PRØST v1

Designers/Submitters

Elif Bilge Kavun[1]
Martin M. Lauridsen[2]
Gregor Leander[1]
Christian Rechberger[2]
Peter Schwabe[3]
Tolga Yalçın[4]

Affiliations

[1] Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
[2] DTU Compute, Technical University of Denmark, Denmark
[3] Digital Security Group, Radboud University Nijmegen, The Netherlands
[4] University of Information Science and Technology, Ohrid, Republic of Macedonia

Contact

proest@cryptojedi.org

March 16, 2014

1 Summary

The PRØST permutation is a strong, fast, amazing permutation WITH strong bounds that is suitable for many platforms and many modes. Thus, we take the mode as a parameter.

2 Specification

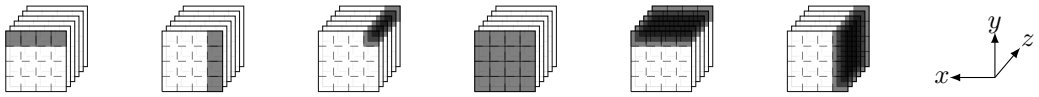
PRØST is a permutation-based AEAD-scheme. Hence, we structure the specification into two parts; first we specify the permutation, then we specify the mode of use.

2.1 PRØST Permutation

We consider an n-bit state $\mathbf{x}$ as a three-dimensional block. We denote the axes by x, y and z c.f. Figure 1. We let $4 \times 4 \times d = n$ be the dimensions of the state along those axes; as such, the state is 4 bits wide, 4 bits high and d bits deep along the z axis. We re-use the nomenclature of Keccak [5] when referring to the terms *row*, *column*, *lane* (which we also call *register*), *slice*, *plane* and *sheet* as indicated in Figure 1. To embed the state into the xy-plane, the registers are arranged into an $h \times w$ matrix

$$\mathbf{x} = \begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} & r_{0,3} \\ r_{1,0} & r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,0} & r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,0} & r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix},$$

where each $r_{x,y}$ is a d-bit register (or lane). We use $r_{x,y,z}$ to refer to the zth bit of register $r_{x,y}$, with $z = 0$ being the most significant bit. The embedding into a single dimension is given by $\mathbb{F}_2^n \ni \mathbf{x} = (r_{0,0}, \cdots, r_{3,3})$.



(a) Row (b) Column (c) Lane (d) Slice (e) Plane (f) Sheet (g) Axes

Figure 1: Nomenclature for state parts

The PRØST permutation consists of compositions of permutations, which we refer to as rounds, borrowing from the design of iterated block cipher constructions. We denote the total number of rounds by T. We use $R_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ to refer to round i, $1 \leq i \leq T$, which is defined as

$$R_i(\mathbf{x}) = (\texttt{AddConstant} \circ \texttt{ShiftPlanes} \circ \texttt{MixSlices} \circ \texttt{SubRows})(\mathbf{x}),$$

where $\mathbf{x}$ is an n-bit three-dimensional block of bits. We give the definition of each of these operations below.

2.1.1 SubRows

The `SubRows` operation substitutes each 4-bit row of the state according to a 4-bit S-box. The cipher uses one 4-bit S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$. More precisely, a state $S = (s)_{x,y,z}$ will be mapped to a state $S' = (s')_{x,y,z}$ where

$$\left(s'_{0,y,z}, s'_{1,y,z}, s'_{2,y,z}, s'_{3,y,z}\right) = \mathcal{S}\left(s_{0,y,z}, s_{1,y,z}, s_{2,y,z}, s_{3,y,z}\right) \quad \forall y, z$$

The action of the S-box is given in Table 1.

Table 1: Action of the PRØST S-box in hexadecimal notation

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	0	4	8	f	1	5	e	9	2	7	a	c	b	d	6	3

2.1.2 MixSlices

The `MixSlices` operation considers, for each j, $0 \leq j < d$, the jth slice as a vector in $\mathbb{F}_2^{16}$ determined as $\mathbf{x} = (r_{0,0,j}, r_{0,1,j}, \ldots, r_{3,3,j})$. Thus, MixSlices applied to a state $R = (r)_{x,y,z}$ will result in a state $R' = (r')_{x,y,z}$ where

$$\begin{pmatrix} r'_{0,0,z} \\ r'_{0,1,z} \\ r'_{0,2,z} \\ r'_{0,3,z} \\ r'_{1,0,z} \\ \vdots \\ r'_{3,2,z} \\ r'_{3,3,z} \end{pmatrix} = M \begin{pmatrix} r_{0,0,z} \\ r_{0,1,z} \\ r_{0,2,z} \\ r_{0,3,z} \\ r_{1,0,z} \\ \vdots \\ r_{3,2,z} \\ r_{3,3,z} \end{pmatrix} \qquad \forall z \in \{0, \ldots, d-1\}.$$

The matrix M used for `MixSlices` is given by (1). It is an MDS, involutive, i.e. self-inverse matrix with 88 ones. It is furthermore the matrix among all the matrices with those criteria which minimizes that number of low weight differences.

$$M = \begin{pmatrix} 1000100100101011 \\ 0100100000011001 \\ 0010010011001000 \\ 0001001001100100 \\ 1001100010110010 \\ 1000010010010001 \\ 0100001010001100 \\ 0010000101000110 \\ 0010101110001001 \\ 0001100101001000 \\ 1100100000100100 \\ 0110010000010010 \\ 1011001010011000 \\ 1001000110000100 \\ 1000110001000010 \\ 0100011000100001 \end{pmatrix}. \tag{1}$$

2.1.3 ShiftPlanes

The `ShiftPlanes` operation cyclically shifts each of the 4 lanes in a plane in the positive z direction. In contrast to many schemes, we have two different shift operations, one for even, and one for odd rounds. The shift amount for plane j, $0 \leq j \leq 3$ is given by the jth entry of a *shift vector* lying in $\mathbb{Z}_d^4$. For round R_i, $1 \leq i \leq T$, the shift vector is denoted π_1 when i is odd, and π_2 when i is even. Both vectors are given in Table 2. Thus, `ShiftPlanes` applied to a state $T = (t)_{x,y,z}$ in round i

will result in a state $T' = (t')_{x,y,z}$ where

$$t'_{x,y,z} = \begin{cases} t_{x,y,(z-\pi_1(x) \bmod d)} & , i \text{ odd} \\ t_{x,y,(z-\pi_2(x) \bmod d)} & , i \text{ even} \end{cases} \qquad \forall x, y \in \{0, \ldots, 3\}.$$

Table 2: Shift vectors π_1 and π_2 for the `ShiftPlanes` operation for $d = 16$ and $d = 32$

d	π_1	π_2
16	$(0, 2, 4, 6)$	$(0, 1, 8, 9)$
32	$(0, 4, 12, 26)$	$(1, 24, 26, 31)$

2.1.4 AddConstant

The `AddConstant` transformation in round i (starting from zero) updates the state $A = A + C_i$ with C_i being the round constants. Let $c_1 = 0x75817b9d$ and $c_2 = 0xb2c5fef0$. With j being the number of the lane, then the round constants are $C_i = c_1 <<< i <<< j)$ for even j and $C_i = c_2 <<< i <<< j)$ for odd j

2.1.5 PRØST-128 and PRØST-256 permutations

We specify two permutations to be used, one of size 256 bits with a security level of 128 bits, and one of size 512 bits with a security level of 256 bits, i.e. PRØST-n is a $2n$-bit permutation with n bits of security.

