SpookChain: Chaining a Sponge-Based AEAD with Beyond-Birthday Security

Gaëtan Cassiers, Chun Guo, Olivier Pereira,
Thomas Peters, and François-Xavier Standaert

UCLouvain – ICTEAM – Crypto Group
B-1348 Louvain-la-Neuve – Belgium

Abstract. We present SpookChain, a new *online authenticated encryption* (OAE) mode that offers several appealing features:

- SpookChain is fully online: it supports the processing of long messages by segments of arbitrary size, and the processing of each segment is online itself, with memory requirements in encryption and decryption being independent of the segment size.
- SpookChain is, to the best of our knowledge, the first concrete mode that is proven to offer dOAE security, a requirement for OAE that, at least guarantees security for new segments as soon as one of the previously processed segments contains a fresh element (nonce, plaintext or associated data).
- SpookChain offers beyond birthday multi-user security (w.r.t. the secret key length), a requirement that we define for the first time in the context of OAE, and which is increasingly appealing in a world where communications are encrypted by default.
- SpookChain is also expected to be remarkably lightweight to implement when protection against side-channel attacks is required.

1 Introduction

Online authenticated encryption. Online authenticated encryption (OAE), first formally studied by Bellare et al. [1], aims at offering confidentiality and authenticity under the constraint that the amount of memory that is needed to encrypt and decrypt messages is limited to a constant that can be much smaller than the size of the messages and ciphertexts that may be processed. Encryption and decryption then need to be possible while only storing small segments of the plaintext or ciphertext.

The design of OAE is motivated by numerous applications. For instance, a user willing to watch a movie stored on a remote server will not wait until the full movie, which may be a few gigabytes big, is downloaded, but rather buffer segments of a few megabytes, which will be decrypted and authenticated before being watched, even if the rest of the movie keeps being downloaded in background.

In switched packet networks, it is also common to use a transport layer that provides an in-order stream communication where data is sent and received by

chunks (which correspond to a packet in the underlying network), and it is then a common approach to secure this stream by enforcing confidentiality and authenticity at the chunk level. An OAE offers here the convenience of securely linking the chunks together as part of a unique stream. For instance, in its latest version, the widely used TLS protocol [10] performs a nonce-based AEAD for each segment (named record) where the nonce contains the record number. Another example is the SSH protocol which uses a similar technique (although it predates the AEAD definition). Online authenticated encryption appears thus as a natural abstraction of the goal of those protocols where, while data is conceptually a stream, it is split into segments (that are sent and received in-order) that match the characteristics of the underlying network for performance reasons.

Eventually, OAE may be required in environments in which the available memory is much smaller than the size of the messages that need to be processed: this may be the case in FPGA or ASIC implementations for instance.

dOAE *security.* Following the tradition of modern authenticated encryption, we are interested in designing a nonce based OAE (that is, deterministic encryption and decryption processes that are initialized with the help of a nonce that is expected to be unique), which still offers some form of security when nonces are repeated (contrary to the normal use of the OAE). Indeed, even if the requirement of a nonce is much less stringent than the requirement of a uniformly random IV, it appears that this uniqueness requirement remains notoriously hard to guarantee in practice [4, 11].

First security definitions for OAE that incorporate some protection against repeated nonces have been proposed by Fleischsmann et al. [6], and have been recently revised and extended by Hoang et al. [8] who propose two new definitions: dOAE, and OAE2 (a third one, nOAE, focuses on nonce-respecting adversaries). OAE2 is the strongest of these notions, and requires two passes on each message segment that cannot be processed online, meaning that the memory available in the encryption and decryption devices must be at least as big as the size of a segment. We are interested in minimizing the memory requirements in order to obtain a scheme that is suitable for the most memory constrained environments. Of course, minimal memory requirements can be obtained by defining very small segments, but this comes at the cost of an increased ciphertext expansion since authentication tags need to be provided per segment. So, our goal is to obtain minimum memory requirements while keeping the flexibility to define the segment size as a function of the latency and/or ciphertext expansion that one is willing to tolerate in a given application. We refer to OAE schemes that offer constant memory requirements, independent of the segment size, as *fully* online.

Fully online encryption is compatible with the second notion proposed by Hoang et al., namely dOAE security, a notion that was inspired by the general duplex design approach [3]. In a nutshell (a more detailed presentation will be offered in the next section), dOAE guarantees security for a segment as long as there is at least one fresh element (nonce, plaintext or associated data) in

a prior segment. So, for instance, even if a nonce is repeated, security remains guaranteed if there is a unique element in the associated data.

Multi-user security. As encryption has become a default choice for communication in an always increasingly connected world, the importance of multi-user security has become paramount: it is not exceptional to have an encryption scheme used with billions of keys.

SpookChain offers beyond birthday multi-user security, that is, it keeps offering security even when using a number of keys that is (much) larger than the square-root of the size of the key space. As dOAE was only defined in the single user setting, we extend the notion to the multi-user setting, a contribution that offers an interest independent of SpookChain itself.

Previous sponge or duplex-based AEs, see e.g., Keyak [2], attempted chaining for online functionality. But SpookChain provides the first rigorous security analysis for such chains in the multi-user setting (even for nOAE). Prior Duplex [5] analysis did not explicitly consider this setting. We leave similar treatments on the other permutation-based AEs as open questions.

Side-channel attack resistant implementations. Side-channel attacks have also become a major concern, as it is more and more common to have devices spread in the nature within adversarial reach, and as it has become standard to have sensitive information belonging to independent organizations being processed on a single hardware platform. As a result, the built-in side-channel attack resistance of authenticated encryption schemes has been pointed out as a core criterion in the ongoing NIST lightweight cryptography competition [9].

SpookChain builds on the TETSponge mode [7], which is a used in the Spook proposal to this NIST competition. SpookChain incorporates many of the features offered by TETSponge, while actually offering some extra efficiency benefits. TETSponge mixes two components: one is a tweakable block cipher (TBC) that is required to be strongly protected against side channel attacks, and is used only twice per encrypted message (independently of the size of the message). The plaintext and associated data are themselves processed using a variant of the sponge-based duplex mode, with minimal protections against side-channel attacks. As the heavily protected TBC is expected to be more expensive than the sponge by 2-3 orders of magnitude, the efficiency benefits of TETSponge are most important when longer messages are processed or, considering an OAE setting, when segments are relatively long. SpookChain mitigates this effect by only requiring a single use of the heavily protected TBC per additional segment (2 TBC calls are still needed for the initial segment).

As a result, in settings where multiple ordered messages need to be sent (e.g., in an SSH or TLS communication), SpookChain offers not only an effective way of splitting the conversation in multiple segments that are linked together, but also comes with efficiency benefits (which are most visible when side-channel attacks are a concern) compared to a solution in which messages would be encrypted independently of each other.

Paper organization. Section 2 introduces our notations and defines the components and security notions that are needed for the presentation of SpookChain. The SpookChain mode itself is presented in Section 3, together with security analysis in Section 4.

2 Preliminaries

We use the following notations. Let $A \in \{0,1\}^*$ be a bitstring and $r > 0$ be an integer. $|A|$ denotes the bit-length of the bit-string A, i.e. the value a such that $A \in \{0,1\}^a$. The concatenation of the bit-strings A and B is $A\|B$. The empty string is ϵ. If $A \neq \epsilon$ (i.e. $|A| > 0$), $A[1]\| \ldots \|A[\alpha] \xleftarrow{r} A$ denotes that A is parsed into r-bit blocks such that $A = A[1]\| \ldots \|A[\alpha]$ and $|A[i]| = r$, for $i = 1, \ldots, \alpha - 1$, and $0 < |A[\alpha]| \leq r$, hence $\alpha = \lceil a/r \rceil$. Moreover, if $r \leq a$, $\mathsf{msb}_r(A)$ (resp. $\mathsf{lsb}_r(A)$) is an r-bit string composed of the r most (resp. least) significant bits of A.

Let $\boldsymbol{A} \in \mathcal{A}^*$. Λ denotes the empty vector, $\Lambda = ()$. $|\boldsymbol{A}|$ denotes the number of components of $\boldsymbol{A}$ in $\mathcal{A}$, i.e. the value L such that $\boldsymbol{A} \in \mathcal{A}^L$. Therefore, $\boldsymbol{A} = (A_1, \ldots, A_{|\boldsymbol{A}|})$ and $\boldsymbol{A}[i] := A_i$ is its i-th component, for $i = 1$ to $|\boldsymbol{A}|$. Moreover, given $A \in \mathcal{A}$, $\boldsymbol{A}\|A = (A_1, \ldots, A_{|\boldsymbol{A}|}, A)$ in $\mathcal{A}^{L+1}$. For instance, if $\mathcal{A} = \{0,1\}^*$, we simply have $\mathcal{A}^* = \{0,1\}^{**}$ such that if $A_i \in \{0,1\}^{a_i}$, then $|\boldsymbol{A}[i]| = a_i$. For $\boldsymbol{A} = (\epsilon, \epsilon)$, we have $|\boldsymbol{A}| = 2$ and then $\boldsymbol{A} \neq \Lambda$, but $|\boldsymbol{A}[1]| = |\boldsymbol{A}[2]| = 0$.

