SCREAM

Side-Channel Resistant Authenticated Encryption with Masking

Vincent Grosso[1*] Gaëtan Leurent[2] François-Xavier Standaert[1†] Kerem Varici[1‡]
Anthony Journault[1*] François Durvaux[1*] Lubos Gaspar[1‡] Stéphanie Kerckhof[1]

[1] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.
[2] Inria, EPI Secret, Rocquencourt, France.

Contact e-mail: scream@uclouvain.be

Version 3 (Second Round Specifications), August 2015.

Abstract

This document defines the authenticated encryption (with associated data) algorithm SCREAM. It is based on Liskov et al.'s Tweakable Authenticated Encryption (TAE) mode with the new tweakable block cipher Scream. The main desirable features of SCREAM are:

- A *simple* and *regular* design allowing excellent performances on a wide range of architectures, in particular if *masking* is implemented *as a side-channel countermeasure*;
- Inheriting from TAE, *security beyond the birthday bound*, i.e. a 128-bit security guarantee with up to 2^{128} bits of data processed with the same 128-bit key;
- *Low overheads* for the authentication mode (e.g. no extra cipher calls to generate masks);
- *Fully parallelisable* authenticated encryption with *minimal ciphertext length*.

Updates from the first round candidate.

- Fixing mistakes in the authenticated encryption mode leading to simple forgeries (as pointed out by Wang Lei and Sim Siang Meng [18]). When no associated data is present, the mode is now identical to TAE as originally described by Liskov *et. al* [19].
- Removal of the involutive family of authenticated encryption algorithm iSCREAM and its underlying block cipher iScream, which were not selected as second-round candidate.
- Extension of the round constants from 8 to 16 bits (motivated by [14, 17]).
- S-box with improved differential properties and algebraic degree (motivated by [4]).
- (*Editorial*) Clarification of the (primary, secondary, ...) recommended parameters.
- (*Editorial*) Removal of the performance evaluation (to be presented in a dedicated report).
- (*Editorial*) Description of the S-box and L-box properties, and rationale.

* PhD student funded by the ERC Project 280141 (acronym CRASH).
† Associate researcher of the Belgian fund for scientific research (FNRS - F.R.S).
‡ Post-doctoral researcher funded by the ERC Project 280141 (acronym CRASH).

1 Design overview

The following cipher and encryption mode aim to allow implementations that are secure against *Side-Channel Attacks* (SCAs) such as Differential Power Analysis (DPA) [16] and Electro-Magnetic Analysis (EMA) [9]. We believe they are important threats to the security of modern computing devices for an increasingly wide class of applications. In this context, numerous countermeasures have been introduced in the literature (a good survey can be found in [20]). Our designs will focus on SCA security based on masking (aka secret sharing) [5, 11] for three main reasons. First, it is a thoroughly investigated countermeasure with well established benefits (a security gain that can be exponential in the number of shares) and limitations (the requirement that the leakage of each share is sufficiently noisy and independent of the others) [7, 8, 28]. Second, it can take advantage of algorithms tailored for this purpose [10, 26]. Third, it can be implemented efficiently and securely both in software [29, 30] and hardware devices [22, 24]. As a result, and without neglecting the need to combine masking with other countermeasures to reach high physical security levels, we believe it is an important building block in the design of side-channel resistant implementations.

Based on these premises, two important additional criteria are *implementation efficiency and design regularity/simplicity*. The first one is motivated by the fact that more operations inevitably mean more leaking operations that may be exploited by a clever adversary (e.g. such as the analytical one in [33]). The second one derives from the observation that physically secure implementations are easier to obtain if computations are performed on well aligned data. For example, manipulating bits and bytes such as in the PRESENT block cipher [3] raises additional challenges for the developers (to guarantee that the bit manipulations do not leak more information than the byte ones). As a result, we also aim for implementation efficiency on various platforms, with performances close to the ones of the AES in an unprotected setting, and significantly improved when the masking countermeasure is activated. Concretely, this includes privileging highly parallel designs.

As far as the block cipher used in our proposal is concerned, the LS-designs recently introduced at FSE 2014 appear as natural candidates to reach the previous goals [12] – we will take advantage of their general structure. As for the authenticated encryption mode, two main options are available. The first one is to directly exploit a block cipher based solution, for which the extra operations required for authentication are as linear (hence, easy to mask) as possible. Depending on the desired implementation and security properties (e.g. parallelism, need of decryption, misuse resistance), modes such as OCB [32], OTR [21], COPA [1] or COBRA [2] could be considered for this purpose. Yet, a drawback of such schemes is that they only guarantee birthday security. Alternatively, one can take advantage of the Tweakable Authenticated Encryption (TAE) proposed by Liskov et al. [19], which looses nothing in terms of its advantage of the underlying tweakable block cipher, hence can provide *beyond birthday security* – we will opt for this second solution.

Instantiating TAE requires a tweakable block cipher, which we achieve by extending the previously proposed block cipher Fantomas. Our main ingredient for this purpose is the addition of a lightweight tweak/key scheduling algorithm. In this respect, our choices were oriented by the conclusions in [15], where it was observed that allowing *round keys (and tweaks)* to be *derived* "*on-the-fly*" both in encryption and decryption significantly improves hardware performances.

Finally, since we care about physical security issues for which developers anyway have to pay attention to implementation aspects, we will not consider misuse resistance as a goal. For similar reasons, we will propose instances of our ciphers with and without security guarantees against related-key attacks – the later ones being the most relevant for our intended case studies.

In the following, we will denote our tweakable block cipher as Scream, and the TAE based on Scream as SCREAM. The designers have not hidden any weakness in any of these ciphers/modes.

2 Security goals

Our security goals are summarized in Table 1. There is no secret message number. The public message number is a nonce. The cipher does not promise integrity or confidentiality if the legitimate key holder uses the same nonce to encrypt two different (plaintext,associated data) pairs under the same key. The numbers in the table correspond to key guesses to find the secret key for confidentiality, and to online forgery attempts for integrity. Any successful forgery or key recovery should be assumed to completely compromise confidentiality and integrity of all messages.

Table 1: Summary of our security goals for SCREAM.

	bits of security
Confidentiality of the plaintext	128
Integrity of the plaintext	128
Integrity of the associated data	128
Integrity of the public message number	128
Side-channel resistance	masking
Related-key security	optional
Misuse resistance	no

The lower part of the table contains qualitative security statements. Side-channel resistant implementations are expected to be achieved with masking. Related-key security is optional and can be obtained with an increased number of rounds. Misuse resistance is not claimed.

3 Specifications

3.1 Tweakable LS-designs

Scream is based on a variant of the LS-designs introduced in [12] that we will denote as Tweakable LS-designs (TLS-designs). They essentially update a n-bit state x by iterating N_s steps, each of them made of N_r rounds. The state is structured as a $l \times s$ matrix, such that $x[i, \star]$ represents a row and $x[\star, j]$ represents a column. The first row contains state bits 0 to $l-1$, the second row contains state bits l to $2l-1$, ... In the following, the number of rounds per step will be fixed to $N_r = 2$. By contrast, the number of steps will vary and will serve as a parameter to adapt the security margins in Section 5. One significant advantage of TLS-designs is their simplicity: they can be described in a couple of lines, as illustrated in Algorithm 1. In this algorithm, P denotes the plaintext, TK a combination of the master key K and tweak T that we will call tweakey. Finally, S and L are the s-bit S-box and l-bit L-box that are used in our TLS-design.

3.2 The tweakable block cipher Scream

Scream is an n=128-bit cipher with s=8-bit S-boxes and l=16-bit L-boxes. Specifying this cipher requires to define these components, together with the round constants. The binary representation of the L-box and the bitslice representation of the S-box are given in Figure 1 and Algorithm 2.

Algorithm 1 TLS-design with l-bit L-box and s-bit S-box ($n = l \cdot s$)

$x \leftarrow P \oplus TK(0)$; ▷ *$x$ is a $l \times s$ bits matrix*
for $0 < \sigma \leq N_s$ **do**
  for $0 < \rho \leq N_r$ **do**
    $r = 2 \cdot (\sigma - 1) + \rho$; ▷ *Round index*
    for $0 \leq j < l$ **do** ▷ *S-box Layer*
      $x[\star, j] = \mathsf{S}[x[\star, j]]$;
    end for
    $x \leftarrow x \oplus C(r)$; ▷ *Constant addition*
    for $0 \leq i < s$ **do** ▷ *L-box Layer*
      $x[i, \star] = \mathsf{L}[x[i, \star]]$;
    end for
  end for
  $x \leftarrow x \oplus TK(\sigma)$; ▷ *Tweakey addition*
end for
return x

Further descriptions are given in Appendix A, and the round constants are in Appendix B. Following the conventions in [12], the S-box has algebraic degree 6 (vs. 5 for our first-round candidate), differential probability 2^{-5} (vs. 2^{-4} for our first-round candidate), and linear probability 2^{-2} (as our first-round candidate). It can be implemented with only 39 gates, including 12 non-linear gates (vs. 36 and 11 for our first-round candidate). The (unchanged) L-box has branch number 8.

$$\begin{pmatrix}
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Figure 1: Scream L-box

