Spook: Sponge-Based Leakage-Resilient Authenticated Encryption with a Masked Tweakable Block Cipher

Davide Bellizia*, Francesco Berti*, Olivier Bronchain*, Gaëtan Cassiers*, Sébastien Duval*, Chun Guo*, Gregor Leander†, Gaëtan Leurent‡, Itamar Levi*, Charles Momin*, Olivier Pereira*, Thomas Peters*, François-Xavier Standaert*, Friedrich Wiemer†.

Version 1, March 29, 2019.

Abstract

This document defines Spook: a sponge-based authenticated encryption with associated data algorithm. It is primarily designed to provide security against side-channel attacks at a low energy cost. For this purpose, Spook is mixing a leakage-resilient mode of operation with bitslice ciphers enabling efficient and low latency implementations. The leakage-resilient mode of operation leverages a re-keying function to prevent differential side-channel analysis, a duplex sponge construction to efficiently process the data, and a tag verification based on a Tweakable Block Cipher (TBC) providing strong data integrity guarantees in the presence of leakages. The underlying bitslice ciphers are optimized for the masking countermeasures against side-channel attacks. Spook is an efficient single-pass algorithm. It provides state-of-the-art black box security with several prominent features: (*i*) nonce misuse-resilience, (*ii*) beyond-birthday security with respect to the TBC size, and (*iii*) multi-user security at minimum cost with a public tweak.

* ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium.
† Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany.
‡ Team SECRET, Inria Paris Research Center, France.

1 Corresponding submitter

- Name: François-Xavier Standaert.
- E-mail: fstandae@uclouvain.be.
- Phone number: +3210472565.
- Organization: Université catholique de Louvain.
- Postal address: Place du Levant, 3, 1348 Louvain-la-Neuve, Belgium.

2 Design rationale and motivation

Spook is an Authenticated Encryption scheme with Associated Data (AEAD). Its primary design goals are *resistance against side-channel analysis* and *low-energy implementations* (jointly).

The motivation for the first goal stems from the observation that lightweight devices may be deployed in environments where they can be under physical control of an adversary, yet be responsible for sensitive tasks, or be the root of critical distributed attacks starting from seemingly non-critical connected objects [RSWO18]. As a result, the ability to provide side-channel resistance (and possibly resistance against fault attacks) easily and at low cost was identified by the NIST as a desirable feature for lightweight cryptography.[1] The motivation for the second goal stems from the observation that energy is a suitable metric to compare the performances of cryptographic algorithms [KDH+12], and a relevant one from the application viewpoint. It is in particular increasingly needed for battery-operated / energy harvesting devices, for example in the IoT [MMGD17].

In order to reach these goals, Spook builds on and specializes two main ingredients.

The first ingredient is a leakage-resilient mode of operation that enables efficient side-channel resistance. We use the S1P mode of operation for this purpose [GPPS19], which stands for "Sponge One-Pass" and is the lightweight variation of a sequence of works aiming at high-physical security guarantees for (authenticated) encryption [PSV15, BKP+18, BPPS17, BGP+19]. For integrity, S1P reaches the top of the definitions' hierarchy established in [GPPS18], namely Ciphertext Integrity with Misuse and Leakage in encryption and decryption (CIML2), in a liberal model where all the intermediate computations are leaked to the adversary, except for a long-term secret key that is only used twice per encrypted or decrypted message. For confidentiality, S1P reaches security against Chosen Ciphertext Adversaries with misuse-resilience and Leakage in encryption (CCAmL1). Compared to related works with constructions additionally achieving CCA security with decryption leakages (i.e., CCAmL2 [GPPS18]), the S1P mode has the significant advantage of being *single-pass* in encryption and in decryption, which we believe is essential for lightweight implementations.[2] Concretely, S1P also encourages so-called *leveled implementations*, where (expensive) protections against side-channel attacks are used in a limited way and independent of the message size, while the bulk of the computation can be executed by cheap and weakly protected components.

The second ingredient is the adoption of regular symmetric primitives to operate the S1P mode of operation, namely the Clyde-128 Tweakable Block Cipher (TBC) and the Shadow-512 permutation, both based on simple extensions of the LS-design framework, which aims at efficient bitslice implementations [GLSV14]. In order to facilitate leveled implementations, those primitives use components that can be efficiently masked against side-channel attacks for the TBC (e.g., with [GMK17] in hardware or [GR17] in software), and enable fast implementations for the permutation. They bring two main improvements compared to earlier proposals of LS-designs. On

[1] https://csrc.nist.gov/projects/lightweight-cryptography.

[2] We consider the definition of misuse-resilience of Ashur et al. [ADL17] rather than the definition of misuse-resistance of Rogaway and Schrimpton [RS06] for a similar reason (i.e., in order to avoid the need of two passes).

the one hand, they leverage the tools introduced by Beierle et al. [BCLR17] in order to prevent the invariant attacks that put several earlier LS-designs at risk [LMR15, TLS16]. On the other hand, they replace the table-based L-boxes used in previous LS-designs by word-level L-boxes that can be efficiently implemented as a sequence of rotation and XOR operations, which is beneficial to prevent cache attacks [TOS10]. As a result, both Clyde-128 and Shadow-512 enable efficient bitslicing and side-channel resistant implementations on a wide range of platforms, (e.g., 32-bit microprocessors such as increasingly used in mobile applications and dedicated hardware or FPGAs).

The motivations for using two symmetric primitives in S1P are twofold. First, an invertible (tweakable) block cipher is instrumental to reach CIML2 security in the unbounded leakage model [BPPS17]. Second, duplex sponge constructions are in general attractive for efficient AE: they can achieve this functionality in a single pass, are highly flexible and ensure nice security bounds in the multi-user setting [BDPA11, DMA17]. Sponge constructions are also believed to provide some level of leakage-resilience by design [DEM+17]. Spook combines the advantages of both.

Eventually, and besides these main features, Spook inherits other interesting properties from the S1P mode of operation: (i) it is secure beyond the birthday bound (with respect to the size n of the TBC), and (ii) it can provide n-bit multi-user security at low cost with a public tweak.