2.1 Primitives

A Tweakable Block Cipher (TBC) with key space $\{0,1\}^\kappa$, tweak space $\{0,1\}^t$, and domain $\{0,1\}^n$, also denoted (κ, t, n)-TBC, is a mapping $\widetilde{\mathsf{E}} : \{0,1\}^\kappa \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that for any key $K \in \{0,1\}^\kappa$ and any tweak $T \in \{0,1\}^t$, $X \mapsto \widetilde{\mathsf{E}}(K,T,X)$ is a permutation of $\{0,1\}^n$. We only focus on (n,n,n)-TBC in this paper. An ideal TBC $\widetilde{\mathsf{IC}} : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a TBC sampled uniformly at random from all (n,n,n)-TBCs: the spirit is the same as ideal (block) ciphers. In this case, $\widetilde{\mathsf{IC}}_K^T$ is a random independent permutation of $\{0,1\}^n$ for each $(K,T) \in \{0,1\}^n \times \{0,1\}^n$ even if the key K is *public*.

A random key-less permutation π, as used in sponge designs, refers to a permutation of $\{0,1\}^\ell$ drawn uniformly at random among the set of all permutations of $\{0,1\}^\ell$. The permutation π is then seen as an ideal object.

Chaining authenticated encryption is made segment by segment as formally defined next.

Definition 1 (Segmented-AE [8]). *A segmented-AE scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where the key space $\mathcal{K}$ is a non-empty set with an associated distribution and both encryption $\mathcal{E} = (\mathcal{E}.\mathsf{init}, \mathcal{E}.\mathsf{next}, \mathcal{E}.\mathsf{last})$ and decryption $\mathcal{D} = (\mathcal{D}.\mathsf{init}, \mathcal{D}.\mathsf{next}, \mathcal{D}.\mathsf{last})$ are specified by triple of deterministic algorithms such that*

$$\begin{array}{ll} \mathcal{E}_K.\mathsf{init} : \mathcal{N} \to \mathcal{S} & \mathcal{D}_K.\mathsf{init} : \mathcal{N} \to \mathcal{S} \\ \mathcal{E}_K.\mathsf{next} : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{S} & \mathcal{D}_K.\mathsf{next} : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\} \times \mathcal{S} \\ \mathcal{E}_K.\mathsf{last} : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} & \mathcal{D}_K.\mathsf{last} : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\} \end{array}$$

where $K \leftarrow \mathcal{K}$, $\mathcal{N}$ is the nonce space, $\mathcal{S}$ is the state space, $\mathcal{A}$ is associated data space, $\mathcal{M}$ is the message space and $\mathcal{C}$ is the ciphertext space.

In order to encrypt a chain $(A_1, M_1), \ldots, (A_L, M_L) \in \mathcal{A} \times \mathcal{M}$, for some L, given a nonce $N \in \mathcal{N}$, we start by generating an initial state $S_0 \leftarrow \mathcal{E}_K.\mathsf{init}(N)$, then we iterate $(S_i, C_i) \leftarrow \mathcal{E}_K.\mathsf{next}(S_{i-1}, A_i, M_i)$, for $i = 1, \ldots, L-1$, and eventually we finalize by $C_L \leftarrow \mathcal{E}_K.\mathsf{last}(S_{L-1}, A_L, M_L)$ resulting in $C_1, \ldots, C_L \in \mathcal{C}$. This process denoted by $\boldsymbol{C} \leftarrow \mathcal{E}_K(N, \boldsymbol{A}, \boldsymbol{M})$, where $\boldsymbol{A} = (A_1, \ldots, A_L) \in \mathcal{A}^L$, $\boldsymbol{M} = (M_1, \ldots, M_L) \in \mathcal{M}^L$ and $\boldsymbol{C} = (C_1, \ldots, C_L) \in \mathcal{C}^L$ is the induced encryption chain. Similarly, we have the induced decryption chain $\boldsymbol{M} \leftarrow \mathcal{D}_K(N, \boldsymbol{A}, \boldsymbol{C})$, where all $\boldsymbol{M}[i] \in \mathcal{M} \cup \{\perp\}$. Since L is not fixed, the induced chain algorithm $\mathcal{E}_K$ (resp. $\mathcal{D}_K$) runs on the domains $\mathcal{N}$, $\mathcal{A}^*$ and $\mathcal{M}^*$ (resp. $\mathcal{C}^*$).

Throughout the paper any segmented-AE is assumed to be correct meaning that, for all $K \leftarrow \mathcal{K}$, the induced chaining is correct: for any $N \in \mathcal{N}$, any $\boldsymbol{A} \in \mathcal{A}^*$ and any $\boldsymbol{M} \in \mathcal{M}^*$, $|\boldsymbol{A}| = |\boldsymbol{M}|$ implies $\boldsymbol{M} = \mathcal{D}_K(N, \boldsymbol{A}, \mathcal{E}_K(N, \boldsymbol{A}, \boldsymbol{M}))$.

Ciphertext expansion The ciphertext expansion τ is the constant value such that, for any ciphertext C computed by a segmented-AE on input (S, A, M), we have $|M| + \tau = |C|$. In our case, we have $\tau = n$ since our ciphertext $C = (\boldsymbol{c}, Z)$, where $\boldsymbol{c} \in \{0,1\}^{|M|}$ and Z is an integrity tag output by an (n, n, n)-TBC.

2.2 dOAE Security

We extend the dOAE security given by Hoang et al [8] to the multi-user (i.e. multi-key) setting. Since we get the original definition in the single-user setting, i.e. when there is only one key in our model, we only proceed with our natural extension. However, we recall the original definition [8] for completeness in Appendix A, Fig. 6.

dOAE Privacy. We recall the intuition behind the dOAE privacy of [8]. A dOAE-secure scheme should produce pseudorandom segments into a chain of segment ciphertexts under partial misuse of prefix chains. More precisely, an adversary can make adaptive encryption queries segment-by-segment. To start a new chain, the adversary makes an *initial* query on a nonce N which may be repeated. The adversary can also make some segment encryption queries to continue an encrypted prefix chain $(N, (A_1, A_2), (C_1, C_2))$ of $(N, (A_1, A_2), (M_1, M_2))$ with a *next* segment (A, M). As long as the prefix $(N, (A_1, A_2, A), (M_1, M_2, \star))$ is fresh, which means that it never appears before, the adversary gets $C \leftarrow \{0,1\}^{|M|+\tau}$, where τ is the ciphertext expansion of the scheme. However, if some prefix $(N, (A_1, A_2, A), (M_1, M_2, M'))$ already exists, the adversary gets $\perp$, if $M \neq M'$, or the already defined segment ciphertext C, otherwise. This latter case allows the adversary to adaptively build two chains with a common prefix that could be forked later from different associated data, like for instance from $(N, (A_1, A_2, A'), (M_1, M_2, \star))$ with A' different than the A above. Eventually, the adversary may also make a *last* segment encryption queries on any chain to add a final segment. If a chain ends with a *last* segment it cannot be further

augmented. The restriction between last segment queries is the same as between next segment queries. Nevertheless, assuming the existence of a prefix chain for $(N, (A_1, A_2), (M_1, M_2))$, it is accepted to make both a next segment encryption query on (A, M) and a last segment encryption query on (A, M'), even if the associated data are common but $M \neq M'$. So there is a full prefix-misuse resistance between next and last segments.

To enhance the dOAE privacy to the multi-user setting, we only have to properly deal with the chaining generated from different keys. Moreover, we merge both types of queries into a single *encryption segment* procedure as the specifications of both queries are very similar. More precisely, the procedure $\mathsf{Enc.seg}(i, j, A, M, 0)$ (resp. $\mathsf{Enc.seg}(i, j, A, M, 1)$) corresponds to an encryption query for a next (resp. last) segment (A, M) using the key K_i, and where j allows keeping track of the current state in the continuing chain. For convenience, even though a last segment has signature $C \leftarrow \mathcal{E}_K.\mathsf{last}(S, A, M)$ we abuse the notation and use $(C, \perp) \leftarrow \mathcal{E}_K.\mathsf{last}(S, A, M)$, where the update state $\perp$ indicates that no further segment is allowed in that chain.

