Security of Full-State Keyed and Duplex Sponge: Applications to Authenticated Encryption

Bart Mennink[1] Reza Reyhanitabar[2] Damian Vizár[2]

[1] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and iMinds, Belgium
bart.mennink@esat.kuleuven.be
[2] EPFL, Lausanne, Switzerland {reza.reyhanitabar,damian.vizar}@epfl.ch

Abstract. We provide a security analysis for *full-state* keyed Sponge and *full-state* Duplex constructions. Our results can be used for making a large class of Sponge-based authenticated encryption schemes more efficient by concurrent absorption of associated data and message blocks. In particular, we introduce and analyze a new variant of SpongeWrap with almost free authentication of associated data. The idea of using *full-state* message absorption for higher efficiency was first made explicit in the Donkey Sponge MAC construction, but without any formal security proof. Recently, Gaži, Pietrzak and Tessaro (CRYPTO 2015) have provided a proof for the *fixed-output-length* variant of Donkey Sponge. Yasuda and Sasaki (CT-RSA 2015) have considered *partially* full-state Sponge-based authenticated encryption schemes for efficient incorporation of associated data. In this work, we unify, simplify, and generalize these results about the security and applicability of full-state keyed Sponge and Duplex constructions; in particular, for designing more efficient authenticated encryption schemes. Compared to the proof of Gaži et al., our analysis directly targets the original Donkey Sponge construction as an *arbitrary-output-length* function. Our treatment is also more general than that of Yasuda and Sasaki, while yielding a more efficient authenticated encryption mode for the case that associated data might be longer than messages.

Keywords: Sponge construction, Duplex construction, full-state absorption, authenticated encryption, associated data.

1 Introduction

Since its introduction, the Sponge construction by Bertoni, Daemen, Peeters and Van Assche [4] has faced an immense increase in popularity. As "simple" hash function mode, it is the fundament of the SHA-3 standard Keccak [5], but also its keyed variants have become very popular modes of operation for a permutation to build a wide spectrum of symmetric-key primitives: reseedable pseudorandom number generators [7], pseudorandom functions and message authentication codes (PRFs/MACs) [9, 11], Extendable-Output Functions ("XOFs") [22] and authenticated encryption (AE) modes [10, 11]. The keyed Sponge principle

also got adopted in Spritz, a new RC4-like stream cipher [24], and in 10 out of 57 submissions to the currently running CAESAR competition on authenticated encryption [1,3]. These use cases reinforce the fact that Sponge-based constructions will continue to play an important role, not only in the new hashing standard SHA-3, but in various next-generation cryptographic algorithms.

The classical Sponge construction consists of a sequential application of a permutation p on a state of b bits. This state is partitioned into an r-bit rate or outer part and a c-bit capacity or inner part, where $b = r + c$. In the absorption phase, message blocks of size r bits are absorbed by the outer part and the state is transformed using p, while in the squeezing phase, digests are extracted from the outer part r bits at a time. In the indifferentiability framework of Maurer, Renner and Holenstein [19], Bertoni et al. [6] proved that the Sponge construction is secure up to the $O(2^{c/2})$ birthday-type bound. The capacity part is left untouched throughout the evaluation of the Sponge construction: a violation of this paradigm would make the indifferentiability security result void.

In this work, we strive for optimality, and investigate the most efficient ways of using Sponges for message authentication and authenticated encryption in a provably secure manner. In both directions, we consider a generalization of the currently known schemes to *full-state absorption*, the most efficient usage of the underlying permutation, and we show that these schemes are secure. Due to the full-state absorption, we cannot anymore rely on the classical indifferentiability result of the Sponge (as was for instance done in [2, 10]), and a new security analysis is required. We will elaborate on both directions in the following.

Message Authentication. Bertoni et al. [9] introduced the keyed Sponge as a simple evaluation of the Sponge function on the key and the message, Sponge($K\|M$), and proved security beyond $O(2^{c/2})$. Chang et al. considered a slight variant of the keyed Sponge where the key is processed in the inner part of the Sponge, and observed that it can be seen as the Sponge based on an Even-Mansour blockcipher. At FSE 2015, Andreeva, Daemen, Mennink and Van Assche [2] considered a generic and improved analysis of both the outer- and inner-keyed Sponge. So far, however, these constructions have only been considered with the classical r-bit absorption.

The idea of using *full-state* message absorption for achieving higher efficiency was first made explicit in the Donkey Sponge MAC construction [11], but without any formal security proof. The recently introduced Donkey-inspired MAC function Chaskey [20] did get a formal security analysis, but its proof is thwarted towards Chaskey and does not apply to the Donkey Sponge.

A thorough analysis of the full-state message absorption keyed Sponge had to wait for Gaži, Pietrzak and Tessaro [17], who prove nearly tight security up to $O(\ell q(q+N)/2^b + q(q+\ell+N)/2^c)$, where the adversary makes q queries of maximal length ℓ, and makes N primitive calls. However, their analysis only applies to the *fixed-output-length* variant, and the proof does not directly seem to extend to the original *arbitrary-output-length* keyed Sponge. In this work, we provide a direct proof for this more general case.

In more detail, we present a generalized scheme, dubbed *Full-state Keyed Sponge (FKS)*, that generalizes the Donkey Sponge construction, and prove that it is secure up to approximately $\frac{2(q\ell)^2}{2^b} + \frac{2q^2\ell}{2^c} + \frac{\mu N}{2^k}$, where k is the size of the key, and μ is a parameter called the "multiplicity". We note that usage of the outer-keyed Sponge makes no longer any difference from the usage of the inner-keyed variant in the presence of full-state absorption (see also Sect. 8). Our proof of FKS follows the modular approach of Andreeva et al., but due to the full-state absorption, we cannot rely on the indifferentiability result of [6], and present a new and more detailed analysis.

Authenticated Encryption. Encryption via the Sponge can be done (and is implicitly done) via the Duplex construction [10], a stateful construction consisting of an initialization interface and a duplexing interface. The initialization interface can be called to initialize an all-zero state; the duplexing interface absorbs a message of size $< r$ bits and squeezes $\leq r$ bits of the outer part. The security of the Duplex Sponge traces back to the indifferentiability of the classical Sponge, yielding a $O(2^{c/2})$ security bound.

Bertoni et al. [10] showed that the Duplex, in turn, allows for authenticated encryption in the form of SpongeWrap. This mode is, de facto, the basis of the majority of Sponge-based submissions to the CAESAR competition. Jovanovic et al. [18] re-investigated Sponge-based authenticated encryption schemes, starring NORX, and derived beyond birthday-bound security. These results are, however, all for the usual r-bit absorption. Yasuda and Sasaki [25] have considered several full-state and *partially* full-state Sponge-based authenticated encryption schemes for efficient incorporation of associated data, directly lifting Jovanovic et al.'s security proofs. The concurrent absorption mode proposed by Yasuda and Sasaki (Fig. 3 in [25]) fails to utilize the full-state absorption when the associated data becomes longer than the message, forcing the mode switch from a full-state mode to the classical r-bit absorbing Sponge mode; hence, we refer to this as a *partially* full-state AE mode. Full-state data absorption was also proposed by Reyhanitabar, Vaudenay and Vizár [23] in their compression function based AE mode p-OMD.

We generically aim to optimize the efficiency in Sponge-based authenticated encryption. To this end, we first formalize the *Full-state Duplex Sponge (FDS)* construction. It differs from the original Duplex in the fact that (i) the key is explicitly used to initialize the state, and (ii) the absorption is performed on the entire state. Note that the possibility to absorb in the entire state enforces the explicit usage of the key. Next, we prove that FDS is provably secure, i.e., indistinguishable from a random oracle with the same interfaces. As before, we cannot rely on the classical indifferentiability proof due to the full-state absorption; however, we show how to adapt the FKS proof to a special case directly related to the security of FDS.

We exemplify the better absorption capabilities of FDS by the introduction of a *Full-state SpongeWrap (FSW)*. The FSW construction is more general than that of Yasuda and Sasaki, who only considered specific AE constructions, and interestingly, our approach also yields a more efficient (truly full-state) authenti-

cated encryption mode irrespective of the relative lengths of messages and their associated data.

Organization of the Paper. Notations and preliminary concepts are presented in Sect. 2. We present the Full-state Keyed Sponge and Full-state Duplex Sponge in Sect. 3. The security model is discussed in Sect. 4. In Sect. 5 we prove security of FKS and in Sect. 6 of FDS. The introduction of the Full-state SpongeWrap, and the application of FDS to this construction is given in Sect. 7. Sect. 8 provides a brief discussion on related-key security and our security models.