3 Specifications

3.1 The S1P mode of operation

Notations. We denote the plaintext as $\boldsymbol{M}$. It is parsed into ℓ blocks $M[0], M[1], \ldots, M[\ell-1]$, where the size of blocks 0 to $\ell-2$ is r and the size of the last block is $1 \leq |M[\ell-1]| \leq r$. We denote the associated data as $\boldsymbol{A}$. It is parsed into λ blocks $A[0], A[1], \ldots, A[\lambda-1]$ in the same way as the plaintext. We denote the τ-bit nonce as N and the key as $K||P$, where K is a long-term secret key of n bits, and P is a public tweak of $n-1$ bits.[3] The secret key K has to be picked up uniformly at random in $\{0,1\}^n$. The public tweak P is set to an $(n-1)$-bit zero vector in case only single-user security is requested, and is set to $P = p||1$ in case multi-user security is requested (i.e., one bit is used to separate the single-user and multi-user security variants). In case multi-user security is requested, p plays the role of a long-term "public key". It is recommended to pick it up uniformly at random like the long-term secret key K. The $\mathsf{S1P}[\mathsf{E}, \pi](\boldsymbol{A}, \boldsymbol{M}, N, K||P)$ mode of operation relies on an Tweakable Block Cipher (TBC) with n-bit blocks, tweaks and keys, denoted as E, and an $(r+c)$-bit permutation π. Our primary parameters are $n = 128$, $r = 256$, $c = 256$ and $\tau = 128$.

Conventions. S1P operates over bitstrings (i.e., each of the manipulated data – the plaintext, associated data, ciphertext, keys and nonce – is a sequence of bits). The Spook cipher is however defined for bytestrings (i.e., each of the manipulated data is a sequence of bytes). For encryption, input data (i.e., plaintext, associated data, keys, nonce) bytestrings are first mapped to bitstrings using the BMAP function defined next, and the ciphertext is converted back to a bytestring using the inverse of the BMAP function. The operations are the same for decryption, except that the plaintext and ciphertext are swapped. BMAP maps bytes to bits in little-endian order. More precisely, it takes as input a sequence of bytes of length q: $(X[0], \ldots, X[q-1])$ and outputs a sequence of bits $(Y[0], \ldots, Y[8q-1])$, where $Y[8i+j] = (X[i]/2^j) \mod 2$ for $0 \leq i < q$ and $0 \leq j < 8$.

As a result, the nonce N, the private part of the key K and (when applicable) the public part of the key p are all 16 bytes long. In order to get the bitstring p (which has a length of 126 bits) from the corresponding bytestring, the last two bits are discarded after application of BMAP.

[3] So in our reference implementations, $K||P$ is the key input string required by the NIST API.

The encryption. As illustrated in Figure 1, the encryption of the 4-string input $(\boldsymbol{A}, \boldsymbol{M}, N, K||P)$ first derives an n-bit initial seed B by using a TBC call $\mathsf{E}_K^{P||0}(N||0^*)$. The initial seed B is then used as a fresh key for an inner keyed duplex sponge construction, to process $\boldsymbol{A}$ and $\boldsymbol{M}$ and produce $\boldsymbol{C}$. Two bits are used for domain separation, in order to distinguish $\boldsymbol{M}$ from $\boldsymbol{A}$ and mark if the last blocks of $\boldsymbol{A}$ and $\boldsymbol{M}$ are of full r bits or not. Let $U||V$ be the first $2n-1$ bits of the final state, with $|U| = n$. The tag Z is produced by using another TBC call $\mathsf{E}_K^{V||1}(U)$, where the 1 concatenated with V guarantees that this tweak is different from the one used to generate B. The ciphertext is made of $\ell - 1$ blocks of r bits, a final block of length $1 \leq |C[\ell-1]| \leq r$ and an n-bit tag. We next denote it as $\boldsymbol{C} := \boldsymbol{c}||Z := C[0]||\ldots||C[\ell-1]||Z$ (i.e., $\boldsymbol{c}$ is the ciphertext excluding the tag).

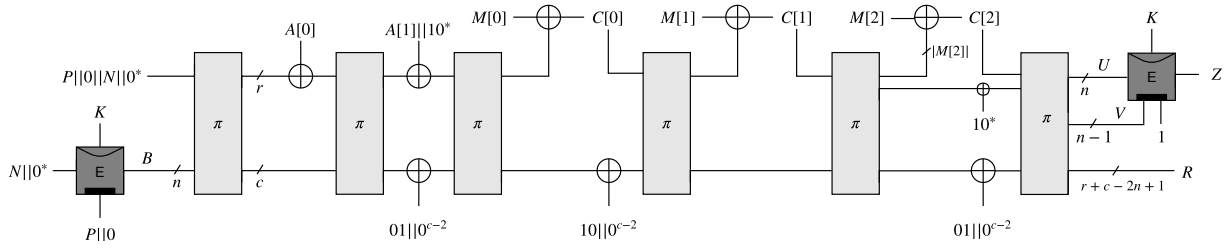


Figure 1: S1P mode with TBC E and permutation π, applied to a 2-block associated data and a 3-block message. The value $01||0^{c-2}$ is inserted only if $|A[\lambda-1]| < r$ (resp., $|M[\ell-1]| < r$)).

The decryption. In order to decrypt the 4-string input $(\boldsymbol{A}, \boldsymbol{C}, N, K||P)$, the mode first derives the initial seed B via $\mathsf{E}_K^{P||0}(N||0^*)$, as when encrypting. It then runs the inner keyed duplex sponge construction on $\boldsymbol{A}$ and $\boldsymbol{c}$ to derive $\boldsymbol{M}$ and the $(2n-1)$-bit truncated state $U||V$. Finally, it makes an inverse TBC call $U^* = \left(\mathsf{E}_K^{V||1}\right)^{-1}(Z)$, and outputs $\boldsymbol{M}$ if and only if there is a match $U^* = U$.[4]

More precisely, $\mathsf{S1P}[\mathsf{E}, \pi].\mathsf{Enc}$ and $\mathsf{S1P}[\mathsf{E}, \pi].\mathsf{Dec}$ are specified in Appendix A, Algorithms 3 and 4. The different cases that the S1P mode can encounter are also illustrated in Appendix B.

