Deoxys v1.41

Designers/Submitters:

Jérémy Jean [1,2], Ivica Nikolić [2], Thomas Peyrin [2], Yannick Seurin [1]

[1] ANSSI, Paris, France

[2] Division of Mathematical Sciences,
School of Physical and Mathematical Science,
Nanyang Technological University, Singapore

{Jeremy.Jean,Yannick.Seurin}@ssi.gouv.fr,
{INikolic,Thomas.Peyrin}@ntu.edu.sg

http://www1.spms.ntu.edu.sg/~syllab/Deoxys

October 12, 2016

Contents

Chapter 1

Introduction

In this note, we propose Deoxys, a new authenticated encryption design based on a tweakable block cipher Deoxys-BC using the well-studied AES round function as a building block. We suggest several sets of parameters that can use different key and tweak sizes, and claim security levels for all the parameters in later sections. Our design uses a particular instantiation of a more general framework (so-called TWEAKEY [23]) allowing designers to unify the vision of key and tweak inputs of a cipher. We plug this cipher into two different yet very close fully parallel and provably secure authenticated encryption modes: one for which the nonce must not be reused, the other one providing security even when the nonce is reused.

In short, Deoxys is an authenticated encryption scheme that provides full 128-bit security (in contrary to AES-GCM [30] or OCB [27]) for both privacy and authenticity. It performs well in software, being faster than AES-GCM [30] on most processors. Moreover, Deoxys performs particularly well for small messages (only $m + 1$ block cipher calls are required for a m block message and no precomputation is required). In the nonce-misuse resistant versions of Deoxys, in addition to a full 128-bit security for unique nonces, we obtain birthday-bound security (not an online nonce-misuse resistance as defined in [17], but a full MRAE security notion [35]) when the nonce is reused. This is done very simply as a tweakable block cipher is a quite handy primitive to build an authenticated encryption scheme. Finally, Deoxys can be lightweight (using existing AES lightweight implementation, the extra area mainly consisting in 192 extra bits of memory for the mode and to store the tweak) and the key can be hardcoded for further smaller area footprint.

Organization of the paper. In Chapter 2, we provide the specification of our proposal Deoxys, including the description of the TWEAKEY framework and the sets of parameters for this proposal. In Chapter 3, we precise the security claims for different scenarios for the various parameters, and in Chapter 4 we perform some security analysis regarding this proposal. In Chapters 5 and 6, we detail some design decisions, and finish with Chapters 8 and 9 where we give notes on intellectual property and consent.

Chapter 2

Specification

In this chapter, we present the full specification of our proposal `Deoxys`. We first give the recommended parameter sets and then proceed with the description of the design. We explain the two authenticated encryption modes `Deoxys-I` (non-repeating nonces) and `Deoxys-II` (nonce-repeating scenario), and then we describe the ad-hoc `AES`-based tweakable block cipher `Deoxys-BC` (which is based on the `TWEAKEY` framework [23]) used to instantiate the modes.

Notations. We first introduce some notations. We denote $E_K(T, P)$ the ciphering of the n-bit plaintext P with the tweakable block cipher `Deoxys-BC` with k-bit key K and t-bit tweak T (similarly, D represents the deciphering process). The concatenation operation is represented by $||$ and $pad10^*$ is the function that applies the 10^* padding on n bits, i.e. $pad10^*(X) = X||1||0^{n-|X|-1}$ when $|X| < n$. For an empty string ϵ, the 10^* padding will not add any bit: $pad10^*(\epsilon) = \epsilon$. The truncation of the word X to the first i bits is given by $\lceil X \rceil_i$, and the truncation to the last i bits by $\lfloor X \rfloor_i$. Moreover, $X \lll a$ will denote the word X rotated by a positions to the left.

Our authenticated encryption scheme `Deoxys` is composed of an encryption part and a verification/decryption part. The encryption part $\mathcal{E}$ takes as input a variable-length plaintext M (with $m = |M|$), a variable-length associated data A (with $a = |A|$), a fixed-length public message number N and a k-bit key K (we deliberately used the same letter K to represent the key in the authenticated encryption scheme and the one in the tweakable block cipher, since they always refer to the same object). It outputs a m-bit ciphertext C and a τ-bit tag, denoted `tag` (with $\tau \in [0, \ldots, n]$), i.e. $(C, \texttt{tag}) = \mathcal{E}_K(N, A, M)$. The verification/decryption part $\mathcal{D}$ takes as input a variable-length ciphertext C (with $m = |C|$), a τ-bit tag `tag` (with $\tau \in [0, \ldots, n]$), a variable-length associated data A (with $a = |A|$), a fixed-length public message number N and a k-bit key K. It outputs either an error string $\perp$ to signify that the verification failed, or a m-bit string $M = \mathcal{D}_K(N, A, C, \texttt{tag})$ when the tag is valid. The maximum message length (in n-bit blocks) is denoted max_l and the maximum number of messages that can be handled with the same key is denoted max_m (the same limitation applies to the associated data material). We have that $max_l = 2^{\lceil t/2 \rceil - 4}$ and $max_m = 2^{\lfloor t/2 \rfloor}$. This will ensure that as long as different fixed-length public message numbers (i.e. nonces) are used, the tweak inputs of all the tweakable block cipher calls are all unique. This also naturally implies that $|N| \geq \log_2(max_m) = \lfloor t/2 \rfloor$. Note that there is a tradeoff possible between max_l and max_m, as long as $max_l \cdot max_m = 2^{t-4}$.

2.1 Parameters

A first parameter for `Deoxys` is the key length k, which is either 128 or 256 bits. We then propose two modes: the first is for nonce-respecting adversaries (denoted with `I`, to represent the fact that

nonces should only be used once), while the second offers nonce misuse-resistance (denoted with II, to show that nonces can appear twice or more). For this reason, we introduce another parameter, that signals the mode of our authenticated encryption scheme. The tag size τ is recommended to be 128 bits, while the public message length $|N|$ is fixed to 64 bits for the first mode, and 120 bits for the second mode.

2.2 Recommended Parameter Sets

The public message number is the nonce. For each of the two modes we recommend two parameter sets (hence in total we have four sets), listed in Table 2.1. The list is sorted from most important to least important. We denote by Deoxys-I the design in the case of the nonce-respecting mode and Deoxys-II the design in the case of the nonce-misuse resistant mode.

The table additionally makes explicit the sorted list of targeted use cases (most important use case first), according to the classification provided in Table 2.2.

Name	k	t	n	$\|N\|$	τ	Use Cases
Deoxys-I-128-128	128	128	128	64	128	2, 1, 3
Deoxys-I-256-128	256	128	128	64	128	2, 1, 3
Deoxys-II-128-128	128	128	128	120	128	3, 2, 1
Deoxys-II-256-128	256	128	128	120	128	3, 2, 1

Table 2.1: Recommended parameter sets for Deoxys. Parameters k, t, $|N|$ and τ are related to the signature of the inner tweakable block cipher of Deoxys.

We note that the two schemes Deoxys-I-128-128 and Deoxys-II-128-128 are based on the internal block cipher Deoxys-BC-256, while Deoxys-I-256-128 and Deoxys-II-256-128 are based on the internal block cipher Deoxys-BC-384.

2.3 The Authenticated Encryption Deoxys

In this section, we provide the high-level description of our proposal. Deoxys uses a tweakable block cipher Deoxys-BC as internal primitive (specified in Section 2.4), and we describe here the simple authenticated encryption modes built on top of it. Deoxys has two main mode variants:

- Deoxys-I, relying on algorithms $\mathcal{E}^{\mathtt{I}}$ and $\mathcal{D}^{\mathtt{I}}$ (see Section 2.3.1): this first variant is for adversaries that are assumed to be nonce-respecting, meaning that the user must ensure that the value N will never be used for encryption twice with the same key. This mode is similar to TAE [29] or ΘCB3 [27] (the tweakable block cipher generalization of OCB3). We denote $\mathcal{E}^{\mathtt{I}}$ the encryption part of this first variant and $\mathcal{D}^{\mathtt{I}}$ the verification/decryption part.
- Deoxys-II, relying on algorithms $\mathcal{E}^{\mathtt{II}}$ and $\mathcal{D}^{\mathtt{II}}$ (see Section 2.3.2): this second variant relaxes the uniqueness constraint and allows the user to reuse the same N with the same key. We call this mode SCT-2 as it heavily relies on SCT, an inverse-free authenticated encryption mode published in [33]. We denote $\mathcal{E}^{\mathtt{II}}$ the encryption part of this first variant and $\mathcal{D}^{\mathtt{II}}$ the verification/decryption part.

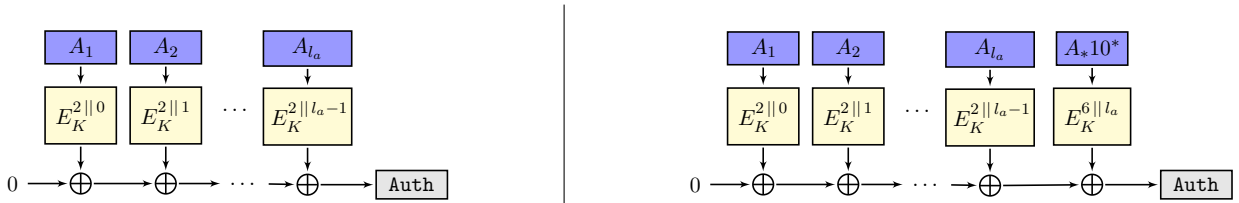
In both modes, we use short 4-bit prefixes for the tweak input in order to properly separate the various types of encryption/authentication blocks. Is is to be noted that the two modes are actually quite similar, the main difference being that the first one applies one pass on the message blocks, while the second performs two passes (which is necessary to obtain a MRAE security notion [35]).