2 Notations and Conventions

The set of all strings of length b is denoted as $\{0,1\}^b$ for any $b \geq 1$ and the set of all finite strings of arbitrary length is denoted as $\{0,1\}^*$. We will denote the empty string of length 0 as ε. For any positive b, we let $\{0,1\}^{<b} = \bigcup_{i=0}^{b-1}\{0,1\}^i$ denote set of all strings of length less than b including ε. For two strings $X, Y \in \{0,1\}^*$ we let $X \parallel Y$ denote the string obtained by concatenation of X and Y. For a string $X \in \{0,1\}^x$ we let $\mathsf{left}_\ell(X)$ denote the ℓ leftmost bits of X and $\mathsf{right}_r(X)$ the r rightmost bits of X such that $X = \mathsf{left}_\chi(X) \parallel \mathsf{right}_{x-\chi}(X)$ for any $0 \leq \chi \leq x$. For integral b, r, c such that $b = r + c$, and for $t \in \{0,1\}^b$, we let $\mathsf{outer}(t) = \mathsf{left}_r(t)$ and $\mathsf{inner}(t) = \mathsf{right}_c(t)$.

For a non-empty finite set $\mathcal{S}$ let $a \xleftarrow{\$} \mathcal{S}$ denote sampling an element a from $\mathcal{S}$ uniformly at random. We let $|Z|$ denote the cardinality if Z is a set and the length if Z is a string. We let $\mathrm{Perm}(b)$ denote the set of all permutations of b-bit strings and $\mathrm{Func}(b)$ the set of all functions over b-bit strings.

Given two strings X, Y, let

$$\mathsf{llcp}_b(X,Y) = \max_{i \geq 0} \{i \ : \ \mathsf{left}_{ib}(X) = \mathsf{left}_{ib}(Y)\}$$

denote the **l**ength of the **l**ongest **c**ommon **p**refix between X and Y in b-bit blocks. For a string X and a non-empty set of strings $\{Y_1, \ldots, Y_n\}$ let

$$\mathsf{llcp}_b(X; Y_1, \ldots, Y_n) = \max\{\mathsf{llcp}_b(X, Y_1), \ldots, \mathsf{llcp}_b(X, Y_n)\}.$$

For any two pairs of integers $(i,j), (i',j')$, we say that $(i',j') < (i,j)$ if either $i' < i$ or if $i' = i$ and $j' < j$. We say that $(i',j') \leq (i,j)$ if $(i',j') < (i,j)$ or if $(i',j') = (i,j)$. In other words, we use lexicographical ordering to determine ordering of integer-tuples.

3 Sponge Constructions

3.1 Full-State Keyed Sponge

We consider the Full-state Keyed Sponge (FKS) construction that is using a public permutation $p : \{0,1\}^b \to \{0,1\}^b$. It is furthermore parameterized with

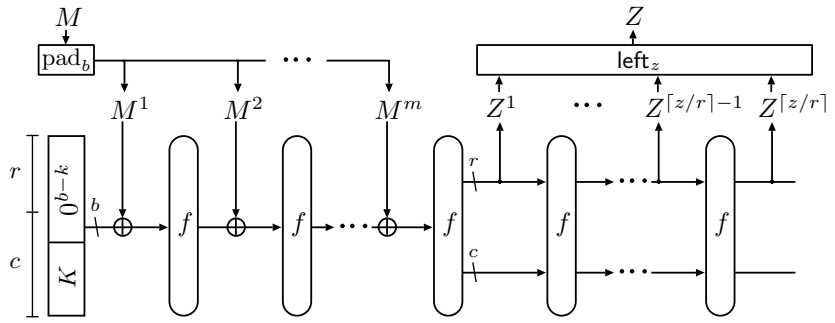

Fig. 1: The FKS construction.

r, k, which are required to satisfy $r < b$ and $k \leq b - r =: c$. The parametrization is sometimes left implicit if it is clear from the context. FKS gets as input a key $K \in \{0,1\}^k$, a message $M \in \{0,1\}^*$, and a natural number z, and it outputs a string $Z \in \{0,1\}^z$:

$$\mathrm{FKS}^p(K, M, z) = \mathrm{FKS}^p_K(M, z) = Z\,.$$

It operates on a state $t \in \{0,1\}^b$, which is initialized using the key K. The message M is first padded to a length a multiple of b bits, using $\mathrm{pad}_b(M) = M \| 10^{b-1-|M| \bmod b}$, which is then viewed as m b-bit message blocks $M^1 \| ... \| M^m$.[3] These message blocks are processed one-by-one, interleaved with evaluations of p. After the absorption of M, the outer r bits of the state are output and the state is processed via p until a sufficient amount of output bits are obtained. FKS is depicted in Fig. 1, and Algo. 1 provides a formal specification of FKS.

Algorithm 1 $\mathrm{FKS}[p, r, k](K, M, z)$

$t \leftarrow 0^{b-k} \| K$
$M^1 \| \cdots \| M^m \xleftarrow{b} \mathrm{pad}_b(M)$
for $i = 1, \ldots, m$ **do**
    $s \leftarrow t \oplus M^i$
    $t \leftarrow p(s)$
$Z \leftarrow \mathsf{left}_r(t)$
while $|Z| < z$ **do**
    $t \leftarrow p(t)$
    $Z \leftarrow Z \| \mathsf{left}_r(t)$
return $\mathsf{left}_z(Z)$

Algorithm 2 $\mathrm{FDS}[p, r, k]$

Interface D.initialize(K)
    $t \leftarrow 0^{b-k} \| K$

Interface D.duplexing(M, z)
    $s \leftarrow t \oplus \mathrm{pad}_b(M)$
    $t \leftarrow p(s)$
    return $\mathsf{left}_z(t)$

[3] In fact, any injective padding function works, as long as the last block is always non-zero.

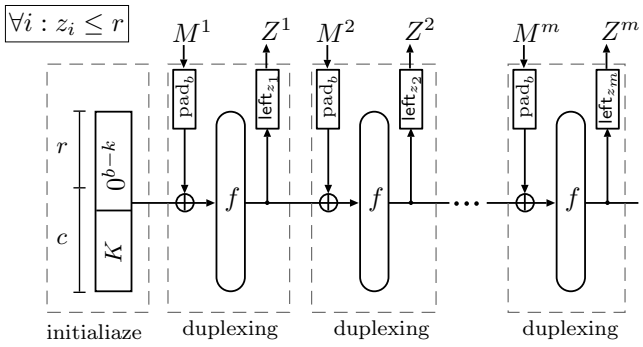

Fig. 2: The FDS construction.

3.2 Full-State Duplex Sponge

We present the Full-state Duplex Sponge (FDS) construction, a generalization of the duplex Sponge of Bertoni et al. [8, 10]. FDS is also parameterized by a public permutation $p : \{0,1\}^b \to \{0,1\}^b$ and values r, k, which are required to satisfy $r < b$ and $k \leq b - r =: c$. Again, the parametrization is sometimes left implicit if clear from the context. An instance of FDS, denoted by D, consists of two interfaces: D.initialize and D.duplexing. D.initialize gets as input a key $K \in \{0,1\}^k$ and outputs nothing, while D.duplexing gets as input a message $M \in \{0,1\}^{<b}$ and a natural number $z \leq r$, and it outputs a string $Z \in \{0,1\}^z$. FDS is depicted in Fig. 2, and the formal specification is given in Algo. 2. FDS is a generalization of FKS where D.initialize is used to initialize the state, and messages are absorbed into the state and/or digests are squeezed out of the state using D.duplexing calls.

4 Security Models and Tools

Multiplicity. Let $\{(x_i, y_i)\}_{i=1}^N$ be a set of N evaluations of a permutation p. Following Andreeva et al. [2], we define the total maximal multiplicity as $\mu = \mu_{\text{fwd}} + \mu_{\text{bwd}}$, where

$$\mu_{\text{fwd}} = \max_a |\{i \in \{1, \ldots, N\} \ : \ \mathsf{outer}(x_i) = a\}|,$$
$$\mu_{\text{bwd}} = \max_a |\{i \in \{1, \ldots, N\} \ : \ \mathsf{outer}(y_i) = a\}|.$$

4.1 Adversaries and Patarin's Coefficient-H Technique

We consider an information-theoretic adversary $\boldsymbol{A}$ that has access to one or more oracles X; this is denoted by $\boldsymbol{A}^X$ and the notation $\boldsymbol{A}^X \Rightarrow 1$ means that $\boldsymbol{A}$, after interaction with X, returns 1. It is a classical fact (for a simple proof see [14]) that in the information-theoretic setting, adversaries can be assumed to be deterministic without loss of generality.

We use Patarin's Coefficient-H technique [21]; more precisely, a revisited formulation of it by Chen and Steinberger [14]. Consider a deterministic information-theoretic adversary $\boldsymbol{A}$ whose goal is to distinguish two oracles X and Y:

$$\Delta_{\boldsymbol{A}}(X;Y) = \left|\Pr\left[\boldsymbol{A}^X \Rightarrow 1\right] - \Pr\left[\boldsymbol{A}^Y \Rightarrow 1\right]\right|.$$

Here, X and Y are randomized algorithms; the randomization depends on the specific scenario and for now is left implicit. The interaction with any of the two systems X or Y is summarized in a transcript τ. Denote by D_X the probability distribution of transcripts when interacting with X, and similarly, D_Y the distribution of transcripts when interacting with Y. A transcript τ is called *attainable* if $\Pr[D_Y = \tau] > 0$, meaning that it can occur during interaction with Y. Denote by $\mathcal{T}$ the set of all attainable transcripts. The Coefficient-H technique states the following, for the proof of which we refer to [14].