3.2 Clyde-128, a Tweakable LS-Design

The S1P mode of operation requires a TBC. We use the Tweakable LS-Design (TLS-design) framework introduced as part of the SCREAM authenticated encryption candidate to the CAESAR competition for this purpose [GLS+14]. TLS-designs are tweakable variants of the LS-designs which specify a family of bitslice ciphers aimed at efficient masked implementations [GLSV14].

Such ciphers work on $n = (s \cdot l)$-bit states, where s is the size of the S-box and l is the size of the L-box. We denote the full cipher state as x, a state row as $x[i, \star]$ $(0 \leq i < s)$ and a state column as $x[\star, j]$ $(0 \leq j < l)$. Concretely, we will consider $s = 4$ and $l = 32$. Although the internal representation of the data is a $(s \cdot l)$-bit matrix, the cipher operates over bitstring inputs and outputs. The mapping between a bitstring B and the corresponding bit matrix x is $x[i, j] = B[i \cdot l + j]$. From an implementation viewpoint, the S-boxes and L-boxes are defined such that they can always be executed thanks to simple operations on the rows (typically corresponding to processor words).

In summary, TLS-designs update the n-bit state x by iterating N_s steps, each of them made of two rounds (so $N_r = 2N_s$). One significant advantage of these designs is their simplicity: they can be described in few lines, as illustrated in Algorithm 1, where μ denotes the plaintext, TK a combination of the master key K and tweak T that we call tweakey [JNP14], $W(r)$ are round constants, and S and L are an s-bit S-box and an l-bit L-box (specified next).[5]

[4] In this way, invalid decryption only leaks meaningless random values U^*, instead of the correct tags.

[5] For regularity (in hardware implementations), we keep the linear L-box in the last round.

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

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

We use SCREAM's lightweight tweakey scheduling algorithm [GLS+14]. It takes the n-bit key K and the n-bit tweak T as input. The tweak is divided into $n/2$-bit halves: $T = t_0 \| t_1$. Then, three different tweakeys are used every three steps as follows:

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

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 $\phi^3(x) = x$):

$$\begin{aligned} \phi : x_0 \| x_1 &\mapsto (x_0 \oplus x_1) \| x_0, \\ \delta_0 &= T, \\ \delta_{i+1} &= \phi(\delta_i), \\ TK(i) &= K \oplus \delta_i. \end{aligned}$$

3.3 Shadow-512, a Multiple LS-Design

The S1P mode of operation also requires a (larger) permutation. We use a simple variant of the LS-designs that we denote as mLS-designs (standing for multiple LS-designs) for this purpose. In summary, mLS-designs mix multiple LS-designs thanks to an additional diffusion layer.

Such ciphers work on $n = (m \cdot s \cdot l)$-bit states, where m is the number of LS-designs considered, s is the size of the S-box and l is the size of the L-box. Taking similar notations as for TLS-designs, we denote the full cipher state as x, each $(s \cdot l)$-bit substate corresponding to an LS-design as a bundle $x[b, \star, \star]$ $(0 \leq b < m)$, a bundle row as $x[b, i, \star]$ $(0 \leq i < s)$ and a bundle column as $x[b, \star, j]$ $(0 \leq j < l)$. Concretely, we will consider $m = 4$, $s = 4$ and $l = 32$. Although the internal representation of the data is an $(m \cdot s \cdot l)$-bit state, the cipher operates over bitstring inputs and outputs. The mapping between a bitstring B and the corresponding state x is $x[b, i, j] = B[b \cdot l \cdot s + i \cdot l + j]$.

In summary, mLS-designs update the n-bit state x by iterating N_s steps, each of them made of two rounds (round A and round B) that respectively apply an L-box to the rows of each bundle independently, and a diffusion layer mixing the rows of different bundles (on top of the S-box layer). An accurate description is given in Algorithm 2, where μ denotes the input, $W(r)$ are round constants, S and L are an s-bit S-box and an l-bit L-box and D is a m-bit diffusion layer.

Algorithm 2 mLS-design with l-bit L-boxes and s-bit S-boxes ($n = m \cdot s \cdot l$)

$x \leftarrow \mu$; ▷ x is a $m \times s \times l$ bits matrix
for $0 \leq \sigma < N_s$ **do**
  for $0 \leq b < m$ **do** ▷ Round A
    for $0 \leq j < l$ **do**
      $x[b, \star, j] = \mathsf{S}(x[b, \star, j])$; ▷ S-box Layer
    for $0 \leq i < s$ **do**
      $x[b, i, \star] = \mathsf{L}(x[b, i, \star])$; ▷ L-box Layer
  $x \leftarrow x \oplus W(2 \cdot \sigma)$; ▷ Constant addition
  for $0 \leq b < m$ **do** ▷ Round B
    for $0 \leq j < l$ **do**
      $x[b, \star, j] = \mathsf{S}(x[b, \star, j])$; ▷ S-box Layer
  for $0 \leq i < s$ **do**
    for $0 \leq j < l$ **do**
      $x[\star, i, j] = \mathsf{D}(x[\star, i, j])$; ▷ Diffusion Layer
  $x \leftarrow x \oplus W(2 \cdot \sigma + 1)$; ▷ Constant addition
return x

3.4 Clyde-128 and Shadow-512 components

We now describe the components S, L and D and the round constants used in Clyde-128 and Shadow-512. Both ciphers are designed to enable simple implementations based on 32-bit word-level operations. For the S-box, we provide its circuit representation (which can be applied in parallel to the 32 bits of a word). For the L-box and diffusion layer, we provide a sequence of 32-bit operations. We denote the bitwise AND as $\odot$ and the left rotation of a word x by α bits as $\mathsf{rot}(x, \alpha)$.