Algorithm 2 Scream S-box and inverse S-box, bitslice implementation.
Input and output is in $(W_0, \ldots, W_7)$ (8 16-bit words)

function SBOX($W_0, \ldots, W_7$)
▷ *Feistel round 1*
$t_0 = (W_1 \wedge W_2) \oplus W_0$
$t_1 = (W_1 \oplus W_3)$
$t_2 = W_2 \oplus t_0$
$W_4 = W_4 \oplus ((W_3 \oplus t_2) \wedge (W_2 \oplus t_1))$
$W_5 = W_5 \oplus t_2$
$W_6 = W_6 \oplus (W_3 \wedge t_0)$
$W_7 = W_7 \oplus (t_1 \wedge t_2)$
▷ *Feistel round 2*
$t_0 = (W_4 \wedge W_5) \oplus W_6$
$t_1 = (W_5 \vee W_6) \oplus W_7$
$t_2 = (W_7 \wedge t_0) \oplus W_4$
$t_3 = (W_4 \wedge t_1) \oplus W_5$
$W_0 = W_0 \oplus t_0$
$W_2 = W_2 \oplus t_1$
$W_1 = W_1 \oplus t_2$
$W_3 = W_3 \oplus t_3$
▷ *Feistel round 3*
$t_0 = \neg((W_1 \wedge W_2) \oplus W_0)$
$t_1 = (W_1 \oplus W_3)$
$t_2 = W_2 \oplus t_0$
$W_4 = W_4 \oplus ((W_3 \oplus t_2) \wedge (W_2 \oplus t_1))$
$W_5 = W_5 \oplus t_2$
$W_6 = W_6 \oplus (W_3 \wedge t_0)$
$W_7 = W_7 \oplus (t_1 \wedge t_2)$
end function

function INVSBOX($W_0, \ldots, W_7$)
▷ *Feistel round 1*
$t_0 = \neg((W_1 \wedge W_2) \oplus W_0)$
$t_1 = (W_1 \oplus W_3)$
$t_2 = W_2 \oplus t_0$
$W_4 = W_4 \oplus ((W_3 \oplus t_2) \wedge (W_2 \oplus t_1))$
$W_5 = W_5 \oplus t_2$
$W_6 = W_6 \oplus (W_3 \wedge t_0)$
$W_7 = W_7 \oplus (t_1 \wedge t_2)$
▷ *Feistel round 2*
$t_0 = (W_4 \wedge W_5) \oplus W_6$
$t_1 = (W_5 \vee W_6) \oplus W_7$
$t_2 = (W_7 \wedge t_0) \oplus W_4$
$t_3 = (W_4 \wedge t_1) \oplus W_5$
$W_0 = W_0 \oplus t_0$
$W_2 = W_2 \oplus t_1$
$W_1 = W_1 \oplus t_2$
$W_3 = W_3 \oplus t_3$
▷ *Feistel round 3*
$t_0 = (W_1 \wedge W_2) \oplus W_0$
$t_1 = (W_1 \oplus W_3)$
$t_2 = W_2 \oplus t_0$
$W_4 = W_4 \oplus ((W_3 \oplus t_2) \wedge (W_2 \oplus t_1))$
$W_5 = W_5 \oplus t_2$
$W_6 = W_6 \oplus (W_3 \wedge t_0)$
$W_7 = W_7 \oplus (t_1 \wedge t_2)$
end function

Finally, Scream has a light tweakey scheduling algorithm that we now detail. It takes the 128-bit key K and the 128-bit tweak T as input. The tweak is divided into 64-bit halves: $T = t_0 \| t_1$. Then, three different tweakeys are used every three steps as follows:

$$\begin{aligned} TK(\sigma = 3i) &= K \oplus (t_0 \| t_1), \\ TK(\sigma = 3i+1) &= K \oplus (t_0 \oplus t_1 \| t_0), \\ TK(\sigma = 3i+2) &= K \oplus (t_1 \| t_0 \oplus t_1). \end{aligned}$$

The tweakeys can also be computed on-the-fly using a simple linear function ϕ, corresponding to multiplication by a primitive element in $GF(4)$ (such that $\phi^2(x) = \phi(x) \oplus x$, and $\phi^3(x) = x$):

$$\begin{aligned} \phi : x_0 \| x_1 &\mapsto (x_0 \oplus x_1) \| x_0, \\ \tau_0 &= T, \\ \tau_{i+1} &= \phi(\tau_i), \\ TK(i) &= K \oplus \tau_i. \end{aligned}$$

3.2.1 Rationale

The construction of the S-box follows ideas of [12] and later results in [4]. In order to reach a good tradeoff between the implementation cost and the security properties, we use a three-round Feistel structure where the first and last round functions are APN, and the middle one is a permutation with differential uniformity 4. The S-box was tweaked to be non-involutive and without fixed-points, in order to limit the effect of some structural attacks (in particular invariant-subspace attacks [17]: we selected an S-box with no invariant subspace containing the direction `1`). However, the direct and inverse S-box are still similar, and a combined implementation has a limited overhead.

The construction of the L-box also follows ideas of [12], but with different optimization goals to ensure security against related-tweak attacks. As discussed in Appendix C, our choice of L-box guarantees a higher number of active S-boxes than the branch number bound alone. Concretely, it was built by randomly permuting the lines and columns of the generating matrix (in systematic form) of a quadratic residue code $QR[32, 16, 8]$, and testing its differential properties. Since the matrix is orthogonal, the differential and linear properties of this L-box are equivalent.

3.3 The encryption mode SCREAM

We use the tweakable block cipher Scream in the TAE mode proposed in [19]. A plaintext $(P_0, \cdots, P_{m-1})$ is encrypted using a nonce (next denoted as N) – the algorithm produces a ciphertext $(C_0, \cdots, C_{m-1})$ and a tag T. Blocks of associated data $(A_0, \cdots, A_{q-1})$ can optionally be authenticated with the message, without being encrypted. During the decryption process, the ciphertext values, tag and associated data are used to recover the plaintext. If the tag is incorrect, the algorithm returns a null output $\perp$. The nonce is supplied by the user, and every message encrypted with a given key must use a different value for N. The nonce bytesize n_b can be chosen by the user between 1 and 15 bytes (the recommended value is 11 bytes), and every message encrypted with a given key must use the same nonce size. The length of the associated data and the length of the message are limited to 2^{124-8n_b} bytes (which corresponds to 2^{120-8n_b} blocks).

There are three main steps in the mode of operation. First, the associated data is processed by dividing it into 128-bit blocks. Each block is encrypted through the tweakable block cipher and the output values are XORed in order to get the main output of this step (denoted as auth.), as

illustrated in Figure 2. If the last block is incomplete, it is padded with a single `1` bit and the rest of the block is filled with zeroes (we denote this padding as $(\texttt{10}^*) := (\texttt{1000}\ldots\texttt{0})$). If the last block is a full block, it is not padded but the encryption uses a different tweak.

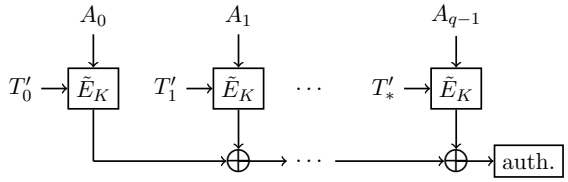

Figure 2: TAE: associated data processing.

Second, plaintext values are encrypted using the tweakable block cipher in order to produce the ciphertext values, as illustrated in Figure 3. If the last block is a partial block, its bitlength is encrypted to generate a mask, which it is then truncated to the partial block size and XORed with the partial plaintext block. Therefore, the ciphertext length is the same as the plaintext length.

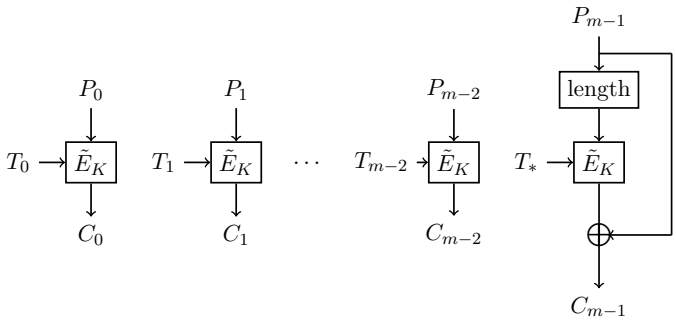

Figure 3: TAE: encryption of the plaintext blocks.

Finally, the tag is generated as represented in Figure 4. That is, the checksum (i.e. the XOR of all plaintext blocks) is first encrypted, and the output of this encryption is then XORed with the output of the associated data processing step (auth.) in order to get the tag.

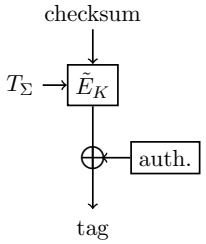

Figure 4: TAE: tag generation.

For the security of the TAE mode, all the calls to the tweakable block cipher must use distinct values of the tweak. In addition, we use some special values for domain separation and we define the tweaks depending on the context. In general, we use tweaks of the form ($N \,\|\, c \,\|$ control byte), where N is the nonce and c is a block counter. The control byte and counter are then specified as follows (with m the number of message blocks, and q the number of associated data blocks):

Plaintext encryption. c is a $120 - 8n_b$-bit block counter.

- All blocks use: $T_c = (N \,\|\, \quad c \quad \|\, \texttt{00000000})$
- In particular, the last block has $c = m - 1$: $T_* = (N \,\|\, m-1 \,\|\, \texttt{00000000})$

Associated data processing. c is a $120 - 8n_b$-bit block counter.

- All blocks but the last one use: $T'_c = (N \,\|\, \quad c \quad \|\, \texttt{00000010})$
- If the last block is a full block, it uses: $T'_* = (N \,\|\, q-1 \,\|\, \texttt{00000100})$
- If the last block is a partial block, it uses: $T'_* = (N \,\|\, q-1 \,\|\, \texttt{00000110})$

Tag generation. Tag generation uses the bitlength $|P|$ of the message: $T_\Sigma = (N \,\|\, |P| \,\|\, \texttt{1})$

with $|P|$ a $127 - 8n_b$-bit integer. Alternatively, this tweak can be written with the block counter c, and the seven low order bits of $|P|$ as control bits (i.e. $\ell = |P| \bmod 128$):

- If the last block is full: $T_\Sigma = (N \,\|\, \quad m \quad \|\, \texttt{00000001})$
- If the last block is partial ($\ell \neq 0$): $T_\Sigma = (N \,\|\, m-1 \,\|\, \quad \ell \,\|\, \texttt{1})$

Our version of the TAE mode is specified in Algorithm 3.