Game dReal

proc initialize
$J_1, \ldots, J_u \leftarrow 0$; $\mathcal{X}_1, \ldots, \mathcal{X}_u \leftarrow \emptyset$;
$(K_1, \ldots, K_u) \xleftarrow{\$} (\mathcal{K})^u$;

proc $\mathsf{Enc.init}(i, N)$
if $N \notin \mathcal{N}$ **then return** $\perp$
$J_i \leftarrow J_i + 1$; $N_{i,J_i} \leftarrow N$;
$\boldsymbol{A}_{i,J_i} \leftarrow \boldsymbol{M}_{i,J_i} \leftarrow \Lambda$
$S_{i,J_i} \leftarrow \mathcal{E}_{K_i}.\mathsf{init}(N)$
return J_i

proc $\mathsf{Enc.seg}(i, j, A, M, b)$ // $b \in \{0, 1\}$
if $j \notin [1..J_i]$ **or** $S_{i,j} = \perp$, **then return** $\perp$
if $(N_{i,j}, \boldsymbol{A}_{i,j} \| A, \boldsymbol{M}_{i,j} \| M', b) \in \mathcal{X}_i$, for some $M' \neq M$, **then return** $\perp$
if $b = 0$, $(C, S_{i,j}) \leftarrow \mathcal{E}_{K_i}.\mathsf{next}(S_{i,j}, A, M)$
if $b = 1$, $(C, S_{i,j}) \leftarrow \mathcal{E}_{K_i}.\mathsf{last}(S_{i,j}, A, M)$
$\boldsymbol{A}_{i,j} \leftarrow \boldsymbol{A}_{i,j} \| A$; $\boldsymbol{M}_{i,j} \leftarrow \boldsymbol{M}_{i,j} \| M$;
$\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b)\}$
return C

Game dRand

proc initialize
$J_1, \ldots, J_u \leftarrow 0$; $\mathcal{X}_1, \ldots, \mathcal{X}_u \leftarrow \emptyset$;
$E_i(x) \leftarrow \mathsf{undef}$ for all x $(1 \leq i \leq u)$

proc $\mathsf{Enc.init}(i, N)$
if $N \notin \mathcal{N}$ **then return** $\perp$
$J_i \leftarrow J_i + 1$; $N_{i,J_i} \leftarrow N$;
$\boldsymbol{A}_{i,J_i} \leftarrow \boldsymbol{M}_{i,J_i} \leftarrow \Lambda$
return J_i

proc $\mathsf{Enc.seg}(i, j, A, M, 0)$ // $b \in \{0, 1\}$
if $j \notin [1..J_i]$ **or** $N_{i,j} = \perp$, **then return** $\perp$
if $(N_{i,j}, \boldsymbol{A}_{i,j} \| A, \boldsymbol{M}_{i,j} \| M', b) \in \mathcal{X}_i$, for some $M' \neq M$, **then return** $\perp$
$\boldsymbol{A}_{i,j} \leftarrow \boldsymbol{A}_{i,j} \| A$; $\boldsymbol{M}_{i,j} \leftarrow \boldsymbol{M}_{i,j} \| M$
$\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b)\}$
if $E_i(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b) = \mathsf{undef}$, **then** $E_i(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b) \xleftarrow{\$} \{0, 1\}^{|M|+\tau}$
$C \leftarrow E_i(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b)$
if $b = 1$, **then** $N_{i,j} \leftarrow \perp$
return C

Fig. 1: Multi-user dOAE privacy experiments.

dOAE Integrity Under the same restriction on the encryption queries it should be unfeasible to output a valid chain $(N, \boldsymbol{A}, \boldsymbol{C})$ for some key K_i, meaning that all the internal segments are valid for K_i, as long as the chain is fresh for i. The multi-user **dForge** experiment given in Fig. 2 follows the multi-user **dReal** experiment where we have to keep track of all the ciphertext chains returned to the adversary, and augmented with a final procedure as in [8] . Formally,

we define the authenticity advantage $\mathbf{Adv}^{\mathsf{doae\text{-}auth}}_{\mathsf{AEAD}}(\mathscr{D}) = \Pr[\mathbf{dForge}_{\mathsf{AEAD}}(\mathscr{D}) \Rightarrow \mathsf{true}]$.

Game dForge

proc initialize

$J_1, \ldots, J_u \leftarrow 0$; $\mathcal{X}_1, \ldots, \mathcal{X}_u \leftarrow \emptyset$; $(K_1, \ldots, K_u) \xleftarrow{\$} (\mathcal{K})^u$;

proc $\mathsf{Enc.init}(i, N)$

if $N \notin \mathcal{N}$ **then return** $\perp$
$J_i \leftarrow J_i + 1$; $N_{i,J_i} \leftarrow N$; $\boldsymbol{A}_{i,J_i} \leftarrow \boldsymbol{M}_{i,J_i} \leftarrow \boldsymbol{C}_{i,J_i} \leftarrow \Lambda$; $S_{i,J_i} \leftarrow \mathcal{E}_{K_i}.\mathsf{init}(N)$
return J_i

proc $\mathsf{Enc.seg}(i, j, A, M, b)$ // $b \in \{0, 1\}$

if $j \notin [1..J_i]$ **or** $S_{i,j} = \perp$, **then return** $\perp$
if $(N_{i,j}, \boldsymbol{A}_{i,j}\|A, \boldsymbol{M}_{i,j}\|M', b) \in \mathcal{X}_i$, for some $M' \neq M$, **then return** $\perp$
if $b = 0$, **then** $(C, S_{i,j}) \leftarrow \mathcal{E}_{K_i}.\mathsf{next}(S_{i,j}, A, M)$, **else** $(C, S_{i,j}) \leftarrow \mathcal{E}_{K_i}.\mathsf{last}(S_{i,j}, A, M)$
$\boldsymbol{A}_{i,j} \leftarrow \boldsymbol{A}_{i,j}\|A$; $\boldsymbol{M}_{i,j} \leftarrow \boldsymbol{M}_{i,j}\|M$; $\boldsymbol{C}_{i,j} \leftarrow \boldsymbol{C}_{i,j}\|C$;
$\mathcal{X}_i \leftarrow \mathcal{X}_i \cup \{(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{M}_{i,j}, b)\}$; $\mathcal{Y}_i \leftarrow \mathcal{Y}_i \cup \{(N_{i,j}, \boldsymbol{A}_{i,j}, \boldsymbol{C}_{i,j}, b)\}$
return C

proc finalize $(i, N, \boldsymbol{A}, \boldsymbol{C}, b)$ // $b \in \{0, 1\}$

if $(N, \boldsymbol{A}, \boldsymbol{C}, b) \in \mathcal{Y}_i$ **or** $|\boldsymbol{A}| \neq |\boldsymbol{C}|$ **or** $|\boldsymbol{A}| = 0$, **then return** false
$S \leftarrow \mathcal{D}_{K_i}.\mathsf{init}(N)$; $m \leftarrow |\boldsymbol{C}|$
$j \leftarrow 1$ **to** $m - b$ **do**
  $(M, S) \leftarrow \mathcal{D}_{K_i}.\mathsf{next}(S, \boldsymbol{A}[j], \boldsymbol{C}[j])$
  if $M = \perp$ **then return** false
if $\mathcal{D}_{K_i}.\mathsf{last}(S, \boldsymbol{A}[m], \boldsymbol{C}[m]) = \perp$ **and** $b = 1$, **then return** false
return true

Fig. 2: Multi-user dOAE integrity experiment.

3 SpookChain Mode

The SpookChain mode is defined upon the TETSponge mode. For completeness, we recall the TETSponge mode of [7] in Fig. 3. We identity the first TBC call which produces B to feed the first permutation call as the "initial" part of TETSponge. The remaining part of the algorithm is named the "Duplex" part and it is denoted as DEnc (resp. DDec) for the encryption (resp. decryption). We stress that the last TBC call is included in these algorithms by convention.

Therefore, we can say that the TETSponge encryption consists of its initial part followed by an encryption of a single segment, processed by DEnc. The basic principle of SpookChain is to "chain" TETSponge by keeping in memory the capacity of the last permutation call which should have been erased after a careful execution of TETSponge. This capacity is denoted by R in Fig. 3. To process a second segment, we simply have to call DEnc again starting with input state $S = 0^r\|R$. This process can be repeated as many times as desired by keeping the next value R generated by DEnc in memory. In a chain, we thus save

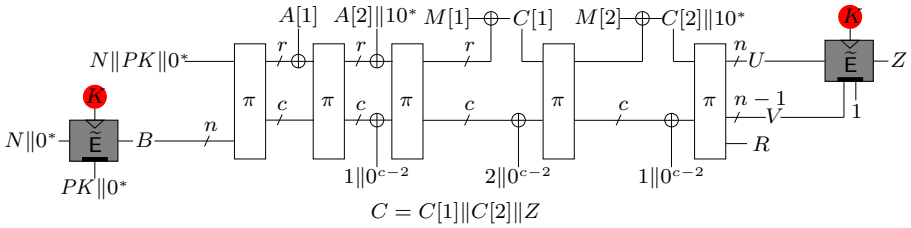

Fig. 3: $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{E}}]_{K,PK}$ AEAD with $\nu = 2$ and $\ell = 2$. Grey square indicates TBC call, where the input to the triangle denotes the key and the input to the dark rectangle denotes the tweak. The block $1\|0^{c-2}$ is inserted only if $|A[\nu]| < r$, resp. $|M[\ell]| < r$.

as many (initial) call to the TBC as the number of segments, and so TETSponge is not used in black-box. To get a fully OAE mode with dOAE security, we also have to treat the "last segment" in another way than the segments that can be further chained. For that purpose, we introduce nothing more than a separation bit in the most significant bit of the input permutation when calling DEnc for the last time. To make the whole SpookChain mode working in all the cases, and so even when the first segment is also the last segment, we just re-order a bit the input permutation of the first call to DEnc, that is the initial part of SpookChain slightly differs from the one of TETSponge. We illustrate SpookChain in Fig. 4.