For PRØST-256, we have $d = 32$, and the number of rounds is $T = 18$. For PRØST-128, we have $d = 16$, and the number of rounds is $T = 16$, and constants c_1 and c_2 are truncated to their first 16 most significant bits.

2.2 Absence of trap-doors

We faithfully declare that we have not inserted any hidden weaknesses in the PRØST permutation.

2.3 PRØST-AE: Authenticated Encryption with PRØST

Authenticated encryption (AE) with associated data (AD) is a symmetric-key cryptographic primitive which combines encryption and authentication in a single algorithm, with focus on both aspects from the design phase. This is contrary to the earlier paradigm of obtaining secrecy and authentication by applying some combination of encryption scheme with a MAC.

AE schemes are most commonly based on block ciphers as the underlying primitive. Examples of such include OCB[1-3] [15, 14, 11], GCM [12], CCM [19], AEGIS [20], COPA [3], COBRA [4], FIDES [7], OTR [13], the McOE family [9] and POET [1].

With the advent of permutation-based designs, notably in the Sponge constructions, the use of permutations as the underlying primitive in AE schemes is becoming increasingly frequent. So far, permutation-based AE schemes include SpongeWrap [6], APE [2] and PPAE [10].

In this section we describe our proposals for the CAESAR competition, which are all specific instantiations of AEAD schemes using our PRØST permutation as the underlying primitive. Each of the instantiations are based on existing AE schemes. The proposals are summarized in Table 3.

For the description of our proposals in the following sections, we use PRØST-n to refer to the PRØST permutation with n-bit security level and internal state of $2n$ bits. Table 3 lists the specific instantiations, while the specifications in the following sections use general n for their descriptions.

Table 3: Our proposals, underlying PRØST permutation and their rankings. Other columns indicate onlineness (in encryption and decryption), parallelizability (P), nonce misuse-resistance (NMR), easy constant-time implementation (CT) and cheap power analysis countermeasures (CM).

			Online					
Rank	Proposal	Permutation	Enc	Dec	P	NMR	CT	CM
1	PRØST-COPA-128	PRØST-128	✓	✓	✓	✓	✓	✓
2	PRØST-COPA-256	PRØST-256	✓	✓	✓	✓	✓	✓
3	PRØST-OTR-128	PRØST-128	✓	✓	✓		✓	✓
4	PRØST-OTR-256	PRØST-256	✓	✓	✓		✓	✓
5	PRØST-APE-256[128, 128]	PRØST-256	✓			✓	✓	✓
6	PRØST-APE-128[64, 64]	PRØST-128	✓			✓	✓	✓

Notation

Throughout our description of the proposals, we use P_n as a shorthand for the PRØST-n permutation, i.e. PRØST with n-bit security level and internal state size of $2n$ bits.

An asterisk in superscript denotes the Kleene star, for e.g. $\mathbb{F}_2^*$ denotes binary strings of any length, including the empty set. A plus in superscript denotes "at least one", so e.g. $(\mathbb{F}_2^{2n})^+$ are the binary strings which have length at least $2n$. We use $\text{msb}_\ell(x)$ to denote the ℓ most significant bits of x. By $X10^*$ we denote a binary string X which is padded with a 1 followed by a number of zeroes, such that the length becomes a multiple of $2n$.

When we write multiplications, we refer to the operation in a finite field $GF(2^{2n})$ defined modulo an irreducible polynomial $f(x)$ over $GF(2)$ of degree $2n$. Constants occurring are also in this field, so e.g. 7 is the polynomial $x^2 + x + 1$.

We use the $\perp$ symbol to denote the output of the decryption algorithm when tag verification fails.

Block cipher-based AE to Permutation-based AE

It is a well-known result that any block cipher based AE scheme can be turned into a permutation-based scheme by plugging a singke-key Even-Mansour construction into the place of the block cipher [8].

For the purpose of using PRØST-n in block cipher-based AE schemes, we define the single-key Even-Mansour block cipher, which takes a single $2n$-bit key K, as $\tilde{P}_{n,K} : \mathbb{F}_2^{2n} \times \mathbb{F}_2^{2n} \to \mathbb{F}_2^{2n}$, defined as $\tilde{P}_{n,K}(x) \stackrel{\text{def}}{=} K \oplus P_n(x \oplus K)$.

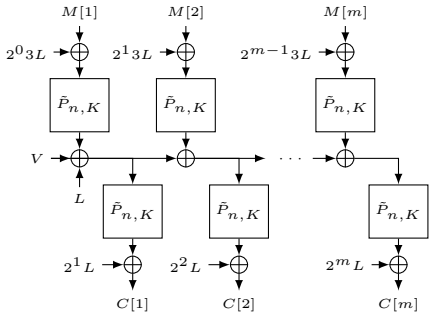
With PRØST, we have designed a permutation which has a simple design, is efficient, has strong security bounds and is inherently timing-attack protected. With this, we next define specific instantiations of authenticated encryption schemes based on PRØST, both when the underlying scheme is block cipher or permutation based, either by using it directly or through $\tilde{P}_{n,K}$.

2.3.1 PRØST-COPA-n

COPA is a design by Andreeva et al. from Asiacrypt 2013 [3]. In this part, we give our specification of PRØST-COPA-n; the instantiation of the COPA AE scheme using $\tilde{P}_{n,K}$ as the underlying block cipher.

The resulting proposal PRØST-COPA-n enjoys the security proofs of COPA, and in particular, COPA is resistant to nonce-misuse, leaking only the length of the common message prefix (which is optimal for single-pass schemes) [3].

Figure 2: Encryption of m message blocks with PRØST-COPA-n



PRØST-COPA-n is a fully parallelizable, online AE scheme, and uses the cheaper doubling as opposed to general multiplication in $GF(2^{2n})$ for the tweak values. The schemes uses two calls to PRØST-n and two doublings in $GF(2^{2n})$ per $2n$-bit block of message.

The encryption of m message blocks of $2n$ bits each is depicted in Figure 2. The complete description of encryption with authentication, decryption with verification and processing of associated data for PRØST-COPA-n is given by Algorithms 1 through 3. The tag length for PRØST-COPA-n equals the message block length of $2n$.