Use Case 1 – Lightweight applications (resource constrained environments)	
critical	fits into small hardware area and/or small code for 8-bit CPUs
desirable	natural ability to protect against side-channel attacks
desirable	hardware performance, especially energy/bit
desirable	speed on 8-bit CPUs
message sizes	usually short (can be under 16 bytes), sometimes longer
Use Case 2 – High-performance applications	
critical	efficiency on 64-bit CPUs (servers) and/or dedicated hardware
desirable	efficiency on 32-bit CPUs (small smartphones)
desirable	constant time when the message length is constant
message sizes	usually long (more than 1024 bytes), sometimes shorter
Use Case 3 – Defense in depth	
critical	authenticity despite nonce misuse
desirable	limited privacy damage from nonce misuse
desirable	authenticity despite release of unverified plaintexts
desirable	limited privacy damage from release of unverified plaintexts
desirable	robustness in more scenarios; e.g., huge amounts of data

Table 2.2: List of typical AE use-cases selected by the CAESAR committee.

2.3.1 Nonce-Respecting Mode: $\mathcal{E}^{\mathtt{I}}$ and $\mathcal{D}^{\mathtt{I}}$

The encryption algorithm $\mathcal{E}^{\mathtt{I}}$ is depicted in Figures 2.1, 2.2 and 2.3, and an algorithmic description is given in Algorithm 1. The verification/decryption algorithmic description of $\mathcal{D}^{\mathtt{I}}$ is given in Algorithm 2. We note that our scheme follows the framework from ΘCB3 [27] and therefore directly benefits from the security proof regarding authentication and privacy.



(a) Without padding. **(b)** With padding.

Figure 2.1: Handling of the associated data for the nonce-respecting mode: in the case where the associated data is a multiple of the block size, no padding is needed.

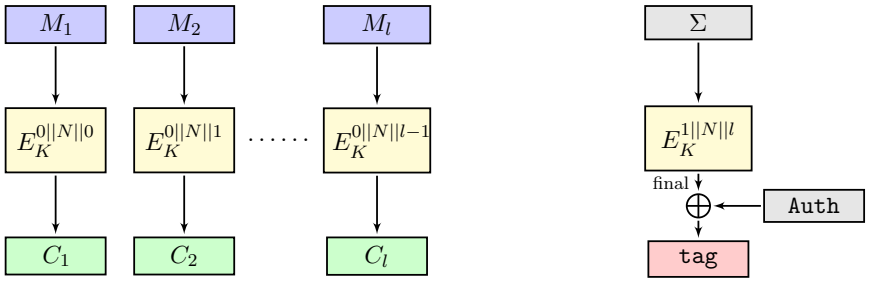


Figure 2.2: Message processing for the nonce-respecting mode: in the case where the message-length is a multiple of the block size, no padding is needed.

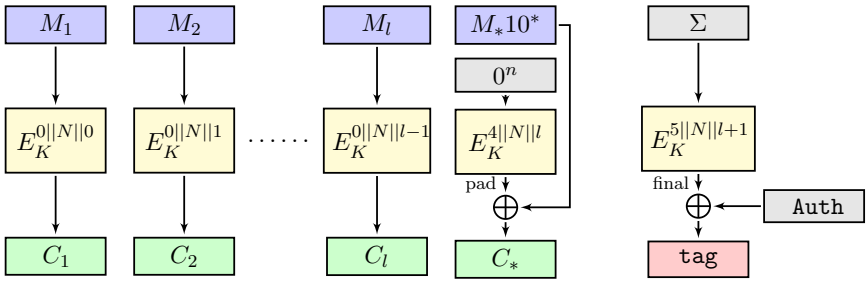


Figure 2.3: Message processing for the nonce-respecting mode: in the case where the message-length is a not multiple of the block size, padding is needed. Note that the checksum Σ is computed with a 10^* padding for block M^*.

Algorithm 1: The encryption algorithm $\mathcal{E}_K^{\mathtt{I}}(N, A, M)$.
In the tweak inputs, the value N is encoded on $\log_2(max_m)$ bits, the integer values j and l are encoded on $\log_2(max_l)$ bits, while the integer values i and l_a are encoded on $\log_2(max_l \cdot max_m) = t - 4$ bits.

/ Associated data */*
$A_1||\dots||A_{l_a}||A_* \leftarrow A$ where each $|A_i| = n$ and $|A_*| < n$
Auth $\leftarrow 0$
for $i = 0$ **to** $l_a - 1$ **do**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0010||i, A_{i+1})$
end
if $A_* \neq \epsilon$ **then**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0110||l_a, pad10^*(A_*))$
end

/ Message */*
$M_1||\dots||M_l||M_* \leftarrow M$ where each $|M_j| = n$ and $|M_*| < n$
Checksum $\leftarrow 0^n$
for $j = 0$ **to** $l - 1$ **do**
  Checksum $\leftarrow$ Checksum $\oplus\, M_j$
  $C_j \leftarrow E_K(0000||N||j, M_{j+1})$
end
if $M_* = \epsilon$ **then**
  Final $\leftarrow E_K(0001||N||l, \text{Checksum})$
  $C_* \leftarrow \epsilon$
else
  Checksum $\leftarrow$ Checksum $\oplus\, pad10^*(M_*)$
  Pad $\leftarrow E_K(0100||N||l, 0^n)$
  $C_* \leftarrow M_* \oplus \lceil \text{Pad} \rceil_{|M_*|}$
  Final $\leftarrow E_K(0101||N||l+1, \text{Checksum})$
end

/ Tag generation */*
tag $\leftarrow$ Final $\oplus$ Auth
return $(C_1||\dots||C_l||C_*, \mathtt{tag})$

Algorithm 2: The verification/decryption algorithm $\mathcal{D}_K^{\mathtt{I}}(N, A, C, \mathtt{tag})$.
In the tweak inputs, the value N is encoded on $\log_2(max_m)$ bits, the integer values j and l are encoded on $\log_2(max_l)$ bits, while the integer values i and l_a are encoded on $\log_2(max_l \cdot max_m) = t - 4$ bits.

/* *Associated data* */
$A_1||\ldots||A_{l_a}||A_* \leftarrow A$ where each $|A_i| = n$ and $|A_*| < n$
Auth $\leftarrow 0$
for $i = 0$ **to** $l_a - 1$ **do**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0010||i, A_{i+1})$
end
if $A_* \neq \epsilon$ **then**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0110||l_a, pad10^*(A_*))$
end

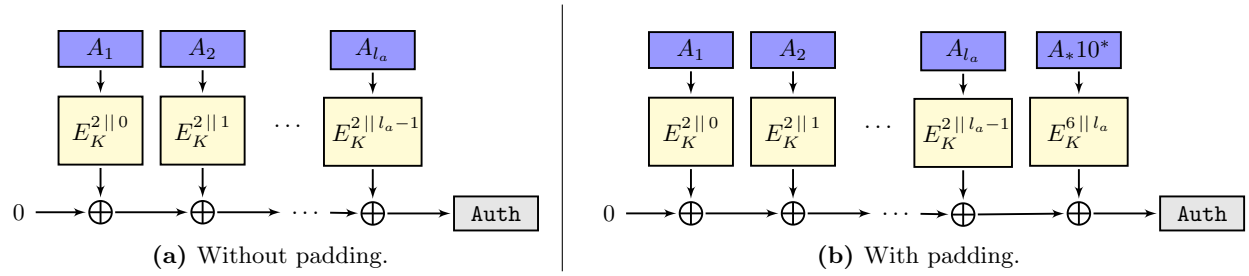
/* *Ciphertext* */
$C_1||\ldots||C_l||C_* \leftarrow C$ where each $|C_j| = n$ and $|C_*| < n$
Checksum $\leftarrow 0^n$
for $j = 0$ **to** $l - 1$ **do**
  $M_j \leftarrow D_K(0000||N||j, C_{j+1})$
  Checksum $\leftarrow$ Checksum $\oplus\, M_j$
end
if $C_* = \epsilon$ **then**
  Final $\leftarrow E_K(0001||N||l, \text{Checksum})$
  $M_* \leftarrow \epsilon$
else
  Pad $\leftarrow E_K(0100||N||l, 0^n)$
  $M_* \leftarrow C_* \oplus \lceil \text{Pad} \rceil_{|C_*|}$
  Checksum $\leftarrow$ Checksum $\oplus\, pad10^*(M_*)$
  Final $\leftarrow E_K(0101||N||l+1, \text{Checksum})$
end

/* *Tag verification* */
$\mathtt{tag}' \leftarrow$ Final $\oplus$ Auth
if $\mathtt{tag}' = \mathtt{tag}$ **then return** $(M_1||\ldots||M_l||M_*)$
else return $\perp$

2.3.2 Nonce-Misuse Resistant Mode: $\mathcal{E}^{\mathtt{II}}$ and $\mathcal{D}^{\mathtt{II}}$

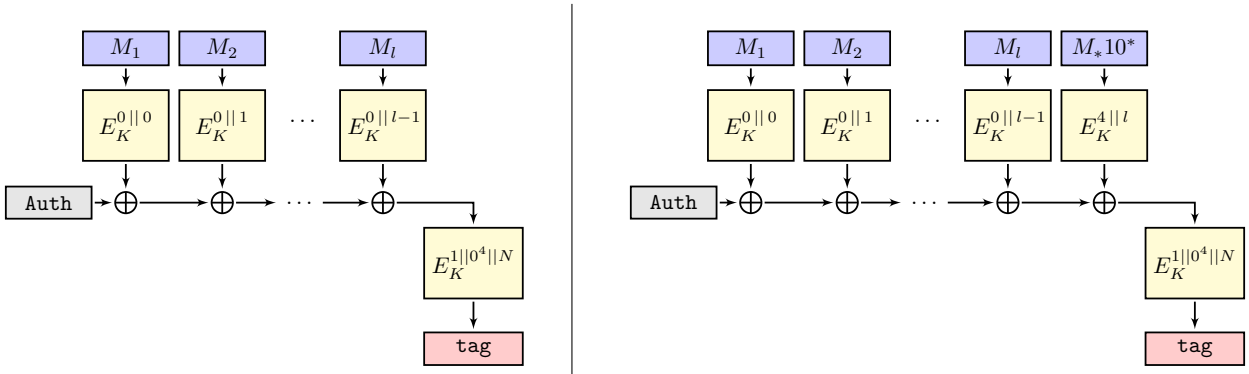
The encryption algorithm $\mathcal{E}^{\mathtt{II}}$ is depicted in Figures 2.4 and 2.5 for the authentication part and in Figure 2.6 for the encryption part. An algorithmic description is given in Algorithm 3. The verification/decryption algorithmic description of $\mathcal{D}^{\mathtt{II}}$ is given in Algorithm 4. Our mode is a variant of SCT (Synthetic Counter in Tweak, [33]) that we call SCT-2. The encryption part is kept unchanged compared with SCT, only the computation of the tag is modified in order to provide graceful degradation of security for authentication with the maximal number of repetitions of nonces [10] (a property that was ensured for confidentiality by SCT, but not for authenticity).

