012)

[6] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic Sponge Functions, available at `http://sponge.noekeon.org/CSF-0.1.pdf`

[7] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography 2011. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2012)

[8] Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In: Bertoni, G., Coron, J.S. (eds.) CHES. Lecture Notes in Computer Science, vol. 8086, pp. 142–158. Springer (2013)

[9] Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-based lightweight authenticated encryption. In: Moriai, S. (ed.) FSE. Lecture Notes in Computer Science, Springer (2013)

[10] Carlet, C., Charpin, P., Zinoviev, V.: Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems. Des. Codes Cryptography 15(2), 125–156 (1998)

[11] Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) IMA Int. Conf. Lecture Notes in Computer Science, vol. 2260, pp. 222–238. Springer (2001)

[12] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)

[13] Fouque, P.A., Joux, A., Martinet, G., Valette, F.: Authenticated On-Line Encryption. In: Matsui, M., Zuccherato, R.J. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 3006, pp. 145–159. Springer (2003)

[14] Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011)

[15] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES. Lecture Notes in Computer Science, vol. 6917, pp. 326–341. Springer (2011)

[16] Handschuh, H., Preneel, B.: Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In: Wagner, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 5157, pp. 144–161. Springer (2008)

[17] IBM: IBM ILOG CPLEX Optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

[18] Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Wu, C., Yung, M., Lin, D. (eds.) Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011)

[19] Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 247–260. Springer (2003)

[20] Saarinen, M.J.O.: Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In: Canteaut, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 7549, pp. 216–225. Springer (2012)

[21] Tsang, P.P., Solomakhin, R.V., Smith, S.W.: Authenticated streamwise on-line encryption. Dartmouth Computer Science Technical Report TR2009-640 (2009)

[22] Wu, S., Wang, M., Wu, W.: Recursive Diffusion Layers for (Lightweight) Block Ciphers and Hash Functions. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 355–371. Springer (2012)

Diagrams

In all figures below the gray dotted box denotes the processing boundaries whenever no associated data is present.

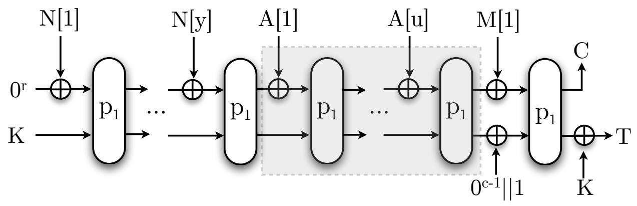


Figure 6: APE AE where $|M[w]| \leq r-1$, $w=1$, fractional A and padded A and M.

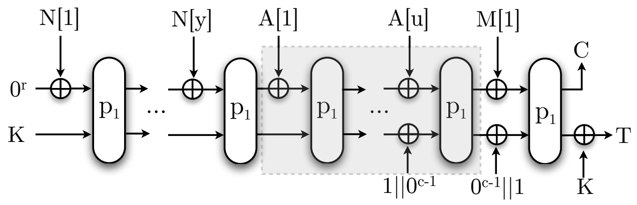


Figure 7: APE AE where $|M[w]| \leq r-1$, $w=1$, integral A and padded M.

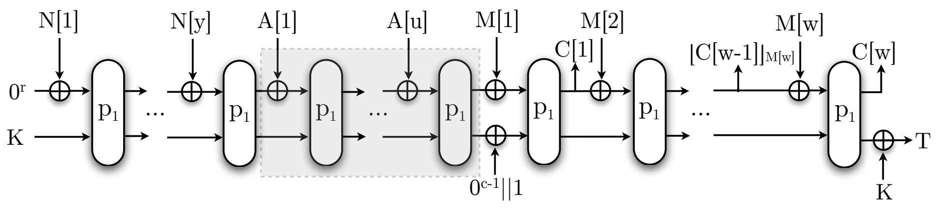


Figure 8: APE AE where $|M[w]| \leq r-1$, $w \geq 2$, fractional A and padded A and M.