Algorithm 1: PRØST-COPA-n-$\mathcal{E}(A, M)$

Data: $A \in \mathbb{F}_2^*, M \in (\mathbb{F}_2^{2n})^+$
$L \leftarrow \tilde{P}_{n,K}(0)$
$V[0] \leftarrow$ PRØST-COPA-n-ProcessAD(A, L)
$\Delta_0 \leftarrow 3L$
$\Delta_1 \leftarrow 2L$
$\Sigma \leftarrow 0$
for $i = 1, \ldots, m$ **do**
  $V[i] \leftarrow \tilde{P}_{n,K}(M[i] \oplus \Delta_0) \oplus V[i-1]$
  $C[i] \leftarrow \tilde{P}_{n,K}(V[i]) \oplus \Delta_1$
  $\Delta_0 \leftarrow 2\Delta_0$
  $\Delta_1 \leftarrow 2\Delta_1$
  $\Sigma \leftarrow \Sigma \oplus M[i]$
end
$T \leftarrow \tilde{P}_{n,K}(\tilde{P}_{n,K}(\Sigma \oplus 2^{m-1}3^2 L) \oplus V[m]) \oplus 2^{m-1}7L$
return (C, T)

Algorithm 2: PRØST-COPA-n-$\mathcal{D}(A, C, T)$

Data: $A \in \mathbb{F}_2^*, C \in (\mathbb{F}_2^{2n})^+, T \in \mathbb{F}_2^{2n}$
$L \leftarrow \tilde{P}_{n,K}(0)$
$V[0] \leftarrow$ PRØST-COPA-n-ProcessAD(A, L)
$\Delta_0 \leftarrow 3L$
$\Delta_1 \leftarrow 2L$
$\Sigma \leftarrow 0$
for $i = 1, \ldots, m$ **do**
  $V[i] \leftarrow \tilde{P}_{n,K}^{-1}(C[i] \oplus \Delta_1)$
  $M[i] \leftarrow \tilde{P}_{n,K}^{-1}(V[i] \oplus V[i-1]) \oplus \Delta_0$
  $\Delta_0 \leftarrow 2\Delta_0$
  $\Delta_1 \leftarrow 2\Delta_1$
  $\Sigma \leftarrow \Sigma \oplus M[i]$
end
$\hat{T} \leftarrow \tilde{P}_{n,K}(\Sigma \oplus 2^{m-1}3^2L)$
if $\hat{T} = \tilde{P}_{n,K}^{-1}(T \oplus 2^{m-1}7L)$ **then**
  return M
end
else
  return $\perp$
end

Algorithm 3: PRØST-COPA-n-ProcessAD(A, L)

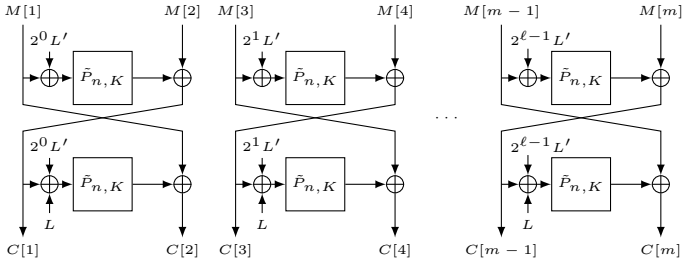
Data: $A \in \mathbb{F}_2^*, L \in \mathbb{F}_2^{2n}$
$\Delta \leftarrow 3^3L$
$V \leftarrow 0$
for $i = 1, \ldots, a-1$ **do**
  $V \leftarrow V \oplus \tilde{P}_{n,K}(A[i] \oplus \Delta)$
  $\Delta \leftarrow 2\Delta$
end
if $|A| \equiv 0 \mod 2n$ **then**
  $V \leftarrow \tilde{P}_{n,K}(V \oplus A[a] \oplus 3\Delta)$
end
else
  $V \leftarrow \tilde{P}_{n,K}(V \oplus A[a]10^* \oplus 3^2\Delta)$
end
return V

2.3.2 PRØST-OTR-n

OTR is an AE scheme designed by Minematsu [13]. Here, we specify PRØST-OTR-n, which is the instantiation of OTR using $\tilde{P}_{n,K}$ as the underlying block cipher.

OTR uses 2-round Feistel ciphers to encrypt two consecutive message blocks. It uses one call to PRØST-n and half a doubling in $GF(2^{2n})$ per $2n$-bit block of message.

Figure 3: Encryption of m message blocks with PRØST-OTR-n, when m is even



PRØST-OTR-n inherits the features of OTR. In particular, it is nonce-based, but not nonce-misuse resistant. This means that security is lost if nonces are re-used. Comparing to schemes which offer nonce-misuse resistance, the gain is performance. The proposal is online and completely parallelizable. It does not require the inverse of the underlying primitive, due to the employment of the Feistel cipher structure.

The encryption of m message blocks, for even m, with PRØST-OTR-n is depicted in Figure 3. In the figure, $\ell = m/2$. The complete description of encryption with authentication, decryption with verification and processing of associated data for PRØST-OTR-n is given by Algorithms 4 through 5. The tag length for PRØST-OTR-n is denoted τ.

Algorithm 4: PRØST-OTR-n-$\mathcal{E}(N, A, M)$

Data: $N \in \mathbb{F}_2^{2n}, A \in \mathbb{F}_2^*, M \in (\mathbb{F}_2^{2n})^+$
$(C, T) \leftarrow$ PRØST-OTR-n-Enc(N, M)
if $|A| = 0$ **then**
  $TA \leftarrow 0$
end
else
  $TA \leftarrow$ PRØST-OTR-n-ProcessAD(A)
end
$T \leftarrow \mathrm{msb}_\tau(T \oplus TA)$
return (C, T)

Algorithm 5: PRØST-OTR-n-ProcessAD(A)

Data: $A \in \mathbb{F}_2^*$
$\Lambda \leftarrow 0$
$Q \leftarrow \tilde{P}_{n,K}(0)$
$Q' \leftarrow 4Q$
$A[1]A[2]\cdots A[a] \leftarrow A$
for $i = 1, \ldots, a-1$ **do**
  $\Lambda \leftarrow \Lambda \oplus \tilde{P}_{n,K}(Q' \oplus A[i])$
  $Q' \leftarrow 2Q'$
end
$\Lambda \leftarrow \Lambda \oplus A[a]10^*$
if $|A[a]| \neq 2n$ **then**
  $TA \leftarrow \tilde{P}_{n,K}(Q' \oplus Q \oplus \Lambda)$
end
else
  $TA \leftarrow \tilde{P}_{n,K}(Q' \oplus 2Q \oplus \Lambda)$
end
return TA