Lemma 1 (Coefficient-H Technique [14,21]). *Consider a fixed deterministic adversary $\boldsymbol{A}$. Let $\mathcal{T} = \mathcal{T}_{\text{good}} \cup \mathcal{T}_{\text{bad}}$ be a partition into* good *transcripts $\mathcal{T}_{\text{good}}$ and* bad *transcripts $\mathcal{T}_{\text{bad}}$. If there exists an ε such that for all $\tau \in \mathcal{T}_{\text{good}}$,*

$$\frac{\Pr[D_X = \tau]}{\Pr[D_Y = \tau]} \geq 1 - \varepsilon,$$

then, $\Delta_{\boldsymbol{A}}(X;Y) \leq \varepsilon + \Pr[D_Y \in \mathcal{T}_{\text{bad}}]$.

The two partitions of $\mathcal{T}$ are labeled as $\mathcal{T}_{\text{good}}$ and $\mathcal{T}_{\text{bad}}$ to aid the intuitiveness of the proof. The transcripts in $\mathcal{T}_{\text{good}}$ are "good" in the sense that they give us a high value of $\Pr[D_X = \tau]/\Pr[D_Y = \tau]$ and thus small ε while the "bad" transcripts from $\mathcal{T}_{\text{bad}}$ fail to do so.

4.2 Security Models for FKS and FDS

Let $RO^\infty : \{0,1\}^* \to \{0,1\}^\infty$ be a random oracle which takes inputs of arbitrary but finite length and returns random infinite strings, where each output bit is selected uniformly and independently for every input M.

Let F be either FKS or FDS, which is based on a permutation $p : \{0,1\}^b \to \{0,1\}^b$ and a key $K \in \{0,1\}^k$. We will define the security of F in two settings: the public permutation setting, where the adversary has query access to the permutation (security comes from the secrecy of K), and the secret permutation setting (with no explicit key K), where the adversary has no access to the underlying permutation and the security comes from the secrecy of the permutation.

We use the notations F_K^p and F_0^π to refer to the public permutation and secret permutation based schemes, respectively; where, π is a secret random permutation.

In both settings, we consider an adversary that aims to distinguish the *real* F from an *ideal* (reference) primitive—an oracle RO with the same interface. For $F = \text{FKS}$ the corresponding ideal primitive RO is defined by $RO_{\text{FKS}}(M, z) = \mathsf{left}_z(RO^\infty(M))$. For $F = \text{FDS}$ the corresponding reference primitive RO_{FDS} is

a stateful oracle with two interfaces: (1) RO^r_{FDS}.initialize() that initializes the state of the oracle, St, to the empty string, and (2) RO^r_{FDS}.duplexing(M, z) that, on input $M \in \{0,1\}^{<b}$ and a natural number z, first updates the state as $\mathrm{St} \leftarrow \mathrm{St}||pad_b(M)$ and then outputs $\mathsf{left}_z\,(RO^\infty(\mathrm{St}))$.

We define the distinguishing advantage of any adversary $\boldsymbol{A}$ against F based on a *public* permutation by

$$\mathbf{Adv}^{\mathrm{ind}}_{F^p_K,p}(\boldsymbol{A}) = \Big|\Pr\left[K \xleftarrow{\$} \{0,1\}^k, p \xleftarrow{\$} \mathrm{Perm}\,(b) \;:\; \boldsymbol{A}^{F^p_K,p,p^{-1}} \Rightarrow 1\right] - \Pr\left[p \xleftarrow{\$} \mathrm{Perm}\,(b) \;:\; \boldsymbol{A}^{RO,p,p^{-1}} \Rightarrow 1\right]\Big|.$$

The distinguishing advantage of $\boldsymbol{A}$ against F based on a *secret* permutation is defined by

$$\mathbf{Adv}^{\mathrm{ind}}_{F^\pi_0}(\boldsymbol{A}) = \Big|\Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}\,(b) \;:\; \boldsymbol{A}^{F^\pi_0} \Rightarrow 1\right] - \Pr\left[\boldsymbol{A}^{RO} \Rightarrow 1\right]\Big|.$$

The resource parameterized advantage functions are defined as usual. Let $\mathbf{Adv}^{\mathrm{ind}}_{F^p_K,p}(q,\ell,\mu,N) = \max_{\boldsymbol{A}} \mathbf{Adv}^{\mathrm{ind}}_{F^p_K,p}(\boldsymbol{A})$ be the maximum advantage over all adversaries that make q queries to the left oracle, all of maximal length ℓ permutation calls if $F = \mathrm{FKS}$ or that make at most q initialize() calls to the left oracle and issue at most ℓ duplexing queries after each initialization if $F = \mathrm{FDS}$ with total maximal multiplicity μ in both cases, and that make N direct queries to the public permutation. To simplify the analysis, we assume that each of the q oracle queries in fact consists of *exactly* ℓ permutation (or that the adversary indeed makes ℓ duplexing calls after each initialization). This is without loss of generality, it can simply be achieved by giving extra squeezing outputs to the adversary. Similarly, we define $\mathbf{Adv}^{\mathrm{ind}}_{F^\pi_0}(q,\ell,\mu) = \max_{\boldsymbol{A}} \mathbf{Adv}^{\mathrm{ind}}_{F^\pi_0}(\boldsymbol{A})$, noticing that in this case $N = 0$, thus it is omitted from the resources.

4.3 Security Model for Even-Mansour

Our proof relies on a reduction to the security of a low-entropy single-key Even-Mansour construction [15, 16]. In more detail, let $p : \{0,1\}^b \to \{0,1\}^b$ be a permutation and $K \in \{0,1\}^k$ be a key. The Even-Mansour blockcipher is defined as

$$E^p_K(M) = p(M \oplus (0^{b-k} \parallel K)) \oplus (0^{b-k} \parallel K).$$

We define the distinguishing advantage of any adversary $\boldsymbol{A}$ against E based on a public permutation p as

$$\mathbf{Adv}^{\mathrm{prp}}_{E^p_K,p}(\boldsymbol{A}) = \Big|\Pr\left[K \xleftarrow{\$} \{0,1\}^k, p \xleftarrow{\$} \mathrm{Perm}\,(b) \;:\; \boldsymbol{A}^{E^p_K,p,p^{-1}} \Rightarrow 1\right] - \Pr\left[\pi, p \xleftarrow{\$} \mathrm{Perm}\,(b) \;:\; \boldsymbol{A}^{\pi,p,p^{-1}} \Rightarrow 1\right]\Big|.$$

Let $\mathbf{Adv}^{\mathrm{prp}}_{E^p_K,p}(q,\mu,N) = \max_{\boldsymbol{A}} \mathbf{Adv}^{\mathrm{prp}}_{E^p_K,p}(\boldsymbol{A})$ be the maximum advantage over all adversaries that make q queries to the left oracle, with total maximal multiplicity μ, and that make N direct queries to the public permutation.

5 Security Analysis of FKS

We prove the following result for FKS:

Theorem 1. *Let $b, r, c, k > 0$ be such that $b = r + c$ and $k \leq c$. Let* FKS *be the scheme of Sect. 3.1. Then,*

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}^p_K,p}(q,\ell,\mu,N) \leq \frac{2(q\ell)^2}{2^b} + \frac{2q^2\ell}{2^c} + \frac{\mu N}{2^k}.$$

The proof follows to a certain extent the modular approach of [2], and in particular also uses the observation that FKS^p_K can alternatively be considered as $\mathrm{FKS}_0^{E^p_K}$, a clever observation used before by Chang et al. [13]. Note that this observation only works for $k \leq c$: it consists of xoring two dummy keys $K \oplus K$ in-between every two adjacent permutation calls, and if $k > c$ this would entail a difference in the squeezing blocks of FKS. This trick splits the security of FKS^p_K into the security of the Even-Mansour blockcipher and the security of FKS with secret primitive. Looking back at [2], the security of IKS/OKS with secret permutations was simply reverted to the classical indifferentiability result of [6]. Because this is a rather loose approach, and additionally because the indifferentiability bound cannot be used for FKS due to its full-state absorption, we consider the security of FKS with secret primitive in more detail and derive an improved bound.