In this SCT-2 variant described below, the nonce N has a size of 120 bits, to include it in the tweak input of the block cipher call producing the tag. If necessary, nonce sizes up to 128 bits can be accommodated at the expense of an additional block cipher call. For this, one simply replaces the finalization of the tag (Lines 20 and 29 in the following algorithms) $\mathtt{tag}' \leftarrow E_K(0001||0^4||N, \mathtt{tag}')$ with $\mathtt{tag}' \leftarrow E_K(0001||0^4||N', \mathtt{tag}')$, where N' are the, say, 120 leftmost bits of the encryption of N with a reserved 4-bit tweak prefix that is used nowhere else in the mode.



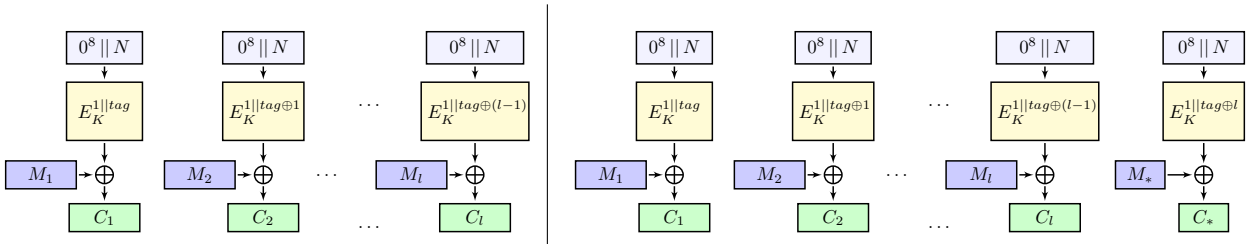
(a) Without padding. **(b)** With padding.

Figure 2.4: Handling of the associated data for the nonce-misuse resisting mode: in the case where the associated data is a multiple of the block size, no padding is needed.



(a) Without padding. **(b)** Without padding.

Figure 2.5: Message processing in the authentication part of the nonce-misuse resisting mode: in the case where the message-length is a multiple of the block size, no padding is needed.



(a) Message-length is a multiple of the block size. **(b)** Message-length is not a multiple of the block size.

Figure 2.6: Message processing for the encryption part of the nonce-misuse resisting mode.

Algorithm 3: The encryption algorithm $\mathcal{E}_K^{\mathtt{II}}(N, A, M)$.
In the tweak inputs, the integer values i, j, l and l_a are encoded on $\log_2(max_l \cdot max_m) = t - 4$ bits. Moreover, $tag + j$ values are encoded on $t - 1$ bits (the most significant bit is truncated since $|\mathtt{tag}| = t$). Recall that the nonce N contains 120 bits.

/* *Associated data* */
$A_1||\ldots||A_{l_a}||A_* \leftarrow A$ where each $|A_i| = n$ and $|A_*| < n$
Auth $\leftarrow 0$
for $i = 0$ **to** $l_a - 1$ **do**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0010||i, A_{i+1})$
end
if $A_* \neq \epsilon$ **then**
  Auth $\leftarrow$ Auth $\oplus\, E_K(0110||l_a, pad10^*(A_*))$
end

/* *Message authentication and tag generation* */
$M_1||\ldots||M_l||M_* \leftarrow M$ where each $|M_j| = n$ and $|M_*| < n$
$\mathtt{tag} \leftarrow$ Auth
for $j = 0$ **to** $l - 1$ **do**
  $\mathtt{tag} \leftarrow \mathtt{tag} \oplus E_K(0000||j, M_{j+1})$
end
if $M_* \neq \epsilon$ **then**
  $\mathtt{tag} \leftarrow \mathtt{tag} \oplus E_K(0100||l, pad10^*(M_*))$
end
$\mathtt{tag} \leftarrow E_K(0001||0^4||N, \mathtt{tag})$

/* *Message encryption* */
for $j = 0$ **to** $l - 1$ **do**
  $C_j \leftarrow M_j \oplus E_K(1||\mathtt{tag} \oplus j, 0^8\,||\,N)$
end
if $M_* \neq \epsilon$ **then**
  $C_* \leftarrow M_* \oplus E_K(1||\mathtt{tag} \oplus l, 0^8\,||\,N)$
end

return $(C_1||\ldots||C_l||C_*, \mathtt{tag})$

Algorithm 4: The verification/decryption algorithm $\mathcal{D}_K^{\mathtt{II}}(N, A, C, \mathtt{tag})$.
In the tweak inputs, the integer values i, j, l and l_a are encoded on $\log_2(max_l \cdot max_m) = t-4$ bits. Moreover, the $tag + j$ values are encoded on $t-1$ bits (the most significant bit is truncated since $|\mathtt{tag}| = t$). Recall that the nonce N contains 120 bits.

/* *Message decryption* */
$C_1||\ldots||C_l||C_* \leftarrow C$ where each $|C_j| = n$ and $|C_*| < n$
for $j = 0$ **to** $l-1$ **do**
  $M_j \leftarrow C_j \oplus E_K(1||\mathtt{tag} \oplus j, 0^8 \,||\, N)$
end
if $C_* \neq \epsilon$ **then**
  $M_* \leftarrow C_* \oplus E_K(1||\mathtt{tag} \oplus l, 0^8 \,||\, N)$
end

/* *Associated data* */
$A_1||\ldots||A_{l_a}||A_* \leftarrow A$ where each $|A_i| = n$ and $|A_*| < n$
Auth $\leftarrow 0$
for $i = 0$ **to** $l_a - 1$ **do**
  Auth $\leftarrow$ Auth $\oplus E_K(0010||i, A_{i+1})$
end
if $A_* \neq \epsilon$ **then**
  Auth $\leftarrow$ Auth $\oplus E_K(0110||l_a, pad10^*(A_*))$
end

/* *Message authentication and tag generation* */
$M_1||\ldots||M_l||M_* \leftarrow M$ where each $|M_j| = n$ and $|M_*| < n$
$\mathtt{tag}' \leftarrow$ Auth
for $j = 0$ **to** $l-1$ **do**
  $\mathtt{tag}' \leftarrow \mathtt{tag}' \oplus E_K(0000||j, M_{j+1})$
end
if $M_* \neq \epsilon$ **then**
  $\mathtt{tag}' \leftarrow \mathtt{tag} \oplus E_K(0100||l, pad10^*(M_*))$
end
$\mathtt{tag}' \leftarrow E_K(0001||0^4||N, \mathtt{tag}')$

/* *Tag verification* */
if $\mathtt{tag}' = \mathtt{tag}$ **then return** $(M_1||\ldots||M_l||M_*)$
else return $\perp$

2.4 The Tweakable Block Cipher Deoxys-BC

Deoxys-BC is an ad-hoc tweakable block cipher so that besides the two standard inputs, a plaintext P (or a ciphertext C) and a key K, it takes an additional input called a tweak T. The cipher $E_K(T, P)$ has 128-bit state and variable size key and tweak. The encryption and decryption are defined in a standard way for tweakable ciphers, i.e. $E_K(T, P) = C$ and $E_K^{-1}(T, C) = P$. We define two ciphers, Deoxys-BC-256 for which the cumulative size of the key and the tweak is 256 bits (and is utilized for Deoxys-I-128-128 and Deoxys-II-128-128), and Deoxys-BC-384 for which the cumulative size of the key and the tweak is 384 bits (and is utilized for Deoxys-I-256-128 and Deoxys-II-256-128).

Deoxys-BC is an AES-like design, i.e. it is an iterative substitution-permutation network (SPN) that transforms the initial plaintext through series of round functions (that depend on the key and the tweak) to a ciphertext. As most AES-like designs, the state of Deoxys-BC is seen as 4×4 matrix of bytes (we denote c the size of a cell, i.e. $c = 8$). We denote $\mathbb{K}$ the base field as $GF(2^8)$ defined by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

The number r of rounds is 14 for Deoxys-BC-256 and 16 for Deoxys-BC-384. One round, similarly to a round in AES, has the following four transformations applied to the internal state in the order specified below:

- AddRoundTweakey – XOR the 128-bit round subtweakey (defined further) to the internal state,
- SubBytes – Apply the 8-bit Sbox AES $\mathcal{S}$ to the 16 bytes of the internal state (see definition in Appendix A.1),
- ShiftRows – Rotate the 4-byte i-th row left by $\rho[i]$ positions, where $\rho = (0, 1, 2, 3)$.
- MixBytes – Multiply the internal state by the 4×4 constant MDS matrix $\mathbf{M}$ defined below whose coefficients lie in $\mathbb{K}$.

After the last round, a final AddRoundTweakey operation is performed to produce the ciphertext.

The MDS matrix $\mathbf{M}$ we use is the one from the AES (coefficients lie in $\mathbb{K}$):

$$\mathbf{M} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}.$$

The round function f^{-1} for a decryption round, naturally, is similar as for the encryption, and the inverse of the four round permutations are applied in a reversed order. We also note that the subtweakeys are used in reverse order. Namely, we perform r times the following operations:

- InvAddRoundTweakey – XOR the 128-bit round subtweakey to the internal state,
- invMixBytes – Multiply the internal state by the 4×4 MDS matrix $\mathbf{M}^{-1}$ (coefficients in $\mathbb{K}$),
- InvShiftRows – Rotate the 4-byte i-th row *right* by $\rho[i]$ positions, where $\rho = (0, 1, 2, 3)$,
- InvSubBytes – Apply the inverse 8-bit Sbox S^{-1} to the 16 bytes of the internal state (see Appendix A.1 for actual values).