Algorithm 6: PRØST-OTR-n-$\mathcal{D}(N, C, A, T)$

Data: $N \in \mathbb{F}_2^{2n}, C \in (\mathbb{F}_2^{2n})^+, A \in \mathbb{F}_2^*, T \in \mathbb{F}_2^\tau$
$(M, T) \leftarrow$ PRØST-OTR-n-Dec(N, C)
if $|A| = 0$ **then**
  $TA \leftarrow 0$
end
else
  $TA \leftarrow$ PRØST-OTR-n-ProcessAD(A)
end
$\hat{T} \leftarrow \mathrm{msb}_\tau(T \oplus TA)$
if $\hat{T} = T$ **then**
  return M
end
else
  return $\perp$
end

Algorithm 7: PRØST-OTR-n-Enc(N, M)

Data: $N \in \mathbb{F}_2^{2n}, M \in (\mathbb{F}_2^{2n})^+$

$\Sigma \leftarrow 0$
$L \leftarrow \tilde{P}_{n,K}(N10^*)$
$L' \leftarrow 4L$
$M[1]M[2]\cdots M[m] \leftarrow M$
for $i = 1, \ldots, \lfloor m/2 \rfloor - 1$ **do**
  $C[2i-1] \leftarrow \tilde{P}_{n,K}(L' \oplus M[2i-1]) \oplus M[2i]$
  $C[2i] \leftarrow \tilde{P}_{n,K}(L' \oplus L \oplus C[2i-1]) \oplus M[2i-1]$
  $\Sigma \leftarrow \Sigma \oplus M[2i]$
  $L' \leftarrow 2L'$
end
if $m \equiv 0 \mod 2$ **then**
  $Z \leftarrow \tilde{P}_{n,K}(L' \oplus M[m-1])$
  $C[m] \leftarrow \mathrm{msb}_{|M[m]|}(Z) \oplus M[m]$
  $C[m-1] \leftarrow \tilde{P}_{n,K}(L' \oplus K \oplus C[m]10^*) \oplus M[m-1]$
  $\Sigma \leftarrow \Sigma \oplus Z \oplus C[m]10^*$ $L_{\mathrm{last}} \leftarrow L' \oplus L$
end
if $m \equiv 1 \mod 2$ **then**
  $C[m] \leftarrow \mathrm{msb}_{|M[m]|}(\tilde{P}_{n,K}(L')) \oplus M[m]$
  $\Sigma \leftarrow \Sigma \oplus M[m]10^*$
  $L_{\mathrm{last}} \leftarrow L'$
end
if $|M[m]| \neq 2n$ **then**
  $T \leftarrow \tilde{P}_{n,K}(3L_{\mathrm{last}} \oplus \Sigma)$
end
else
  $T \leftarrow \tilde{P}_{n,K}(3L_{\mathrm{last}} \oplus L \oplus \Sigma)$
end
$C \leftarrow C[1]C[2]\cdots C[m]$
return (C, T)

Algorithm 8: PRØST-OTR-n-Dec(N, C)

Data: $N \in \mathbb{F}_2^{2n}, C \in (\mathbb{F}_2^{2n})^+$

$\Sigma \leftarrow 0$
$L \leftarrow \tilde{P}_{n,K}(N10^*)$
$L' \leftarrow 4L$
$C[1]C[2]\cdots C[m] \leftarrow C$
for $i = 1, \ldots, \lfloor m/2 \rfloor - 1$ **do**
  $M[2i-1] \leftarrow \tilde{P}_{n,K}(L' \oplus L \oplus C[2i-1]) \oplus C[2i]$
  $M[2i] \leftarrow \tilde{P}_{n,K}(L' \oplus C[2i-1]) \oplus C[2i-1]$
  $\Sigma \leftarrow \Sigma \oplus M[2i]$
  $L' \leftarrow 2L'$
end
if $m \equiv 0 \mod 2$ **then**
  $M[m-1] \leftarrow \tilde{P}_{n,K}(L' \oplus L \oplus C[m]10^*) \oplus C[m-1]$
  $Z \leftarrow \tilde{P}_{n,K}(L' \oplus M[m-1])$
  $M[m] \leftarrow \mathrm{msb}_{|C[m]|}(Z) \oplus C[m]$
  $\Sigma \leftarrow \Sigma \oplus Z \oplus C[m]10^*$ $L_{\mathrm{last}} \leftarrow L'$
end
if $m \equiv 1 \mod 2$ **then**
  $M[m] \leftarrow \mathrm{msb}_{|C[m]|}(\tilde{P}_{n,K}(L')) \oplus C[m]$
  $\Sigma \leftarrow \Sigma \oplus M[m]10^*$
  $L_{\mathrm{last}} \leftarrow L'$
end
if $|C[m]| \neq 2n$ **then**
  $T \leftarrow \tilde{P}_{n,K}(3L_{\mathrm{last}} \oplus \Sigma)$
end
else
  $T \leftarrow \tilde{P}_{n,K}(3L_{\mathrm{last}} \oplus L \oplus \Sigma)$
end
$M \leftarrow M[1]M[2]\cdots M[m]$
return (M, T)

2.3.3 PRØST-APE-$n[r, c]$

APE is a permutation-based AE scheme by Andreeva et al. from FSE 2014 [2]. It does not require a block cipher, but rather a permutation as the underlying primitive. Here, we give the specification of our instantiation of APE using PRØST-n as the underlying permutation.

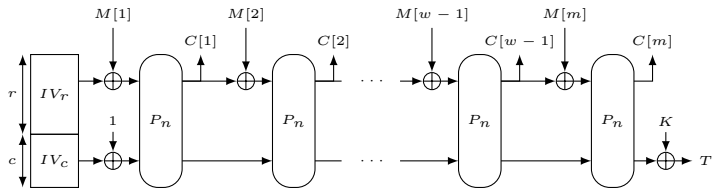


Figure 4: Encrypting and authenticating m message blocks of r bits each using PRØST-APE-$n[r, c]$

PRØST-APE-$n[r, c]$ is online in encryption, i.e. one does not need to know all message blocks, and in particular the number of message blocks, before one can start encrypting. Indeed, the ciphertext block $C[i]$ depends only on message blocks $M[1], \ldots, M[i]$. For decryption, however, one starts decrypting the last ciphertext block first, and hence decryption is *not* online. The fact that one decrypts in reverse implies the need for the inverse the underlying permutation as well.

PRØST-APE-$n[r, c]$ has two important parameters: the *rate* denoted r and the *capacity* denoted c. As the underlying permutation PRØST-n has a state size of $2n$ bits, we have that $r + c = 2n$. It uses a single c-bit key $K \in \mathbb{F}_2^c$. The scheme encrypts r-bit message blocks to r-bit ciphertext blocks. In our specification, subscript r or c denotes the rate, respectively capacity part of the $(r + c)$-bit operand. Figure 4 illustrates the encryption and tag generation for m message blocks of r bits each. When there is no associated data, i.e. $|A| = 0$, the IV is set to $IV = (IV_r \| IV_c) = (0^r \| K)$, and otherwise the IV is generated by the associated data processing algorithm 13. The complete description of encryption with authentication, decryption with verification and processing of associated data for PRØST-APE-$n[r, c]$ is given by Algorithms 9 through 13.