Proof (Proof of Theorem 1). Consider any adversary $\boldsymbol{A}$ with resources (q,ℓ,μ,N). Note that $\mathrm{FKS}^p_K = \mathrm{FKS}_0^{E^p_K}$. Therefore, by a modular argument,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}^p_K,p}(\boldsymbol{A}) &= \Delta_{\boldsymbol{A}}\left(\mathrm{FKS}_0^{E^p_K}, p; RO_{\mathrm{FKS}}, p\right) \\
&\leq \Delta_{\boldsymbol{B}}\left(\mathrm{FKS}_0^{\pi}, p; RO_{\mathrm{FKS}}, p\right) + \Delta_{\boldsymbol{C}}\left(E^p_K, p; \pi, p\right) \\
&= \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^{\pi}}(\boldsymbol{B}) + \mathbf{Adv}^{\mathrm{prp}}_{E^p_K,p}(\boldsymbol{C})
\end{aligned}$$

for some adversary $\boldsymbol{B}$ with resources (q,ℓ,μ) and adversary $\boldsymbol{C}$ with resources $(q\ell,\mu,N)$. Note that $\boldsymbol{B}$ also has access to p, but queries to this oracle are meaningless as its left oracle (FKS_0^{π} or RO_{FKS}) is independent of p.

In [2], it is proven that $\mathbf{Adv}^{\mathrm{prp}}_{E^p_K,p}(\boldsymbol{C}) \leq \frac{\mu N}{2^k}$ for any $\boldsymbol{C}$. In Lem. 2, we prove that $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^{\pi}}(\boldsymbol{B}) \leq \frac{2(q\ell)^2}{2^b} + \frac{2q^2\ell}{2^c}$ for any adversary $\boldsymbol{B}$. □

Lemma 2. *Let $b, r, c > 0$ be such that $b = r + c$. Let* FKS *be the scheme of Sect. 3.1. Then,*

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^{\pi}}(q,\ell,\mu) \leq \frac{2(q\ell)^2}{2^b} + \frac{2q^2\ell}{2^c}.$$

Proof. Given that the padding is publicly known and injective, we can generalize the setting, and assume that the i^{th} query M_i has length divisible by b and that $M_i^{m_i} \neq 0^b$. More detailed, for $1 \leq i \leq q$, we let $m_i = |M_i|/b$ and $M_i =$

$M_i^1 \parallel M_i^2 \parallel \ldots \parallel M_i^{m_i}$ s.t. $|M_i^j| = b$ for $1 \le j \le m_i$. We further assume, that the adversary always asks for output of length divisible by r and that every query induces exactly ℓ primitive calls. This is without loss of generality: we can simply output "free bits" to the adversary. We will denote the b-bit state of FKS just *before* the j^{th} application of π is made when processing the i^{th} query as s_i^j for $1 \le j \le \ell$. Similarly, we will denote the b-bit state of FKS just *after* the j^{th} application of π in i^{th} query as t_i^j for $1 \le j \le \ell$. We will call the former in-states and the latter out-states. Note that every in-state s_i^j is determined by the out-state t_i^{j-1} and the block of query M_i^j as $s_i^j = t_i^{j-1} \oplus M_i^j$ in the absorbing phase or just by t_i^j in the squeezing phase as depicted in Fig. 3.

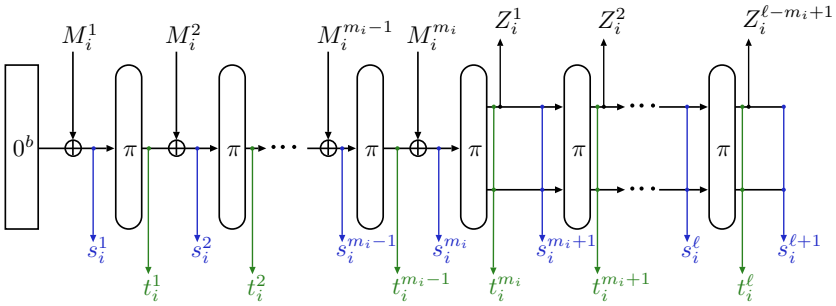

Fig. 3: Processing the i^{th} query.

To aid the simplicity of further analysis we additionally define initial dummy out-states $t_i^0 = 0^b$ and extended queries $\bar{M}_i = M_i \parallel 0^{(\ell-m_i)b}$ for $1 \le i \le q$. Now we can express every in-state, be it absorbing or squeezing, as $s_i^j = t_i^{j-1} \oplus \bar{M}_i^j$. We will group the out-states of i^{th} query as $T_i = \{t_i^0, t_i^1, \ldots, t_i^\ell\}$. Because each query induces exactly ℓ calls to π, we know that a query M_i will be answered by a string $Z_i = Z_i^1 \parallel \ldots \parallel Z_i^{z_i}$ with $z_i = \ell - m_i + 1$ and $|Z_i^j| = r$ for $1 \le j \le z_i$. In particular, we have that $Z_i^j = \mathsf{outer}\left(t_i^{m_i+j-1}\right)$.

The RP-RF Switch. We start by replacing the random permutation $\pi \xleftarrow{\$} \mathrm{Perm}(b)$ by a random function $f \xleftarrow{\$} \mathrm{Func}(b)$ in the experiment. This will contribute the term $(q\ell)^2/2^b$ to the final bound by a standard hybrid argument so we have $\mathbf{Adv}^{\text{ind}}_{\text{FKS}_0^\pi}(q,\ell,\mu) \le \mathbf{Adv}^{\text{ind}}_{\text{FKS}_0^f}(q,\ell,\mu) + (q\ell)^2/2^b$.

Patarin's Coefficient-H Technique. We will use the coefficient-H technique to show that $\mathbf{Adv}^{\text{ind}}_{\text{FKS}_0^f}(q,\ell,\mu) \le (q\ell)^2/2^b + 2q^2\ell/2^c$. The two systems an adversary is trying to distinguish are FKS_0^f and RO_{FKS}. We will refer to the former as X and to the latter as Y. In either of the worlds, the adversary makes q queries $M_1, \ldots, M_q$ and learns the responses $Z_1, \ldots, Z_q$. The transition from queries M_i to $\bar{M}_i$ is injective, and additionally the length m_i of M_i is implicit

from $\bar{M}_i$. Therefore, we can summarize the interaction of the adversary with its oracle (X or Y) with a transcript $(\bar{M}_1, \ldots, \bar{M}_q, Z_1, \ldots, Z_q)$.

To facilitate the analysis, we will disclose additional information $T_1, \ldots, T_q$ to the adversary at the end of the experiment. In the real world, these are the out-states $T_i = \{t_i^0, t_i^1, \ldots, t_i^\ell\}$ as discussed in the beginning of the proof. In the ideal world, these are dummy variables that satisfy the following intrinsic properties of the Sponge construction:

1. $t_i^0 = 0^b$ for $1 \le i \le q$,
2. if $\mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i'}\right) = n$ for $1 \le i, i' \le q$ then $t_i^j = t_{i'}^j$ for $1 \le j \le n$,
3. $\mathsf{outer}\left(t_i^{j+m_i-1}\right) = Z_i^j$ for $1 \le i \le q$ and $1 \le j \le z_i$,

but are perfectly random otherwise. Note that in both worlds, $Z_1, \ldots, Z_q$ are fully determined by $T_1, \ldots, T_q$, so we can drop them from the transcript. Thus a transcript of adversary's interaction with FKS will be $\tau = (\bar{M}_1, \ldots, \bar{M}_q, T_1, \ldots, T_q)$.

With respect to Lem. 1, we will show that there exists a definition of bad transcripts $\mathcal{T}_{\text{bad}}$, such that $\Pr[D_X = \tau] / \Pr[D_Y = \tau] = 1$ for any $\tau \in \mathcal{T}_{\text{good}} = \mathcal{T} \backslash \mathcal{T}_{\text{bad}}$, and thus $\mathbf{Adv}^{\text{ind}}_{\text{FKS}_0^f}(q, \ell, \mu) \le \Pr[D_Y \in \mathcal{T}_{\text{bad}}]$.

Definition of a Bad Transcript. Stated formally, a transcript τ is labeled as *bad* if

$$\begin{aligned} &\exists (1,1) \le (i,j), (i',j') \le (q,\ell) \text{ such that:} \\ &\qquad j \ne j' \vee \mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i'}\right) < j = j' \le \ell, \\ &\qquad t_i^{j-1} \oplus \bar{M}_i^j = t_{i'}^{j'-1} \oplus \bar{M}_{i'}^{j'}. \end{aligned} \tag{1}$$

This formalization of a bad transcript comes with an intuitive, informal interpretation; as long as all relevant *inputs* $s_i^j = t_i^{j-1} \oplus \bar{M}_i^j$ to the random function f induced by the Sponge function are distinct the output of the Sponge will be distributed uniformly. We do not require uniqueness of all in-states because the adversary can trivially force their repetition by issuing queries with common prefixes, as we have argued earlier. However these collisions are not a problem because uniqueness of the queries implies that $\mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i'}\right) < \max\{m_i, m_{i'}\}$ for any two queries $\bar{M}_i, \bar{M}_{i'}$. Even if the adversary truncates an old query and thus forces an old absorbing in-state s to be squeezed for output, it is still not a problem because the adversary *has not seen* the image $f(s)$ before. Note that albeit in-states do not exist in the ideal world, they can be defined by the same relation as in the real world, i.e. $s_i^j = t_i^{j-1} \oplus \bar{M}_i^j$.