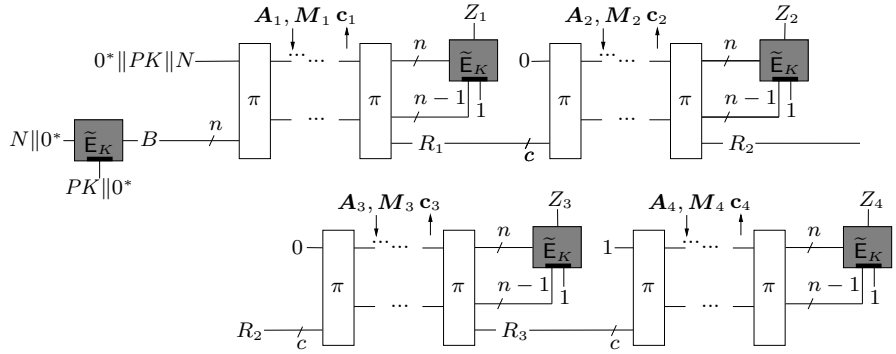

Fig. 4: $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{E}}]_{K,PK}$ with 4 segments, the final one being a *last* segment, then starting with 1. The TBC is in grey. See Fig. 3 for the details of each segment.

In the description of SpookChain, the input of the first permutation of DEnc corresponds to the state S in the sense of the segmented-AE definition, see Def. 1. Therefore, it also corresponds to the full input state of the permutation π. However, from a space complexity standpoint, only the c bits of the capacity must be stored (additionally to K) to process the next segments both in encryption and decryption. We now turn to the full specification of SpookChain.

Parameters. $\widetilde{\mathsf{E}}$ is an (n,n,n)-TBC and π is an $r+c$ bit keyless permutation. The encryption is made by r-bit block of the message. The key of SpookChain is $K\|PK$, where $|K| = n$ and $|PK| = n_p$. We stress that only K has to be kept *secret*, but PK can be public as in TETSponge. The secret key K is picked *uniformly* at random in $\{0,1\}^n$. The public key PK only needs to be *distinct* for each session (pair of users). For simplicity, in this paper we focus on uniform $PK \in \{0,1\}^{n_p}$. Moreover, we have $\mathcal{A} = \mathcal{M} = \mathcal{C} = \{0,1\}^*$ and $\mathcal{N} = \{0,1\}^{n_N}$, where the nonce length n_N is fixed. We require that $n_N, n_p + 1 \leq n$ (initial $\widetilde{\mathsf{E}}$ input), $1 + n_p + n_N + n \leq r + c$ (π input), and $2n \leq r + c + 1$ (tag $\widetilde{\mathsf{E}}$ input). Yet, we recommend $n_p \approx n$ and $c \approx 2n$ and we actually choose $n_p = n-1$ and $c = 2n$ as this leads to a security up to $2^n/n^2$ complexity. There is no recommendation for n_N, but when $n = 128$ one could take $n_N = 96$ which is a standard choice. The ciphertext expansion of SpookChain is $\tau = n$.

---

algorithm $\mathcal{E}_{K,PK}.\mathsf{init}(N)$

1. $B \leftarrow \widetilde{\mathsf{E}}_K^{PK\|0^{n-n_P}}(N\|0^{n-n_N})$
2. **return** $0^{r+c-n_P-n_N-n}\|PK\|N\|B$

algorithm $\mathcal{D}_{K,PK}.\mathsf{init}(N)$

1. $B \leftarrow \widetilde{\mathsf{E}}_K^{PK\|0^{n-n_P}}(N\|0^{n-n_N})$
2. **return** $0^{r+c-n_P-n_N-n}\|PK\|N\|B$

algorithm $\mathcal{E}_{K,PK}.\mathsf{next}(S,A,M)$

1. $(C, capa) \leftarrow \mathsf{DEnc}_K(S,A,M)$
2. **return** $(C, 0^r\|capa)$

algorithm $\mathcal{D}_{K,PK}.\mathsf{next}(S,A,M)$

1. $(M, capa) \leftarrow \mathsf{DDec}_K(S,A,M)$
2. **return** $(M, 0^r\|capa)$

algorithm $\mathcal{E}_{K,PK}.\mathsf{last}(S,A,M)$

1. $S \leftarrow S \oplus 10^{r+c-1}$ // i.e. $1\|\mathsf{lsb}_{r+c-1}(S)$
2. $(C, capa) \leftarrow \mathsf{DEnc}_K(S,A,M)$
3. **return** C

algorithm $\mathcal{D}_{K,PK}.\mathsf{last}(S,A,M)$

1. $S \leftarrow S \oplus 10^{r+c-1}$ // i.e. $1\|\mathsf{lsb}_{r+c-1}(S)$
2. $(M, capa) \leftarrow \mathsf{DDec}_K(S,A,M)$
3. **return** M

---

algorithm $\mathsf{DEnc}_K(S_0,A,M)$

1. $\ell \leftarrow \lceil |M|/r \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
2. $S_1 \leftarrow \pi(S_0)$
3. **if** $\nu \geq 1$ **then**
4. $\quad A[1]\|\ldots\|A[\nu] \xleftarrow{r} A$
5. $\quad$ **for** $i = 1$ **to** $\nu - 1$ **do**
6. $\quad\quad S_i \leftarrow S_i \oplus (A[i]\|0^c)$
7. $\quad\quad S_{i+1} \leftarrow \pi(S_i)$
8. $\quad$ **if** $|A[\nu]| < r$ **then**
9. $\quad\quad A[\nu] \leftarrow A[\nu]\|10^{r-|A[\nu]|-1}$
10. $\quad\quad S_\nu \leftarrow S_\nu \oplus (0^r\|[1]_2\|0^{c-2})$
11. $\quad S_\nu \leftarrow S_\nu \oplus (A[\nu]\|0^c)$
12. $\quad S_{\nu+1} \leftarrow \pi(S_\nu)$
13. **if** $\ell \geq 1$ **then**
14. $\quad M[1]\|\ldots\|M[\ell] \xleftarrow{r} M$
15. $\quad S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r\|[2]_2\|0^{c-2})$
16. $\quad$ **for** $i = 1$ **to** $\ell - 1$ **do**
17. $\quad\quad j \leftarrow i + \nu$
18. $\quad\quad C[i] \leftarrow \mathsf{msb}_r(S_j) \oplus M[i]$
19. $\quad\quad S_j \leftarrow C[i]\|\mathsf{lsb}_c(S_j)$
20. $\quad\quad S_{j+1} \leftarrow \pi(S_j)$
21. $\quad C[\ell] \leftarrow \mathsf{msb}_{|M[\ell]|}(S_{\nu+\ell}) \oplus M[\ell]$
22. $\quad$ **if** $|C[\ell]| < r$ **then**
23. $\quad\quad S_{\nu+\ell} \leftarrow S_{\nu+\ell} \oplus (0^r\|[1]_2\|0^{c-2})$
24. $\quad\quad S_{\nu+\ell} \leftarrow C[\ell]\|10^{r-|C[\ell]|-1}\|\mathsf{lsb}_c(S_{\nu+\ell})$
25. $\quad$ **else** $S_{\nu+\ell} \leftarrow C[\ell]\|\mathsf{lsb}_c(S_{\nu+\ell})$
26. $\quad S_{\nu+\ell+1} \leftarrow \pi(S_{\nu+\ell})$
27. $U\|V \xleftarrow{n} \mathsf{msb}_{2n-1}(S_{\nu+\ell+1})$
28. $capa \leftarrow \mathsf{lsb}_c(S_{\nu+\ell+1})$
29. $Z \leftarrow \widetilde{\mathsf{E}}_K^{V\|1}(U)$
30. **if** $\ell = 0$, **return** $(Z, capa)$
31. $\mathbf{c} \leftarrow C[1]\|\ldots\|C[\ell]$, $C \leftarrow \mathbf{c}\|Z$
32. **return** $(C, capa)$

algorithm $\mathsf{DDec}_K(S_0,A,C)$

1. Parse $\mathbf{c}\|Z \leftarrow C$, where $Z \leftarrow \mathsf{lsb}_n(C)$
2. $\ell \leftarrow \lceil |\mathbf{c}|/r \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
3. $S_1 \leftarrow \pi(S_0)$
4. **if** $\nu \geq 1$ **then**
5. $\quad A[1]\|\ldots\|A[\nu] \xleftarrow{r} A$