S-box. We use a variant of the 4-bit S-box used in Skinny [BJK+16], modified by replacing the NOR gates by AND gates. It is given in Table 1, with numbers representing bitstrings encoded in little-endian. That is, $x = \sum_{i=0}^{3} 2^i \cdot x[i]$ and $S(x) = \sum_{i=0}^{3} 2^i \cdot y[i]$. It has linear and differential

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathsf{S}(x)$	0	8	1	15	2	10	7	9	4	13	5	6	14	3	11	12

Table 1: S-box in table representation.

probabilities 2^{-2} and algebraic degree 3. Concretely, $y = \mathsf{S}(x)$ can be implemented serially with 4 AND gates and 4 XOR gates in the direct and inverse directions. In the direct sense, it has an AND depth of two and allows computing the two first (and two last) AND gates in parallel:

- $y[1] = (x[0] \odot x[1]) \oplus x[2]$;
- $y[0] = (x[3] \odot x[0]) \oplus x[1]$;
- $y[3] = (y[1] \odot x[3]) \oplus x[0]$;
- $y[2] = (y[0] \odot y[1]) \oplus x[3]$;

An inverse implementation (with 4 AND gates & 4 XOR gates) is given in Appendix C.

L-box. We use an interleaved L-box that applies jointly to pairs of 32-bit words and has branch number 16 over those pairs. Denoting the two words on which it is applied as x and y:

$$(a, b) = \mathsf{L}'(x, y) = \begin{pmatrix} \mathsf{circ}(\texttt{0xec045008}) \cdot x^{\mathsf{T}} \oplus \mathsf{circ}(\texttt{0x36000f60}) \cdot y^{\mathsf{T}} \\ \mathsf{circ}(\texttt{0x1b0007b0}) \cdot x^{\mathsf{T}} \oplus \mathsf{circ}(\texttt{0xec045008}) \cdot y^{\mathsf{T}} \end{pmatrix},$$

where circ denotes the circulant matrix whose first line is given in hexadecimal notation, so that the number $b = \sum_{i=0}^{31} 2^i b_i$ corresponds to the row vector $(b_0, \ldots, b_{31})$. Concretely, this L-box can be efficiently implemented (in the direct and inverse directions) thanks to six word-level (left) rotations and six 32-bit XORs per word as follows:

- $a = x \oplus \mathsf{rot}(x, 12)$;
- $b = y \oplus \mathsf{rot}(y, 12)$;
- $a = a \oplus \mathsf{rot}(a, 3)$;
- $b = b \oplus \mathsf{rot}(b, 3)$;
- $a = a \oplus \mathsf{rot}(x, 17)$;
- $b = b \oplus \mathsf{rot}(y, 17)$;
- $c = a \oplus \mathsf{rot}(a, 31)$;
- $d = b \oplus \mathsf{rot}(b, 31)$;
- $a = a \oplus \mathsf{rot}(d, 26)$;
- $b = b \oplus \mathsf{rot}(c, 25)$;
- $a = a \oplus \mathsf{rot}(c, 15)$;
- $b = b \oplus \mathsf{rot}(d, 15)$;

The inverse implementation is in Appendix D. Note that this L-box requires a minor modification of the TLS-designs and mLS-designs in Sections 3.2 and 3.3. For TLS-designs, the loop:

for $0 \leq i < s$
  $x[i, \star] = \mathsf{L}(x[i, \star])$;

applying L independently on each word, has to be replaced by the following one:

for $0 \leq i < s/2$
  $(x[2i, \star], x[2i+1, \star]) = \mathsf{L}'(x[2i, \star], x[2i+1, \star])$;

A similar change applies to mLS-designs. The L' notation reflects this application to pairs of words. In LS-designs, such interleaved L-boxes are only applicable to S-boxes with even number of bits.

Diffusion layer (for Shadow-512 only). We use the diffusion layer of the low-energy cipher Midori [BBI+15], which is based on a near-MDS 4×4 matrix defined as follows:

$$(a, b, c, d) = \mathsf{D}(w, x, y, z) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}.$$

It has branch number 4 (an MDS diffusion would provide 5), and can be efficiently implemented with six 32-bit XORs as a circuit with gate depth 2 as follows:

- $u = w \oplus x$;
- $v = y \oplus z$;
- $a = x \oplus v$;
- $b = w \oplus v$;
- $c = u \oplus z$;
- $d = u \oplus y$;

Round constants. Round constants for Clyde-128 are generated from a 4-bit LFSR. Each state of the LFSR is used as the constant for a single round. The four bits are XORed with the first bit of the four state rows. Precisely, the round constants are:

- Round 0: (1,0,0,0); • Round 1: (0,1,0,0); • Round 2: (0,0,1,0); • Round 3: (0,0,0,1);
- Round 4: (1,1,0,0); • Round 5: (0,1,1,0); • Round 6: (0,0,1,1); • Round 7: (1,1,0,1);
- Round 8: (1,0,1,0); • Round 9: (0,1,0,1); • Round 10: (1,1,1,0); • Round 11: (0,1,1,1);

For Shadow-512, we take exactly the same constants, but for each bundle $b = 0, \ldots, m-1$, we add the round constant on the bth bit of the four bundle rows, that is $x[b, \star, b]$.

4 Security analysis and claims

Our claims are in the single-key setting. Related-key attacks should be avoided at the protocol level.

4.1 The S1P mode of operation

The black box security analysis of S1P is proven under the assumptions that the TBC is a secure tweakable strong pseudo-random permutation and that the permutation is a random permutation. CIML2 is proven under the additional assumption that the long-term key of the TBC cannot be leaked (but all other intermediate values can be leaked in full). CCAmL1 security is proven under a bounded leakage assumption. We refer to [GPPS19] for details on these assumptions and proofs.

Based on the above, the single-user security claims of the mode are summarized in Table 2. These bounds imply that the leakage security of Spook depends on the concrete strength of its implementation. Informally, the CIML2 bound guarantees that message integrity reduces to the side-channel security of the TBC. The best forgery attack is a Differential Power Analysis (DPA) against Clyde-128 – the complexity of which is expected to be smaller than $\frac{2^n}{n^2}$. The CCAmL1 analysis is more subtle, but essentially guarantees that the confidentiality of long messages reduces to the confidentiality of single-block messages. The best attacks are then a DPA against Clyde-128 (as for CIML2) and a Simple Power Analysis (SPA) against the ephemeral secrets (i.e., the secret part of the permutation state), the complexity of which is expected to be smaller than $2^{n/2}$.[6]

The security claims for the multi-user variant of S1P are summarized in Table 3.
Note that no additional restrictions are imposed on the message length. The security bounds in both tables only depend on the total number of message and associated data blocks to encrypt.

[6] The birthday bound for CCAmL1 could possibly be improved. There is no matching attack we are aware of.