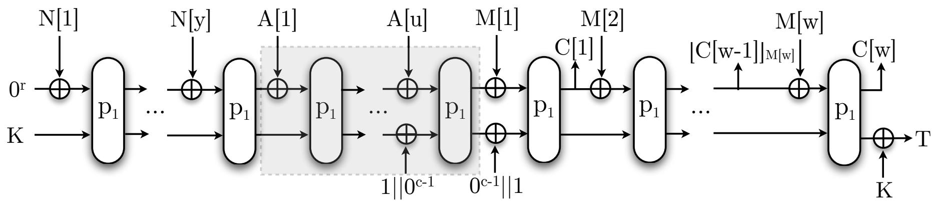


Figure 9: APE AE where $|M[w]| \leq r-1$, $w \geq 2$, integral A and padded M.

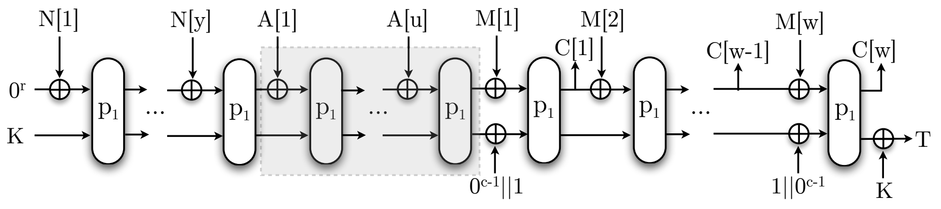


Figure 10: APE AE where $|M[w]| = r$, fractional and padded A.

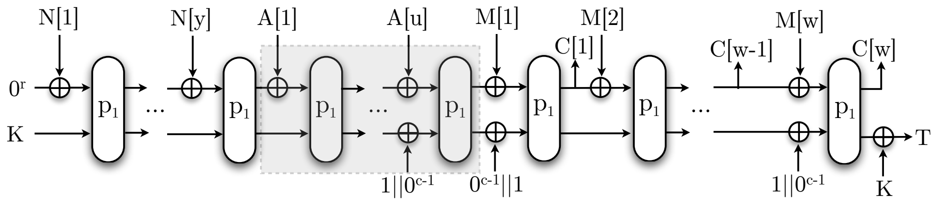


Figure 11: APE AE where $|M[w]| = r$ and integral A.

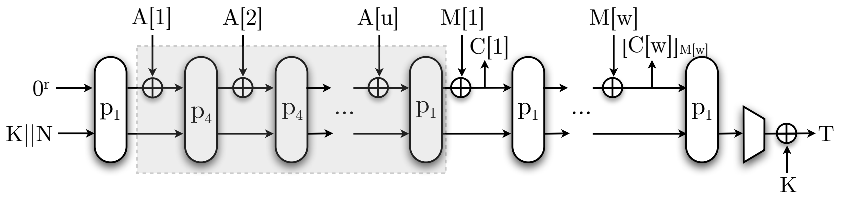


Figure 12: HANUMAN AE where $|M[w]| \leq r - 1$, fractional A and padded A and M.

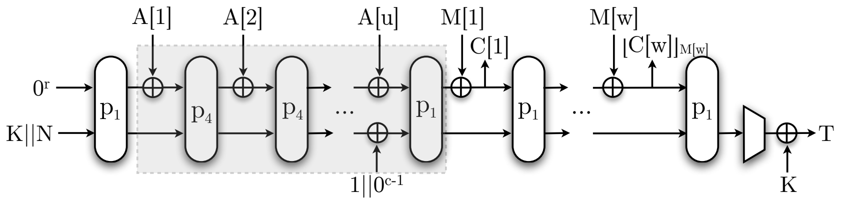


Figure 13: HANUMAN AE where $|M[w]| \leq r - 1$, integral A and padded M.