6. **for** $i = 1$ **to** $\nu - 1$ **do**
7. $\quad S_i \leftarrow S_i \oplus (A[i] \| 0^c)$
8. $\quad S_{i+1} \leftarrow \pi(S_i)$
9. **if** $|A[\nu]| < r$ **then**
10. $\quad A[\nu] \leftarrow A[\nu] \| 10^{r-|A[\nu]|-1}$
11. $\quad S_\nu \leftarrow S_\nu \oplus (0^r \| [1]_2 \| 0^{c-2})$
12. $S_\nu \leftarrow S_\nu \oplus (A[\nu] \| 0^c)$
13. $S_{\nu+1} \leftarrow \pi(S_\nu)$
14. **if** $\ell \geq 1$ **then**
15. $C[1] \| \ldots \| C[\ell] \xleftarrow{r} \mathbf{c}$
16. $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r \| [2]_2 \| 0^{c-2})$
17. **for** $i = 1$ **to** $\ell - 1$ **do**
18. $\quad j \leftarrow i + \nu$
19. $\quad M[i] \leftarrow \mathsf{msb}_r(S_j) \oplus C[i]$
20. $\quad S_j \leftarrow C[i] \| \mathsf{lsb}_c(S_j)$
21. $\quad S_{j+1} \leftarrow \pi(S_j)$
22. $M[\ell] \leftarrow \mathsf{msb}_{|C[\ell]|}(S_{\nu+\ell}) \oplus C[\ell]$
23. **if** $|C[\ell]| < r$ **then**
24. $\quad S_{\nu+\ell} \leftarrow S_{\nu+\ell} \oplus (0^r \| [1]_2 \| 0^{c-2})$
25. $\quad S_{\nu+\ell} \leftarrow C[\ell] \| 10^{r-|C[\ell]|-1} \| \mathsf{lsb}_c(S_{\nu+\ell})$
26. **else** $S_{\nu+\ell} \leftarrow C[\ell] \| \mathsf{lsb}_c(S_{\nu+\ell})$
27. $S_{\nu+\ell+1} \leftarrow \pi(S_{\nu+\ell})$
28. $U \| V \xleftarrow{n} \mathsf{msb}_{2n-1}(S_{\nu+\ell+1})$
29. $capa \leftarrow \mathsf{lsb}_c(S_{\nu+\ell+1})$
30. $U^* \leftarrow (\widetilde{\mathsf{E}}_K^{V\|1})^{-1}(Z)$
31. **if** $U \neq U^*$, **return** $(\perp, \perp)$
32. **if** $\ell \geq 1$, **return** $(M[1] \| \ldots \| M[\ell], capa)$
33. **return** $(\mathbf{true}, capa)$

Forward secrecy against full state recovery As an interesting additional feature SpookChain provides forward secrecy if a full state of a permutation call is exposed. As a chain might be very long it is of great important to measure the degradation implied by the exposure of the rate and the capacity of a permutation call. Hopefully, if such an exposure occurs in a segment, its prefix in the chain is not affected. Indeed, by inverting the permutation calls, the adversary is able to compute backward the state of the segment, i.e. the first input of π in that segment containing the capacity value R, See Fig. 4, called *capa* in the specification. However, in order to carry on this backward process to adversary has to guess the output rate of the last permutation call of the previous segment (the rate which feeds the TBC call of the previous segment, resulting in Z). So, as long as r is long enough in $O(n)$, the adversary can no longer "rewind" the chain. In contrary, in the forward direction the adversary can easily breaks the privacy of all the next segments. Nevertheless, a full state exposure does not allow computing new valid authentication tags as the key (of the TBC) is still hidden.

Finally, it is worth noticing that the damage caused by a full state recovery in a session is only isolated in the chain where it happens. The security of the other chains of the same session is not impacted in any manner. Clearly, other sessions are neither impacted.

4 Security Analysis

We move on to show the beyond-birthday dOAE-security of SpookChain in the multi-user setting. In the dOAE experiments of privacy (Fig. 1) and integrity (Fig. 2) the adversary can make numerous calls to many different procedures. We say the adversary is $(\overrightarrow{q}, \sigma, \sigma_d)$-bounded, where $\overrightarrow{q} = (q_e, q_{\widetilde{\mathsf{IC}}}, q_\pi)$, q_e denotes the maximal number of adversarial queries to init, $q_{\widetilde{\mathsf{IC}}}$ and q_π denote the number of ideal TBC and permutation queries, σ denotes the total number of blocks in encrypted segments, and σ_d denotes the number of blocks in the decryption queries/adversarial forgery trial (for dOAE authenticity only).

Theorem 1. *Assuming that* $u \le 2^{n_p}$, $n_p \le n$, $n \ge 5$, *and* $2\sigma + 2\sigma_d + q_\pi \le \min\{2^n/4, 2^{r+c}/2\}$. *Then, in the ideal TBC and permutation model, for any* $(\overrightarrow{q}, \sigma)$*-adversary* $\mathscr{A}$ *it holds*

$$\mathbf{Adv}^{\mathsf{doae\text{-}priv}}_{\mathsf{SpookChain}} \le \frac{3u}{2^{n_p}} + \frac{4n(2\sigma + q_\pi) + nq_{\widetilde{\mathsf{IC}}} + n^2 q_e}{2^n} + \frac{17(2\sigma + q_\pi)^2}{2^c}, \tag{1}$$

$$\mathbf{Adv}^{\mathsf{doae\text{-}auth}}_{\mathsf{SpookChain}} \le \frac{3u}{2^{n_p}} + \frac{4n(2\sigma + 2\sigma_d + q_\pi) + nq_{\widetilde{\mathsf{IC}}} + 2}{2^n} + \frac{16(2\sigma + 2\sigma_d + q_\pi)^2}{2^c}. \tag{2}$$

Intuition of the proofs. We will consider the security game capturing the interaction between the adversary and the SpookChain mode. In a nutshell, our arguments proceed in two steps. First, we replace the ideal TBC $\widetilde{\mathsf{IC}}$ in SpookChain by another ideal TBC $\widetilde{\mathsf{IC}}^*$ that *cannot be queried* by the adversary. This modification is "invisible" to the adversary as long as certain types of collisions do not occur, and thus we are able to bound the difference. As the second step, we study the somewhat ideal game with $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{IC}}^*]$. For privacy, we further identify some bad collisions, and we are able to show that the outputs of $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{IC}}^*]$ are random as long as these collisions do not occur. Hence the collision probability, which can be bounded, cinches the privacy security bound. For integrity, we show that the probability to reach a forgery for $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{IC}}^*]$ is sufficiently small, and thus this forgery probability plus the already bounded gap between $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{IC}}^*]$ and $\mathsf{SpookChain}[\pi, \widetilde{\mathsf{IC}}]$ gives the integrity security bound. See the two subsequent subsections for details.

4.1 Proof of dOAE Privacy

Proof (Sketch). We denote by G_0 the game capturing the interaction between $\mathscr{D}$ and **dReal**, and G_2 the game capturing $\mathscr{D}$'s interaction with **dRand**. We transit G_0 to G_2 via several hybrids.

Intermediate game G_1. First, we replace the ideal TBC used in SpookChain by another ideal TBC $\widetilde{\mathsf{IC}}^*$ that's independent from the $\widetilde{\mathsf{IC}}$ accessible to $\mathscr{D}$. This yields the game G_1. We formally describe G_1 in Fig. 5, in which we define four sets $\tau_{\widetilde{\mathsf{IC}}}, \tau^*_{\widetilde{\mathsf{IC}}}, \tau_\pi, \tau^*_\pi$ for subsequent arguments.

It can be seen that, as long as the following bad event does not happen in G_0, the behaviors of G_0 and G_1 are the same:

– (B-1) At any time, there exists $(i, N) \in \tau_{ie}$ such that $(K_i, PK_i \| 0^*, \star, \star) \in \tau_{\widetilde{\mathsf{IC}}}$, where τ_{ie} is the set of all earlier queries to $\mathsf{Enc.init}(i, N)$.

To analyze this event, we define

$$\mu_{PK} := \max_{pk \in \{0,1\}^{n_p}} \big|\{i \in \{1, \ldots, u\} : PK_i = pk\}\big|. \tag{3}$$

We consider the event $\mu_{PK} \ge n+1$. As $PK_1, \ldots, PK_u$ are uniformly distributed, it holds

$$\Pr[\mu_{PK} \ge n+1] \le \binom{u}{n+1} \cdot \frac{1}{(2^{n_p})^n} \le \left(\frac{u}{2^{n_p}}\right)^{n+1} \cdot \frac{2^{n_p}}{(n+1)!} \le \left(\frac{u}{2^{n_p}}\right)^{n+1},$$

where the last inequality comes from $(n+1)! \geq \left(\frac{n+1}{e}\right)^{n+1} \geq 2^{n+1} \geq 2^{n_p}$ since $n+1 \geq 6 > 2e$. Furthermore, when $u \leq 2^{n_p}$ and $n_p \leq n$, we have

$$\Pr[\mu_{PK} \geq n+1] \leq \left(\frac{u}{2^{n_p}}\right)^{n+1} \leq \frac{u}{2^{n_p}}.$$

Conditioned on $\mu_{PK} \leq n$, we could bound $\Pr[\text{(B-1)}]$. Note that in G_1, for any $(i, N) \in \tau_{ie}$, the "user key" K_i is uniformly sampled. Then, using an auxiliary set

$$\tau_{\widetilde{\mathsf{IC}}}[T] := \left\{K \in \{0,1\}^n : (K, T, \star, \star) \in \tau_{\widetilde{\mathsf{IC}}}\right\},$$

it holds

$$\begin{aligned}\Pr[\text{(B-1)}] \leq & \sum_{(i,N)\in\tau_{ie}} \Pr\left[K_i \in \tau_{\widetilde{\mathsf{IC}}}[PK_i\|0^*]\right] \\ \leq & \mu_{PK} \cdot \sum_{PK\in\{0,1\}^{n_p}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[PK\|0^*]\right|}{2^n} \leq \frac{nq_{\widetilde{\mathsf{IC}}}}{2^n}.\end{aligned}$$

Therefore,

$$\left|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]\right| \leq \frac{u}{2^{n_p}} + \frac{nq_{\widetilde{\mathsf{IC}}}}{2^n}.$$

Gap between G_1 and G_2. Second, we define several bad conditions in G_1 that will trigger "abort". See Fig. 5 for details. Roughly speaking, they capture the following conditions:

- (C-1) When an initial key B is derived, it results in a $(r+c)$-bit initial state that collides with the adversarial π-queries nor the other earlier $(r+c)$-bit internal states;
- (C-2) When an internal $(r+c)$-bit state is derived, it collides with the adversarial π-queries nor the other earlier internal states in certain senses;
- (C-3) When a final $(r+c)$-bit state is derived for a query to $\mathcal{E}.\mathsf{next}$ or $\mathcal{E}.\mathsf{last}$, it collides with the earlier internal calls to $\widetilde{\mathsf{IC}}^*$ nor the π-queries in certain senses.