Security model	security (bits)
Plaintext confidentiality with nonce misuse-resilience (mR)	$n - \log n$
Ciphertext integrity with misuse-resistance (MR) but no leakage	$n - \log n$
Plaintext confidentiality with encryption leakages and mR	$\approx n/2$
Ciphertext integrity with full leakages and MR	$\approx n - \log n$

Table 2: Single-user security claims.

Security model	security (bits)	# of users
Plaintext confidentiality with nonce misuse-resilience (mR)	$n - 2\log n$	$\approx 2^{n-2}$
Ciphertext integrity with misuse-resistance (MR) but no leakage	$n - 2\log n$	$\approx 2^{n-2}$
Plaintext confidentiality with encryption leakages and mR	$\approx n/2$	$\approx 2^{n-2}$
Ciphertext integrity with full leakages and MR	$\approx n - 2\log n$	$\approx 2^{n-2}$

Table 3: Multi-user security claims.

4.2 The Clyde-128 (tweakable) block cipher

The security of Clyde-128 against linear and differential attacks can be analyzed thanks to the wide-trail strategy [DR01]. Two rounds activate 16 S-boxes and the linear/differential probability of our S-box is 2^{-2}. As a result, eight rounds (four steps) lead to a bound on the probability of the best linear/differential characteristics of $(2^{-2})^{4 \cdot 16} = 2^{-128}$. Our recommended parameters add four rounds (two steps) in order to prevent improvements of these standard attacks.

According to the upper bound in [BCC11], at least five rounds of Clyde-128 are necessary to reach the maximum algebraic degree. We expect that the recommended twelve rounds (six steps) should prevent risks of algebraic cryptanalysis [CP02] and related attacks (e.g., cube [DS09] or division property [Tod15]) – our experiments in this regard did not reveal any weaknesses.

Besides, and following the security arguments recently put forward in [BCLR17], the round constants are chosen so as to maximize the dimension of the smallest invariant subspace over the linear layer that contains all round constants. To achieve this, we need at least ten rounds. This ensures that no invariants exist simultaneously for the S-box layer and the L-box layer.

Note that while Clyde-128 is built from the TLS-designs introduced in [GLS+14] and analyzed as an ideal TBC in [GPPS19], the way it is used in Spook implies that its tweak input is either constant (as a zero vector or a public value) or pseudo-random and out of adversarial control. So while a standard TBC would require security against chosen-tweak attacks, the number of rounds selected for Clyde-128 only corresponds to single-key and random-tweak security, which is the minimum required for the analysis of the S1P mode of operation to hold. Chosen-tweak security for Clyde-128 could be obtained by doubling the number of rounds, following the approach in [GPPR11].

4.3 The Shadow-512 permutation

The exact requirements for the Shadow-512 permutation are more difficult to specify.

A minimum is to reach 128-bit security against linear cryptanalysis. This can be analyzed by considering the super S-box structure of Shadow-512. Two rounds activate 16 S-boxes and four rounds activate 16×4 S-boxes thanks to the branch number of the diffusion layer. Hence, a probability bound of 2^{-128} for the best linear characteristic is reached after four rounds.

Another minimum requirement is to reach an algebraic degree 128. According to the upper bound in [BCC11], this can be reached after at least five rounds of Shadow-512.

Besides, one important requirement for the permutation in the analysis of the S1P mode of operation is that it ensures collision resistance for the 255 bits that are used to generate the tag. Hence, a more specific requirement is to prevent truncated differentials with probability larger than 2^{-128} for those 255 bits. A conservative heuristic for this purpose is to require that no differential characteristic has probability better than 2^{-385}, which happens after twelve rounds (six steps).

5 Primary candidate and variants

Underlying primitives. We consider two sets of parameters for the Clyde-128 TBC and Shadow-512 permutation. The *recommended parameters* are 12 rounds for Clyde-128 and 12 rounds for Shadow-512. We additionally provide *aggressive parameters*, with 12 rounds for Clyde-128 and 8 rounds for Shadow-512, as an interesting target for cryptanalysis. Note that our reference implementations and test vectors are based on the recommended parameters.

Full algorithm. We denote as Spook$[128, 512, \mathsf{su}]$ the AEAD algorithm operating the S1P mode in the single user setting with Clyde-128 as TBC and Shadow-512 as permutation, and as Spook$[128, 512, \mathsf{mu}]$ its multi-user version. Based on these notations, we define our:

- **Primary candidate** as Spook$[128, 512, \mathsf{su}]$ with recommended parameters.
- **First variant** as Spook$[128, 512, \mathsf{mu}]$ with recommended parameters.

We recall that the only difference between the single-user and multi-user versions is that the public tweak p is stuck at zero in the first case (i.e., the key is limited to 128 secret bits), and picked up at random in the second one (i.e., the key is made of 128 secret bits and 126 public bits).

We additionally define two versions of Spook with a 384-bit state. They are obtained by turning the 512-bit permutation into a 384-bit one. We do so by defining Shadow-384 as a 3LS-design (rather than a 4LS-design) where the diffusion layer $(a, b, c) = \mathsf{D}(x, y, z)$ is specified as:

- $a = x \oplus y \oplus z$;
- $b = x \oplus z$;
- $c = x \oplus y$;

The rest of the permutation and all the other elements of the mode are adapted so that $r = 128$, with the same number of rounds for the recommended and aggressive parameters, leading to our:

- **Second variant** as Spook$[128, 384, \mathsf{su}]$ with recommended parameters.
- **Third variant** as Spook$[128, 384, \mathsf{mu}]$ with recommended parameters.

6 Design trade-offs: advantages and limitations

Spook is an AEAD algorithm with state-of-the-art guarantees in the black box setting. It ensures beyond-birthday security with respect to the size of its long-term key, can be extended to multi-user security with a public tweak, and provides nonce misuse-resilience in the sense of Ashur et al [ADL17]. Thanks to its one-pass structure, Spook should allow efficient implementations on a wide range of platforms. Its design is in particular well-suited to 32-bit software implementations (thanks to an intensive exploitation of 32-bit word-level operations), and to dedicated hardware and FPGA implementations (thanks to the low gate complexity of its underlying components).