Algorithm 3 Tweakable Authenticated Encryption with Associated Data

function TAE(N, A, P)

▷ *Initialisation*
$auth. \leftarrow 0$;
$C \leftarrow \varnothing$;
$\Sigma \leftarrow 0$;
▷ *TAE: associated data*
if $|A| > 0$ **then**
  for $0 \leq i < \lfloor (|A| - 1)/128 \rfloor$ **do**
    $auth. \leftarrow auth. \oplus \tilde{E}(T'_i, A_i)$;
  end for
  if $|A| \nmid 128$ **then**
    $A_{i+1} \leftarrow A_{i+1} \,\|\, 10^*$;
    $auth. \leftarrow auth. \oplus \tilde{E}(T'_*, A_{i+1})$;
  else
    $auth. \leftarrow auth. \oplus \tilde{E}(T'_*, A_{i+1})$;
  end if
end if

▷ *TAE: Encryption*
for $0 \leq i < \lfloor |P|/128 \rfloor$ **do**
  $C \leftarrow C \,\|\, \tilde{E}(T_i, P_i)$;
  $\Sigma \leftarrow \Sigma \oplus P_i$;
end for
if $|P| \nmid 128$ **then**
  $C \leftarrow C \,\|\, Trunc(\tilde{E}(T_*, |P_{i+1}|)) \oplus P_{i+1}$;
  $\Sigma \leftarrow \Sigma \oplus (P_{i+1} \,\|\, 0^*)$;
end if

▷ *TAE: Tag generation*
$tag \leftarrow \tilde{E}(T_\Sigma, \Sigma) \oplus auth.$;
return C, tag
end function

4 Reference implementation

We use the bitslice representation of Scream in the reference implementation, and consider the 128-bit input values as 8×16 binary matrices on which we apply the step functions. We first process the S-boxes row-wise by using the descriptions of Algorithm 2 in Section 3.2. We then add the round constants before the linear layer, which is performed column-wise using the pre-computed tables given in Appendix A.3.1 and A.4.2. This completes the execution of one round, which is iterated and completed with a tweakey addition to conclude one step. The number of step functions depends on the security level (recommended parameters are in Section 5.3).

5 Security analysis

5.1 The tweakable block cipher Scream

Scream is a tweakable block cipher derived from the LS-cipher Fantomas. The security analysis in [12] shows that such a design has good properties and follows the wide-trail strategy, which allows deriving simple bounds on the probability of differential and linear trails. However, the security notion for a tweakable block cipher is much stronger than for a standard block cipher. Indeed, the family of keyed permutations indexed by the tweak must be secure against adversaries who can query every member of the family. In particular, she can perform a differential attack between two members of the family with a different tweak value. In the following we focus our attention on this scenario, and evaluate the best differential trails in this context. This analysis is mostly dependent on the tweakey scheduling algorithms used in Scream. In Fantomas, the lack of key scheduling combined with key additions every round allows simple related-key trails with a single active S-box per round. In Scream, we avoid this weakness by using a construction similar to the one of LED [13]: the tweakey is used every second round, and we argue that related-tweak trails over σ steps must have at least $\lfloor \sigma/2 \rfloor$ active steps, where each active step has at least 8 active S-boxes. More precisely, the tweakey scheduling of Scream is designed to allow improved bounds in several scenarios, by mixing the bits corresponding to the same S-box. It allows to reuse some of the analysis for fixed tweak/key in a context with differences in the tweak/key. In particular, the L-box has been selected in order to avoid simple iterative trails with a low number of active S-boxes. Our results are listed in Table 2, and the detailed analysis is available in Appendix C.

Setting	Steps:	1	2	3	4	5	6	7	8	9
Single-key, fixed-tweak	Scream	8	20	30	40					
Single-key, chosen-tweaks	Scream	0	8	14	20	28	35			
Related-keys, chosen-tweaks	Scream	0	0	8	14	14	22	28	28	36

Table 2: Minimum number of active S-boxes for Scream.

Since differential trails with 26 active S-boxes or more have a probability below 2^{-128}, the maximum number of steps that can be reached with a related-tweak trail is 4 with a single key, and 6 with related keys. Linear trails can have up to 32 active S-boxes, but there is no simple way to leverage different tweaks for such trails, as done in a related-key differential attack.

5.2 The encryption modes SCREAM

The TAE encryption mode provides a tight security reduction to the security of its underlying tweakable block ciphers that we assume to have 128-bit security (see [19] for the details).

5.3 Suggested and recommended parameters

Based on our previous security analysis, we suggest the parameters listed below, corresponding to lightweight security, single-key security and related-key security. For each type of security, we provide two sets of (tight and safe) parameters. Some of these suggestions being redundant, this makes a total of four sets of parameters for SCREAM (with 6, 8, 10 and 12 steps).

Lightweight security. 80-bit security, with a protocol avoiding related keys
Tight parameters: 6 steps, *Safe parameters*: 8 steps.

Single-key security. 128-bit security, with a protocol avoiding related keys
Tight parameters: 8 steps, *Safe parameters*: 10 steps.

Related-key security. 128-bit security, with possible related keys
Tight parameters: 10 steps, *Safe parameters*: 12 steps.

Our recommended parameters are the safe ones with single-key or related-key security. Lightweight parameters are given for possible comparisons with other schemes. Tight parameters define interesting targets for further cryptanalysis efforts and could lead to additional performance gains.

More precisely, we order our recommended sets of parameters as follows:

- First set of recommended parameters: SCREAM with 10 steps, single-key security.
- Second set of recommended parameters: SCREAM with 12 steps, related-key security.

6 Performance evaluation

Due to its simplicity, Scream is expected to allow excellent performances on a wide variety of platforms. We will provide implementation figures for this algorithm in software – for high-end CPUs and small microcontrollers – and hardware – for Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs), in a dedicated report.

7 Intellectual property

To the extent permitted under law, the designers have released all copyright neighboring rights related to the tweakable block cipher Scream and the authenticated encryption algorithm SCREAM to the public domain. The submitters do not hold any patent related to the designs, nor will apply for any. To the best of their knowledge, the TAE mode of operation is free of patents. Yet, as acknowledged by its authors, it can be viewed as a paraphrase or re-statement of the OCB encryption mode proposed by Rogaway et al. [32], which is patented (US patents 7,046,802, 7,200,227, 7,949,129, and 8,321,675). We note that versions of OCB using a tweakable block cipher, and related patents, are more recent than the publication of TAE by Liskov *et al.* [19]. Referring to Phil Rogaway's webpage [31], "*there are further patents in the authenticated encryption space. I would single out those of Gligor and Donescu (VDG) and Jutla (IBM): 6,963,976, 6,973,187, 7,093,126, and 8,107,620. Do the claims of these patents read against OCB? It is difficult to answer such a question. In fact, I suspect that nobody can give an answer. It seems extremely subjective.*" If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

8 Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each

selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

References

[1] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, E. Tischhauser, and K. Yasuda, *Parallelizable and authenticated online ciphers*, in ASIACRYPT (1), K. Sako and P. Sarkar, eds., vol. 8269 of Lecture Notes in Computer Science, Springer, 2013, pp. 424–443.

[2] E. Andreeva, A. Luykx, B. Mennink, and K. Yasuda, *COBRA: A parallelizable authenticated online cipher without block cipher inverse*, in Cid and Rechberger [6], pp. 187–204.

[3] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, *PRESENT: An ultra-lightweight block cipher*, in CHES, P. Paillier and I. Verbauwhede, eds., vol. 4727 of Lecture Notes in Computer Science, Springer, 2007, pp. 450–466.

[4] A. Canteaut, S. Duval, and G. Leurent, *Construction of Lightweight S-Boxes using Feistel and MISTY structures*, in SAC 2015 (to appear), 2015.

[5] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, *Towards sound approaches to counteract power-analysis attacks*, in Wiener [34], pp. 398–412.

[6] C. Cid and C. Rechberger, eds., *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, vol. 8540 of Lecture Notes in Computer Science, Springer, 2015.

[7] A. Duc, S. Dziembowski, and S. Faust, *Unifying leakage models: From probing attacks to noisy leakage*, in Nguyen and Oswald [23], pp. 423–440.

[8] A. Duc, S. Faust, and F. Standaert, *Making masking security proofs concrete - or how to evaluate the security of any leaking device*, in Oswald and Fischlin [25], pp. 401–429.

[9] K. Gandolfi, C. Mourtel, and F. Olivier, *Electromagnetic analysis: Concrete results*, in CHES, Çetin Kaya Koç, D. Naccache, and C. Paar, eds., vol. 2162 of Lecture Notes in Computer Science, Springer, 2001, pp. 251–261.

[10] B. Gérard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert, *Block ciphers that are easier to mask: How far can we go?*, in CHES, G. Bertoni and J.-S. Coron, eds., vol. 8086 of Lecture Notes in Computer Science, Springer, 2013, pp. 383–399.

[11] L. Goubin and J. Patarin, *DES and differential power analysis (the "duplication" method)*, in Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99,

Worcester, MA, USA, August 12-13, 1999, Proceedings, Ç. K. Koç and C. Paar, eds., vol. 1717 of Lecture Notes in Computer Science, Springer, 1999, pp. 158–172.

[12] V. Grosso, G. Leurent, F. Standaert, and K. Varici, *Ls-designs: Bitslice encryption for efficient masked software implementations*, in Cid and Rechberger [6], pp. 18–37.