First, let us consider the condition (C-1). Consider the ℓ-th initializing call $\mathsf{Enc.init}(i_\ell, N_\ell)$ that internally results in the state $[0]\|PK\|N\|B_\ell$. Define

$$\tau_\pi[N_i, PK_i] := \left\{B \in \{0,1\}^n : (\star\|PK_i\|N_i\|B, \star) \in \tau_\pi\right\},$$

then this call causes abort if $B_\ell \in \tau_\pi[N_i, PK_i]$. We next argue that w.h.p., B_ℓ is uniform. For this, consider any earlier call $\mathsf{init}(i_{\ell'}, N_{\ell'})$ resulting in the n-bit seed $B_{\ell'}$ with $\ell' < \ell$. If $i_{\ell'} = i_\ell$, then $N_{\ell'} \neq N_\ell$ since abortion is only check in "new" calls, and thus B_ℓ is uniform in at least $2^n - q_e$ values given $B_{\ell'}$. Otherwise, conditioned on that the *key collision* event $\exists i \neq j : K_i\|PK_i = K_j\|PK_j$ in **initialize** did not happen, the two init-calls would induce two distinct TBC-calls $\widetilde{\mathsf{IC}}_{K_{i_\ell}}^{PK_{i_\ell}\|0^*}(N_\ell\|0^*) \to B_\ell$ and $\widetilde{\mathsf{IC}}_{K_{i_{\ell'}}}^{PK_{i_{\ell'}}\|0^*}(N_{\ell'}\|0^*) \to B_{\ell'}$, and thus B_ℓ is

proc initialize

$\tau_{\widetilde{\mathsf{IC}}}, \tau^*_{\widetilde{\mathsf{IC}}}, \tau_\pi, \tau^*_\pi \leftarrow \emptyset$; $\quad J_1, \ldots, J_u \leftarrow 0$;
$\mathcal{X}_1, \ldots, \mathcal{X}_u \leftarrow \emptyset$; $\quad \mathcal{Y}_1, \ldots, \mathcal{Y}_u \leftarrow \emptyset$
$(K_1 \| PK_1, \ldots, K_u \| PK_u) \xleftarrow{\$} (\mathcal{K})^u$
if $\exists i \neq j : K_i \| PK_i = K_j \| PK_j$ **then abort**

proc $\mathsf{Enc.init}(i, N)$

1. ...the same as Fig. 1 (Left).

proc $\mathsf{Enc.seg}(i, j, A, M, b)$ $\quad$ // $b \in \{0,1\}$

1. ...the same as Fig. 1 (Left).

algorithm $\mathcal{E}_{K,PK}.\mathsf{init}(N)$

1. $B \leftarrow (\widetilde{\mathsf{IC}}^*)_K^{PK \| 0^{n-n_P}}(N \| 0^{n-n_N})$
2. **if** the query is new **and** $([0] \| PK \| N \| B, \star) \in (\tau_\pi \cup \tau^*_\pi)$ **then abort**
3. **return** $[0] \| PK \| N \| B$

algorithm $\mathcal{E}_{K,PK}.\mathsf{next}(S, A, M)$

1. ...the same as SpookChain

algorithm $\mathcal{E}_{K,PK}.\mathsf{last}(S, A, M)$

1. ...the same as SpookChain

algorithm $\mathcal{D}_{K,PK}.\mathsf{init}(N)$

1. $B \leftarrow (\widetilde{\mathsf{IC}}^*)_K^{PK \| 0^{n-n_P}}(N \| 0^{n-n_N})$
2. **return** $[0] \| PK \| N \| B$

algorithm $\mathcal{D}_{K,PK}.\mathsf{next}(S, A, M)$

1. ...the same as SpookChain

algorithm $\mathcal{D}_{K,PK}.\mathsf{last}(S, A, M)$

1. ...the same as SpookChain

algorithm $\widetilde{\mathsf{IC}}_K^T(X)$

1. **if** $(K, T, X, \star) \notin \tau_{\widetilde{\mathsf{IC}}}$ **then**
2. $\quad Y \xleftarrow{\$} \{0,1\}^n$ s.t. $(K, T, \star, Y) \notin \tau_{\widetilde{\mathsf{IC}}}$
3. $\quad \tau_{\widetilde{\mathsf{IC}}} \leftarrow \tau_{\widetilde{\mathsf{IC}}} \cup (K, T, X, Y)$
4. **return** Y s.t. $(K, T, X, Y) \in \tau_{\widetilde{\mathsf{IC}}}$

algorithm $(\widetilde{\mathsf{IC}}_K^T)^{-1}(Y)$

1. **if** $(K, T, \star, Y) \notin \tau_{\widetilde{\mathsf{IC}}}$ **then**
2. $\quad X \xleftarrow{\$} \{0,1\}^n$ s.t. $(K, T, X, \star) \notin \tau_{\widetilde{\mathsf{IC}}}$
3. $\quad \tau_{\widetilde{\mathsf{IC}}} \leftarrow \tau_{\widetilde{\mathsf{IC}}} \cup (K, T, X, Y)$
4. **return** X s.t. $(K, T, X, Y) \in \tau_{\widetilde{\mathsf{IC}}}$

algorithm $(\widetilde{\mathsf{IC}}^*)_K^T(X)$
if $(K, T, X, \star) \notin \tau^*_{\widetilde{\mathsf{IC}}}$ **then**
$\quad Y \xleftarrow{\$} \{0,1\}^n$ s.t. $(K, T, \star, Y) \notin \tau^*_{\widetilde{\mathsf{IC}}}$
$\quad \tau^*_{\widetilde{\mathsf{IC}}} \leftarrow \tau^*_{\widetilde{\mathsf{IC}}} \cup (K, T, X, Y)$
return Y s.t. $(K, T, X, Y) \in \tau^*_{\widetilde{\mathsf{IC}}}$

algorithm $((\widetilde{\mathsf{IC}}^*)_K^T)^{-1}(X)$
if $(K, T, \star, Y) \notin \tau^*_{\widetilde{\mathsf{IC}}}$ **then**
$\quad X \xleftarrow{\$} \{0,1\}^n$ s.t. $(K, T, X, \star) \notin \tau^*_{\widetilde{\mathsf{IC}}}$
$\quad \tau^*_{\widetilde{\mathsf{IC}}} \leftarrow \tau^*_{\widetilde{\mathsf{IC}}} \cup (K, T, X, Y)$
return X s.t. $(K, T, X, Y) \in \tau^*_{\widetilde{\mathsf{IC}}}$

algorithm $\pi(S^{in})$
if $(S^{in}, \star) \notin \tau_\pi$ **then**
$\quad S^{out} \xleftarrow{\$} \{0,1\}^{r+c}$ s.t. $(S^{in}, S^{out}) \notin \tau_\pi$
$\quad$ **if** $(\star \| \mathsf{lsb}_{c-2}(S^{out}), \star) \in \tau_\pi$ **then abort**
$\quad$ **if** $(\star, \star \| \mathsf{lsb}_{c-2}(S^{out})) \in \tau_\pi$ **then abort**
$\quad U \| V \leftarrow \mathsf{msb}_{2n-1}(S^{out})$
$\quad$ **if** $(\star, V \| 1, U, \star) \in (\tau_{\widetilde{\mathsf{IC}}} \cup \tau^*_{\widetilde{\mathsf{IC}}})$ **then abort**
$\quad \tau_\pi \leftarrow \tau_\pi \cup (S^{in}, S^{out})$
return S^{out} s.t. $(S^{in}, S^{out}) \in \tau_\pi$

algorithm $\pi^{-1}(S^{out})$
if $(\star, S^{out}) \notin \tau_\pi$ **then**
$\quad S^{in} \xleftarrow{\$} \{0,1\}^{r+c}$ s.t. $(S^{in}, S^{out}) \notin \tau_\pi$
$\quad \tau_\pi \leftarrow \tau_\pi \cup (S^{in}, S^{out})$
return S^{in} s.t. $(S^{in}, S^{out}) \in \tau_\pi$

Fig. 5: Game G_1 with abort conditions.