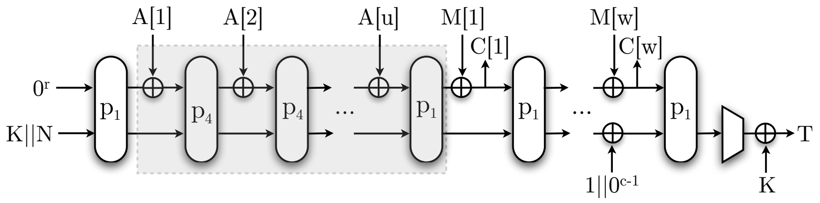


Figure 14: HANUMAN AE where $|M[w]| = r$ and fractional and padded A.

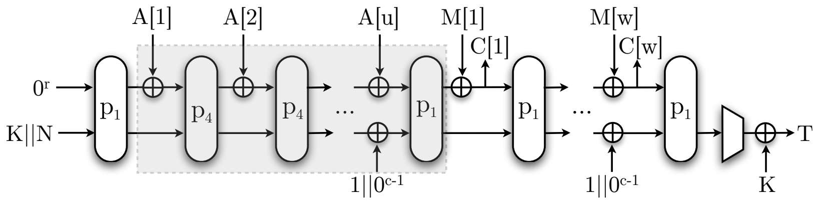


Figure 15: HANUMAN AE where $|M[w]| = r$ and integral A.

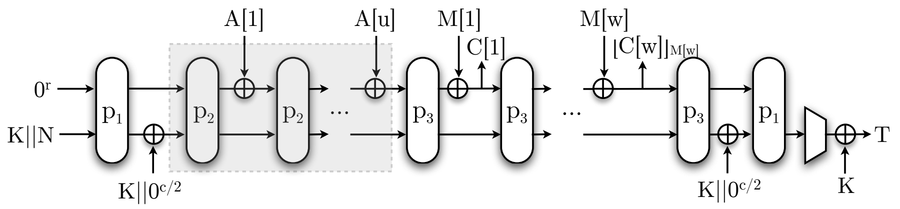


Figure 16: GIBBON AE where $|M[w]| \leq r - 1$, fractional A and padded A and M.

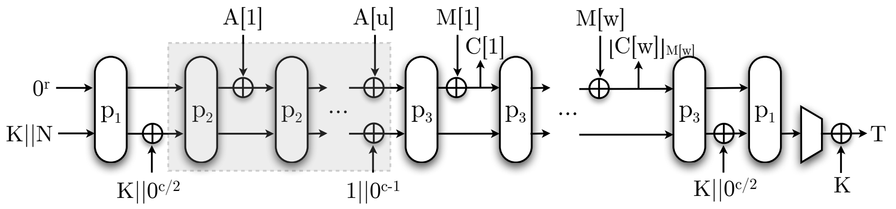


Figure 17: GIBBON AE where $|M[w]| \leq r - 1$, integral A and padded M.

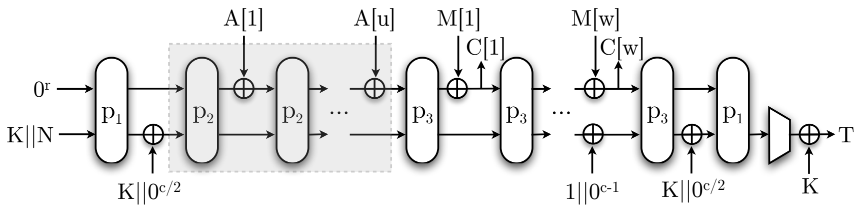


Figure 18: GIBBON AE where $|M[w]| = r$ and fractional and padded A.

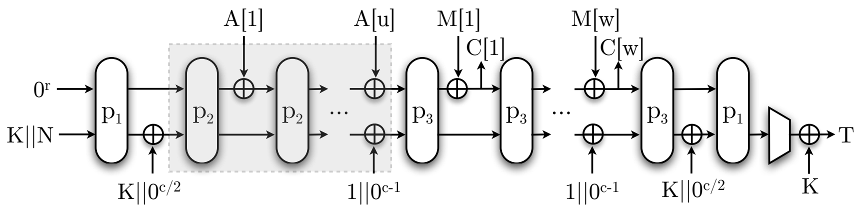


Figure 19: GIBBON AE where $|M[w]| = r$ and integral A.