Spook provides excellent opportunities to mitigate physical attacks efficiently thanks to its leakage-resilient features. In particular, the general rationale behind its design enables leveled implementations, where the Clyde-128 TBC is well protected against side-channel attacks and the Shadow-512 (or Shadow-384) permutation is implemented with cheaper protections (or even no protections at all). It is in the specific contexts where physical attacks are an important concern that Spook is expected to exhibit significant performance gains compared to modes without leakage-resilient features. Concretely, protecting the TBC can be achieved thanks to the masking countermeasure, both in hardware [GMK17] and in software [GR17]. For this purpose, Clyde-128 is designed both with low AND complexity (as previous LS-designs) and low AND depth (which is important to limit the latency of so-called glitch-free implementations [NRS11, FGP+18]). As for the permutation, low-latency / low-energy implementations in the sense of [KDH+12] are natural candidates in hardware, while some minimum countermeasures to prevent SPA (e.g., low-order masking, or time randomization [VMKS12]) should be sufficient in software. For this purpose, the Shadow-512 (or Shadow-384) permutation is designed with low-latency components. Leveled implementations of Spook can also benefit from pre-computing the (expensive) generation of fresh seeds.

The main price to pay for the leakage-resilient features of Spook is that it suffers from some overheads in case of short messages. This seems unavoidable in any mode leveraging a re-keying process. However, and as evaluated in [BGP+19], these overheads are amortized as soon as the data to process is a few blocks long, and the gains of leveled implementations can reach factors 10 to 100 (e.g., in energy) if a high physical security level is required by an application.

A secondary drawback is the need of two primitives (a TBC and a permutation), which implies a larger cost (i.e., area) in hardware. However, this drawback vanishes for the intended case studies, since leveled implementations require implementations with different physical security levels anyway. Also, in case uniformly (un)protected implementations are considered, the use of the same S-box and L-box in Clyde-128 and Shadow-512 (or Shadow-384) should allow resource sharing.

Eventually, we list a couple of additional interesting features of Spook.

First, the S1P mode is compatible with solutions for the encryption of long messages segmented into several smaller packets, as for example proposed by Bertoni et al. [BDPA11] and formalized by Hoang et al. [HRRV15]. Such a "session feature" can be used as a partial tagging mechanism which allows the decryption of long messages when only a limited memory is available (i.e., smaller than the size of the message), and saves the execution of one TBC per segment (i.e., the highly protected part and therefore more expensive part in a leveled implementation of S1P). As such modes are not directly compatible with the NIST API, we leave their discussion for a separate report.

Second, and since leveraging a re-keying process, the Spook algorithm inherently provides good resistance against various Differential Fault Attacks, as discussed in [MSGR10, DEM+17]).

Finally, an inverse-free variant of Spook can be obtained by doing the tag verification in the direct sense. It only satisfies CIML1 in the unbounded leakage model, yet can provide good concrete security against bounded leakages if the tag verification is sufficiently protected (e.g., masked).

Acknowledgments. Gaëtan Cassiers, Thomas Peters and François-Xavier Standaert are respectively PhD Student, Post-Doctoral Researcher and Senior Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by EU and the Walloon Region through the ERC Project 724725 (SWORD), the FEDER Project USERMedia (convention 501907-379156), the H2020 project REASSURE and the Wallinov TRUSTEYE project.

References

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting Authenticated Encryption Robustness with Minimal Modifications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, 2017.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.

[BCLR17] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *LNCS*, pages 647–678. Springer, 2017.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a Leakage-Resilient AEAD Mode for High (Physical) Security Applications. *IACR Cryptology ePrint Archive*, 2019:137, 2019.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 37–50. ACM, 2018.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[CP02] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *LNCS*, pages 267–287. Springer, 2002.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - Towards Side-Channel Secure Authenticated Encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.

[DMA17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[DR01] Joan Daemen and Vincent Rijmen. The Wide Trail Design Strategy. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.

[DS09] Itai Dinur and Adi Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.

[FGP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.

[GLS+14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, rançois Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM & iSCREAM Side-Channel Resistant Authenticated Encryption with Masking, 2014.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *LNCS*, pages 18–37. Springer, 2014.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, volume 10159 of *LNCS*, pages 95–112. Springer, 2017.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.

[GPPS18] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-Resilient Authenticated Encryption with Misuse in the Leveled Leakage Setting: Definitions, Separation Results, and Constructions. *IACR Cryptology ePrint Archive*, 2018:484, 2018.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards Lightweight Side-Channel Security and the Leakage-Resilience of the Duplex Sponge. *IACR Cryptology ePrint Archive*, 2019:133, 2019.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *LNCS*, pages 567–597, 2017.

[HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *LNCS*, pages 493–517. Springer, 2015.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *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 II*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014.

[KDH+12] Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *LNCS*, pages 390–407. Springer, 2012.

[LMR15] Gregor Leander, Brice Minaud, and Sondre Rønjom. A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9056 of *LNCS*, pages 254–283. Springer, 2015.

[MMGD17] Elodie Morin, Mickael Maman, Roberto Guizzetti, and Andrzej Duda. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. *IEEE Access*, 5:7097–7114, 2017.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *LNCS*, pages 279–296. Springer, 2010.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *LNCS*, pages 373–390. Springer, 2006.

[RSWO18] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Security & Privacy*, 16(1):54–62, 2018.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *LNCS*, pages 3–33, 2016.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.

[TOS10] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient Cache Attacks on AES, and Countermeasures. *J. Cryptology*, 23(1):37–71, 2010.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *LNCS*, pages 740–757. Springer, 2012.

A S1P mode of operation: specifications

Notation. For a bitstring $S = b_0 \dots b_{m-1}$, we denote the bitstring of first bits $b_0 \dots b_{x-1}$ as $S[: x]$ and we denote the bitstring of last bits $b_x \dots b_{m-1}$ as $S[x :]$.

Algorithm 3 $\mathsf{S1P}[\mathsf{E}, \pi].\mathsf{Enc}(\boldsymbol{A}, \boldsymbol{M}, N, K||P)$.