Finally, a final InvAddRoundTweakey operation is performed to produced the plaintext value. For

the sake of completeness, we provide the inverse of the $\mathbf{M}$ matrix (coefficients are in $\mathbb{K}$):

$$\mathbf{M}^{-1} = \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}.$$

Definition of the Subtweakeys. So far, the description of the cipher has followed the classical construction of an `AES`-like block cipher. The operation AddRoundTweakey, and in particular the production of the subtweakeys, is where `Deoxys-BC` differs from the `AES`.

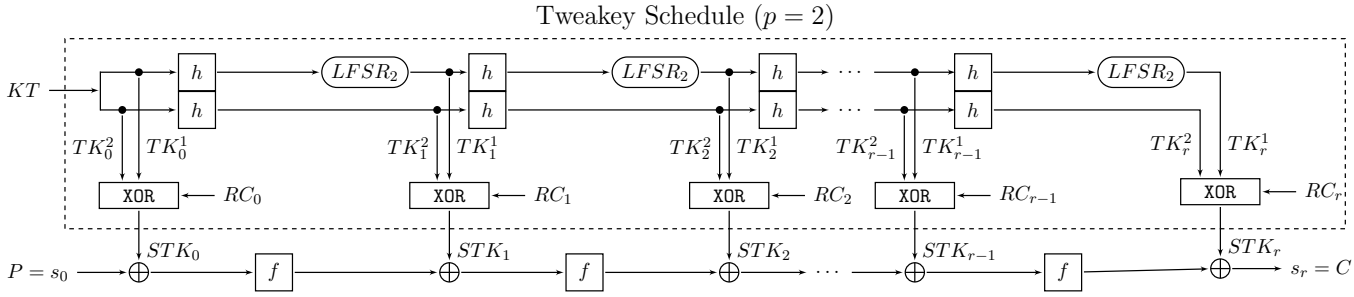


Figure 2.7: Instantiation of the `TWEAKEY` framework for `Deoxys-BC`.

We denote the concatenation of the key K and the tweak T as KT, i.e. $KT = K||T$. Then, the tweakey state is divided into words of 128 bits. More precisely, in `Deoxys-BC-256`, the size of KT is 256 bits with the first (most significant) 128 bits of KT being denoted W_1, while the second W_2. For `Deoxys-BC-384`, the size of KT is 384 bits, with the first (most significant) 128 bits of KT being denoted W_1, the second W_2 and the third W_3. Finally, we denote with STK_i the subtweakey (a 128-bit word) that is added to the state at Round i of the cipher during the AddRoundTweakey operation. For `Deoxys-BC-256`, a subtweakey is defined as:

$$STK_i = TK_i^1 \oplus TK_i^2 \oplus RC_i,$$

whereas for the case of `Deoxys-BC-384` it is defined as:

$$STK_i = TK_i^1 \oplus TK_i^2 \oplus TK_i^3 \oplus RC_i.$$

The 128-bit words TK_i^1, TK_i^2, TK_i^3 are outputs produced by a special tweakey schedule algorithm, initialized with $TK_0^1 = W_1$ and $TK_0^2 = W_2$ for `Deoxys-BC-256` and with $TK_0^1 = W_1$, $TK_0^2 = W_2$ and $TK_0^3 = W_3$ for `Deoxys-BC-384`. The tweakey schedule algorithm is defined as

$$\begin{aligned} TK_{i+1}^1 &= h(TK_i^1), \\ TK_{i+1}^2 &= h(LFSR_2(TK_i^2)), \\ TK_{i+1}^3 &= h(LFSR_3(TK_i^3)), \end{aligned}$$

where the byte permutation h is defined as:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 1 & 6 & 11 & 12 & 5 & 10 & 15 & 0 & 9 & 14 & 3 & 4 & 13 & 2 & 7 & 8 \end{pmatrix},$$

and where we number the 16 bytes of a 128-bit tweakey word by the usual ordering:

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix}.$$

The $LFSR_2$ and $LFSR_3$ functions are simply the application of an LFSR to each of the 16 bytes of a tweakey 128-bit word. More precisely, the two LFSRs used are given in Table 2.3 (x_0 stands for the LSB of the cell).

Table 2.3: The two LFSRs used in `Deoxys-BC` tweakey schedule.

$LFSR_2$	$(x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0\|\|x_7 \oplus x_5)$
$LFSR_3$	$(x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_0 \oplus x_6\|\|x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1)$

Finally, RC_i are the key schedule round constants, and are defined as:

$$RC_i = \begin{pmatrix} 1 & \texttt{RCON}[i] & 0 & 0 \\ 2 & \texttt{RCON}[i] & 0 & 0 \\ 4 & \texttt{RCON}[i] & 0 & 0 \\ 8 & \texttt{RCON}[i] & 0 & 0 \end{pmatrix}$$

where `RCON`$[i]$ denotes the i-th key schedule constants of the `AES`. We report their actual values in Appendix A.2.

Cipher Instances Separation. We note that the tweak and key material are not made explicitly distinct in `Deoxys-BC`, and one might argue that since the tweakable block cipher is always the same whatever is the amount of key or tweak inputs, there are some obvious relations between these different cipher variants. Only considering the primitive `Deoxys-BC`, a simple distinction between the instances could be to encode the parameter sizes into the round constants. We however chose to not consider such related-cipher attacks [38], but instead leave this distinction at the discretion of the protocol or mode calling the primitive. Especially, for the two `Deoxys` operating modes, the separation between the tweakable block cipher instances is naturally and safely done since the tweak and key sizes are fixed and the placement of key and tweak material is fully determined.

Security of `Deoxys-BC`. We strongly encourage third-party analysis of the (possibly round-reduced) internal tweakable block cipher `Deoxys-BC` of our proposal, which appears to be of independent interest from the CAESAR competition. We emphasize that we claim security up to 2^{128} computations for `Deoxys-BC-256` and up to 2^{256} computations for `Deoxys-BC-384`, since `Deoxys-BC-256` and `Deoxys-BC-384` are used to build authenticated encryption algorithms with 128-bit and 256-bit key respectively.

Chapter 3

Security Claims

3.1 Claims

We provide our security claims for the different variants of Deoxys in Table 3.1. We recall that the variants are defined in part by the bit size k of key and the bit size t of the tweak in Section 2.2. We give below the security goals expressed in terms of k and the block size n.

One can see that we do claim full k-bit security for both Deoxys-I and Deoxys-II for a nonce-respecting user, in contrary to other modes like AES-GCM [30] or OCB3 [27], which only ensure birthday-bound security. In the nonce-misuse scenario, we claim a birthday-bound security concerning Deoxys-II.

	Security (bits)	
Goal (nonce-respecting user)	Deoxys-I	Deoxys-II
Key recovery	k	k
Confidentiality for the plaintext	n	$n-1$
Integrity for the plaintext	n	$n-1$
Integrity for the associated data	n	$n-1$
Integrity for the public message number	n	$n-1$

	Security (bits)	
Goal (nonce-misuse user)	Deoxys-I	Deoxys-II
Key recovery	k	k
Confidentiality for the plaintext	none	$n/2$
Integrity for the plaintext	none	$n/2$
Integrity for the associated data	none	$n/2$
Integrity for the public message number	none	$n/2$

Table 3.1: Security goals of Deoxys. The upper table stands for the situation where the user will never repeat the same value N for the same key (nonce-respecting user). The lower table stands for the situation where such repetitions in N for the same key are allowed (nonce-misuse user). The bit security of our designs is expressed in terms of calls to the internal tweakable block cipher, up to a small logarithmic factor.

In the table, we assume that the public message number is a nonce and there is no secret message

number. We also assume that for the nonce-respecting mode, the total size of the associated data and the total size of the message do not exceed $16 \cdot 2^{max_l}$ bytes, thus 2^{64} bytes for all variants of `Deoxys-I`. For the nonce-misuse resistant mode, the total size of the associated data and the total size of the message do not exceed $16 \cdot 2^{max_l} \cdot 2^{max_m}$ bytes, thus 2^{128} bytes for all variants of `Deoxys-II`. Moreover, the maximum number of messages that can be handled for a same key is 2^{max_m}, that is 2^{64} for all variants of `Deoxys`.

We recommend to use a tag size $\tau = n$. However, in case a smaller tag size is required, the security claims will drop according to τ. We explicitly exclude related-cipher attacks, for example when an attacker would try to find some correlations between different versions of `Deoxys` (we assume that such a separation, if needed, will be handled by the protocol using the authenticated encryption primitive).

3.2 Comparison of `Deoxys` Modes With Other Modes

`Deoxys-I` provides "full" security in the nonce-respecting setting; more precisely, confidentiality is perfectly guaranteed and the forgery probability is 2^{-t}, independently of the number of blocks of data in encryption/decryption queries made by the adversary. In comparison, `OCB` only provides security up to the birthday bound, more precisely up to roughly $2^{n/2}$ *blocks of data* since it relies on `XE/XEX` (a construction of a tweakable block cipher from a standard block cipher with security only up to the birthday bound). In Figure 3.1, we represent the security of several nonce-respecting modes present in the third round of the CAESAR competition.

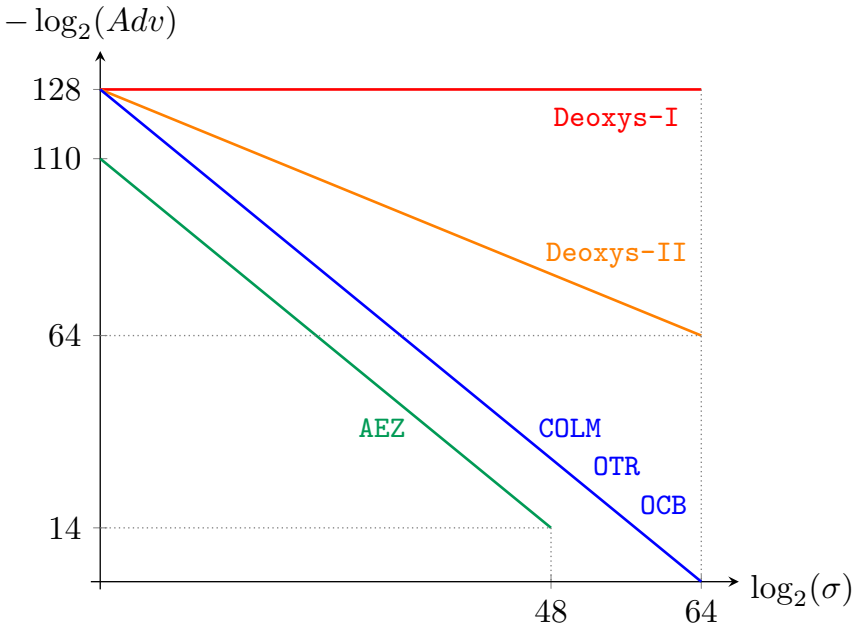


Figure 3.1: Security comparisons of some nonce-respecting modes from the third round of the CAESAR competition. The figure plots the advantage of an adversary as a function of the total number of queried blocks σ.