APE achieves privacy and integrity up to the bound $2^{c/2}$, both in the ideal permutation model and the standard model. In other words, the choice of rate and capacity influence performance and security, in the sense that increasing capacity and decreasing rate will increase security and decrease performance, and vice versa. We refer to Table 3 for the specific parameters of our proposals.

APE is the first permutation-based AE scheme to obtain nonce-misuse resistance. In particular, when nonces are repeated, it leaks only the XOR of the common prefix of the message.

Algorithm 9: PRØST-APE-n-$\mathcal{E}(A, M)$

Data: $K \in \mathbb{F}_2^c, A \in (\mathbb{F}_2^r)^*, M \in (\mathbb{F}_2^r)^+$
Result: $C \in (\mathbb{F}_2^r)^+, T \in \mathbb{F}_2^c$
if $A = \emptyset$ **then**
  $IV \leftarrow (0^r \| K) \in \mathbb{F}_2^{r+c}$
end
else
  $IV \leftarrow$ PRØST-APE-n-ProcessAD$((0^r \| K), A)$
end
$(C, \hat{V}_c) \leftarrow$ PRØST-n-APE-Enc(IV, M)
$T \leftarrow \hat{V}_c \oplus K$
return (C, T)

Algorithm 10: PRØST-APE-n-$\mathcal{D}(A, C, T)$

Data: $K \in \mathbb{F}_2^c, A \in (\mathbb{F}_2^r)^*, C \in (\mathbb{F}_2^r)^+, T \in \mathbb{F}_2^c$
Result: $M \in (\mathbb{F}_2^r)^+$ or $\perp$
if $A = \emptyset$ **then**
  $IV \leftarrow (0^r \| K) \in \mathbb{F}_2^{r+c}$
end
else
  $IV \leftarrow$ PRØST-APE-n-ProcessAD$((0^r \| K), A)$
end
$(M, V_c) \leftarrow$ PRØST-APE-n-Dec$(T \oplus K, C)$
if $IV_c = V_c$ **then**
  return M
end
else
  return $\perp$
end

Algorithm 11: PRØST-APE-n-Enc(IV, M)

Data: $IV \in \mathbb{F}_2^{r+c}, M \in (\mathbb{F}_2^r)^+$
Result: $C \in (\mathbb{F}_2^r)^+, \hat{V}_c \in \mathbb{F}_2^c$
$V \leftarrow IV \oplus (0^r \| 1)$
$M[1]M[2] \cdots M[m] \leftarrow M$
for $i = 1, \ldots, m$ **do**
  $\hat{V} \leftarrow P_n((M[i] \oplus V_r) \| V_c)$
  $C[i] \leftarrow \hat{V}_r$
  $V \leftarrow \hat{V}$
end
return $(C[1]C[2] \cdots C[m], \hat{V}_c)$

Algorithm 12: PRØST-APE-n-Dec$(\hat{V}_c, C)$

Data: $IV_r \in \mathbb{F}_2^r, Cin(\mathbb{F}_2^r)^+, \hat{V}_c \in \mathbb{F}_2^c$
Result: $M \in (\mathbb{F}_2^r)^+, V_c \in \mathbb{F}_2^c$
$C[1]C[2] \cdots C[m] \leftarrow C$
$C[0] \leftarrow IV_r$
for $i = m, \ldots, 1$ **do**
  $V \leftarrow P_n^{-1}(C[i] \| \hat{V}_c)$
  $M[i] \leftarrow C[i-1] \oplus V_r$
  $\hat{V}_c \leftarrow V_c$
end
return $(M[1]M[2] \cdots M[m], V_c \oplus 1)$

Algorithm 13: PRØST-APE-n-ProcessAD(V, A)

Data: $V \in \mathbb{F}_2^{r+c}, A \in (\mathbb{F}_2^r)^+$
Result: $\hat{V} \in \mathbb{F}_2^{r+c}$

$A[1]A[2] \cdots A[u] \leftarrow A$
for $i = 1, \ldots, u$ **do**
    $\hat{V} \leftarrow P_n((A[i] \oplus V_r) \| V_c)$
    $V \leftarrow \hat{V}$
end
return $\hat{V}$

3 Security Goals

Table 4: Ranks and security claims for our proposals for plaintext confidentiality (PT_{CONF}), plaintext integrity (PT_{INT}) and associated data integrity (AD_{INT})

Rank	Proposal	PT_{CONF}	PT_{INT}	AD_{INT}
1	PRØST-COPA-128	64	64	64
2	PRØST-COPA-256	128	128	128
3	PRØST-OTR-128	64	64	64
4	PRØST-OTR-256	128	128	128
5	PRØST-APE-256[128, 128]	128	128	128
6	PRØST-APE-128[64, 64]	64	64	64

Remarks regarding additional security

The COPA and APE proposals offer a level of resistance against nonce-misuse. For the details of the nonce-misuse resistance of APE and COPA parameter sets, we refer to their respective mode papers[2, 3]. Additionally, all proposals have the following additional security goals:

- Any straight-forward implementation will run in constant time.
- Due to the choice of a lightweight 4-bit Sbox as the only non-linear element, countermeasures against power/EM side-channel attacks are much cheaper.

4 Design Rationale

The main design rational for the PRØST permutation was efficiency and strong, easily verifiable, security arguments.

Strong arguments are possible as we follow the wide-trail strategy. Here we modified the usual AES-like structure by interleaving two different shift-rows like operation. This results in significantly improved bounds on the best linear and differential characteristics.

Efficieny is a result of mainly two efforts. First we optimized every single component with respect to implementation cost and second the strong arguments mentioned above allowed us to keep the number of rounds rather small.

Below, we describe the design rational behind the separate components within the PRØST permutation.

4.1 SubRow

For the `SubRow` operation, we use a very simple (in terms of hardware/software-efficiency – the formulation is given in Appendix B), 10-instruction, 4-bit $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ S-box. The concrete S-box was the result of a hardware assisted search though a significant subset of all possible S-boxes. Besides being very efficient in terms of cycle count, this S-box is also optimal with respect to linear and differential attacks.

We chose S among all S-boxes fulfilling the following criteria.

1. S is an involution, which prevents the encryption/decryption overhead.
2. The maximal probability of a differential is 1/4.
3. There are exactly 24 differentials with probability 1/4.
4. The maximal absolute bias of a linear approximation is 1/4.
5. There are exactly 36 linear approximations with absolute bias 1/4.
6. Output bits have algebraic degrees of 2,2,3, and 3, respectively.

Having only one single S-box within one plane allows to implement the S-box application using bit-slicing. On top, keeping the S-boxes identical for all planes and all rounds reduced the code space and avoids additional overhead. The increased danger of symmetries throughout the cipher is countered by relatively heavy round constants.