Bounding the Ratio of Probabilities of Good Transcripts. In the ideal world, the out-states $\{t_i^0\}_{i=0}^q$ are always assigned a value trivially. Beside that, we will also trivially assign a single randomly sampled value to multiple state variables, that are affected by the common prefixes of the queries. The remaining out-states are sampled uniformly at random. It follows that there are exactly $\eta(\tau) = \sum_{i=1}^q \ell - \mathsf{llcp}_b(M_i; M_1, \ldots, M_{i-1})$ b-bit values in any transcript τ, that are sampled independently and uniformly. We thus have $\Pr[D_Y = \tau] = 2^{-\eta(\tau)b}$ for any τ.

Let Ω_X be the set of all possible real-world oracles. We have that $|\Omega_X| = 2^{b2^b}$. Let $\mathsf{comp}_X(\tau) \subseteq \Omega_X$ be the set of all oracles compatible with the transcript τ, i.e. number of the real-world oracles that are capable of producing τ in an experiment. We will compute the probability of seeing τ in the real world as $\Pr[D_X = \tau] = |\mathsf{comp}_X(\tau)|/|\Omega_X|$. Note that a real-world oracle is completely determined by the underlying function f.

If $\tau \in \mathcal{T}_{\text{good}}$, then every in-state $s_i^j = t_i^{j-1} \oplus \bar{M}_i^j$ that does not trivially collide with some other in-state $s_{i'}^{j'}$ due to common prefix of $\bar{M}_i^j$ and $\bar{M}_{i'}^{j'}$ must be distinct. The number of domain points of f that have an image assigned by τ is easily seen to be $\eta(\tau) = \sum_{i=1}^{q} \ell - \mathsf{llcp}_b(M_i; M_1, \ldots, M_{i-1})$. A compatible function f can therefore have arbitrary image values on the remaining $2^b - \eta(\tau)$ domain points. Thus we compute $|\mathsf{comp}_X(\tau)| = 2^{b\left(2^b - \eta(\tau)\right)}$ and

$$\Pr[D_X = \tau] = \frac{|\mathsf{comp}_X(\tau)|}{|\Omega_X|} = \frac{2^{b\left(2^b-\eta(\tau)\right)}}{2^{b2^b}} = 2^{-\eta(\tau)b} = \Pr[D_Y = \tau].$$

Bounding the Probability of a Bad Transcript in the Ideal World. We can bound the probability of τ being bad (cf. (1)) by first bounding the collision probability of an arbitrary but fixed pair of in-states $s_i^j, s_{i'}^{j'}$ (i.e. the event $s_i^j = s_{i'}^{j'}$ occurs) and then summing this probability for all possible values of $(i,j),(i',j')$ with $(i',j') \neq (i,j)$. Because this probability varies significantly, we will split all in-states into three classes and bound probabilities of individual collisions between these classes.

We will associate to each in-state s_i^j a label stamp_i^j. We set $\mathsf{stamp}_i^j = \texttt{free}$ if $1 < j = \mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right) + 1 \leq m_i$ such that $m_{i^*} < j$ for some $i^* < i$. We will set $\mathsf{stamp}_i^1 = \texttt{initial}$ for $1 \leq i \leq q$ and $\mathsf{stamp}_i^j = \texttt{fixed}$ in the remaining cases. Informally, we have $\mathsf{stamp}_i^j = \texttt{free}$ whenever the adversary forces $\mathsf{outer}\left(t_i^{j-1}\right) = Z_{i^*}^{j-m_{i^*}-1}$ by reusing exactly first $j-1$ blocks of a previous query $\bar{M}_{i^*}$ in $\bar{M}_i$ and sets $\bar{M}_i^j \neq \bar{M}_{i^*}^j = 0^b$. By doing this, it freely but non-trivially chooses $\mathsf{outer}\left(s_i^j\right) = \mathsf{outer}\left(s_{i*}^j \oplus \bar{M}_{i*}^j \oplus \bar{M}_i^j\right)$. Note that if the adversary puts $\bar{M}_i^j = \bar{M}_{i^*}^j$, this is not counted as a free state (the states will in fact be the same). We have $\mathsf{stamp}_i^j = \texttt{initial}$ for the initial in-state of every query.

As the condition (1) is symmetrical w.r.t. (i,j) and (i',j'), and as it cannot be satisfied if $(i,j) = (i',j')$, it can be rephrased as

$$\begin{aligned} &\exists (1,1) \leq (i',j') < (i,j) \leq (q,\ell) \text{ such that:} \\ &\qquad \mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right) < j \leq \ell,\ s_i^j = s_{i'}^{j'}. \end{aligned} \tag{2}$$

Doing so is without loss of generality, as each s_i^j with $j \leq \mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right)$ is identical with some previous state that has already been checked for collisions with $s_{i'}^{j'}$ for every possible (i',j'). In the further analysis, we will be working with (2) rather than with (1).

We will now bound the probability of collision of an arbitrary pair of in-states $(s_i^j, s_{i'}^{j'}) = (t_i^{j-1} \oplus \bar{M}_i^j, t_{i'}^{j'-1} \oplus \bar{M}_{i'}^{j'})$ with $\mathsf{stamp}_i^j = \texttt{fixed}$. We fix arbitrary i and investigate the following three cases for j. In each case we treat every $(i', j') < (i, j)$.

Case 1: $\mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right) + 1 < j \leq m_i$. In this case, t_i^{j-1} is undetermined when the adversary issues the query $\bar{M}_i$. This implies that it will be independent from all $t_{i'}^{j'-1}$ for any $(i', j') < (i, j)$. The probability of the collision $t_i^{j-1} \oplus \bar{M}_i^j = t_{i'}^{j'-1} \oplus \bar{M}_{i'}^{j'}$ is easily seen to be 2^{-b}.

Case 2: $\max\left\{\mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right) + 1, m_i\right\} < j \leq \ell$. Here $t_i^{j-1} = Z_i^{j-m_i} \parallel \mathsf{inner}\left(t_i^{j-1}\right)$ and $\bar{M}_i^j = 0^b$. Although the adversary learns the value of $Z_i^{j-m_i}$ during the experiment, this is independent of all $s_{i'}^{j'}$ with $(i', j') < (i, j)$ (because $j + 1 > \mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right)$). Even if $\mathsf{stamp}_{i'}^{j'} \in \{\texttt{free}, \texttt{initial}\}$ and $\mathsf{outer}\left(s_{i'}^{j'}\right) = \alpha$ for some value α chosen by the adversary, the collision $Z_i^{j-m_i} \parallel \mathsf{inner}\left(t_i^{j-1}\right) = \alpha \parallel \mathsf{inner}\left(s_{i'}^{j'}\right)$ happens with probability 2^{-b}.

Case 3: $j = \mathsf{llcp}_b\left(\bar{M}_i; \bar{M}_1, \ldots, \bar{M}_{i-1}\right) + 1$. If $j = \mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i'}\right) + 1$, the in-state $s_{i'}^{j'=j}$, call it a twin-state of s_i^j, cannot collide with s_i^j, as by the second trivial property $t_i^{j-1} = t_{i'}^{j-1}$ and by $j - 1 = \mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i'}\right)$ we have $\bar{M}_i^j \neq \bar{M}_{i'}^j$. Note that if there was an $i^* < i$ with $m_{i^*} \leq \mathsf{llcp}_b\left(\bar{M}_i, \bar{M}_{i^*}\right) = j-1$ and $j \leq m_i$ then we would have $\mathsf{stamp}_i^j = \texttt{free}$. However if we had the same situation but with $j > m_i$ then $\bar{M}_i$ and $\bar{M}_{i^*}$ would be identical. So $\mathsf{outer}\left(t_i^{j-1}\right)$ has not been set and revealed to the adversary by any previous output value and for any non-twin, `fixed` in-state $s_{i'}^{j'}$, the probability of collision is at most 2^{-b} by a similar argument as in **Case 1**.

There are no more than $q\ell$ choices for (i, j) and no more than $q\ell$ possible (i', j') for every (i, j) so the overall probability that the condition (2) will be evaluated due to a pair of in-states with $\mathsf{stamp}_i^j = \texttt{fixed}$ is at most $(q\ell)^2/2^b$.

If $\mathsf{stamp}_i^j = \texttt{free}$ then $\mathsf{outer}\left(s_i^j\right)$ is under adversary's control. However the value of $\mathsf{inner}\left(t_i^{j-1}\right)$ is always generated at the end of the experiment. By a case analysis similar to the previous one we can verify that the probability of a collision due to a pair of in-states with $\mathsf{stamp}_i^j = \texttt{free}$ is not bigger than 2^{-c}. It is apparent from the definition of a `free` in-state that there is at most one such in-state for each query. Having $q\ell$ in-states in total, there are at most $q(q\ell)$ pairs with $\mathsf{stamp}_i^j = \texttt{free}$ and the probability of $\tau \in \mathcal{T}_{\text{bad}}$ due to such a pair is at most $q^2\ell/2^c$.