`Deoxys-II` provides "full" security in the nonce-misuse setting, in the MRAE sense of [35] (but its bond depends on the number of queries made by the adversary). Moreover, security degrades gracefully (both for confidentiality and authenticity) with respect to the maximal number of nonce repetitions (i.e., repeating nonces only a few times does not affect the security bound much, and the same nonce must be repeated close to $2^{n/2}$ times for an adversary to have noticeable advantage). In comparison, `COLM` only provides the weaker notion of *online* nonce-misuse resistance [17], while

`AEZ` provides nonce-misuse resistance (and even the stronger notion of robustness, which guarantees security against release of unverified plaintext) but only up to the birthday bound.

To give an numerical example, with 2^{40} blocks ciphered (about 16 TeraBytes), one gets an advantage of about 2^{-48} to generate a valid tag for most operating modes in the nonce-respecting scenario (actually 2^{-30} in the case of `AEZ`). For the same amount of data, the advantage becomes 2^{-88} for `Deoxys-II`, and remains 2^{-128} for `Deoxys-I`.

Chapter 4

Security Analysis

In the past two decades, the Advanced Encryption Standard (AES) and AES-type ciphers have been the subject of extensive analysis. As a result, the security of these ciphers against the most popular forms of cryptanalysis, the differential and the linear attacks, is well understood in the single-key model. Important progress in AES security analysis has been provided in the past several years, and it involved careful study of the key schedule of AES-type ciphers. In other words, the latest attacks rely on how the key is processed in the rounds of the ciphers. Two such notable examples are the related-key differential attacks [3, 4] and the Meet-in-the-Middle (MITM) attacks [11, 13, 15, 28] on AES.

Our TWEAKEY framework allows a dual view of the whole constructions. The first is as described previously, i.e. in each round a subkey and a subtweak are added to the state. In the second view, however, one can treat the XOR of the subkey and the subtweak as one single subkey called *subtweakey*, which is produced from a more complex key schedule (composition of the original key schedule and tweak schedule). This way the security analysis of TWEAKEY reduces to the security analysis of a block cipher with more complex key schedule, and where one part of the key is secret, and the second is public.

We view the Deoxys tweakey schedule as an improvement over the AES key schedule: not only it is much simpler and more efficient, but it also provides stronger security guarantees against related-key differential cryptanalysis for example.

4.1 Differential Cryptanalysis

Designing a SPN cipher resistant against single-key differential attacks is fairly simple and can be done by carefully choosing the diffusion layer (ensuring that its branch number is high enough). For example, in the case of AES, because its diffusion matrix has a branching number of 5, one can prove at least 25 differentially active Sboxes for four rounds of AES in the single-key model. Assuming that each of the active Sboxes can reach the maximal differential probability of the AES Sbox $p_{max} = 2^{-6}$, one directly deduce that four rounds already provide sufficient protection against simple differential cryptanalysis in the single-key model.

Since our tweakable block cipher Deoxys-BC is directly built on the AES round function and since the number of active Sboxes is independent of the key schedule in the single-key model, the very same analysis can be applied to Deoxys-BC. Thus, four rounds of Deoxys-BC already provide sufficient protection against simple differential cryptanalysis in the single-key model.

While the single-key case is straightforward, it is much harder to guarantee resistance against related-key differential attacks and the STK construction is an elegant way to tackle this problem.

There exists search algorithms and tools [5, 6, 16, 18, 31, 36] that given a key schedule can return the upper bound on the probability of the best related-key differential characteristics, and in the case when such a bound is low, practically provide and prove the resistance against related-key differential attacks. The STK construction greatly facilitates the applicability of these tools and we used precisely these algorithms in our security analysis against related-key attacks.

These search tools have been designed to look for related-key characteristics, however, we allow the adversary to operate in a stronger setting of related-key and possibly related-tweak (or both at the same time) attacks. Nonetheless, we can accommodate and modify the tools to search for such characteristics. Although the modification can be done easily, the feasibility (expressed in the time complexity required the search algorithm to finish) is the real problem. To cope with this, we use several different tools, each chosen to provide the probability bounds in the shortest time. More precisely, we alternate between the search algorithm based on Matsui's approach [5], split approach [6], and extended split approach [16]. We omit the details on how these search algorithms operate due to their complexity, and further, give only the final results produced by the tools.

Rounds	**Active Sboxes**	**Upper bound on probability**	**Method used**
1	0	2^{0}	Trivial
2	0	2^{0}	Trivial
3	1	2^{-6}	Matsui's
4	5	2^{-30}	Matsui's
5	9	2^{-54}	Matsui's
6	12	2^{-72}	Matsui's
8	≥ 17	2^{-108}	extended split (4R+4R)
10	≥ 22	2^{-132}	extended split (5R+5R)

Table 4.1: Deoxys-BC-256: Upper bounds on the probability of the best round-reduced related-key related-tweak differential characteristics for Deoxys-BC-256.

Rounds	**Active Sboxes**	**Upper bound on probability**	**Method used**
1	0	2^{0}	Trivial
2	0	2^{0}	Trivial
3	0	2^{0}	Trivial
4	1	2^{-6}	Matsui's
5	4	2^{-24}	Matsui's
6	8	2^{-48}	Matsui's
12	≥ 22	2^{-132}	extended split (6R+6R)

Table 4.2: Deoxys-BC-384: Upper bounds on the probability of the best round-reduced related-key related-tweak differential characteristics for Deoxys-BC-384

In Table 4.1, we give the results for our tweakable block cipher Deoxys-BC where the sum of the key and tweak sizes is 256 bits. Again assuming that each of the active Sboxes can reach the maximal differential probability of the AES Sbox $p_{max} = 2^{-6}$, we get the upper bound given in the third column. Using at least $r = 10$ rounds for this cipher, the number of active Sboxes is lower-bounded by 22, meaning that the probability of the associated differential characteristic is

upper-bounded by $2^{-6\times 22} = 2^{-132}$. Thus, such characteristics cannot be exploited due to the state size of 128 bits (the attacker cannot construct more than 2^{128} plaintext pairs to start the attack).

We perform the same analysis when the sum of the key and tweak sizes is 384 bits and the results from Table 4.2 show that we reach more than $\lceil \frac{128}{6} \rceil = 22$ Sboxes after 12 rounds.

Thus, all versions of `Deoxys-BC` have a security margin of at least four rounds and thus highly resistant against related-key related-tweak attacks.

4.2 Linear Cryptanalysis

Similarly to the differential cryptanalysis case, the security guarantees of `AES` with regards to linear cryptanalysis in the single-key model directly translate to `Deoxys-BC`. Therefore, we can easily prove that four rounds of `Deoxys-BC` already provide sufficient protection against simple linear cryptanalysis in the single-key model.

Moreover, since there is no cancellation of linearly active Sboxes in the related-key model [26], the analysis of single-key directly translates to the related-key model as well. Therefore, four rounds of `Deoxys-BC` already provide sufficient protection against simple linear cryptanalysis also in the related-key model.

One might argue that the linearity of the `Deoxys-BC` tweakey schedule will help linear cryptanalysis. However, we emphasize that most analysis of `AES` with regards to linear cryptanalysis have been done assuming that the subkeys are independent. In addition, unlike for `AES`, the tweakey schedule of `Deoxys-BC` has been chosen to maximize the number of active Sboxes and the key bytes diffusion. This will very likely render the cryptanalysis of `Deoxys-BC` even harder for an attacker.

4.3 Meet-in-the-Middle Attacks

Additionally, we scrutinize the resistance of our design in regard to the recent advanced meet-in-the-middle attack on `AES` conducted in [13]. Indeed, this attack strongly relies on the `AES` key schedule to propagate linear equations in the MITM strategy to spare some guesses in both the offline and online phases. As the design we propose introduces a new tweakey schedule, we have analyzed how it interacts with the `AES` round function.

For a given tweak value, `Deoxys-BC` behaves as the `AES` with a new schedule with partially known values (the subtweakeys) XORed between each round, without additional input values. This tweakey schedule is fully linear as it first applies a byte permutation and then an LFSR on some bytes of the state. In that context, a first analysis shows that the meet-in-the-middle technique from [13] can attack up to 8 rounds, where the `AES` key schedule for 128-bit keys stops the attack at 7 rounds.

4.4 Security Against Other Attacks

As mentioned earlier in the chapter, the security bound of `Deoxys-BC` against most of the other attacks matches the bounds of `AES`, i.e. all the attacks that do not exploit the key schedule will have the same success on `Deoxys-BC` as on `AES`. This gives us a security reduction from `AES`, however, we note that as `Deoxys-BC` has more rounds, for these particular attacks the security margin of `Deoxys-BC` is higher.

Besides the above types of attacks, we encourage to investigate attack vectors that rely on some additional property of the AES key schedule of Deoxys-BC, for instance impossible differential attacks. We emphasize that other attack techniques like slide [7], rotational [24] and the internal differential attacks [32] are prevented by the usage of the constants in the key schedule, as done in the AES.

As previously described, since by design there is no distinction between key and tweak material for Deoxys-BC (rather the key+tweak inputs are treated as one tweakey input), trivial so-called related-cipher attacks [38] would apply to two different versions of the Deoxys-BC. As the practical threat coming from this type of attack framework is very limited and can be avoided at the operating mode level, we decided not to put different constants RC_i in order to prevent the attack. Moreover, we recall that for the two Deoxys operating modes, the separation between the tweakable block cipher instances is naturally and safely done since the tweak and key sizes are fixed and the placement of key and tweak material is fully determined.