4.2 MixSlices

We had three major requirements for the `MixSlices` operation.

- Linear and differential branch number 5
- Low density
- As a heuristic to minimize implementation characteristics for both encryption and decryption, `MixSlices` is its own inverse

We elaborate on the first two requirements below.

High Branch Number The main design goal of the `MixSlices` transformation is to follow the wide trail strategy. Hence, the `MixSlices` transformation is related to an $\mathbb{F}_2$-linear error-correcting code over $\mathbb{F}_2^4$ with minimum distance 5. Note that in our setting linear and differential branch number are identical.

In other words, a difference in $k > 0$ rows of a slice will result in a difference of at least $5 - k$ rows after one `MixSlices` application. While this is a good bound for 2 rounds, only the interaction with the `ShiftPlanes` operation guarantees a secure overall design.

Low density The density roughly corresponds to the number of XOR operations that have to be preformed when implementing the matrix. It is therefore a suitable metric when optimizing performance – both in software and harware.

We searched among all 16×16 binary matrices with are involutions and have branch number 5 for ones with a particular small number of ones. The optimal solution, i.e. the lowest density of such a matrix, we where able to find with our hardware assisted search had 88 ones. Note that we cannot guarantee that our matrix is actually optimal (as the minimal number of ones in such a matrix is unknown).

4.3 ShiftPlanes

We had two design criteria for this transformation. Firstly, we needed shift values which result in "optimal diffusion".

The (π_1, π_2) pairs for PRØST (see Table 2) were found to be optimal in the sense that they give the best diffusion, number of active S-boxes (see also Sections 5.1 and 5.2) and implementation cost, for the specified register lengths d.

For register length $d = 16$ we obtain full diffusion after 2 rounds and for $d = 32$ after 3 rounds. For lower bounds on the number of active S-boxes over various number of rounds, we refer to Section 5.2.

With respect to implementation cost, we have optimized to have as many multiples of 8 as possible, as these are free on 8-bit platforms and cheap on larger platforms. For the constants that are not multiples of 8, we minimize their sum, as the implementation cost is proportional to the constant. This sum is 22 for the $d = 16$ case and 88 for the $d = 32$ case, which roughly translates to the implementation cost in cycles.

4.4 AddConstants

The purpose of adding round constants is to make each round different. If the rounds are all the same, then fixed points x such that $R(x) = x$ for the round function R extend to the entire permutation. For example, if $P = R^{10}$, then fixed points for R^2 and R^5 would also extend to P. Therefore, one can expect several fixed points for P, whereas for an ideal permutation only a single fixed point is expected. By choosing round-dependent constants for `AddConstant`, we expect the number of fixed points to be close to 1.

We use two 32-bit constants c_1 and c_2 from which all round constants are derived through rotations. Round i uses $c_1 \ggg i$ and $c_2 \ggg i$. The two constants $c_1 = 0x75817b9d$ and $c_2 = 0xb2c5fef0$ are generated from the first 64 digits after the decimal point of π as illustrated by the following C code:

```
#include <stdint.h>
#include <stdio.h>

#define ROR32(X,N) ((X>>N) | (X<<(32-N)))

const uint32_t pi[64] = {1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,
                         3,8,4,6,2,6,4,3,3,8,3,2,7,9,5,0,
                         2,8,8,4,1,9,7,1,6,9,3,9,9,3,7,5,
                         1,0,5,8,2,0,9,7,4,9,4,4,5,9,2,3};

int main(void)
{
  uint32_t c1=0;
  uint32_t c2=0;
  int i;
  for(i=0;i<32;i++)
  {
    c1 |= (pi[i]&1) << i;
    c2 |= (pi[i+32]&1) << i;
  }
  printf("c1 = 0x%08x\n", c1);
  printf("c2 = 0x%08x\n", c2);
  return 0;
}
```

5 Security Analysis

In here, we summarize some of our findings in the security analysis. A more extensive treatment will be available from[18].

5.1 Diffusion and Strict Avalanche Criterion

The well-known concept of diffusion was first defined by Shannon in his 1949 seminal work [17]. In its original meaning, the goal of diffusion is to lessen the short-term statistical properties from the plaintext in the resulting ciphertext, such than an attacker would need to obtain a lot of ciphertext to infer knowledge of the statistical properties of the plaintext. The modern definition of diffusion slightly different, and we take diffusion to mean the extent to which output bits depend on input bits. We say we have *full diffusion* when each bit of the output depends on each bit of the input.

A condition somewhat parallel to diffusion is the *Strict Avalanche Criterion* (SAC). A cipher or permutation is said to satisfy the SAC when flipping any bit of the input results in flipping each bit of the output with probability 1/2 (for a block cipher, one averages over all keys). In this section we present our findings on diffusion and the SAC for the PRØST permutation.

Lastly, we consider the *avalanche effect*, which is the property that on average, half the bits of the output are flipped when a single input bit is flipped.

5.1.1 Diffusion

For our analysis, we assume that for the 4-bit S-box used, each of the 4 output bits depend on each of the 4 input bits. Also, we assume that the `MixSlices` operation mixes the bits in a column, so each of the h output bits depend on each of the input bits.

After applying `SubRows` of the first round, all bits within a row are interdependent by assumption. Applying `MixSlices` of the first round implies that we get bit interdependency between all bits in the same slice. The `ShiftPlanes` operation cyclically shifts the planes by a round-dependent amount, essentially mixing the slices.

As `SubRows` and `MixSlices` take care of mixing bits in slices in each round, the question becomes how many rounds are required to make the plane shifting of `ShiftPlanes` create bit-interdependency between all planes.

When determining the choice of (π_1, π_2) vectors, we conducted an experiment trying out all combinations of $\pi_1, \pi_2 \subset \{0, \ldots, d-1\}$ with 4 elements, and investigate the required number of rounds to obtain full diffusion. For $d = 16$ we found that the best pairs (π_1, π_2) require a minimum of 2 rounds for full diffusion, and there are 4096 such pairs. For $d = 32$ a minimum of 3 rounds are required for full diffusion, and there are 729088 pairs (π_1, π_2) obtaining this minimum. Unfortunately, there is no pair (π_1, π_2) which obtains the best diffusion for both $d = 16$ and $d = 32$, hence the two parameter sets in Table 2.