If $\mathsf{stamp}_i^j = \texttt{initial}$ then s_i^j cannot non-trivially collide with any other `initial` in-state. A collision with a non-`initial` state $s_{i'}^{j'}$ implies that $t_{i'}^{j'-1} = \bar{M}_{i'}^{j'} \oplus \bar{M}_i^1$. If $j' > m_{i'}$ or if there is some M_{i^*} with $m_{i^*} < j' <= \mathsf{llcp}_b\left(M_{i'}, \bar{M}_{i^*}\right) +$

1, then $\mathsf{outer}\left(t_{i'}^{j'-1}\right)$ is known to the adversary. However $\mathsf{inner}\left(t_{i'}^{j'-1}\right)$ is always generated at the end of the experiment. By a case analysis similar to the one we carried out earlier, it can be verified that the collision $s_i^1 = s_{i'}^{j'}$ occurs with probability no bigger than 2^{-c}. There is exactly one `initial` in-state in each query, so similarly as with the `free` in-states, the overall probability of a transcript being bad due to a pair with an `initial` in-state is at most $q^2\ell/2^c$. By summing all the partial collision probabilities we obtain that $\Pr[D_Y \in \mathcal{T}_{bad}] \leq (q\ell)^2/2^b + 2q^2\ell/2^c$. □

6 Security Analysis of FDS

For FDS, we prove the following result:

Theorem 2. *Let $b, r, c, k > 0$ be such that $b = r + c$ and $k \leq c$. Let* FDS *be the scheme of Sect. 3.2. Then,*

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}^p_K,p}(q,\ell,\mu,N) \leq \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c} + \frac{\mu N}{2^k}.$$

The proof uses Lem. 3 to transform a FDS adversary into an FKS adversary, similarly to [8, 10]. While this would be sufficient to prove the security of the Duplex construction, the bound induced solely by Lem. 3 suffers from a quantitative degradation: we have that $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}^p_K,p}(q,\ell,\mu,N) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}^p_K,p}(q\ell,\ell,\mu,N)$, resulting in a bound $\frac{2q^2\ell^4}{2^b} + \frac{2q^2\ell^3}{2^c} + \frac{\mu N}{2^k}$ according to Thm. 1. In reality, there *will be* a quantitative gap between the security of FDS construction and that of FKS will be present, but it will be smaller. This is because an FKS adversary constructed from an FDS adversary issues queries of a specific structure which is far from general. In below proof for FDS, we use this property. In more detail, we derive a specific class of "constrained adversaries" and generalize the proof of Lem. 2 to these adversaries.

Proof (Proof of Theorem 2). Consider any adversary $\boldsymbol{A}$ with resources (q,ℓ,μ,N). We have that $\mathrm{FDS}^p_K = \mathrm{FDS}_0^{E^p_K}$. Therefore, by a modular argument,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}^p_K,p}(\boldsymbol{A}) &= \Delta_{\boldsymbol{A}}\left(\mathrm{FDS}_0^{E^p_K}, p; RO_{\mathrm{FDS}}, p\right) \\
&\leq \Delta_{\boldsymbol{B}}\left(\mathrm{FDS}_0^{\pi}, p; RO_{\mathrm{FDS}}, p\right) + \Delta_{\boldsymbol{C}}\left(E^p_K, p; \pi, p\right) \\
&\leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}_0^{\pi}}(\boldsymbol{B}) + \mathbf{Adv}^{\mathrm{sprp}}_{E^p_K,p}(\boldsymbol{C})
\end{aligned}$$

for some adversary $\boldsymbol{B}$ with resources (q,ℓ,μ) and adversary $\boldsymbol{C}$ with resources $(q\ell,\mu,N)$. Note that $\boldsymbol{B}$ also has access to p, but these queries are meaningless as its left oracle (FDS_0^{π} or RO_{FDS}) is independent of p.

In [2], it is proven that $\mathbf{Adv}^{\mathrm{sprp}}_{E^p_K,p}(\boldsymbol{C}) \leq \mu N/2^k$. In Cor. 3 we show that any FDS adversary $\boldsymbol{B}$ can be turned into a special "constrained" adversary $\boldsymbol{B'}$ against FKS with resources $(q\ell,\ell,\mu)$:

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}_0^{\pi}}(\boldsymbol{B}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^{\pi}}(\boldsymbol{B'}).$$

In Lem. 4, we prove that $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^\pi}(\boldsymbol{B'}) \leq (q\ell)^2/2^b + (q\ell)^2/2^c$ for any such adversary $\boldsymbol{B'}$. □

For the remainder of the proof, we introduce the mapping $Q_{\mathrm{FKS}} : (\{0,1\}^{<b})^+ \to \{0,1\}^*$. For any $b > 0$ and for all $X_1, \ldots, X_n \in \{0,1\}^{<b}$ we let

$$Q_{\mathrm{FKS}}(X_1, \ldots, X_n) = \mathrm{pad}_b(X_1) \parallel \ \ldots \ \parallel \mathrm{pad}_b(X_{n-1}) \parallel X_n.$$

Lemma 3. *Let $b, r, c, k > 0$ be such that $b = r + c$ and $k \leq c$. Let $D = \mathrm{FDS}^p$ as defined in Sect. 3.2. Then for the i^{th} duplexing query (M_i, z_i) made after the last D.*initialize(K) *we have*

$$Z_i = D.\mathrm{duplexing}\,(M_i, z_i) = \mathrm{FKS}^p(K, Q_{\mathrm{FKS}}(M_1, \ldots, M_i), z_i).$$

Moreover, the mapping $Q_{\mathrm{FKS}} : (\{0,1\}^{<b})^+ \to \{0,1\}^*$ is injective.

Proof. We will show the first claim by induction. For $i = 1$, the internal state of FDS is updated to $t_1 = p\left((0^{b-k} \parallel K) \oplus \mathrm{pad}_b(M_1)\right)$, which is exactly the same as the state of FKS evaluated on M_1 only. Then both FDS and FKS output the same value $Z_1 = \mathsf{left}_{z_1}(t_1)$. For every $i > 1$, FDS updates its state to $t_i = p\,(t_{i-1} \oplus \mathrm{pad}_b(M_i))$. By the induction argument, t_{i-1} is also the state of FKS after processing the first $i-1$ padded blocks. Then the final state of FKS is easily seen to be t_i as well. The equality of outputs follows trivially.

To verify the injectivity of Q_{FKS}, we will show how to invert it. For any image $X = Q_{\mathrm{FKS}}(X_1, \ldots, X_n)$, we can start recovering the input arguments from the left to right. Firstly, we have $n = \lceil |X|/b \rceil$. While $|X| > b$, we keep removing the leftmost b bits of X and applying the inverse of pad_b to them to recover the next component X_i. What remains is the unpadded block X_n. □

The result of Lem. 3 can be used to reduce any FDS adversary to a constrained FKS adversary. More specifically, any adversary $\boldsymbol{A}$ against FDS that makes q initialize calls and duplexes ℓ blocks after each initialization can be reduced to a constrained FKS adversary $\boldsymbol{A'} = R_{\mathrm{FKS}}(\boldsymbol{A})$. To answer the j^{th} duplexing query (M_i^j, z_i^j) made by $\boldsymbol{A}$ after the i^{th} initialize call, $\boldsymbol{A'}$ queries its own oracle with $(Q_{\mathrm{FKS}}(M_i^1, \ldots, M_i^j), z_i^j)$. $\boldsymbol{A'}$ copies the output of $\boldsymbol{A}$ at the end of the experiment.

Corollary 3. *Let $\boldsymbol{A}$ be an adversary against FDS that makes q initialize calls and duplexes ℓ blocks after each initialization and $R_{\mathrm{FKS}}(\boldsymbol{A})$ the constrained FKS adversary as defined above. It follows from Lem. 3, that* $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FDS}_0^\pi}(\boldsymbol{A}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^\pi}(R_{\mathrm{FKS}}(\boldsymbol{A}))$.

We denote by $\mathcal{A}'_{q,\ell}$ the set of constrained adversaries against FKS, that were induced by some FDS adversary that makes q initialize calls and duplexes ℓ blocks after each initialization:

$$\mathcal{A}'_{q,\ell} = \{R_{\mathrm{FKS}}(\boldsymbol{A}) \ : \ \boldsymbol{A} \text{ an FDS adversary with resources } (q,\ell)\}.$$

Lemma 4. *Let $b, r, c > 0$ be such that $b = r + c$. Let* FKS *be the scheme of Sect. 3.1. Then,*

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^\pi}(\boldsymbol{A'}) \leq \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c},$$

for any constrained adversary $\boldsymbol{A'} \in \mathcal{A}'_{q,\ell}$

Proof. We will to large extent follow the notation and conventions from the proof of Lem. 2. We assume that every query is already padded and ends with a non-zero final b-bit block with m_i being the number of b-bit blocks in the query M_i. The structure of the queries and the number of squeezed bits will however differ.