[13] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, *The LED block cipher*, in Preneel and Takagi [27], pp. 326–341.

[14] A. Journault, F.-X. Standaert, and K. Varici, *Improving the Security and Efficiency of Block Ciphers based on LS-Designs (Extended Abstract)*. In WCC, P. Charpin and N. Sendrier eds., Paris, France, April 2015.

[15] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.-X. Standaert, *Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint*, in CHES, E. Prouff and P. Schaumont, eds., vol. 7428 of Lecture Notes in Computer Science, Springer, 2012, pp. 390–407.

[16] P. C. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, in Wiener [34], pp. 388–397.

[17] G. Leander, B. Minaud, and S. Rønjom, *A generic approach to invariant subspace attacks: Cryptanalysis of robin, iscream and zorro*, in Oswald and Fischlin [25], pp. 254–283.

[18] W. Lei and S. S. Meng, *Practical Forgery Attacks on SCREAM and iSCREAM*. Message on the `crypto-competitions` mailing list. Available at http://www1.spms.ntu.edu.sg/~syllab/m/images/b/b3/ForgeryAttackonSCREAM.pdf, March 2014.

[19] M. Liskov, R. L. Rivest, and D. Wagner, *Tweakable block ciphers*, J. Cryptology, 24 (2011), pp. 588–613.

[20] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*, Springer, 2007.

[21] K. Minematsu, *Parallelizable rate-1 authenticated encryption from pseudorandom functions*, in Nguyen and Oswald [23], pp. 275–292.

[22] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, *Pushing the limits: A very compact and a threshold implementation of AES*, in EUROCRYPT, K. G. Paterson, ed., vol. 6632 of Lecture Notes in Computer Science, Springer, 2011, pp. 69–88.

[23] P. Q. Nguyen and E. Oswald, eds., *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, vol. 8441 of Lecture Notes in Computer Science, Springer, 2014.

[24] S. Nikova, V. Rijmen, and M. Schläffer, *Secure hardware implementation of nonlinear functions in the presence of glitches*, J. Cryptology, 24 (2011), pp. 292–321.

[25] E. Oswald and M. Fischlin, eds., *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, vol. 9056 of Lecture Notes in Computer Science, Springer, 2015.

[26] G. Piret, T. Roche, and C. Carlet, *PICARO - a block cipher allowing efficient higher-order side-channel resistance*, in ACNS, F. Bao, P. Samarati, and J. Zhou, eds., vol. 7341 of Lecture Notes in Computer Science, Springer, 2012, pp. 311–328.

[27] B. Preneel and T. Takagi, eds., *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, vol. 6917 of Lecture Notes in Computer Science, Springer, 2011.

[28] E. Prouff and M. Rivain, *Masking against side-channel attacks: A formal security proof*, in EUROCRYPT, T. Johansson and P. Q. Nguyen, eds., vol. 7881 of Lecture Notes in Computer Science, Springer, 2013, pp. 142–159.

[29] E. Prouff and T. Roche, *Higher-order glitches free implementation of the AES using secure multi-party computation protocols*, in Preneel and Takagi [27], pp. 63–78.

[30] M. Rivain and E. Prouff, *Provably secure higher-order masking of AES*, in CHES, S. Mangard and F.-X. Standaert, eds., vol. 6225 of Lecture Notes in Computer Science, Springer, 2010, pp. 413–427.

[31] P. Rogaway, *http://www.cs.ucdavis.edu/~rogaway/ocb/*.

[32] P. Rogaway, M. Bellare, and J. Black, *OCB: A block-cipher mode of operation for efficient authenticated encryption*, ACM Trans. Inf. Syst. Secur., 6 (2003), pp. 365–403.

[33] N. Veyrat-Charvillon, B. Gérard, and F. Standaert, *Soft analytical side-channel attacks*, in Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, P. Sarkar and T. Iwata, eds., vol. 8873 of Lecture Notes in Computer Science, Springer, 2014, pp. 282–296.

[34] M. J. Wiener, ed., *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, vol. 1666 of Lecture Notes in Computer Science, Springer, 1999.

A Scream components' descriptions

A.1 Scream S-box

A.1.1 Table representation

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

Table 3: Scream S-box, table representation.

A.1.2 Algebraic Normal Form

$y_0 = x_1x_2x_3x_5 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_4 + x_0x_1x_5 + x_0x_2x_5 + x_1x_2x_5 + x_0x_3x_5 + x_1x_3x_5 + x_0x_1 + x_0x_2 + x_1x_2 + x_0x_3 + x_2x_3 + x_0x_4 + x_2x_4 + x_2x_5 + x_3x_5 + x_4x_5 + x_0 + x_2 + x_6$

$y_1 = x_0x_1x_2x_3x_5 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_4 + x_0x_1x_2x_5 + x_0x_1x_3x_5 + x_0x_1x_4x_5 + x_0x_3x_4x_5 + x_2x_3x_4x_5 + x_1x_2x_3x_6 + x_0x_1x_3x_7 + x_0x_2x_3x_7 + x_1x_2x_4x_7 + x_0x_1x_5x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_0x_3x_5x_7 + x_1x_3x_5x_7 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + x_0x_1x_4 + x_0x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_0x_1x_6 + x_0x_3x_6 + x_2x_3x_6 + x_0x_1x_7 + x_0x_2x_7 + x_1x_2x_7 + x_0x_3x_7 + x_2x_3x_7 + x_0x_4x_7 + x_2x_4x_7 + x_2x_5x_7 + x_3x_5x_7 + x_4x_5x_7 + x_0x_2 + x_1x_2 + x_1x_3 + x_2x_7 + x_6x_7 + x_1 + x_2 + x_3 + x_4$

$y_2 = x_1x_2x_3x_5 + x_0x_2x_3 + x_0x_3x_5 + x_1x_2x_6 + x_0x_1 + x_1x_2 + x_0x_3 + x_2x_3 + x_0x_6 + x_2x_6 + x_5x_6 + x_0 + x_5 + x_6 + x_7$

$y_3 = x_0x_1x_2x_3x_5 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_5x_6 + x_0x_2x_3x_4 + x_0x_3x_4x_5 + x_0x_1x_3x_6 + x_0x_2x_3x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_1x_2x_3x_7 + x_0x_1x_2 + x_0x_2x_3 + x_0x_1x_4 + x_1x_2x_4 + x_0x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_0x_2x_5 + x_1x_2x_5 + x_0x_3x_5 + x_1x_3x_5 + x_0x_3x_6 + x_1x_3x_6 + x_2x_3x_6 + x_0x_4x_6 + x_2x_4x_6 + x_2x_5x_6 + x_3x_5x_6 + x_4x_5x_6 + x_0x_1x_7 + x_0x_2x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_0x_2 + x_2x_3 + x_0x_4 + x_2x_4 + x_2x_5 + x_3x_5 + x_4x_5 + x_3x_6 + x_4x_6 + x_2x_7 + x_3x_7 + x_4x_7 + x_0 + x_3 + x_5$

$y_4 = x_0x_1x_2x_3x_4 + x_0x_1x_2x_3x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_2x_5x_6 + x_0x_2x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_1x_2x_3x_7 + x_0x_1x_2x_4x_7 + x_0x_2x_3x_4x_7 + x_1x_2x_3x_4x_7 + x_0x_1x_2x_5x_7 + x_1x_2x_3x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_3x_4 + x_0x_2x_3x_4 + x_1x_2x_3x_4 + x_0x_1x_3x_5 + x_0x_1x_4x_5 + x_1x_2x_4x_5 + x_0x_3x_4x_5 + x_0x_1x_3x_6 + x_0x_1x_4x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_0x_4x_5x_6 + x_2x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_1x_2x_7 + x_0x_1x_3x_7 + x_0x_2x_3x_7 + x_1x_2x_3x_7 + x_0x_2x_4x_7 + x_1x_2x_4x_7 + x_0x_1x_5x_7 + x_1x_3x_5x_7 + x_2x_3x_5x_7 + x_0x_4x_5x_7 + x_3x_4x_5x_7 + x_0x_1x_6x_7 + x_1x_2x_6x_7 +$

$x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 + x_2x_5x_6x_7 + x_0x_1x_2 + x_0x_1x_5 + x_1x_2x_5 + x_2x_3x_5 + x_0x_4x_5 + x_1x_4x_5 + x_2x_4x_5 + x_0x_1x_6 + x_0x_2x_6 + x_1x_2x_6 + x_1x_3x_6 + x_0x_4x_6 + x_2x_4x_6 + x_3x_4x_6 + x_0x_5x_6 + x_1x_5x_6 + x_4x_5x_6 + x_1x_3x_7 + x_2x_4x_7 + x_3x_4x_7 + x_1x_5x_7 + x_3x_5x_7 + x_4x_6x_7 + x_5x_6x_7 + x_0x_4 + x_2x_4 + x_3x_4 + x_1x_5 + x_3x_5 + x_4x_5 + x_0x_6 + x_3x_6 + x_4x_6 + x_5x_6 + x_0x_7 + x_1x_7 + x_4x_7 + x_6x_7 + x_1 + x_2 + x_3 + x_6$