5.1.2 SAC and Avalanche Effect

For the purpose of investigating the SAC and avalanche effect, we conduct randomized experiments to measure the degree to which these two properties are obtained. For the $2n$-bit permutation PRØST-n, the statements of the avalanche effect and SAC are given by Equations (2) and (3), respectively, where $\mathbf{e}_i$ denotes $2n$-bit string with a 1 on position i and zeroes elsewhere, and

subscript j denotes the jth bit.

$$\forall i \in \{1, \ldots, 2n\} \left(2^{-2n} \sum_{x \in \mathbb{F}_2^{2n}} \text{wt}\left(\text{PRØST-}n(x) \oplus \text{PRØST-}n(x \oplus \mathbf{e}_i)\right) = n \right) \tag{2}$$

$$\forall i \in \{1, \ldots, 2n\} \left(\forall j \in \{1, \ldots, 2n\} \left(\mathbf{Pr}[\text{PRØST-}n(x)_j \neq \text{PRØST-}n(x \oplus \mathbf{e}_i)_j] = \frac{1}{2} \right) \right) \tag{3}$$

To experimentally measure the extent to which PRØST meets these two criteria, we sample a random $X \subset \mathbb{F}_2^{2n}$ and define the degree of avalanche effect $\deg_{ava}$ and degree of SAC $\deg_{SAC}$ as defined by Serf in his investigation of the AES finalists [16]. Our results show that we obtain SAC and avalanche effect degrees of 1.0 (which is ideal) after 2-3 rounds for $d = 16$ and 3-4 rounds for $d = 32$.

5.2 Bounds on Active S-boxes

Here we give lower bounds on the number of active S-boxes for various PRØST parameters. These bounds are of paramount importance as, combined with the differential- and linear properties of the S-box, provide upper bounds on the differential- and linear trail probabilities for the full PRØST permutation.

To lower bound the number of active S-boxes for $d = 16$ and $d = 32$, we model the propagation of active S-boxes over a particular number of rounds as an integer programming problem. A part of choosing from the (π_1, π_2) shift vectors giving optimal diffusion, for $d = 16$ and $d = 32$, has been to solve this program for randomly chosen subsets of (π_1, π_2) and choosing thos giving the best bounds.

The findings for the number of active S-boxes for the (π_1, π_2) from Table 2, for various number of rounds, are given in Table 5. The integer programming problem is modeled in Appendix A.

Table 5: Lower bounds on the number of active S-boxes for $d \in \{16, 32\}$ for various number of rounds. In the $d = 32, T = 8$ case, the number in parenthesis was the best obtained when the solver stopped due to memory limitations.

	Rounds T				
d	4	5	6	7	8
16	25	41	85	96	105
32	25	41	105	169	(210)

6 Features

All the proposed parameter sets have the following features of the permutation in common.

- Designed for straight-forward bit-sliced implementations.
- Even the most straight-forward implementation of the permutation leads to constant execution times.
- Due to the choice of a lightweight 4-bit Sbox as the only non-linear element, countermeasures against power/EM side-channel attacks are much cheaper than ARX or AES-based designs.
- Fast and compact in software on a wide range of platforms.

- Fast, compact, energy-efficient, and allows for low-latency implementations in hardware implementations.

We will back up these claim in upcoming supplementary documents on implementation aspects of PRØST.
All the proposed modes have the following features in common:

- They are based on a large cryptographically secure permutation. This is arguable one of the simplest, if not the simplest way to build primitives for symmetric key cryptography, including authenticated encryption primitives. In particular, this avoids complicated and often very inelegant considerations related to key schedules for traditional block cipher based constructions.

- They are based on a *single* permutation. Having a single permutation further avoids considerations related to the independence of permutations.

All of the above described features are advantages over AES and AES-GCM. In addition, we'd like to point out the following features which is shared with AES-GCM, but other proposals are often lacking:

- Simple and clean design.

- Strong bounds against large classes of cryptanalytic attacks, like linear and differential cryptanalysis which are among the most powerful attack vectors known. This has two advantages. Firstly it gives assurance that these attack vectors in their basic form do not lead to attacks on the proposal. On top of that we chose a number of rounds that gives a large security margin. Secondly, it safes valuable cryptanalysis time. Finding and improving upon known statistical properties for linear and differential attacks is a very time consuming task for cryptanalysts. Knowledge of bounds saves a lot of this time, which can instead be spent on other attack vectors or more advanced attacks. Most of our bounds are tight, i.e. we are able to give matching characteristics/trails[18]. This will be of independent interest for external cryptanalysts.

Now we discuss more specific features of the parameter sets. We start with PRØST-COPA, our main proposal, that comes with two security levels. On top of the advantages mentioned above, it offers a level of nonce-misuse resistance. We feel this is an important property as in many environments this seemingly simple task of keeping track of nonces and making sure they are unique, e.g. by implementing a counter, can actually be very difficult. Resets in virtualized environments, or very resource constrained environments, are examples of such situations.

For environments where respecting the requirements for unique nonces is easy to achieve, we propose PRØST-OTR, again with two security levels. This comes with improved implementation characteristics.

Both classes of parameter sets allow for parallelization, which, together with the bit-sliced nature of our permutation, allows for very efficient implementations on modern SIMD architectures.

As a complementary set of parameters, we propose PRØST-APE. It's advantages are the possibility of a very fine-grained adjustment of the security against generic attack, by allowing different performance/security trade-offs. To focus cryptanalyst attention, we limit ourselves to the two mean security levels though. Also, in contrast to the PRØST-COPA and PRØST-OTR proposals, for PRØST-APE we do not need twice the key length for the claimed security level.

For the details of the nonce-misuse resistance of APE and COPA parameter sets, we refer to their respective mode papers[2, 3].

7 Intellectual Property

The design team behind the PRØST submission for the CAESAR competition are not aware of any known patents, patent applications, planned patent applications or other intellectual-property constraints pertaining to the use of the cipher. If any of this information changes, the submitter/submitters will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

8 Consent

The submitter/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 submitter/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 analyses that led to the selection of the algorithm. The submitter/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 submitter/submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter/submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter/submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

References

[1] F. Abed, S. Fluhrer, C. Forler, E. List, S. Lucks, D. McGrew, and J. Wenzel. Pipelineable On-Line Encryption. In C. Cid and C. Rechberger, editors, *Fast Software Encryption*, LNCS. Springer, to appear. 3

[2] E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda. Ape: Authenticated permutation-based encryption for lightweight cryptography. In C. Cid and C. Rechberger, editors, *Fast Software Encryption*, LNCS. Springer, to appear. http://eprint.iacr.org/2013/791/. 3, 11, 14, 19

[3] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda. Parallelizable and Authenticated Online Ciphers. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8269 of *LNCS*, pages 424–443. Springer, 2013. eprint.iacr.org/2013/790/. 3, 4, 5, 14, 19

[4] E. Andreeva, A. Luykx, B. Mennink, and K. Yasuda. COBRA: A Parallelizable Authenticated Online Cipher Without Block Cipher Inverse. In C. Cid and C. Rechberger, editors, *Fast Software Encryption*, LNCS. Springer, to appear. 3

[5] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Keccak specifications. Submission to NIST, 2008. http://keccak.noekeon.org/Keccak-specifications.pdf. 1