1. $\ell \leftarrow \lceil |\boldsymbol{M}|/r \rceil$, $\lambda \leftarrow \lceil |\boldsymbol{A}|/r \rceil$;
2. Parse $\boldsymbol{M}$ as $M[0] \| \dots \| M[\ell-1]$, with $|M[0]| = \dots = |M[\ell-2]| = r$ and $1 \leq |M[\ell-1]| \leq r$;
3. Parse $\boldsymbol{A}$ as $A[0] \| \dots \| A[\lambda-1]$, with $|A[0]| = \dots = |A[\lambda-2]| = r$ and $1 \leq |A[\lambda-1]| \leq r$;
4. $B \leftarrow \mathsf{E}_K^{P\|0}(N\|0^*)$;
5. $IV \leftarrow P\|0\|N\|0^*$ (with size $r + c - n$);
6. $S_0 \leftarrow \pi(IV\|B)$;
7. **if** $\lambda \geq 1$ **then**
   (a) **for** $i = 0$ **to** $\lambda - 2$ **do**
      - $S_i \leftarrow S_i \oplus (A[i]\|0^c)$;
      - $S_{i+1} \leftarrow \pi(S_i)$;

   (b) **if** $|A[\lambda-1]| < r$ **then**
      - $A[\lambda-1] \leftarrow A[\lambda-1]\|10^{r-|A[\lambda-1]|-1}$;
      - $S_{\lambda-1} \leftarrow S_{\lambda-1} \oplus (0^r\|01\|0^{c-2})$;

   (c) $S_{\lambda-1} \leftarrow S_{\lambda-1} \oplus (A[\lambda-1]\|0^c)$;
   (d) $S_\lambda \leftarrow \pi(S_{\lambda-1})$;
8. **if** $\ell \geq 1$ **then**
   (a) $S_\lambda \leftarrow S_\lambda \oplus (0^r\|10\|0^{c-2})$;
   (b) **for** $i = 0$ **to** $\ell - 2$ **do**
      - $j \leftarrow i + \lambda$;
      - $C[i] \leftarrow S_j[: r] \oplus M[i]$;
      - $S_j \leftarrow C[i]\|S_j[r :]$;
      - $S_{j+1} \leftarrow \pi(S_j)$;

   (c) $C[\ell-1] \leftarrow S_{\lambda+\ell-1}[: |M[\ell-1]|] \oplus M[\ell-1]$;
   (d) **if** $|C[\ell-1]| < r$ **then**
      - $S_{\lambda+\ell-1} \leftarrow S_{\lambda+\ell-1} \oplus (0^{|C[\ell-1]|}\|10^{r-|C[\ell-1]|-1}\|01\|0^{c-2})$;
      - $S_{\lambda+\ell-1} \leftarrow C[\ell-1]\|S_{\lambda+\ell-1}[|C[l-1]| :]$;

   (e) **else** $S_{\lambda+\ell-1} \leftarrow C[\ell-1]\|S_{\lambda+\ell-1}[r :]$;
   (f) $S_{\lambda+\ell} \leftarrow \pi(S_{\lambda+\ell-1})$;
9. $U||V \leftarrow S_{\lambda+\ell}[: 2n-1]$;
10. $Z \leftarrow \mathsf{E}_K^{V||1}(U)$;
11. $\boldsymbol{c} \leftarrow C[0]\| \dots \|C[\ell-1]$, $\boldsymbol{C} \leftarrow \boldsymbol{c}||Z$;
12. **return** $\boldsymbol{C}$;

Algorithm 4 $\mathsf{S1P}[\mathsf{E},\pi].\mathsf{Dec}(\boldsymbol{A},\boldsymbol{C},N,K||P)$.

1. $\ell \leftarrow \lceil \frac{|C|-n}{r} \rceil$, $\lambda \leftarrow \lceil |A|/r \rceil$;
2. Parse $\boldsymbol{C}$ as $C[0]\| \ldots \|C[\ell-1]||Z$, with $|C[0]| = \ldots = |C[\ell-2]| = r$, $1 \leq |C[\ell-1]| \leq r$ and $|Z| = n$;
3. Parse $\boldsymbol{A}$ as $A[0]\| \ldots \|A[\lambda-1]$, with $|A[0]| = \ldots = |A[\lambda-2]| = r$ and $1 \leq |A[\lambda-1]| \leq r$;
4. $B \leftarrow \mathsf{E}_K^{P\|0}(N\|0^*)$;
5. $IV \leftarrow P\|0\|N\|0^*$ (with size $r+c-n$);
6. $S_0 \leftarrow \pi(IV\|B)$;
7. **if** $\lambda \geq 1$ **then**
   (a) **for** $i = 0$ **to** $\lambda - 2$ **do**
      - $S_i \leftarrow S_i \oplus (A[i]\|0^c)$;
      - $S_{i+1} \leftarrow \pi(S_i)$;