$y_5 = x_0x_1x_2x_3x_5 + x_0x_1x_2x_4x_5 + x_0x_2x_3x_4x_5 + x_1x_2x_3x_5x_6 + x_0x_1x_2x_3x_7 + x_0x_1x_2x_5x_7 + x_0x_2x_3x_5x_7 + x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_1x_2x_3x_4 + x_0x_1x_3x_5 + x_1x_2x_3x_5 + x_0x_3x_4x_5 + x_2x_3x_4x_5 + x_0x_1x_2x_6 + x_0x_1x_3x_6 + x_1x_2x_3x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_0x_1x_2x_7 + x_1x_2x_3x_7 + x_0x_2x_4x_7 + x_1x_2x_4x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_2x_3x_5x_7 + x_2x_4x_5x_7 + x_1x_2x_3 + x_0x_1x_4 + x_0x_3x_4 + x_0x_3x_5 + x_0x_1x_6 + x_1x_2x_6 + x_0x_3x_6 + x_1x_3x_6 + x_0x_4x_6 + x_2x_4x_6 + x_1x_5x_6 + x_2x_5x_6 + x_3x_5x_6 + x_4x_5x_6 + x_0x_1x_7 + x_0x_3x_7 + x_1x_3x_7 + x_2x_3x_7 + x_2x_4x_7 + x_2x_5x_7 + x_2x_6x_7 + x_0x_2 + x_2x_4 + x_1x_5 + x_0x_6 + x_1x_6 + x_2x_6 + x_3x_6 + x_4x_6 + x_5x_6 + x_1x_7 + x_3x_7 + x_4x_7 + x_0 + x_7 + 1$

$y_6 = x_0x_1x_2x_3x_4x_5 + x_0x_1x_2x_3x_5x_7 + x_0x_1x_2x_3x_4 + x_0x_1x_2x_4x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_3x_5x_6 + x_0x_2x_3x_5x_6 + x_1x_2x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_2x_3x_4x_7 + x_0x_1x_2x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_0x_1x_3x_4 + x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_0x_1x_3x_6 + x_0x_2x_3x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_2x_3x_5x_6 + x_0x_4x_5x_6 + x_1x_4x_5x_6 + x_2x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_1x_3x_7 + x_1x_2x_3x_7 + x_0x_2x_4x_7 + x_0x_3x_4x_7 + x_2x_3x_4x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_2x_3x_5x_7 + x_2x_4x_5x_7 + x_0x_1x_6x_7 + x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 + x_2x_5x_6x_7 + x_0x_1x_3 + x_1x_2x_3 + x_0x_1x_4 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4 + x_0x_2x_5 + x_0x_4x_5 + x_1x_4x_5 + x_0x_2x_6 + x_1x_3x_6 + x_2x_3x_6 + x_1x_4x_6 + x_3x_4x_6 + x_2x_5x_6 + x_3x_5x_6 + x_0x_1x_7 + x_0x_3x_7 + x_2x_3x_7 + x_0x_4x_7 + x_1x_4x_7 + x_3x_4x_7 + x_1x_5x_7 + x_2x_5x_7 + x_3x_6x_7 + x_4x_6x_7 + x_0x_1 + x_2x_3 + x_2x_4 + x_0x_5 + x_1x_5 + x_2x_5 + x_3x_5 + x_4x_5 + x_0x_6 + x_3x_6 + x_4x_6 + x_5x_6 + x_3x_7 + x_3 + x_5 + x_6$

$y_7 = x_0x_1x_2x_3x_5 + x_0x_1x_2x_4x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_2x_5x_6 + x_0x_1x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_1x_2x_4x_7 + x_0x_2x_3x_4x_7 + x_0x_1x_3x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_3x_4 + x_0x_1x_2x_5 + x_0x_2x_3x_5 + x_0x_1x_4x_5 + x_1x_2x_4x_5 + x_0x_2x_3x_6 + x_1x_2x_3x_6 + x_0x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_1x_3x_5x_6 + x_0x_4x_5x_6 + x_1x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_2x_3x_7 + x_0x_3x_4x_7 + x_2x_3x_4x_7 + x_0x_1x_5x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_2x_3x_5x_7 + x_0x_4x_5x_7 + x_2x_4x_5x_7 + x_0x_1x_6x_7 + x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 + x_2x_5x_6x_7 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + x_0x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_1x_2x_5 + x_0x_4x_5 + x_2x_4x_5 + x_1x_2x_6 + x_1x_3x_6 + x_2x_3x_6 + x_0x_4x_6 + x_1x_4x_6 + x_2x_4x_6 + x_3x_4x_6 + x_2x_5x_6 + x_3x_5x_6 + x_4x_5x_6 + x_0x_1x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_1x_4x_7 + x_2x_4x_7 + x_3x_4x_7 + x_1x_5x_7 + x_2x_5x_7 + x_0x_6x_7 + x_3x_6x_7 + x_4x_6x_7 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_3 + x_2x_3 + x_0x_4 + x_0x_5 + x_1x_5 + x_2x_5 + x_4x_5 + x_0x_6 + x_1x_6 + x_2x_6 + x_4x_6 + x_5x_6 + x_0x_7 + x_2x_7 + x_3x_7 + x_4x_7 + x_5x_7 + x_0 + x_1 + x_4 + x_7$

A.2 Inverse Scream S-box

A.2.1 Table representation

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

Table 4: Inverse Scream S-box, table representation.

A.2.2 Algebraic Normal Form

$y_0 = x_1x_2x_3x_5 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_4 + x_0x_1x_5 + x_0x_2x_5 + x_1x_2x_5 + x_0x_3x_5 + x_1x_3x_5 + x_0x_1 + x_0x_2 + x_1x_2 + x_0x_3 + x_1x_3 + x_0x_4 + x_2x_4 + x_1x_5 + x_4x_5 + x_0 + x_1 + x_3 + x_4 + x_6$

$y_1 = x_0x_1x_2x_3x_5 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_5x_7 + x_1x_2x_3x_4 + x_0x_1x_2x_5 + x_0x_1x_3x_5 + x_1x_2x_3x_5 + x_0x_1x_4x_5 + x_0x_3x_4x_5 + x_2x_3x_4x_5 + x_1x_2x_3x_6 + x_0x_1x_3x_7 + x_0x_2x_3x_7 + x_1x_2x_4x_7 + x_0x_1x_5x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_0x_3x_5x_7 + x_1x_3x_5x_7 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + x_0x_1x_4 + x_0x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_1x_2x_5 + x_1x_3x_5 + x_1x_4x_5 + x_3x_4x_5 + x_0x_1x_6 + x_0x_3x_6 + x_2x_3x_6 + x_0x_1x_7 + x_0x_2x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_0x_4x_7 + x_2x_4x_7 + x_1x_5x_7 + x_4x_5x_7 + x_0x_2 + x_1x_3 + x_2x_3 + x_1x_4 + x_3x_4 + x_1x_5 + x_1x_6 + x_3x_6 + x_1x_7 + x_3x_7 + x_4x_7 + x_6x_7 + x_1 + x_3 + x_4$

$y_2 = x_1x_2x_3x_5 + x_0x_2x_3 + x_0x_3x_5 + x_1x_2x_6 + x_0x_1 + x_1x_2 + x_0x_3 + x_3x_5 + x_0x_6 + x_2x_6 + x_5x_6 + x_0 + x_1 + x_3 + x_5 + x_7 + 1$

$y_3 = x_0x_1x_2x_3x_5 + x_1x_2x_3x_4x_5 + x_1x_2x_3x_5x_6 + x_0x_2x_3x_4 + x_1x_2x_3x_5 + x_0x_3x_4x_5 + x_0x_1x_3x_6 + x_0x_2x_3x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_1x_2x_3x_7 + x_0x_1x_2 + x_0x_2x_3 + x_0x_1x_4 + x_1x_2x_4 + x_0x_3x_4 + x_0x_1x_5 + x_0x_2x_5 + x_1x_2x_5 + x_0x_3x_5 + x_1x_3x_5 + x_3x_4x_5 + x_0x_3x_6 + x_0x_4x_6 + x_2x_4x_6 + x_1x_5x_6 + x_4x_5x_6 + x_0x_1x_7 + x_0x_2x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_0x_2 + x_1x_2 + x_0x_4 + x_1x_4 + x_2x_4 + x_3x_4 + x_1x_5 + x_4x_5 + x_1x_7 + x_4x_7 + x_0 + x_2 + x_3 + x_4 + x_5 + 1$

$y_4 = x_0x_1x_2x_3x_4 + x_0x_1x_2x_3x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_2x_5x_6 + x_0x_2x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_1x_2x_3x_7 + x_0x_1x_2x_4x_7 + x_0x_2x_3x_4x_7 + x_1x_2x_3x_4x_7 + x_0x_1x_2x_5x_7 + x_1x_2x_3x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_3x_4 + x_0x_2x_3x_4 + x_0x_1x_3x_5 + x_1x_2x_3x_5 + x_0x_1x_4x_5 + x_1x_2x_4x_5 + x_0x_3x_4x_5 + x_1x_3x_4x_5 + x_2x_3x_4x_5 + x_0x_1x_3x_6 + x_1x_2x_3x_6 + x_0x_1x_4x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_2x_3x_5x_6 + x_0x_4x_5x_6 + x_2x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_1x_2x_7 + x_0x_1x_3x_7 + x_0x_2x_3x_7 + x_0x_2x_4x_7 + x_2x_3x_4x_7 + x_0x_1x_5x_7 + x_1x_2x_5x_7 + x_1x_3x_5x_7 + x_2x_3x_5x_7 + x_0x_4x_5x_7 + x_3x_4x_5x_7 + x_0x_1x_6x_7 + x_1x_2x_6x_7 + x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 +$