[6] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In A. Miri and S. Vaudenay, editors,

Selected Areas in Cryptography, volume 7118 of *LNCS*, pages 320–337. Springer, 2012. http://eprint.iacr.org/2011/499.pdf. 3

[7] B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, and Q. Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer-Verlag Berlin Heidelberg, 2013. 3

[8] O. Dunkelman, N. Keller, and A. Shamir. Minimalism in cryptography: The even-mansour scheme revisited. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 336–354, 2012. http://eprint.iacr.org/2011/541/. 4

[9] E. Fleischmann, C. Forler, S. Lucks, and J. Wenzel. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. Cryptology ePrint Archive, Report 2011/644, 2011. http://eprint.iacr.org/2011/644/. 3

[10] D. Khovratovich. PPAE: parallelizable permutation-based authenticated encryption. Presentation at Directions in Authenticated Ciphers – DIAC 2013, 2013. http://2013.diac.cr.yp.to/slides/khovratovich.pdf. 3

[11] T. Krovetz and P. Rogaway. The Software Performance of Authenticated-Encryption Modes. In *FSE*, pages 306–327, 2011. 3

[12] D. A. McGrew and J. Viega. The Galois/Counter Mode of operation (GCM), 2004. http://www.cryptobarn.com/papers/gcm-spec.pdf. 3

[13] K. Minematsu. Parallelizable Authenticated Encryption from Functions. Cryptology ePrint Archive, Report 2013/628, 2013. http://eprint.iacr.org/2013/628/. 3, 7

[14] P. Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In P. J. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, 2004. http://www.cs.ucdavis.edu/~rogaway/papers/offsets.pdf. 3

[15] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption. In *ACM Conference on Computer and Communications Security*, pages 196–205, 2001. 3

[16] P. Serf. The degrees of completeness, of avalanche effect, and of strict avalanche criterion for MARS, RC6, Rijndael, Serpent, and Twofish with reduced number of rounds. 2000. 18

[17] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949. http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf. 17

[18] The Prøst team. Security Analysis of Prost, 2014 (to appear). 17, 19

[19] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Proposed Standard), 2008. http://www.ietf.org/rfc/rfc3610.txt. 3

[20] H. Wu and B. Preneel. AEGIS: A Fast Authenticated Encryption Algorithm. Cryptology ePrint Archive, Report 2013/695, 2013. http://eprint.iacr.org/2013/695/. 3

A Bounds on Active S-boxes

To determine lower bounds on the number of active S-boxes for a given number of rounds R, we employ the modeling of the permutation and its propagation of active S-boxes as a mixed integer programming problem. Consider the n-bit state as viewed from the (y, z)-plane, which yields a two-dimensional structure of h rows and d columns (we use here the more general size of h rows in each slice). We now let $x_{i,j}^t$ be a binary variable with $0 \leq i < h$, $0 \leq j < d$ and $0 \leq t \leq 2R$. We let $x_{i,j}^t = 1$ if and only if row i in sheet j is active when going operation t, starting from index 0, of the chain $(\texttt{ShiftPlanes} \circ \texttt{MixSlices})^R$. For example x^0 is the input to the first `MixSlices` operation, x^1 is the input to the first `ShiftPlanes` which uses π_1, etc.

We can now model the propagation of active S-boxes as the propagation of ones in x over time t. First, to require at least one active S-box in the input, we use the constraint given by (4):

$$\sum_{i=0}^{h-1} \sum_{j=0}^{d-1} x_{i,j}^1 > 0. \tag{4}$$

For every 4th value of t, we apply `ShiftPlanes` with π_1 starting from $t = 1$, and similarly for `ShiftPlanes` using π_2 starting with $t = 3$. To model the behaviour of these operations we use the following constraints:

$$\forall t \in \{1, 5, \ldots, 2R-3\}, i \in \{0, \ldots, h-1\}, j \in \{0, \ldots, d-1\} : x_{i,j}^t = x_{i,(j+\pi_1[i] \mod d)}^{t+1} \tag{5}$$

$$\forall t \in \{3, 7, \ldots, 2R-1\}, i \in \{0, \ldots, h-1\}, j \in \{0, \ldots, d-1\} : x_{i,j}^t = x_{i,(j+\pi_2(i) \mod d)}^{t+1} \tag{6}$$

The constraint (5) models the behaviour of `ShiftPlanes` with π_1 and constraint (6) models the behaviour of `ShiftPlanes` with π_2.

For every other value of t, starting with $t = 0$, x^t is input to the `MixSlices` operation. Here, we want to model that

$$\forall t \in \{0, 2, \ldots, 2R-2\}, j \in \{0, \ldots, d-1\} : \sum_{i=0}^{h-1} x_{i,j}^t + \sum_{i=0}^{h-1} x_{i,j}^{t+1} \geq \begin{cases} \mathcal{B} & , \sum_{i=0}^{h-1} x_{i,j}^t > 0 \\ 0 & , \sum_{i=0}^{h-1} x_{i,j}^t = 0 \end{cases},$$

where $\mathcal{B}$ is the branch number. To model the two cases (either or), we introduce a binary variable a_j^t which is 1 if and only if $\sum_{i=0}^{h-1} x_{i,j}^t > 0$. Using this, we model the operation of `MixSlices` by constraints (7) through (10):

$$\forall t \in \{0, 2, \ldots, 2R-2\}, j \in \{0, \ldots, d-1\} : \sum_{i=0}^{h-1} x_{i,j}^t + \sum_{i=0}^{h-1} x_{i,j}^{t+1} \leq 2h \cdot a_j^t \tag{7}$$

$$\forall t \in \{0, 2, \ldots, 2R-2\}, j \in \{0, \ldots, d-1\} : \sum_{i=0}^{h-1} x_{i,j}^t + \sum_{i=0}^{h-1} x_{i,j}^{t+1} \geq \mathcal{B} \cdot a_j^t \tag{8}$$

$$\forall t \in \{0, 2, \ldots, 2R-2\}, j \in \{0, \ldots, d-1\} : \sum_{i=0}^{h-1} x_{i,j}^t \leq h \cdot a_j^t \tag{9}$$

$$\forall t \in \{0, 2, \ldots, 2R-2\}, j \in \{0, \ldots, d-1\} : \sum_{i=0}^{h-1} x_{i,j}^t \geq a_j^t. \tag{10}$$

Finally, we introduce the variable z which we want to minimise:

$$z = \sum_{t \in \{0,2,\ldots,2R-2\}} \sum_{i=0}^{h-1} \sum_{j=0}^{d-1} x_{i,j}^t. \tag{11}$$

B S-box Formulation

$$
\begin{aligned}
&p = a;\\
&q = b;\\
&a = c \oplus (p\&q);\\
&b = d \oplus (q\&c);\\
&c = p \oplus (a\&b);\\
&d = q \oplus (b\&c);
\end{aligned}
$$