Finally, we note that a possible increment in the number of attacked rounds might happen in the scenario of open-key distinguishers (even though we have not been able to improve the known attacks [12, 20, 22] using this extra tweak input). However, we emphasize that we do not claim any resistance of Deoxys-BC in this attack model.

4.5 Comparing Deoxys-BC with AES

The Deoxys-BC tweakable block cipher is directly built upon the AES round function, but its key schedule has been updated. We see this new key schedule as a direct improvement over the AES key schedule. We recall that AES-192 and AES-256 have been shown to be weak against related-key attacks [3, 4], thus indicating that their respecting key schedule is not strong enough against certain type of attacks.

Table 4.3: Proved bounds on the minimal number of differential active Sboxes for AES-256 and for Deoxys-BC-256. Model **SK** denotes the single-key scenario and model **RTK** denotes the related-tweakey scenario where differences can be inserted in the tweakey state.

Cipher	Model	Rounds							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Deoxys-BC-256	**SK**	1	5	9	25	26	30	34	50
(14 rounds)	**RTK**	0	0	1	5	9	12	≥ 17	≥ 22
AES-256	**SK**	1	5	9	25	26	30	34	50
(14 rounds)	**RTK**	0	0	1	3	5	5	5	10

We compare in Table 4.3 the minimum number of differentially active Sboxes of AES-256 and Deoxys-BC-256 in the related-key model (we compare these two primitives as they have the same tweakey size). One can see that Deoxys-BC-256 tweakey schedule guarantees much more active Sboxes than AES-256 (and thus an expected higher resistance against related-key attacks), while being much more efficient. Note that the bounds for Deoxys-BC-256 are not tight and can probably be improved, for instance using MILP modeling.

We finally emphasize that Deoxys-BC-256 is used in the 128-bit key mode of Deoxys, and thus attacks requiring more than 2^{128} computations are not a concern (in contrary to AES-256). This further increases the security margin Deoxys-BC-256 provides when compared to AES-256.

Chapter 5

Features

The starting point of our design is to provide a sound ad-hoc tweakable block cipher based on AES. Indeed, the main idea heavily exploited in the design of Deoxys is the introduction of an efficient tweakable block cipher Deoxys-BC, belonging to the family of the well-known AES-based primitives. Speed benchmarks show that Deoxys achieves almost the same speed as OCB while offering much higher security than OCB.

Deoxys-BC is a secure instantiation of the more general TWEAKEY framework [23] and does not rely on big field (e.g., $GF(2^{128})$) multiplications as previous tweakable ciphers proposals. Structurally, Deoxys-BC can be seen as a standalone primitive, whereas previous attempts at building tweakable block ciphers use a given block cipher as a black box and use it in a particular mode. The design is in particular very efficient on latest Intel processors, where we can reach much less than a cycle per byte for scenario with nonce-respecting users.

We detail more precisely below the main features of Deoxys.

- **Security margin.** Deoxys has a good security margin for all the recommended parameters. We measure the security margin in terms of number of rounds: the small variant of Deoxys counts 14 rounds and the large variant 16 rounds. The best known attacks on AES-based design in secret-key models for similar size of keys reach 7 to 9 rounds. For a stronger adversarial model in the related-key scenario, the two large variants of the AES are theoretically broken by known results from [3,4]. With 14 rounds or more, the two versions of Deoxys offer a confident security margin regarding this class of attacks. Interestingly, Deoxys-BC-256-128 is very similar to AES-256 for a fixed tweak value, but we have shown that a good key schedule can significantly reduce the number of rounds required for a secure cipher: Deoxys-BC-256-128 only needs 12 rounds to be secure to a related-key attack, whereas AES-256 on 14 rounds is already subject to a theoretical related-key distinguisher.
- **Security proofs.** The security arguments of Deoxys are directly inherited from the two modes used in our design. Indeed, for nonce-respecting users, Deoxys benefits from the proofs of the OCB3 mode, while for nonce-repeating users, we rely on the provably secure authenticated encryption mode SCT [33].
- **Software implementations.** Deoxys achieves good performances for software implementations. As most of the AES-based designs, it hugely benefits from the new AES-NI instruction set added in the latest processors. In addition, as we use fully parallelizable modes, the cycles per byte count drops significantly. The current speed of our nonce-respecting design is faster than AES-GCM on Intel Sandy Bridge (although AES-GCM ensures only birthday-bound security).
- **Small messages.** Deoxys is efficient for small messages, which is particularly important in many lightweight applications where messages sent are usually composed of a few dozens

of bytes. This is common disadvantage of sponge-based or stream cipher based lightweight designs like FIDES [2] (broken in [14]) or ALE [8] (broken in [25]), which will usually require a costly initialization. This is also important for Internet traffic as packets sizes can be rather small. Deoxys is efficient for small messages mainly because it is based on a tweakable block cipher: it allows to avoid any precomputation (like in OCB, AES-GCM, etc.).

- **Theoretical performances.** The number of calls to the internal primitive is minimized. For Deoxys-I, the first 128-bit message block is handled directly, and taking in account the tag generation, one needs only $m + 1$ internal cipher calls to process messages of m blocks of n bits each, which is optimal. For Deoxys-II, taking in account the tag generation, one needs only $2m + 1$ internal cipher calls to process messages of m blocks of n bits each (which is close to be optimal, as at least $2m$ calls are required to obtain MRAE security), which is smaller than existing MRAE-secure designs.
- **Well-understood design.** Deoxys also benefits from the vast research literature on the cryptanalysis of the AES. Indeed, being an AES-based primitive, the tweakable block cipher Deoxys-BC is subject to the same class of attacks than AES, which consists of an active research line since 15 years. It is highly probable that any significant attack on either one would yield a similar cryptanalytic improvement on the other.
- **Simplicity.** Deoxys is simple for both the construction of the internal tweakable block cipher and for the authentication mode. It uses well-studied building blocks and is arguably easy to analyze. The implementation of Deoxys is also easy, and can reuse the design strategies, implementations and optimizations available for the AES. Moreover, our two operating modes are very similar (for example, the way the associated data is handled is exactly the same), thus further simplifying the analysis and the use of Deoxys.
- **Flexibility.** Deoxys has smooth parameters handling. We define some recommended parameter sets in this document, but any user can pick its own variant of the inner tweakable block cipher Deoxys-BC by adapting the key and tweak sizes at his/her convenience. This flexibility comes from the unified vision of the key and tweak material brought by the TWEAKEY framework. It means that one implementation of the cipher is sufficient to support all versions with different key and tweak sizes (with the same cumulative size). This feature extends to the whole Deoxys design.
- **Side channels.** Deoxys can resists to side-channel attacks with the same techniques as AES. Literature in this area available for the AES can be very easily adapted to the case of any AES-based designs, including Deoxys.
- **Beyond-birthday-bound security.** The nonce-misuse resistant mode Deoxys-II provides graceful (more precisely, linear) degradation of security with the maximal number of nonce repetitions. This means that security does not collapse to the birthday bound when nonces are repeated just a few times. In other words, any attack with birthday complexity against Deoxys-II must repeat the same nonce roughly $2^{n/2}$ times.

Chapter 6

Design Rationale

The starting point of our design is to provide a sound ad-hoc AES-based tweakable block cipher that has full security. Having such a primitive is beneficial for many authenticated encryption modes that are secure beyond the birthday bound, but loose this feature when instantiated with the current constructions that use a permutation or a cipher as a black box and surround it with addition of words produced by a finite field multiplications (beyond-birthday security authenticated encryption modes that use a block cipher remain quite slow).

Therefore, designing a secure tweakable block cipher would enable us to reach full 128-bit security for both confidentiality and authenticity. This direction of research was not explored yet as it was believed that ad-hoc tweaking of AES-like ciphers is not an easy task from points of view of both security and efficiency (adding some extra freedom to the attacker seems to enable more powerful attacks and thus implies many more rounds). As our design is lightweight, we are also careful in choosing the internal permutations of the cipher and the mode that provides the authenticated encryption.

The designer/designers have not hidden any weaknesses in this cipher.

6.1 Details for the STK Construction

Designing a secure round function for block ciphers has become a fairly easy task: an Sbox layer and a diffusion layer based on MDS code immediately provide good security margin against differential and linear attacks even when the number of rounds in the cipher is small. The problem when designing ciphers, however, lies in how to choose the key schedule – for the cipher to be secure the number of rounds has often to be very large. The complexity of this task increases manifold if the key size is larger and if the key schedule is supposed to be simple (no non-linear operations, and as few linear operations as possible).

We provide a solution to tackle the above two main points in the form of the STK construction. This construction gives a simple key schedule for arbitrary length keys and with an additional checks on related-key attacks, ensures that the cipher is secure. The number of total rounds in the cipher is kept fairly small because of a special trick we use in the key schedule. We split the master key on equal key sizes, each with its own (but similar to the other) simple schedule that produces subkeys that are added simultaneously to the state. Due to the similarity of the schedules, and the use of simple linear layers to differentiate them, we can control the number of difference cancellations happening in the subkeys in a related-key attack. Thus, the security against these type of attacks can be proven with a number of rounds that is not necessarily very high.

In detail, we denote with TK-p the cipher where the master key is p times larger than the state

(and thus is divided into p keys). In Deoxys, we work only with TK-2, and TK-3, but the same strategy would work when $p > 3$. Let us choose an arbitrary position of a byte at the beginning of each of the p key schedules. For instance, we fix the position $(1, 1)$ and we investigate how the p bytes at this position at the beginning of the p key schedules, change during the production of the subtweakeys. What is interesting is that as all key schedules apply the same permutation h, the initial p bytes will always be XORed at the same position in the subtweakeys (taking into account the definition of the permutation h we can see that the positions through the rounds change as: (1,1) in the first, (2,1) in the second, (3,2) in the third, (4,4) in the fourth, etc). From the initial p-tuple of byte values $\mathbf{x} = [x_1^0, \ldots, x_p^0]$, the STK key schedule (which can be seen as p similar key schedules that differ only in the linear layers used to update them) produces r tuples $[x_1^1, \ldots, x_p^1], \ldots, [x_1^r, \ldots, x_p^r]$, such that $x_i^{k+1} = L_j(x_i^k)$, where L_i represents the linear layer that updates the i-th word of the schedule. All of them are integrated to the internal state by considering the $r + 1$ XOR values $\bigoplus_{i=1}^{p} x_i^k$, for $0 \leq k \leq r$.