uniform given $B_{\ell'}$. By these, the probability of abortion in the call to $\mathsf{init}(i_\ell, N_\ell)$ is at most $\left|\tau_\pi[N_{i_\ell}, PK_{i_\ell}]\right|/(2^n - q_e)$. Summing over the q_e calls results in

$$\begin{aligned}\Pr[\text{(C-1)} \mid \neg \textit{key collision}] &\le \sum_{\ell=1}^{q_e} \frac{\left|\tau_\pi[N_{i_\ell}, PK_{i_\ell}]\right|}{2^n - q_e} \le \sum_{\ell=1}^{q_e} \frac{2\left|\tau_\pi[N_{i_\ell}, PK_{i_\ell}]\right|}{2^n} \\ &\le \mu_{PK} \cdot \sum_{N\|PK \in \{0,1\}^{n_N+n_p}} \frac{2\left|\tau_\pi[N, PK]\right|}{2^n} \\ &\le \frac{2\mu_{PK}|\tau_\pi|}{2^n},\end{aligned}$$

where μ_{PK} is as defined in Eq. (3). Note that when the total number of queried blocks is σ, the number of internal π-calls can't exceed 2σ. Thus $|\tau_\pi| \le 2\sigma + q_\pi$. Furthermore, as we've proved that $\Pr[\mu_{PK} \ge n+1] \le \frac{u}{2^{n_p}}$, and clearly

$$\Pr[\textit{key collision}] = \Pr[\exists i \ne j : K_i\|PK_i = K_j\|PK_j] \le \frac{u^2}{2^{n+n_p}} \le \frac{u}{2^{n_p}},$$

we obtain

$$\Pr[\text{(C-1)}] \le \frac{2n(2\sigma + q_\pi)}{2^n} + \frac{2u}{2^{n_p}}.$$

For (C-2), for each of the internal π-calls, the probability of colliding on the least significant $c-2$ bits is $\le 2 \cdot 2|\tau_\pi|/2^{c-2} \le 2(2\sigma + q_\pi)/2^{c-2}$. Thus we have

$$\Pr[\text{(C-2)}] \le \frac{16(2\sigma + q_\pi)^2}{2^c}.$$

Noticing that $|\tau_{\widetilde{\mathsf{IC}}}| + |\tau^*_{\widetilde{\mathsf{IC}}}| \le q_{\widetilde{\mathsf{IC}}} + 2q_e$. Thus a similar analysis gives rise to

$$\Pr[\text{(C-3)}] \le (2\sigma + q_\pi) \cdot \frac{2(q_{\widetilde{\mathsf{IC}}} + 2q_e)}{2^{2n-1}} \le \frac{4(2\sigma + q_\pi)(q_{\widetilde{\mathsf{IC}}} + 2q_e)}{2^{2n}}.$$

Summing over the above, we obtain

$$\begin{aligned}\Pr[\mathsf{G}_1 \text{ aborts}] &\le \frac{2n(2\sigma + q_\pi)}{2^n} + \frac{2u}{2^{n_p}} + \frac{16(2\sigma + q_\pi)^2}{2^c} + \frac{4(2\sigma + q_\pi)(q_{\widetilde{\mathsf{IC}}} + 2q_e)}{2^{2n}} \\ &\le \frac{4n(2\sigma + q_\pi)}{2^n} + \frac{2u}{2^{n_p}} + \frac{16(2\sigma + q_\pi)^2}{2^c}.\end{aligned}$$

The inequality leverages $4(2\sigma + q_\pi)(q_{\widetilde{\mathsf{IC}}} + 2q_e)/2^{2n} \le 2(2\sigma + q_\pi)/2^n \le 2n(2\sigma + q_\pi)/2^n$ assuming $q_{\widetilde{\mathsf{IC}}} + 2q_e \le 2^n/2$.

To complete the analysis for $\mathbf{Adv}^{\mathsf{doae\text{-}priv}}_{\mathsf{AEAD}}$, we replace the freshly generated internal states with values uniformly distributed in $\{0,1\}^{r+c}$ and the tags Z with values uniformly distributed in $\{0,1\}^n$, to obtain the game $\mathsf{G}_2 = \mathbf{dRand}$. It is easy to see as long as G_1 does not abort, replacing internal states induces a

gap of at most $\frac{\sigma^2}{2^{r+c}}$ (a somewhat standard RP-RF switch). On the other hand, replacing tags may cause a gap of $\frac{q_e^2}{2^n}$. However, define

$$\mu_V := \max_{v\in\{0,1\}^{n-1}} \left|\{\pi(S^{in}) \to S^{out}, \mathsf{msb}_{2n-1}(S^{out}) = \star\|V\}\right|,$$

then conditioned on $\mu_V \leq n$, replacing tags only induces a gap of $q_e \cdot \frac{n^2}{2^n}$ since the number of TBC-calls under each tweak $V\|1$ does not exceed n.

It remains to prove that the ciphertext blocks produced in G_1 are "as random as" those in G_2. This essentially requires to prove that in G_1, all new queries to $\mathcal{E}.\mathsf{next}$ and $\mathcal{E}.\mathsf{last}$ give rise to uniform output (as it is also the case in G_2). We show that it is the case in G_1 conditioned on the absence of abortion. For this, we first consider a new call to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 0)$, assume that its corresponding nonce is N_ℓ and initial key is B_ℓ, and distinguish two cases:

- Case 1: it is the first segment for the nonce N_ℓ. We show that all earlier queries to $\mathsf{Enc.seg}$ will not affect the randomness of the output. We only need to consider earlier calls of the form $\mathsf{Enc.seg}(i_\ell, j_\ell, \cdot, \cdot, \cdot)$, as the other calls must have a different 1st call to π, which give rise to completely independent computation flows. Earlier calls to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_{\ell'}, M_{\ell'}, 1)$ will not affect $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 0)$ either, as they only query $\pi(1\|\star)$ which does not affect $\pi([0]\|PK_{i_\ell}\|N_{i_\ell,j_\ell}\|\star)$. For earlier calls to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_{\ell'}, M_{\ell'}, 0)$, it has to be $A_{\ell'} \neq A_\ell$ by the security definitions, and this necessarily make the two computation flows diverge after processing A_ℓ and $A_{\ell'}$. Since the abort conditions around queries to π were never fulfilled, this means all the π-calls made during processing M_ℓ and $M_{\ell'}$ are distinct. A similar argument applies to the resulted tag Z_ℓ. Therefore, the call $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 0)$ gives rise to a ciphertext segment C_ℓ that is uniform and independent from all the previous ones, as well as a new state $capa_\ell$ that is fresh and not contained by any entries in τ_π.
- Case 2: it is not the first segment. Assume that the corresponding $2n$-bit chaining state is S. The subsequent discussion is similar: earlier calls to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_{\ell'}, M_{\ell'}, 1)$ only query $\pi([1]\| \star \|S)$ and will not influence $\pi([0]_{r+c-2n}\|S)$, while earlier calls to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_{\ell'}, M_{\ell'}, 0)$ result in a computation flow that is distinct from $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 0)$ after processing A_ℓ and $A_{\ell'}$. A similar argument applies to the resulted tag Z_ℓ, and thus the call to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 0)$ gives rise to a ciphertext segment C_ℓ that is uniform and independent from all the previous ones.

The analysis a new call to $\mathsf{Enc.seg}(i_\ell, j_\ell, A_\ell, M_\ell, 1)$ is similar. Therefore, the statistical distance between the ciphertext segments produced by G_1 and G_2 *is* the aforementioned gap $\frac{\sigma^2}{2^{r+c}} + \frac{n^2 q_e}{2^n} \leq \frac{(2\sigma+q_\pi)^2}{2^c} + \frac{n^2 q_e}{2^n}$ plus the abort probability of G_1, i.e.,

$$\left|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]\right| \leq \frac{4n(2\sigma+q_\pi) + n^2 q_e}{2^n} + \frac{2u}{2^{n_p}} + \frac{17(2\sigma+q_\pi)^2}{2^c}.$$

This finally concludes with Eq. (1), i.e.,

$$\begin{aligned}\mathbf{Adv}^{\mathsf{doae\text{-}priv}}_{\mathsf{AEAD}} &\leq \frac{u}{2^{n_p}} + \frac{nq_{\widetilde{\mathsf{IC}}}}{2^n} + \frac{4n(2\sigma+q_\pi)+n^2q_e}{2^n} + \frac{2u}{2^{n_p}} + \frac{17(2\sigma+q_\pi)^2}{2^c} \\ &\leq \frac{3u}{2^{n_p}} + \frac{4n(2\sigma+q_\pi)+nq_{\widetilde{\mathsf{IC}}}+n^2q_e}{2^n} + \frac{17(2\sigma+q_\pi)^2}{2^c}.\end{aligned}$$

4.2 Proof of the dOAE Authenticity

Proof (Sketch). For authenticity, we follow the above proof flow and transit to the intermediate game G_1. However, with the additional blocks in the final decryption query, the technical results shall be updated as

$$\begin{aligned}\left|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]\right| &\leq \frac{u}{2^{n_p}} + \frac{nq_{\widetilde{\mathsf{IC}}}}{2^n}, \\ \Pr[\mathsf{G}_1 \text{ aborts}] &\leq \frac{4n(2\sigma+2\sigma_d+q_\pi)}{2^n} + \frac{2u}{2^{n_p}} + \frac{16(2\sigma+2\sigma_d+q_\pi)^2}{2^c}.\end{aligned}$$