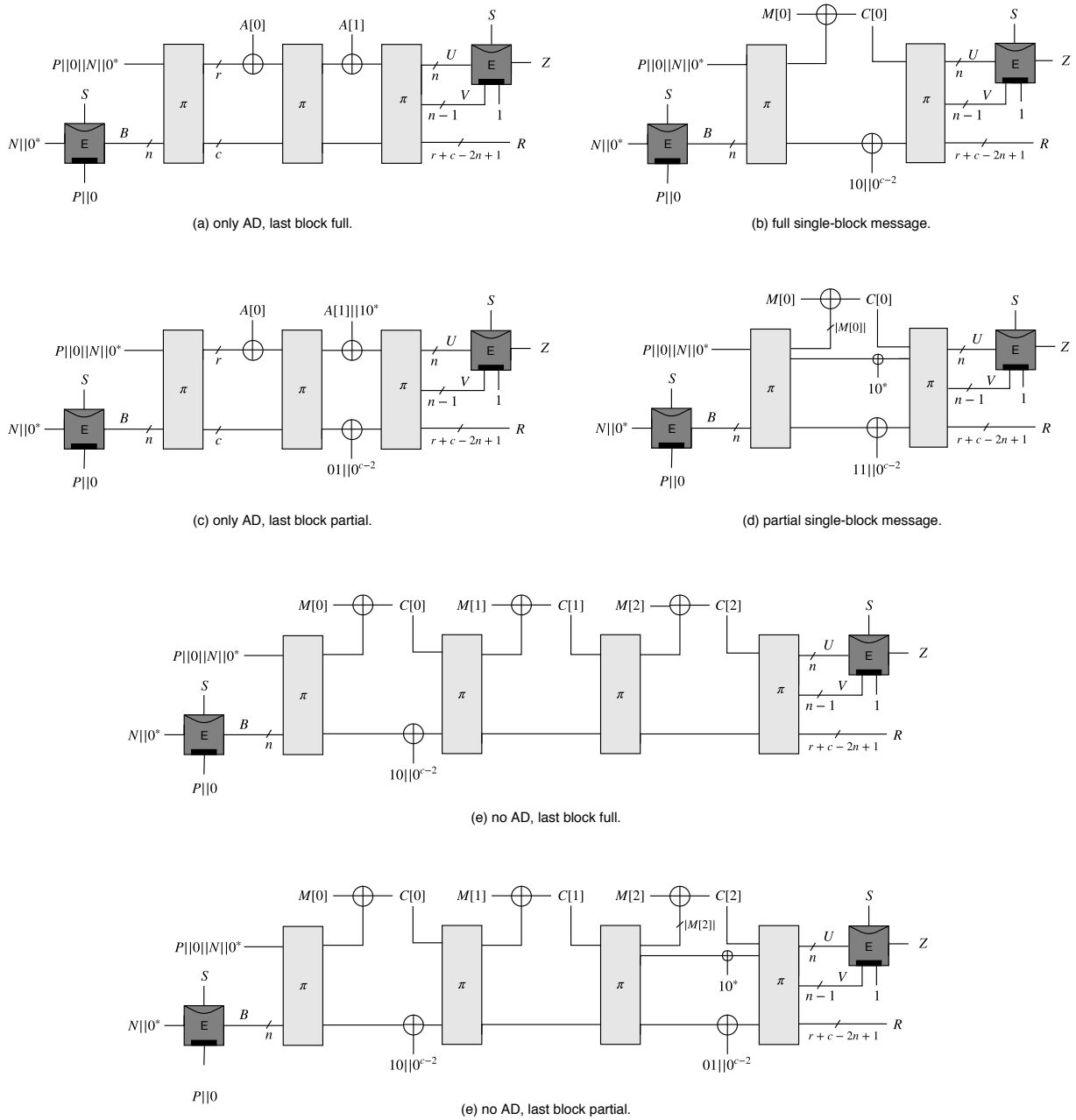
   (b) **if** $|A[\lambda-1]| < r$ **then**
      - $A[\lambda] \leftarrow A[\lambda-1]\|10^{r-|A[\lambda-1]|-1}$;
      - $S_{\lambda-1} \leftarrow S_{\lambda-1} \oplus (0^r\|01\|0^{c-2})$;

   (c) $S_{\lambda-1} \leftarrow S_{\lambda-1} \oplus (A[\lambda-1]\|0^c)$;
   (d) $S_\lambda \leftarrow \pi(S_{\lambda-1})$;
8. **if** $\ell \geq 1$ **then**
   (a) $S_\lambda \leftarrow S_\lambda \oplus (0^r\|10\|0^{c-2})$;
   (b) **for** $i = 0$ **to** $\ell - 2$ **do**
      - $j \leftarrow i + \lambda$;
      - $M[i] \leftarrow S_j[:r] \oplus C[i]$;
      - $S_j \leftarrow C[i]\|S_j[r:]$;
      - $S_{j+1} \leftarrow \pi(S_j)$;

   (c) $M[\ell-1] \leftarrow S_{\lambda+\ell-1}[:|C[\ell-1]|] \oplus C[\ell-1]$;
   (d) **if** $|C[\ell-1]| < r$ **then**
      - $S_{\lambda+\ell-1} \leftarrow S_{\lambda+\ell-1} \oplus (0^{|C[\ell-1]|}\|10^{r-|C[\ell-1]|-1}\|01\|0^{c-2})$;
      - $S_{\lambda+\ell-1} \leftarrow C[\ell-1]\|S_{\lambda+\ell-1}[|C[l-1]|:]$;

   (e) **else** $S_{\lambda+\ell-1} \leftarrow C[\ell-1]\|S_{\lambda+\ell-1}[r:]$;
   (f) $S_{\lambda+\ell} \leftarrow \pi(S_{\lambda+\ell-1})$;
9. $U||V \leftarrow S_{\lambda+\ell}[:2n-1]$;
10. $U^* \leftarrow \left(\mathsf{E}_K^{V||1}\right)^{-1}(Z)$;
11. **if** $U \neq U^*$ **then return** $\perp$;
12. **else if** $\ell > 0$ **then return** $M[0]\| \ldots \|M[\ell-1]$;
13. **else return true**;

B Cases of the S1P mode of operation



(a) only AD, last block full.

(b) full single-block message.

(c) only AD, last block partial.

(d) partial single-block message.

(e) no AD, last block full.

(e) no AD, last block partial.

Figure 2: Different cases of the S1P mode of operation.

C Inverse S-box implementation

- $y[3] = (x[0] \odot x[1]) \oplus x[2]$;
- $y[0] = (x[1] \odot y[3]) \oplus x[3]$;
- $y[1] = (y[3] \odot y[0]) \oplus x[0]$;
- $y[2] = (y[0] \odot y[1]) \oplus x[1]$;

D Inverse L-box implementation

- $a = x \oplus \mathsf{rot}(x, 25)$;
- $b = y \oplus \mathsf{rot}(y, 25)$;
- $c = x \oplus \mathsf{rot}(a, 31)$;
- $d = y \oplus \mathsf{rot}(b, 31)$;
- $c = c \oplus \mathsf{rot}(a, 20)$;
- $d = d \oplus \mathsf{rot}(b, 20)$;
- $a = c \oplus \mathsf{rot}(c, 31)$;
- $b = d \oplus \mathsf{rot}(d, 31)$;
- $c = c \oplus \mathsf{rot}(b, 26)$;
- $d = d \oplus \mathsf{rot}(a, 25)$;
- $a = a \oplus \mathsf{rot}(c, 17)$;
- $b = b \oplus \mathsf{rot}(d, 17)$;
- $a = \mathsf{rot}(a, 16)$;
- $b = \mathsf{rot}(b, 16)$;