The goal was then to choose the linear layers L_i such that the number of difference cancellations in the sequence $\bigoplus_{i=1}^{p} x_i^k$ is minimized. By choosing the simple LFSRs given in Section 2 (and already used in the lightweight block cipher SKINNY [1]), we have checked with a computer program that cancellation of values (and differences in general as the key schedule is linear) in a chosen byte of TK-p cannot occur more than $p - 1$ times over 15 consecutive subkeys. For TK-2, this means that the cumulative difference coming from the p tweakey words can be canceled only once by XOR for 15 consecutive subkeys. For TK-3, this cancellation event can happen twice for 15 consecutive subkeys. We note that for the case of Deoxys-BC-384, the number of rounds for which we can control the difference cancellations (15) is slightly smaller than the total number of rounds of the cipher (16). However, this has no impact since the security proofs we will aim only apply to a rather small number of rounds $r' < 16$.

The above strategy of designing the key schedule is only the first step that ensures the schedule is not trivially insecure against related-key attacks (and that does not require a huge number of rounds to make the cipher secure). The step that follows is the choice of the bytes position permutation h. It was done after trying several of them. We settled down on the one that provides security against related-key attacks in the least number of rounds (we inspected the security with the tools specified in Section 4).

6.2 From Block Cipher to Tweakable Block Cipher

The STK construction (with specified permutation h and linear layers L_i) provides only a secure block cipher with an arbitrary length key. However, turning this block cipher into a tweakable block cipher is trivial: some bits of the master key are announced as tweak, while the remaining bits are kept as secret key bits. As the key and the tweak are treated in the same way in our designs, we give them the general name *tweakey*. From the TK-2 block cipher that in our case has 256-bit key and 128-bit block, we were able to obtain tweakable block ciphers with 128-bit key and 128-bit tweak (called Deoxys-BC-128-128). A similar transition was made from the TK-3 block cipher (with 384-bit tweakey) to Deoxys-BC-256-128.

During this transition, it is important to note that the security of the cipher against related-key (and now related-tweak) attacks does not drop, even though parts of the original master key become available to the attacker. The reason for this is twofold: first, the key schedule is linear, it never has any active Sboxes; and second, the XOR of all subkeys/subtweaks in each round to the state is secret (as long as one of them is secret), and also the state is secret (thus the attacker cannot reduce the number of active Sboxes by controlling the tweak).

Chapter 7

Implementations

7.1 Software Performances

As Deoxys is based on the AES, it allows very efficient software implementations on the processors that support AES-NI. In addition, the modes allow complete parallelization of the AES-NI calls. The actual overhead compared to AES mostly comes from the increased number of rounds and slightly from the tweak schedule. The key schedule plays role only for very short messages, but even then, it is quite efficient and much faster than the key schedule of AES. Note that the key schedule uses lightweight LFSR to update the subkey bytes, but not the tweak schedule. This was made on purpose, as the subkeys are computed once, while the subtweaks change in each call to the block cipher. For a fixed key, the overhead of the tweak schedule is one XOR and one permutation of 16 bytes, and arguably it is one of the most efficient tweak schedules in the framework of TWEAKEY.

We used the three Intel processor families Intel Sandy Bridge, Intel Haswell and Intel Skylake with AES-NI enabled to obtain the benchmarks. The code was compiled on Linux with gcc v4.8.1. The reported speed was taken as an average over multiple execution of the code with the same fixed message length. We did take into account the key schedule as well as the loading the bytes from the memory and storing them back to memory.

Our benchmarks for encryption and authentication (simultaneously) using Deoxys-I-128-128 are reported in Table 7.1. For complete and consistent benchmark comparisons with other authenticated encryption schemes, we refer the interested reader to SUPERCOP available online.[1]

Platform	128B	256B	512B	1024B	2048B	4096B	8192B	65536B
Intel Sandy Bridge	6.57	3.93	2.60	1.96	1.61	1.45	1.37	1.29
Intel Haswell	4.74	2.85	1.90	1.43	1.18	1.07	1.01	0.96
Intel Skylake	4.01	2.51	1.67	1.29	1.09	0.98	0.93	0.89

Table 7.1: Benchmarks for Deoxys-I-128-128 expressed in cycles per byte on AES-NI enabled platforms (disable Turbo Boost) for increasing numbers of processed bytes.

We can see that our nonce-respecting mode Deoxys-I performs well, and is faster than AES-GCM on Intel Sandy Bridge. The second mode Deoxys-II is expected to be around twice slower as it requires about twice more calls to the internal cipher. As expected, the cycle per byte count in the first mode stabilizes after the message length reaches 256-512 bytes or more. This is due to the fact that no preprocessing step is required to start the cipher calls.

[1] https://bench.cr.yp.to/ebaead.html

Although the above benchmark rely on the AES-NI instruction set, the simplicity of the tweak schedule guarantees that the speed ratio compared to AES will remain the same even if we used a simple table look-up implementation of AES. In fact, the overhead of the tweak schedule in this case compared to AES, will be very small, and the speed of Deoxys-BC will be very close to the speed of AES.

7.2 Hardware Performances

7.2.1 ASIC implementations

We report preliminary ASIC implementations from Axel Poschmann and Marc Stöttinger [34]. *Xlinx ISE DesignSuite 13.3* and *Synopsys DesignCompiler E-2010.12-SP2* were used for functional simulation and synthesis of the designs to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3* [37], which is based on the *UMC L180 0.18µm 1P6M* logic process with a typical voltage of 1.8 V. For synthesis and for power estimation the compiler was advised to keep the hierarchy and use a clock frequency of 100 KHz.

Different variants of Deoxys in VHDL have been implemented and their post-synthesis performance simulated. Two architectures were designed: one is fully serialized, i.e. performing operations on one cell per clock cycle, and aims for the smallest area possible; the second one is a round-based implementation, thus performing one round in one clock cycle, resulting in a significant speed-up. Only the encryption and authentication parts have been implemented (no decryption capability).

The following tables give hardware performance results independently for the internal tweakable block ciphers and when they are plugged in the modes. Table 7.2 gives hardware performances of Deoxys-BC-256 and Deoxys-BC-384, while Table 7.3 considers the higher level where the primitives are plugged into the two modes I and II previous described depending on some test cases (Table 7.4).

Table 7.2: Overview of the ASIC performances of the underlying tweakable block ciphers.

Internal Primitive	Architecture	Clk	Area [GE]
Deoxys-BC-256	Round	14	8,005
	Serial	338	2,860
Deoxys-BC-384	Round	16	9,362
	Serial	384	3,575

7.2.2 FPGA implementations

We report round-based FPGA implementations from the GMU research team [19], which are available on the ATHENA website.[2] These implementations are CAESAR Hardware API-compliant implementations (thus with both encryption and decryption capabilities). On Virtex 6, Deoxys-I-128-128 requires 3250 LUTs for a throughput of 2816 Mbit/s, while on Virtex 7, Deoxys-I requires 3283 LUTs for a throughput of 2844 Mbit/s. As expected, these results locates Deoxys in the same range as other AES-based candidates.

[2] https://cryptography.gmu.edu/athena

Table 7.3: Overview of the ASIC performances for the two modes `I` and `II` for the two tweakable block cipher versions.

Mode	TBC	Arch.	Clock Cycles for Test Case							Area [GE]
			I	II	III	IV	V	VI	VII	
I	Deoxys-BC-256	**Round**	**22**	**60**	**79**	**62**	**82**	**81**	**120**	**12,496**
		Round*	19	57	76	59	79	78	117	11,936
	Deoxys-BC-384	**Round**	**24**	**66**	**87**	**68**	**90**	**89**	**132**	**13,872**
		Round*	21	63	84	65	87	86	129	13,326
II	Deoxys-BC-256	**Round**	**58**	**96**	**115**	**137**	**176**	**156**	**214**	**12,744**
		Round*	55	93	112	134	173	153	211	12,068
	Deoxys-BC-384	**Round**	**64**	**106**	**127**	**151**	**194**	**172**	**236**	**14,107**
		Round*	61	103	124	148	191	169	233	13,422

* slightly modified API.

Table 7.4: Test cases and length (in bytes) of Associated Data, Message, Key, and Nonce.

Test Case	AD	Message	Key	Nonce
I	0	0	16	8
II	32	0	16	8
III	33	0	16	8
IV	0	32	16	8
V	0	33	16	8
VI	16	32	16	8
VII	17	33	16	8

Chapter 8

Intellectual Property

Deoxys is not patented and is free for use in any application. 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. We note that since Deoxys-I uses a mode that presents similarities with the generic ΘCB3 framework, it is unclear if patents relative to OCB (such as United States Patent No. 7,046,802; United States Patent No. 7,200,227; United States Patent No. 7,949,129; United States Patent No.8,321,675) apply to our proposal.

Chapter 9

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.