Then, assume that abortion never happens, we show that the final decryption of $(i, N, \boldsymbol{A}, \boldsymbol{C}, b)$ gives rise to $\perp$ except with a low probability. Assume that $\boldsymbol{A} = (\boldsymbol{A}[1], \ldots, \boldsymbol{A}[m])$ and $\boldsymbol{C} = (\boldsymbol{C}[1], \ldots, \boldsymbol{C}[m])$. Further assume that the tuple in $\mathcal{Y}_i$ that has the longest common prefix with $(\boldsymbol{A}, \boldsymbol{C})$ is $(N, \boldsymbol{A}', \boldsymbol{C}', b')$, where $\boldsymbol{A}' = (\boldsymbol{A}'[1], \ldots, \boldsymbol{A}'[m])$ and $\boldsymbol{C}' = (\boldsymbol{C}'[1], \ldots, \boldsymbol{C}'[m])$. We distinguish two cases:

- Case 1: $b = b'$. Then there must necessarily exist an index ℓ such that $(\boldsymbol{A}[1], \boldsymbol{C}[1]) \neq (\boldsymbol{A}'[1], \boldsymbol{C}'[1])$. This means the computation flows of processing $(\boldsymbol{A}[1], \boldsymbol{C}[1])$ and $(\boldsymbol{A}'[1], \boldsymbol{C}'[1])$ necessarily deviate at some point, and thus processing $(\boldsymbol{A}[1], \boldsymbol{C}[1])$ eventually gives rise to a fresh π-call at the end. This gives rise to a random $2n-1$ bit value $U\|V$, which means the integrity checking condition $U = U^*$ (line 31. in DDec) is fulfilled with probability at most $\frac{2}{2^n}$. As such, with probability at least $1 - \frac{2}{2^n}$, the final decryption returns $\perp$.
- Case 2: $b \neq b'$. Regardless of the contents, the call to $\mathcal{D}.\mathsf{next}(S, \boldsymbol{A}[m], \boldsymbol{C}[m])$ (when $b = 0$) or $\mathcal{D}.\mathsf{last}(S, \boldsymbol{A}[m], \boldsymbol{C}[m])$ (when $b = 1$) necessarily started with a new state (similarly to that argued before). Similarly to Case 1, this gives rise to a random $2n-1$ bit value $U\|V$, and thus the probability that it passes the integrity checking is at most $\frac{2}{2^n}$.

By the above, Eq. (2) is established:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{doae\text{-}auth}}_{\mathsf{SpookChain}} &\leq \frac{u}{2^{n_p}} + \frac{nq_{\widetilde{\mathsf{IC}}}}{2^n} + \frac{4n(2\sigma+2\sigma_d+q_\pi)}{2^n} + \frac{2u}{2^{n_p}} + \frac{16(2\sigma+2\sigma_d+q_\pi)^2}{2^c} + \frac{2}{2^n} \\ &\leq \frac{3u}{2^{n_p}} + \frac{4n(2\sigma+2\sigma_d+q_\pi)+nq_{\widetilde{\mathsf{IC}}}+2}{2^n} + \frac{16(2\sigma+2\sigma_d+q_\pi)^2}{2^c},\end{aligned}$$

which concludes the proof. □

Acknowledgments

Gaëtan Cassiers is a Research Fellow and François-Xavier Standaert is a Senior Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the EU through the ERC project SWORD (Grant 724725), by the EU and the Walloon Region through the FEDER project USERMedia (convention number 501907-379156), and by the Walloon Region through the project DIGITRANS.

References

1. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online ciphers and the Hash-CBC construction. In: Advances in Cryptology - CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer (2001)
2. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: Caesar submission: Keyak v2 (2015), https://keccak.team/obsolete/Keyak-2.0.pdf
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: SAC 2011. pp. 320–337 (2011)
4. Böck, H., Zauner, A., Devlin, S., Somorovsky, J., Jovanovic, P.: Nonce-disrespecting adversaries: Practical forgery attacks on GCM in TLS. In: 10th USENIX WOOT (2016)
5. Daemen, J., Mennink, B., Assche, G.V.: Full-state keyed duplex with built-in multi-user support. In: Asiacrypt. LNCS, vol. 10625, pp. 606–637. Springer (2017)
6. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: FSE 2012. pp. 196–215 (2012)
7. Guo, C., Pereira, O., Peters, T., Standaert, F.: Towards Lightweight Side-Channel Security and the Leakage-Resilience of the Duplex Sponge. IACR Cryptology ePrint Archive **2019**, 133 (2019), https://eprint.iacr.org/2019/133
8. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In: CRYPTO 2015, Part I. pp. 493–517. LNCS, Springer (2015)
9. NIST: Lightweight cryptography. https://csrc.nist.gov/projects/lightweight-cryptography
10. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC **8446**, 1–160 (2018). https://doi.org/10.17487/RFC8446, https://doi.org/10.17487/RFC8446
11. Vanhoef, M., Piessens, F.: Key reinstallation attacks: Forcing nonce reuse in WPA2. In: Proceedings of ACM CCS 2017. pp. 1313–1328. ACM (2017)

A Original dOAE Definition

Formally, Hoang et. al define the dOAE privacy advantage $\mathbf{Adv}^{\mathsf{doae\text{-}priv}}_{\mathsf{AEAD}}(\mathscr{D}) = |\Pr[\mathscr{D}^{\mathbf{dReal}_{\mathsf{AEAD}}} \Rightarrow 1] - \Pr[\mathscr{D}^{\mathbf{dRand}_{\mathsf{AEAD}}} \Rightarrow 1]|$ and the dOAE authenticity advantage $\mathbf{Adv}^{\mathsf{doae\text{-}auth}}_{\mathsf{AEAD}}(\mathscr{D}) = \Pr[\mathbf{dForge}_{\mathsf{AEAD}}(\mathscr{D}) \Rightarrow \mathsf{true}]$, where the experiments dReal, dRand and dForge are given in Fig. 6.

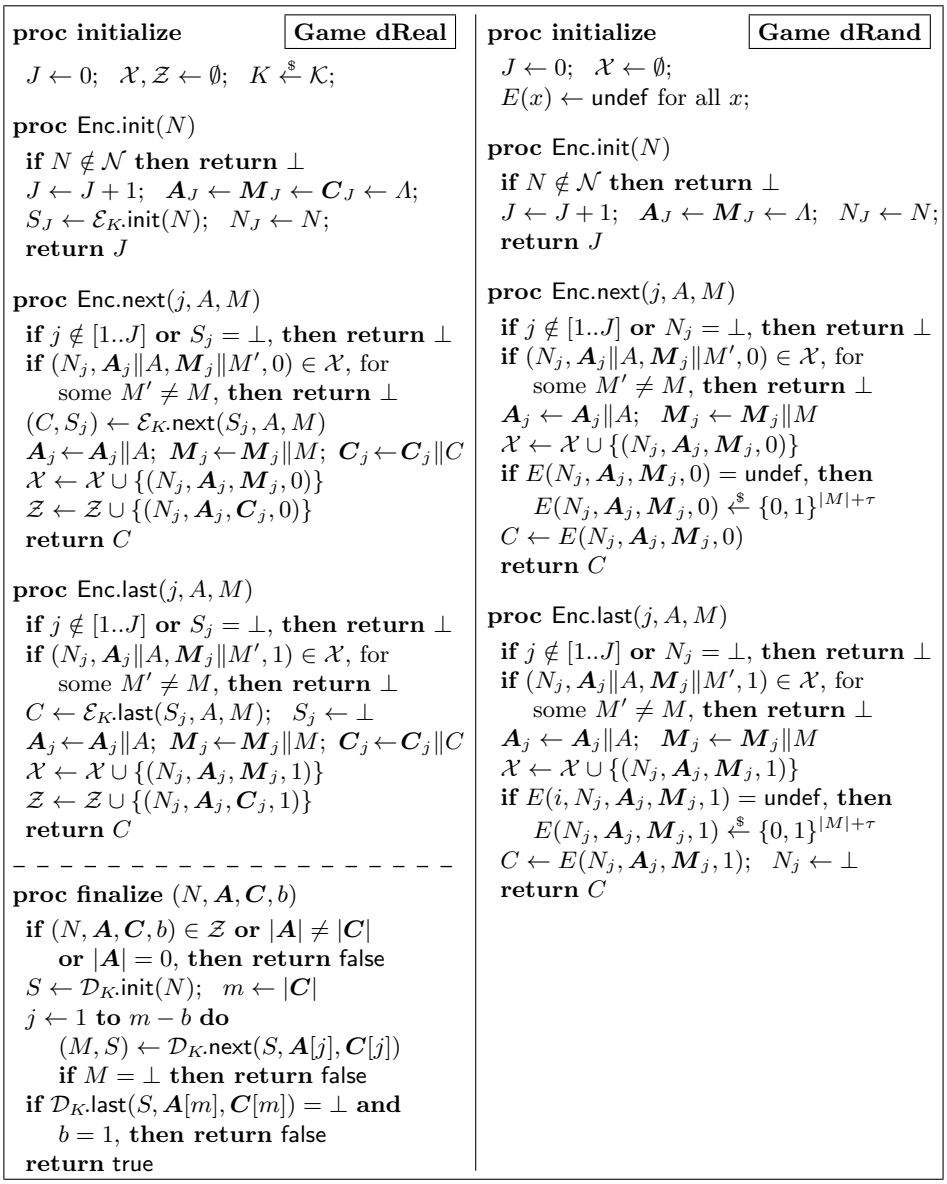

Fig. 6: (dOAE). First column: **dReal** experiment, finalize procedure excluded; **dForge** experiment includes all the procedures of the column. Second column: game **dRand**.