$x_2x_5x_6x_7 + x_0x_1x_2 + x_1x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_1x_2x_5 + x_1x_3x_5 + x_2x_3x_5 + x_0x_4x_5 + x_2x_4x_5 + x_3x_4x_5 + x_0x_1x_6 + x_0x_2x_6 + x_1x_2x_6 + x_0x_4x_6 + x_1x_4x_6 + x_2x_4x_6 + x_3x_4x_6 + x_0x_5x_6 + x_3x_5x_6 + x_1x_2x_7 + x_2x_3x_7 + x_3x_4x_7 + x_3x_5x_7 + x_4x_5x_7 + x_1x_6x_7 + x_3x_6x_7 + x_4x_6x_7 + x_5x_6x_7 + x_1x_2 + x_0x_4 + x_2x_4 + x_3x_4 + x_3x_5 + x_0x_6 + x_1x_6 + x_2x_6 + x_3x_6 + x_0x_7 + x_1x_7 + x_4x_7 + x_6x_7 + x_1 + x_2 + x_3 + x_4 + x_7$

$y_5 = x_0x_1x_2x_3x_5 + x_0x_1x_2x_4x_5 + x_0x_2x_3x_4x_5 + x_1x_2x_3x_5x_6 + x_0x_1x_2x_3x_7 + x_0x_1x_2x_5x_7 + x_0x_2x_3x_5x_7 + x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_1x_2x_3x_4 + x_0x_1x_3x_5 + x_1x_2x_4x_5 + x_0x_3x_4x_5 + x_0x_1x_2x_6 + x_0x_1x_3x_6 + x_1x_2x_3x_6 + x_1x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_0x_1x_2x_7 + x_0x_2x_4x_7 + x_1x_2x_4x_7 + x_0x_2x_5x_7 + x_2x_4x_5x_7 + x_0x_1x_4 + x_1x_2x_4 + x_0x_3x_4 + x_0x_3x_5 + x_1x_3x_5 + x_3x_4x_5 + x_0x_1x_6 + x_0x_3x_6 + x_0x_4x_6 + x_2x_4x_6 + x_4x_5x_6 + x_0x_1x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_2x_3x_7 + x_2x_6x_7 + x_0x_2 + x_1x_4 + x_2x_4 + x_3x_4 + x_1x_5 + x_3x_5 + x_0x_6 + x_2x_6 + x_5x_6 + x_4x_7 + x_0 + x_2 + x_6 + x_7$

$y_6 = x_0x_1x_2x_3x_4x_5 + x_0x_1x_2x_3x_5x_7 + x_0x_1x_2x_3x_4 + x_0x_1x_2x_4x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_1x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_3x_5x_6 + x_0x_2x_3x_5x_6 + x_1x_2x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_2x_3x_4x_7 + x_0x_1x_2x_5x_7 + x_1x_2x_3x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_2x_3 + x_0x_1x_2x_4 + x_0x_1x_3x_4 + x_1x_2x_3x_5 + x_1x_3x_4x_5 + x_2x_3x_4x_5 + x_0x_1x_3x_6 + x_0x_2x_3x_6 + x_1x_2x_3x_6 + x_0x_2x_5x_6 + x_1x_2x_5x_6 + x_0x_3x_5x_6 + x_1x_3x_5x_6 + x_0x_4x_5x_6 + x_1x_4x_5x_6 + x_2x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_1x_3x_7 + x_1x_2x_3x_7 + x_0x_2x_4x_7 + x_0x_3x_4x_7 + x_0x_2x_5x_7 + x_2x_3x_5x_7 + x_2x_4x_5x_7 + x_0x_1x_6x_7 + x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 + x_2x_5x_6x_7 + x_0x_1x_3 + x_0x_1x_4 + x_0x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_0x_2x_5 + x_0x_4x_5 + x_1x_4x_5 + x_0x_2x_6 + x_1x_4x_6 + x_3x_4x_6 + x_4x_5x_6 + x_0x_1x_7 + x_0x_3x_7 + x_1x_3x_7 + x_2x_3x_7 + x_0x_4x_7 + x_1x_4x_7 + x_2x_4x_7 + x_1x_5x_7 + x_1x_6x_7 + x_4x_6x_7 + x_0x_1 + x_1x_3 + x_2x_3 + x_1x_4 + x_0x_5 + x_1x_5 + x_3x_5 + x_0x_6 + x_2x_6 + x_3x_6 + x_4x_6 + x_5x_6 + x_1x_7 + x_4x_7 + x_1 + x_3$

$y_7 = x_0x_1x_2x_3x_5 + x_0x_1x_2x_4x_5 + x_0x_1x_3x_4x_5 + x_0x_2x_3x_4x_5 + x_0x_1x_2x_3x_6 + x_0x_1x_2x_5x_6 + x_0x_1x_3x_5x_6 + x_1x_2x_4x_5x_6 + x_0x_1x_2x_4x_7 + x_0x_2x_3x_4x_7 + x_0x_1x_3x_5x_7 + x_2x_3x_4x_5x_7 + x_1x_2x_3x_6x_7 + x_0x_1x_3x_4 + x_0x_1x_2x_5 + x_0x_2x_3x_5 + x_1x_2x_3x_5 + x_0x_1x_4x_5 + x_1x_3x_4x_5 + x_2x_3x_4x_5 + x_0x_2x_3x_6 + x_0x_2x_4x_6 + x_0x_1x_5x_6 + x_0x_2x_5x_6 + x_0x_4x_5x_6 + x_1x_4x_5x_6 + x_3x_4x_5x_6 + x_0x_2x_3x_7 + x_1x_2x_4x_7 + x_0x_3x_4x_7 + x_0x_1x_5x_7 + x_0x_2x_5x_7 + x_1x_2x_5x_7 + x_1x_3x_5x_7 + x_2x_3x_5x_7 + x_0x_4x_5x_7 + x_2x_4x_5x_7 + x_0x_1x_6x_7 + x_0x_3x_6x_7 + x_1x_3x_6x_7 + x_2x_3x_6x_7 + x_2x_5x_6x_7 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + x_0x_3x_4 + x_1x_3x_4 + x_2x_3x_4 + x_0x_1x_5 + x_2x_3x_5 + x_0x_4x_5 + x_1x_4x_5 + x_2x_4x_5 + x_1x_2x_6 + x_1x_3x_6 + x_0x_4x_6 + x_1x_4x_6 + x_3x_4x_6 + x_1x_5x_6 + x_3x_5x_6 + x_0x_1x_7 + x_1x_2x_7 + x_0x_3x_7 + x_1x_3x_7 + x_2x_3x_7 + x_1x_4x_7 + x_2x_4x_7 + x_4x_5x_7 + x_0x_6x_7 + x_1x_6x_7 + x_4x_6x_7 + x_0x_1 + x_0x_2 + x_1x_2 + x_0x_3 + x_1x_3 + x_0x_4 + x_3x_4 + x_0x_5 + x_2x_5 + x_0x_6 + x_1x_6 + x_2x_6 + x_5x_6 + x_0x_7 + x_1x_7 + x_2x_7 + x_4x_7 + x_5x_7 + x_6x_7 + x_0 + x_2 + x_3 + x_5 + x_6 + 1$

A.3 Scream L-box

A.3.1 8-bit table representation

$\mathsf{L}(b_0 \,\|\, b_1) = \big(\mathsf{L}_{1,1}(b_1) \oplus \mathsf{L}_{2,1}(b_0)\big) \,\|\, \big(\mathsf{L}_{1,2}(b_1) \oplus \mathsf{L}_{2,2}(b_0)\big)$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	38	52	6A	7B	43	29	11	96	AE	C4	FC	ED	D5	BF	87
10	D7	EF	85	BD	AC	94	FE	C6	41	79	13	2B	3A	02	68	50
20	3A	02	68	50	41	79	13	2B	AC	94	FE	C6	D7	EF	85	BD
30	ED	D5	BF	87	96	AE	C4	FC	7B	43	29	11	00	38	52	6A
40	E5	DD	B7	8F	9E	A6	CC	F4	73	4B	21	19	08	30	5A	62
50	32	0A	60	58	49	71	1B	23	A4	9C	F6	CE	DF	E7	8D	B5
60	DF	E7	8D	B5	A4	9C	F6	CE	49	71	1B	23	32	0A	60	58
70	08	30	5A	62	73	4B	21	19	9E	A6	CC	F4	E5	DD	B7	8F
80	FE	C6	AC	94	85	BD	D7	EF	68	50	3A	02	13	2B	41	79
90	29	11	7B	43	52	6A	00	38	BF	87	ED	D5	C4	FC	96	AE
A0	C4	FC	96	AE	BF	87	ED	D5	52	6A	00	38	29	11	7B	43
B0	13	2B	41	79	68	50	3A	02	85	BD	D7	EF	FE	C6	AC	94
C0	1B	23	49	71	60	58	32	0A	8D	B5	DF	E7	F6	CE	A4	9C
D0	CC	F4	9E	A6	B7	8F	E5	DD	5A	62	08	30	21	19	73	4B
E0	21	19	73	4B	5A	62	08	30	B7	8F	E5	DD	CC	F4	9E	A6
F0	F6	CE	A4	9C	8D	B5	DF	E7	60	58	32	0A	1B	23	49	71

Table 5: $\mathsf{L}_{1,1}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	5C	A9	F5	B3	EF	1A	46	C1	9D	68	34	72	2E	DB	87
10	6D	31	C4	98	DE	82	77	2B	AC	F0	05	59	1F	43	B6	EA
20	E0	BC	49	15	53	0F	FA	A6	21	7D	88	D4	92	CE	3B	67
30	8D	D1	24	78	3E	62	97	CB	4C	10	E5	B9	FF	A3	56	0A
40	24	78	8D	D1	97	CB	3E	62	E5	B9	4C	10	56	0A	FF	A3
50	49	15	E0	BC	FA	A6	53	0F	88	D4	21	7D	3B	67	92	CE
60	C4	98	6D	31	77	2B	DE	82	05	59	AC	F0	B6	EA	1F	43
70	A9	F5	00	5C	1A	46	B3	EF	68	34	C1	9D	DB	87	72	2E
80	A5	F9	0C	50	16	4A	BF	E3	64	38	CD	91	D7	8B	7E	22
90	C8	94	61	3D	7B	27	D2	8E	09	55	A0	FC	BA	E6	13	4F
A0	45	19	EC	B0	F6	AA	5F	03	84	D8	2D	71	37	6B	9E	C2
B0	28	74	81	DD	9B	C7	32	6E	E9	B5	40	1C	5A	06	F3	AF
C0	81	DD	28	74	32	6E	9B	C7	40	1C	E9	B5	F3	AF	5A	06
D0	EC	B0	45	19	5F	03	F6	AA	2D	71	84	D8	9E	C2	37	6B
E0	61	3D	C8	94	D2	8E	7B	27	A0	FC	09	55	13	4F	BA	E6
F0	0C	50	A5	F9	BF	E3	16	4A	CD	91	64	38	7E	22	D7	8B