Acknowledgments

The authors would like to thank the anonymous reviewer of the CAESAR commitee for their helpful comments and suggestions. Moreover, we would like to individually thank Tetsu Iwata, Guo Jian, Gaëtan Leurent and Wang Lei for very fruitful discussions on authenticated encryption designs. Moreover, we are very grateful to Christof Beierle and Anne Canteaut for pointing to us issues in our first general formulations of the bound on the number of active Sboxes coming from the subtweakeys schedule in the `STK` construction. Finally, the authors would like to thank Axel Poschmann and Marc Stöttinger for their hardware implementations of `Deoxys` presented in Section 7.2. This work is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

Bibliography

[1] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Robshaw, M., Katz, J., eds.: Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Volume 9815 of Lecture Notes in Computer Science., Springer (2016) 123–153

[2] Bilgin, B., Bogdanov, A., Knežević, M., Mendel, F., Wang, Q.: Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In Bertoni, G., Coron, J.S., eds.: CHES 2013. Volume 8086 of LNCS., Springer (August 2013) 142–158

[3] Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: ASIACRYPT 2009. Volume 5912 of LNCS., Springer (December 2009) 1–18

[4] Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In Halevi, S., ed.: CRYPTO 2009. Volume 5677 of LNCS., Springer (August 2009) 231–249

[5] Biryukov, A., Nikolić, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In Gilbert, H., ed.: EUROCRYPT. Volume 6110 of Lecture Notes in Computer Science., Springer (2010) 322–344

[6] Biryukov, A., Nikolić, I.: Search for Related-Key Differential Characteristics in DES-Like Ciphers. In Joux, A., ed.: FSE. Volume 6733 of Lecture Notes in Computer Science., Springer (2011) 18–34

[7] Biryukov, A., Wagner, D.: Slide Attacks. In Knudsen, L.R., ed.: FSE'99. Volume 1636 of LNCS., Springer (March 1999) 245–259

[8] Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption. In Moriai, S., ed.: FSE 2013. Volume 8424 of LNCS., Springer (March 2014) 447–466

[9] Canteaut, A., ed.: FSE 2012. In Canteaut, A., ed.: FSE 2012. Volume 7549 of LNCS., Springer (March 2012)

[10] Cogliati, B., Seurin, Y.: An Efficient and Nonce-Misuse Resistant MAC Based on a Tweakable Block Cipher. Draft manuscript (2016)

[11] Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In Nyberg, K., ed.: FSE 2008. Volume 5086 of LNCS., Springer (February 2008) 116–126

[12] Derbez, P., Fouque, P.A., Jean, J.: Faster Chosen-Key Distinguishers on Reduced-Round AES. In Galbraith, S.D., Nandi, M., eds.: INDOCRYPT 2012. Volume 7668 of LNCS., Springer (December 2012) 225–243

[13] Derbez, P., Fouque, P.A., Jean, J.: Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In Johansson, T., Nguyen, P.Q., eds.: EUROCRYPT 2013. Volume 7881 of LNCS., Springer (May 2013) 371–387

[14] Dinur, I., Jean, J.: Cryptanalysis of FIDES. In Cid, C., Rechberger, C., eds.: FSE 2014. Volume 8540 of LNCS., Springer (March 2015) 224–240

[15] Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Abe, M., ed.: ASIACRYPT 2010. Volume 6477 of LNCS., Springer (December 2010) 158–176

[16] Emami, S., Ling, S., Nikolić, I., Pieprzyk, J., Wang, H.: The resistance of PRESENT-80 against related-key differential attacks. Cryptography and Communications (2013) 1–17

[17] Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. [9] 196–215

[18] Fouque, P.A., Jean, J., Peyrin, T.: Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In Canetti, R., Garay, J.A., eds.: CRYPTO 2013, Part I. Volume 8042 of LNCS., Springer (August 2013) 183–203

[19] Gaj, K., Kaps, J.P., Amirineni, V., Rogawski, M., Homsirikamol, E., Brewster, B.Y.: Athena-automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using fpgas. In: 2010 International Conference on Field Programmable Logic and Applications, IEEE (2010) 414–421

[20] Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. [21] 365–383

[21] Hong, S., Iwata, T., eds.: FSE 2010. In Hong, S., Iwata, T., eds.: FSE 2010. Volume 6147 of LNCS., Springer (February 2010)

[22] Jean, J., Naya-Plasencia, M., Peyrin, T.: Improved Rebound Attack on the Finalist Grøstl. [9] 110–126

[23] Jean, J., Nikolic, I., Peyrin, T.: Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Sarkar, P., Iwata, T., eds.: ASIACRYPT 2014, Part II. Volume 8874 of LNCS., Springer (December 2014) 274–288

[24] Khovratovich, D., Nikolic, I.: Rotational Cryptanalysis of ARX. [21] 333–346

[25] Khovratovich, D., Rechberger, C.: The LOCAL Attack: Cryptanalysis of the Authenticated Encryption Scheme ALE. In Lange, T., Lauter, K., Lisonek, P., eds.: SAC 2013. Volume 8282 of LNCS., Springer (August 2014) 174–184

[26] Kranz, T., Leander, G., Wiemer, F.: Linear Cryptanalysis: On Key Schedules and Tweakable Block Ciphers. Preprint (2016)

[27] Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In Joux, A., ed.: FSE 2011. Volume 6733 of LNCS., Springer (February 2011) 306–327

[28] Li, L., Jia, K., Wang, X.: Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE. Cryptology ePrint Archive, Report 2013/573 (2013)

[29] Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. Journal of Cryptology **24**(3) (July 2011) 588–613

[30] McGrew, D., Viega, J.: The Galois/Counter mode of operation (GCM). Submission to NIST. http://csrc. nist. gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec. pdf (2004)

[31] 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. Volume 7537 of Lecture Notes in Computer Science., Springer (2011) 57–76

[32] Peyrin, T.: Improved Differential Attacks for ECHO and Grøstl. In Rabin, T., ed.: CRYPTO 2010. Volume 6223 of LNCS., Springer (August 2010) 370–392

[33] Peyrin, T., Seurin, Y.: Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In Robshaw, M., Katz, J., eds.: Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Volume 9814 of Lecture Notes in Computer Science., Springer (2016) 33–63

[34] Poschmann, A., Stöttinger, M. Personal communication

[35] Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In Vaudenay, S., ed.: EUROCRYPT 2006. Volume 4004 of LNCS., Springer (May / June 2006) 373–390

[36] Sun, S., Hu, L., Wang, P.: Automatic Security Evaluation for Bit-oriented Block Ciphers in Related-key Model: Application to PRESENT-80, LBlock and Others. Cryptology ePrint Archive, Report 2013/676 (2013)

[37] Virtual Silicon Inc.: 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm (July 2004)

[38] Wu, H.: Related-Cipher Attacks. In Deng, R.H., Qing, S., Bao, F., Zhou, J., eds.: Information and Communications Security, 4th International Conference, ICICS 2002, Singapore, December 9-12, 2002, Proceedings. Volume 2513 of Lecture Notes in Computer Science., Springer (2002) 447–455

Appendix A

AES Sbox and constants

A.1 AES Sbox and its inverse

We define here the AES Sbox $\mathcal{S}$ and its inverse $\mathcal{S}^{-1}$, as an array where the value of $\mathcal{S}(x)$ can be found at the position x in the array.

	y0	y1	y2	y3	y4	y5	y6	y7	y8	y9	yA	yB	yC	yD	yE	yF
0x	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1x	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2x	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3x	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4x	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5x	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6x	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7x	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8x	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9x	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
Ax	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
Bx	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
Cx	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
Dx	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
Ex	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
Fx	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table A.1: The AES Sbox $\mathcal{S}$. The retrieve the value of $\mathcal{S}(x)$, convert x to its hexadecimal representation, and use its four leftmost bits x and four rightmost bits y as coordinates in the table. For example $\mathcal{S}(0x25) = 0x3F$.

	y0	y1	y2	y3	y4	y5	y6	y7	y8	y9	yA	yB	yC	yD	yE	yF
0x	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1x	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2x	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3x	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4x	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5x	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6x	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7x	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8x	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9x	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
Ax	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
Bx	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
Cx	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
Dx	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
Ex	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
Fx	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Table A.2: The AES inverse Sbox $\mathcal{S}^{-1}$. The retrieve the value of $\mathcal{S}(x)$, convert x to its hexadecimal representation, and use its four leftmost bits x and four rightmost bits y as coordinates in the table. For example $\mathcal{S}(0x3F) = 0x25$.

A.2 RCON constants

The Table A.3 below gives the values of constants RCON used in the tweakey scheduling algorithm of the Deoxys.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RCON[i]	2f	5e	bc	63	c6	97	35	6a	d4	b3	7d	fa	ef	c5	91	39	72

Table A.3: The RCON constants used in the key scheduling algorithm. The constants are written on lines from left to right, from top to bottom. For example, RCON[1] = 0x2f, RCON[2] = 0x5e, and RCON[11])=0xb3.

Appendix B

Changelog

B.1 Changelog from v1.3 to v1.41

We detail here the differences between v1.3 and v1.41 of this document.

1. Applied four minor tweaks on the `Deoxys` design:
   (a) Replaced in the tweakey schedule of `Deoxys-BC` the multiplications in GF(2^8) by simpler and cheaper linear layers.
   (b) Changed the way the nonce is handled in the authentication part of `Deoxys-II`, leading to a more efficient construction with higher security guarantees.
   (c) (from v1.4 to v1.41) Replaced the modular addition by an XOR in the tweak inputs of the encryption part of `Deoxys-II` for faster implementations.
   (d) (from v1.4 to v1.41) Increased the message length counter by one when handling the checksum in `Deoxys-I` for faster implementations.
2. Renamed the modes `Deoxys-`$\neq$ and `Deoxys-`$=$ to `Deoxys-I` and `Deoxys-II`, respectively.
3. Removed the STK construction security explanation, added instead the exact cancellations periods for the `Deoxys-BC` tweakey schedule.
4. Added use-cases in recommended parameter set of Section 2.2.
5. Added a paragraph on cipher instances separation.
6. Completed the hardware implementation Section 7.2.
7. Completed the security analysis Section 4.
8. Added a section on comparing `Deoxys-BC` with `AES`.

B.2 Changelog from v1.2 to v1.3

We detail here the differences between v1.2 and v1.3 of this document.

1. the most important modification is the replacement of the `COPA`-based mode by the new Synthetic Counter in Tweak (`SCT`) mode when nonce-misuse resistance is required.
2. for the the nonce-respecting mode $\mathcal{E}^{\neq}$ and $\mathcal{D}^{\neq}$, the nonce N is removed from the tweak input during the processing of the associated data, offering considerable speed-up when the associated data is fixed.

3. made the security claims more precise.

B.3 Changelog from v1.1 to v1.2

We detail here the differences between v1.1 and v1.2 of this document.

1. Removed one block cipher call in the associated data in the case this input is empty.
2. Changed the wrong "nibble" wording to "byte".

B.4 Changelog from v1 to v1.1

1. Complete specifications of the nonce-misuse resistant mode.
2. Website link added.
3. Acknowledgments section added.
4. Typos and minor inconsistencies corrected.
5. Absence of hidden weaknesses statement added.