Any adversary $\boldsymbol{A'} \in \mathcal{A}'_{q,\ell}$ makes exactly $q\ell$ FKS queries but these queries comprise at most $q\ell$ unique b-bit blocks. Moreover, these queries follow a certain pattern. We have that for every $1 \leq i \leq q$ and every $2 \leq j \leq \ell$:

$$M_{\ell(i-1)+1} = M_i^1 \text{ and } M_{\ell(i-1)+j} = M_{\ell(i-1)+j-1} \parallel M_i^j$$

where all $M_i^j \in \{0,1\}^b$ are non-zero (due to padding). Note that we have $m_{\ell(i-1)+j} = j$. For every query, $\boldsymbol{A'}$ asks for no more than r output bits.

Because we know the specific structure of the adversarial queries made by $\boldsymbol{A'}$, the extended queries are now identical with the original queries. Indeed we have for $1 \leq i \leq q$ and $1 \leq j \leq \ell$ that $\bar{M}_{\ell(i-1)+j} = M_{\ell(i-1)+j}$. The internal in-states s_i^j and out-states t_i^j are defined the same way as before.

The RP-RF Switch. We will replace the random permutation $\pi \xleftarrow{\$} \mathrm{Perm}(b)$ by a random function $f \xleftarrow{\$} \mathrm{Func}(b)$ in the experiment. Although there are $q\sum_{j=1}^{\ell} j = q\ell(\ell+1)/2$ calls to π made throughout the experiment, the structure of the queries implies, that there will be at most $q\ell$ calls to π with unique input. Thus the switching will contribute the term $(q\ell)^2/2^b$ to the final bound by a standard hybrid argument. We have $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^\pi}(\boldsymbol{A'}) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^f}(\boldsymbol{A'}) + (q\ell)^2/2^b$.

Patarin's Coefficient-H Technique. This part of the proof relies heavily on the corresponding part of the proof of Lem. 2. We will show that $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{FKS}_0^f}(\boldsymbol{A'}) \leq (q\ell)^2/2^c$.

The two systems an adversary is trying to distinguish are FKS_0^f and RO_{FKS}. We will use the same definition of a transcript $\tau = (\bar{M}_1, \ldots, \bar{M}_{q\ell}, T_1, \ldots, T_{q\ell})$ where $T_{\ell(i-1)+j}$ holds all the $j+1$ out-states appearing due to $\bar{M}_{\ell(i-1)+j}$ (including the dummy state $t^0_{\ell(i-1)+j}$). We will also use the same definition of a bad state (q.v. (1)). This will immediately give us $\Pr[D_X = \tau] / \Pr[D_Y = \tau] = 1$ for any $\tau \in \mathcal{T}_{\mathrm{good}}$ by a similar argument as in the proof of Lem. 2. The probability $\Pr[D_Y \in \mathcal{T}_{bad}]$ needs new investigation.

Bounding the Probability of a Bad Transcript in the Ideal World. We define the three possible labels of in-states, `free`, `initial` and `fixed` in the same way as before and we will work with the re-expressed definition of a bad state (2). Since the definitions of `free`, `initial` and `fixed` states are unchanged,

the probabilities of collision due to a pair of in-states $s_i^j, s_{i'}^{j'}$ with $\mathsf{stamp}_i^j = \texttt{free}$, $\mathsf{stamp}_i^j = \texttt{initial}$ and $\mathsf{stamp}_i^j = \texttt{fixed}$ do not change. The only thing that really changes is the final counting.

For any $1 \leq i \leq q$, the query $\bar{M}_{\ell(i-1)+1} = M_i^1$ consists of a single block. Thus it only induces a single in-state with $\mathsf{stamp}^1_{\ell(i-1)+1} = \texttt{initial}$. Then for any $2 \leq j \leq \ell$, we have $\mathsf{llcp}_b\left(\bar{M}_{\ell(i-1)+j}, \bar{M}_{\ell(i-1)+j-1}\right) = j-1$, so there is at most one new in-state induced by $\bar{M}_{\ell(i-1)+j}$ and unaffected by the common prefix with previous queries. It is $s_{\ell(i-1)+j}$ and we always have $\mathsf{stamp}^j_{\ell(i-1)+j} = \texttt{free}$.

We see that, w.r.t. (2), for every value of i, ℓ states need to be considered, giving us a total amount of $q\ell$ possible tuples (i, j). For any such state $s_{\ell(i-1)+j}$, we need to count all other states (visited by (i', j') in (2)) with which it can collide. For any $i' < i$, it suffices to check equality of $s_{\ell(i-1)+j}$ with all ℓ in-states induced by $\bar{M}_{\ell(i'-1)+\ell}$, as every other query $\bar{M}_{\ell(i'-1)+j'}$ is its prefix. For $i' = i$, it suffices to look at in-states induced by $\bar{M}_{\ell(i-1)+j-1}$. Thus for any state $s_{\ell(i-1)+j}$, there are no more than $q\ell$ unique states, with which it can collide. Using the collision probabilities from the proof of Lem. 2, we conclude that $\Pr\left[D_Y \in \mathcal{T}_{bad}\right] \leq (q\ell)^2/2^c$. □

7 Full-State SpongeWrap and its Security

Our results from Sect. 6 can be used to prove security of modified, more efficient versions of existing Sponge-based AE schemes. As an interesting instance, we introduce Full-state SpongeWrap, a variant of the authenticated encryption mode SpongeWrap [8,10], offering improved efficiency with respect to processing of associated data (AD).

7.1 Authenticated Encryption for Sequences of Messages

We will focus on authenticated encryption schemes that act on sequences of AD-message pairs. Following Bertoni et al. [8, 10]we will think of an authenticated encryption scheme as an object W surfacing three APIs:

- W.initialize(K, N): calling this function will initialize W with a secret key from the set of keys $\mathcal{K}$ and a nonce from the set of nonces $\mathcal{N}$.
- W.wrap(A, M): this function inputs an AD-message pair (A, M) and outputs a ciphertext-tag pair (C, T), where $|C| = |M|$ and T is a τ-bit tag authenticating (A, M) and all the queries processed by W so far (i.e. since the last initialization call).
- W.unwrap(A, C, T): this function accepts a triple of AD, ciphertext and tag, and outputs a message M if C is an encryption of M and T is a valid tag for (A, M), and all the previous queries processed by W so far; otherwise it outputs an error symbol $\perp$.

Here, the AD, messages and ciphertexts are finite strings and we have $|C| = |M|$. τ is a positive integer and we call it the expansion of W. We require that W

is initialized before making the first wrapping or unwrapping call. For a given key K, we will use W_K to refer to the corresponding keyed instance, omitting K from the list of inputs; that is, $W.\text{initialize}(K, N) = W_K.\text{initialize}(N)$.

Security of Authenticated Encryption. We follow Bertoni et al. [8, 10] for defining the security of AE. We split the twofold security goal of AE into two separate requirements: privacy and authenticity.

Let W be a scheme for authenticated encryption, as described above, that internally makes calls to a public random permutation p. We formalize the privacy of W by an experiment in which an adversary $\boldsymbol{A}$ is given access to p, p^{-1} and an oracle O that provides two interfaces: $O.\text{initialize}(N)$ and $O.\text{wrap}(A, M)$. We have $O \in \{W_K, RO_W\}$, where W_K is an instance of the real scheme with the key K, and RO_W is an ideal primitive that acts as follows: it keeps a *list* of strings $St \in (\{0,1\}^*)^*$ as its internal state. On calling $RO_W.\text{initialize}(N)$ the list St is set to the empty list and then the nonce N is added to the list (denote this operation by $St \leftarrow St||N$); now each call $RO_W.\text{wrap}(A, M)$ will first update the list as $St \leftarrow St||(A, M)$ and then will output $\mathsf{left}_{|M|+\tau}\left(RO^{\infty}(\langle St\rangle)\right)$, where $\langle St\rangle$ denotes an *injective* encoding of the list St into a *string* in $\{0,1\}^*$. (Note that the list St preserves the boundaries between N and all the queried AD-message pairs.)

The adversary must distinguish between the two worlds: the real world where it is interacting W_K and the ideal world where it is interacting with RO_W. The advantage of the adversary in doing so is defined as

$$\mathbf{Adv}^{\text{priv}}_{W[p]}(\boldsymbol{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} \;:\; \boldsymbol{A}^{W_K, p, p^{-1}} \Rightarrow 1\right] - \Pr\left[\boldsymbol{A}^{RO_W, p, p^{-1}} \Rightarrow 1\right]\right|.$$

It is assumed that the adversary meets the following *nonce-requirement*: if it has queried the sequence $(N, (A_1, M_1), \ldots, (A_{n-1}, M_{n-1}), (A_n, M_n))$ then it will make no subsequent query of $(N, (A_1, M_1), \ldots, (A_{n-1}, M_{n-1}), (A'_n, M'_n))$ with $(A'_n, M'_n) \neq (A_n, M_n)$. Note that this nonce-requirement is more relaxed than the traditional nonce-respecting requirement mandating that the nonce N must not be repeated for any individual AD-message pair. To put it another way, the nonce-requirement here means that N must not be repeated for a whole sequence of AD-message pairs.