Table 6: $\mathsf{L}_{1,2}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	46	F1	B7	A1	E7	50	16	7F	39	8E	C8	DE	98	2F	69
10	67	21	96	D0	C6	80	37	71	18	5E	E9	AF	B9	FF	48	0E
20	7A	3C	8B	CD	DB	9D	2A	6C	05	43	F4	B2	A4	E2	55	13
30	1D	5B	EC	AA	BC	FA	4D	0B	62	24	93	D5	C3	85	32	74
40	70	36	81	C7	D1	97	20	66	0F	49	FE	B8	AE	E8	5F	19
50	17	51	E6	A0	B6	F0	47	01	68	2E	99	DF	C9	8F	38	7E
60	0A	4C	FB	BD	AB	ED	5A	1C	75	33	84	C2	D4	92	25	63
70	6D	2B	9C	DA	CC	8A	3D	7B	12	54	E3	A5	B3	F5	42	04
80	8A	CC	7B	3D	2B	6D	DA	9C	F5	B3	04	42	54	12	A5	E3
90	ED	AB	1C	5A	4C	0A	BD	FB	92	D4	63	25	33	75	C2	84
A0	F0	B6	01	47	51	17	A0	E6	8F	C9	7E	38	2E	68	DF	99
B0	97	D1	66	20	36	70	C7	81	E8	AE	19	5F	49	0F	B8	FE
C0	FA	BC	0B	4D	5B	1D	AA	EC	85	C3	74	32	24	62	D5	93
D0	9D	DB	6C	2A	3C	7A	CD	8B	E2	A4	13	55	43	05	B2	F4
E0	80	C6	71	37	21	67	D0	96	FF	B9	0E	48	5E	18	AF	E9
F0	E7	A1	16	50	46	00	B7	F1	98	DE	69	2F	39	7F	C8	8E

Table 7: $\mathsf{L}_{2,1}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	4B	AF	E4	33	78	9C	D7	74	3F	DB	90	47	0C	E8	A3
10	12	59	BD	F6	21	6A	8E	C5	66	2D	C9	82	55	1E	FA	B1
20	6F	24	C0	8B	5C	17	F3	B8	1B	50	B4	FF	28	63	87	CC
30	7D	36	D2	99	4E	05	E1	AA	09	42	A6	ED	3A	71	95	DE
40	1B	50	B4	FF	28	63	87	CC	6F	24	C0	8B	5C	17	F3	B8
50	09	42	A6	ED	3A	71	95	DE	7D	36	D2	99	4E	05	E1	AA
60	74	3F	DB	90	47	0C	E8	A3	00	4B	AF	E4	33	78	9C	D7
70	66	2D	C9	82	55	1E	FA	B1	12	59	BD	F6	21	6A	8E	C5
80	B1	FA	1E	55	82	C9	2D	66	C5	8E	6A	21	F6	BD	59	12
90	A3	E8	0C	47	90	DB	3F	74	D7	9C	78	33	E4	AF	4B	00
A0	DE	95	71	3A	ED	A6	42	09	AA	E1	05	4E	99	D2	36	7D
B0	CC	87	63	28	FF	B4	50	1B	B8	F3	17	5C	8B	C0	24	6F
C0	AA	E1	05	4E	99	D2	36	7D	DE	95	71	3A	ED	A6	42	09
D0	B8	F3	17	5C	8B	C0	24	6F	CC	87	63	28	FF	B4	50	1B
E0	C5	8E	6A	21	F6	BD	59	12	B1	FA	1E	55	82	C9	2D	66
F0	D7	9C	78	33	E4	AF	4B	00	A3	E8	0C	47	90	DB	3F	74

Table 8: $\mathsf{L}_{2,2}$.

A.4 Inverse Scream L-box

A.4.1 Binary representation

$$\begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

A.4.2 8-bit table representation

$\mathsf{L}(b_0 \parallel b_1) = \big(\mathsf{L}_{1,1}(b_1) \oplus \mathsf{L}_{2,1}(b_0)\big) \parallel \big(\mathsf{L}_{1,2}(b_1) \oplus \mathsf{L}_{2,2}(b_0)\big)$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	E7	77	90	2A	CD	5D	BA	63	84	14	F3	49	AE	3E	D9
10	DC	3B	AB	4C	F6	11	81	66	BF	58	C8	2F	95	72	E2	05
20	AE	49	D9	3E	84	63	F3	14	CD	2A	BA	5D	E7	00	90	77
30	72	95	05	E2	58	BF	2F	C8	11	F6	66	81	3B	DC	4C	AB
40	29	CE	5E	B9	03	E4	74	93	4A	AD	3D	DA	60	87	17	F0
50	F5	12	82	65	DF	38	A8	4F	96	71	E1	06	BC	5B	CB	2C
60	87	60	F0	17	AD	4A	DA	3D	E4	03	93	74	CE	29	B9	5E
70	5B	BC	2C	CB	71	96	06	E1	38	DF	4F	A8	12	F5	65	82
80	82	65	F5	12	A8	4F	DF	38	E1	06	96	71	CB	2C	BC	5B
90	5E	B9	29	CE	74	93	03	E4	3D	DA	4A	AD	17	F0	60	87
A0	2C	CB	5B	BC	06	E1	71	96	4F	A8	38	DF	65	82	12	F5
B0	F0	17	87	60	DA	3D	AD	4A	93	74	E4	03	B9	5E	CE	29
C0	AB	4C	DC	3B	81	66	F6	11	C8	2F	BF	58	E2	05	95	72
D0	77	90	00	E7	5D	BA	2A	CD	14	F3	63	84	3E	D9	49	AE
E0	05	E2	72	95	2F	C8	58	BF	66	81	11	F6	4C	AB	3B	DC
F0	D9	3E	AE	49	F3	14	84	63	BA	5D	CD	2A	90	77	E7	00

Table 9: $\mathsf{L}_{1,1}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	9E	04	9A	D1	4F	D5	4B	13	8D	17	89	C2	5C	C6	58
10	05	9B	01	9F	D4	4A	D0	4E	16	88	12	8C	C7	59	C3	5D
20	F6	68	F2	6C	27	B9	23	BD	E5	7B	E1	7F	34	AA	30	AE
30	F3	6D	F7	69	22	BC	26	B8	E0	7E	E4	7A	31	AF	35	AB
40	39	A7	3D	A3	E8	76	EC	72	2A	B4	2E	B0	FB	65	FF	61
50	3C	A2	38	A6	ED	73	E9	77	2F	B1	2B	B5	FE	60	FA	64
60	CF	51	CB	55	1E	80	1A	84	DC	42	D8	46	0D	93	09	97
70	CA	54	CE	50	1B	85	1F	81	D9	47	DD	43	08	96	0C	92
80	AE	30	AA	34	7F	E1	7B	E5	BD	23	B9	27	6C	F2	68	F6
90	AB	35	AF	31	7A	E4	7E	E0	B8	26	BC	22	69	F7	6D	F3
A0	58	C6	5C	C2	89	17	8D	13	4B	D5	4F	D1	9A	04	9E	00
B0	5D	C3	59	C7	8C	12	88	16	4E	D0	4A	D4	9F	01	9B	05
C0	97	09	93	0D	46	D8	42	DC	84	1A	80	1E	55	CB	51	CF
D0	92	0C	96	08	43	DD	47	D9	81	1F	85	1B	50	CE	54	CA
E0	61	FF	65	FB	B0	2E	B4	2A	72	EC	76	E8	A3	3D	A7	39
F0	64	FA	60	FE	B5	2B	B1	2F	77	E9	73	ED	A6	38	A2	3C

Table 10: $\mathsf{L}_{1,2}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	1E	B9	A7	19	07	A0	BE	A8	B6	11	0F	B1	AF	08	16
10	6A	74	D3	CD	73	6D	CA	D4	C2	DC	7B	65	DB	C5	62	7C
20	7E	60	C7	D9	67	79	DE	C0	D6	C8	6F	71	CF	D1	76	68
30	14	0A	AD	B3	0D	13	B4	AA	BC	A2	05	1B	A5	BB	1C	02
40	7B	65	C2	DC	62	7C	DB	C5	D3	CD	6A	74	CA	D4	73	6D
50	11	0F	A8	B6	08	16	B1	AF	B9	A7	00	1E	A0	BE	19	07
60	05	1B	BC	A2	1C	02	A5	BB	AD	B3	14	0A	B4	AA	0D	13
70	6F	71	D6	C8	76	68	CF	D1	C7	D9	7E	60	DE	C0	67	79
80	86	98	3F	21	9F	81	26	38	2E	30	97	89	37	29	8E	90
90	EC	F2	55	4B	F5	EB	4C	52	44	5A	FD	E3	5D	43	E4	FA
A0	F8	E6	41	5F	E1	FF	58	46	50	4E	E9	F7	49	57	F0	EE
B0	92	8C	2B	35	8B	95	32	2C	3A	24	83	9D	23	3D	9A	84
C0	FD	E3	44	5A	E4	FA	5D	43	55	4B	EC	F2	4C	52	F5	EB
D0	97	89	2E	30	8E	90	37	29	3F	21	86	98	26	38	9F	81
E0	83	9D	3A	24	9A	84	23	3D	2B	35	92	8C	32	2C	8B	95
F0	E9	F7	50	4E	F0	EE	49	57	41	5F	F8	E6	58	46	E1	FF

Table 11: $\mathsf{L}_{2,1}$.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	54	BE	EA	D8	8C	66	32	A5	F1	1B	4F	7D	29	C3	97
10	BF	EB	01	55	67	33	D9	8D	1A	4E	A4	F0	C2	96	7C	28
20	E5	B1	5B	0F	3D	69	83	D7	40	14	FE	AA	98	CC	26	72
30	5A	0E	E4	B0	82	D6	3C	68	FF	AB	41	15	27	73	99	CD
40	D6	82	68	3C	0E	5A	B0	E4	73	27	CD	99	AB	FF	15	41
50	69	3D	D7	83	B1	E5	0F	5B	CC	98	72	26	14	40	AA	FE
60	33	67	8D	D9	EB	BF	55	01	96	C2	28	7C	4E	1A	F0	A4
70	8C	D8	32	66	54	00	EA	BE	29	7D	97	C3	F1	A5	4F	1B
80	D8	8C	66	32	00	54	BE	EA	7D	29	C3	97	A5	F1	1B	4F
90	67	33	D9	8D	BF	EB	01	55	C2	96	7C	28	1A	4E	A4	F0
A0	3D	69	83	D7	E5	B1	5B	0F	98	CC	26	72	40	14	FE	AA
B0	82	D6	3C	68	5A	0E	E4	B0	27	73	99	CD	FF	AB	41	15
C0	0E	5A	B0	E4	D6	82	68	3C	AB	FF	15	41	73	27	CD	99
D0	B1	E5	0F	5B	69	3D	D7	83	14	40	AA	FE	CC	98	72	26
E0	EB	BF	55	01	33	67	8D	D9	4E	1A	F0	A4	96	C2	28	7C
F0	54	00	EA	BE	8C	D8	32	66	F1	A5	4F	1B	29	7D	97	C3

Table 12: $\mathsf{L}_{2,2}$.

B Round Constants

We use simple constants in order to limit their implementation cost, but expect them to avoid any kind of slide attack of self-similarity property. The round constants in Scream are defined as: $C(\rho) = 2199 \cdot \rho \bmod 2^{16}$. We consider constants of the form $C(\rho) = C \cdot \rho \bmod 2^{16}$ because they can implemented by incrementing a counter by steps of C. However, we would rather avoid the trivial choice $C = 1$ because this implies simple linear relations between the constants, such as $C(2\rho + 1) = C(2\rho) \oplus 1$. Concretely, we built 16×16 matrices with the binary representation of $C(1), C(2), \ldots, C(16)$, and computed the rank of these matrices. There are a few values that give a full rank matrix, and we decided to use the smallest value with this property: 2199.

C Analysis of differential trails

In this section we give lower bounds on the number of active S-boxes for differential trails on Scream. We use a simple notation to describe trails, where '`x`' denotes an active step, and '`-`' an inactive step. With this notation, trails using the tweak to limit the number of active steps to $\lfloor \sigma/2 \rfloor$ are denoted as '`-x-x-x-`'.

C.1 Single key, fixed tweak

With a fixed tweak, Scream is a slightly modified version of Fantomas, with a different L-box. Therefore, we can compute bounds on its number of active S-boxes for differential and linear trails following [12].

C.2 Single key, chosen tweaks

The L-box of Scream has been chosen so that trails over one step (2 rounds) with the same input and output difference pattern have at least 14 active S-boxes. Due to the structure of the tweakey scheduling, a pair of tweaks with a truncated difference δ gives tweakeys with the difference pattern δ in every round. In particular a trail '`-x-`' using a difference in the tweak leads to the same pattern of active S-boxes δ at the beginning and at the end of the second round; this implies at least 14 active S-boxes.

We also use bounds from single key trails in our analysis of related-key trails. For instance, a trail '`-xx-`' gives truncated differences $\delta \rightsquigarrow a$, $b \rightsquigarrow \delta$ assuming a difference δ in the tweakeys, and the truncated difference a must be transformed into b after a tweakey addition with difference δ. This can transformed into a single key trail over two steps: $b \rightsquigarrow \delta \rightsquigarrow a$, and we know that such a trail has at least 20 active S-boxes. Moreover, we can enumerate the trails with 20 active S-boxes, and we found than there is a single trail where the patterns a, b, and δ are compatible:

$$b = \texttt{0010011100101001} \rightsquigarrow \texttt{0100001001101000} \rightsquigarrow \texttt{1100000000000001} = \delta,$$
$$\delta = \texttt{1100000000000001} \rightsquigarrow \texttt{0010001000000111} \rightsquigarrow \texttt{1110011100101001} = a.$$

This leads to the following analysis of differential trails:

5-step trails. We can list all the possible 5-step trails with 3 active steps or less (we use symmetries to consider only half of the patterns), and compute a lower bound on the corresponding number of active S-boxes:
`-x-x-`**:** At least 28 active S-boxes.
`-x-xx`**:** At least 30 active S-boxes $(14 + 8 + 8)$.
`-xx-x`**:** At least 28 active S-boxes $(20 + 8)$.
`-xxx-`**:** At least 28 active S-boxes $(20 + 8$; 20 for steps 1 and 3 combined$)$.
`x-x-x`**:** At least 30 active S-boxes $(8 + 14 + 8)$.
If there are 4 or more active steps, this implies at least $8 \times 4 = 32$ active S-boxes.

6-step trails. Similarly, we can list all the possible 6-step trails with 4 active steps or less:
`-x-x-x`**:** At least 36 active S-boxes $(14 + 14 + 8)$.
`-x-xx-`**:** At least 34 active S-boxes $(14 + 20)$.
`-x-xxx`**:** At least 38 active S-boxes $(14 + 8 + 8 + 8)$.
`-xx-xx`**:** At least 36 active S-boxes $(20 + 8 + 8)$.
`-xxx-x`**:** At least 36 active S-boxes $(20 + 8 + 8$; 20 for steps 1 and 3 combined$)$.
`-xxxx-`**:** At least 36 active S-boxes $(20 + 8 + 8$; 20 for steps 1 and 4 combined$)$.

`x-x-xx`: At least 38 active S-boxes $(8 + 14 + 8 + 8)$.
`x-xx-x`: At least 36 active S-boxes $(8 + 20 + 8)$.
If there are 5 or more active steps, this implies at least $8 \times 5 = 40$ active S-boxes.

In addition, we verified that there is no valid trail with only 34 active S-boxes following `-x-xx-`: the only trail with weight 20 for steps 3 and 4 gives a tweak difference δ with no valid trails $\delta \rightsquigarrow \delta$ for step 1. This proves that 6-step trails have at least 35 active S-boxes.

C.3 Related keys, chosen tweaks

All tweakeys active. First, let us consider trails with a difference in all the tweakeys. Using a difference in the key and in the tweak, it could be possible to have tweakey differences with only 8 active S-boxes per step. However, such trails must still activate at least $\lfloor \sigma/2 \rfloor$ steps. Therefore, they have at least $8 \cdot \lfloor \sigma/2 \rfloor$ active S-boxes.

We can improve this bound using the property of the tweakey scheduling that the same tweakeys are used every three rounds. In particular, a trail '`-x-x-`' would have tweakey differences δ_0, δ_1, δ_2, δ_0, δ_1, δ_2, and transitions $\delta_1 \rightsquigarrow \delta_2$ and $\delta_0 \rightsquigarrow \delta_1$. This can be turned into a 2-step single key trail $\delta_0 \rightsquigarrow \delta_1 \rightsquigarrow \delta_2$ that has at least 20 active S-boxes. This implies the following bounds:

- The only 7-step trail with 3 active steps is '`-x-x-x-`'; it has at least 28 active S-boxes. Trails with 4 or more active steps have at least 32 active S-boxes.
- The only 9-step trail with 4 active steps is '`-x-x-x-x-`'; it has at least 40 active S-boxes. Trails with 5 or more active steps have at least 40 active S-boxes.

Some tweakeys inactive. The tweakey scheduling of Scream further allows to cancel some key and tweak differences. Let us now consider trails where some tweakeys are inactive.

Without loss of generality, we assume that $\delta[K] = \delta[T]$. The tweakey differences for the following rounds will be:

$$\delta[K \oplus \phi(T)] = \delta[T] \oplus \phi(\delta[T]) = \phi^2(\delta[T]),$$
$$\delta[K \oplus \phi^2(T)] = \delta[T] \oplus \phi^2(\delta[T]) = \phi(\delta[T]).$$

In particular, the truncated pattern δ will be the same for all the active tweakeys, because ϕ is computed on each state line independently.

This leads to the following analysis of trails:

2-step trails. They can have no active step if the second tweakey is inactive.

3-step trails. A 3-step trail with a single active step has at least 8 active S-boxes. This can be reached by following '`--x`'.

4-step trails. A 4-step trail with a single active step must follow '`--x-`'. The third step must have the same input and output pattern, therefore it has at least 14 active S-boxes. Trails with two or more active steps have at least 16 active S-boxes.

More generally, a trail over σ steps has at least:

- $14 \cdot \lfloor \sigma/3 \rfloor - 6$ active S-boxes if $\sigma \bmod 3 = 0$.
- $14 \cdot \lfloor \sigma/3 \rfloor$ active S-boxes otherwise.