For the definition of authenticity property, consider an experiment where an adversary $\boldsymbol{A}$ is given access to the oracle W_K and is allowed to ask the queries $W_K.\text{initialize}(N)$ and $W_K.\text{wrap}(A, M)$. It is assumed that $\boldsymbol{A}$ respects the *nonce-requirement* in the wrapping queries. $\boldsymbol{A}$ is again allowed to query p. Adversary's goal is to forge; i.e. to output a sequence $(N, (A_1, C_1, T_1), \ldots, (A_n, C_n, T_n))$ at the end of the experiment such that after calling $W.\text{initialize}(K, N)$ and then $W.\text{unwrap}(A_i, C_i, T_i)$ for $1 \leq i \leq n-1$, $W.\text{unwrap}(A_n, C_n, T_n)$ does not return $\perp$. The sequence $(N, (A_1, C_1, T_1), \ldots, (A_n, C_n, T_n))$ must be such that the adversary has not obtained (C_n, T_n) from a wrapping query that followed an initialization with N and a series of wrapping queries $(A_1, M_1), \ldots, (A_n, M_n)$ with some $M_1, \ldots, M_n$. Note that the adversary does not have to use a unique

nonce in the forgery. We define the advantage of $\boldsymbol{A}$ as

$$\mathbf{Adv}^{\mathrm{auth}}_{W[p]}(\boldsymbol{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} \,:\, \boldsymbol{A}^{W_K, p, p^{-1}} \text{ forges}\right].$$

We let $\mathbf{Adv}^{\mathrm{priv}}_{W[p]}(q, \ell, \mu, N) = \max_{\boldsymbol{A}} \mathbf{Adv}^{\mathrm{priv}}_{W[p]}(\boldsymbol{A})$ be the maximum advantage over all adversaries that make q initialize queries to the left oracle, and after each initialization do wrapping queries that induce at most ℓ permutation calls (including the initialization) and with total maximal multiplicity μ, and that make N direct queries to the public permutation. We similarly let $\mathbf{Adv}^{\mathrm{auth}}_{W[p]}(q, \ell, \mu, N) = \max_{\boldsymbol{A}} \mathbf{Adv}^{\mathrm{auth}}_{W[p]}(\boldsymbol{A})$.

7.2 Full-State SpongeWrap

The Full-State SpongeWrap (FSW) is a permutation mode for authenticated encryption of AD-message sequences as described in Sect. 7.1. It is parameterized by a b-bit permutation p, the maximal message block size r, the key size k, the nonce size n, and the tag size $\tau > 0$. We require that $k \leq b - r =: c$ and $n < r$. The set of keys is $\mathcal{K} = \{0,1\}^k$ and the set of nonces is $\mathcal{N} = \{0,1\}^n$. The FSW construction uses an instance of FDS internally to process the inputs block by block. To ensure domain separation of different stages of processing a query, we use three *frame bits* placed at the same position in each duplexing call to FDS as explained in Table 1.

To keep the description of FSW compact, we introduce the following notations. For any $L \in \{0,1\}^{\leq r}$, $R \in \{0,1\}^{\leq c-5}$ and $F \in \{0,1\}^3$, we let

$$Q(L, F, R) = \mathrm{pad}_{r+1}(L) \parallel F \parallel R. \tag{3}$$

Note that $r+4 \leq |Q(L,F,R)| \leq b-1$ for any L, F, R. We let $(L, R) = \mathsf{lsplit}(X, n)$ for any $X \in \{0,1\}^*$ such that $L = \mathsf{left}_{\min(|X|,n)}(X)$ and $\mathsf{right}_{|X|-|L|}(X)$. We will use the abbreviation $D.\mathrm{dpx}(M, z)$ for the interface $D.\mathrm{duplexing}(M, z)$ of an FDS D. The interfaces of FSW$[p, r, k, n, \tau]$ are defined in Algo. 3.

label	value	usage
F_{N}	000	process nonce, derive initial mask of a query
F_{AM}	001	block of A and M inside query
F_{M}	010	block of M inside query
F_{A}	011	block of A inside query
$F_{\mathrm{AM}\vert}$	100	last block of A and M inside query
$\bar{F}_{\mathrm{AM}}$	101	last block of A and M, query ends, produces tag
$\bar{F}_{\mathrm{M}}$	110	last block of M, query ends, produces tag
$\bar{F}_{\mathrm{A}}$	111	last block of A, query ends, produces tag

Table 1: Labeling and usage of the frame bits in FSW.

Algorithm 3 FSW$[p, r, k, n, \tau]$

Interface W.initialize(K, N)
  D.initialize(K)
  $S \leftarrow \mathrm{pad}_r(N) \parallel 0 \parallel F_{\mathrm{N}} \parallel 0^{c-5}$
  $Z \leftarrow D.\mathrm{dpx}(S, r)$

Interface W.wrap(A, M)
  $M_1 \parallel \ldots \parallel M_m \xleftarrow{r} M$
  $(A', A^*) \leftarrow \mathsf{lsplit}(A, m(c-5))$
  $A'_1 \parallel \ldots \parallel A'_{a'} \xleftarrow{c-5} A'$
  $A^*_1 \parallel \ldots \parallel A^*_{a^*} \xleftarrow{b-5} A^*$
  if $m = a' = a^* = 0$ **then**
    $T \leftarrow \varepsilon$
    $F \leftarrow \bar{F}_{\mathrm{A}}$
  for $i \leftarrow 1$ **to** $a' - 1$ **do**
    $C_i \leftarrow M_i \oplus Z$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_i, F_{\mathrm{AM}}, A'_i), r)$
  if $0 < a' < m$ **or** $0 < a', a^*$ **then**
    $C_{a'} \leftarrow M_{a'} \oplus \mathsf{left}_{|M_{a'}|}(Z)$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_{a'}, F_{\mathrm{AM}|}, A'_{a'}), r)$
  else if $0 < m = a'$ **and** $a^* = 0$ **then**
    $C_{a'} \leftarrow M_{a'} \oplus \mathsf{left}_{|M_{a'}|}(Z)$
    $T \leftarrow D.\mathrm{dpx}(Q(M_{a'}, \bar{F}_{\mathrm{AM}}, A'_{a'}), r)$
    $F \leftarrow \bar{F}_{\mathrm{AM}}$
  for $i \leftarrow a' + 1$ **to** $m - 1$ **do**
    $C_i \leftarrow M_i \oplus Z$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_i, F_{\mathrm{M}}, \varepsilon), r)$
  if $a' < m$ **then**
    $C_m \leftarrow M_m \oplus \mathsf{left}_{|M_m|}(Z)$
    $T \leftarrow D.\mathrm{dpx}(Q(M_m, \bar{F}_{\mathrm{M}}, \varepsilon), r)$
    $F \leftarrow \bar{F}_{\mathrm{M}}$
  for $i \leftarrow 1$ **to** $a^* - 1$ **do**
    $(L, R) \leftarrow \mathsf{lsplit}(A^*_i, r)$
    $D.\mathrm{dpx}(Q(L, F_{\mathrm{A}}, R), 0)$
  if $a^* > 0$ **then**
    $(L, R) \leftarrow \mathsf{lsplit}(A^*_{a^*}, r)$
    $T \leftarrow D.\mathrm{dpx}(Q(L, \bar{F}_{\mathrm{A}}, R), r)$
    $F \leftarrow \bar{F}_{\mathrm{A}}$
  while $|T| < \tau$ **do**
    $T \leftarrow T \parallel D.\mathrm{dpx}(Q(\varepsilon, F, \varepsilon), r)$
  $Z \leftarrow D.\mathrm{dpx}(Q(\varepsilon, F_{\mathrm{N}}, \varepsilon), r)$
  $C \leftarrow C_1 \parallel \ldots \parallel C_m$
  return $C, \mathsf{left}_\tau(T)$

Interface W.unwrap(A, C, T)
  $C_1 \parallel \ldots \parallel C_m \xleftarrow{r} C$
  $(A', A^*) \leftarrow \mathsf{lsplit}(A, m(c-5))$
  $A'_1 \parallel \ldots \parallel A'_{a'} \xleftarrow{c-5} A'$
  $A^*_1 \parallel \ldots \parallel A^*_{a^*} \xleftarrow{b-5} A^*$
  if $m = a' = a^* = 0$ **then**
    $T' \leftarrow \varepsilon$
    $F \leftarrow \bar{F}_{\mathrm{A}}$
  for $i \leftarrow 1$ **to** $a' - 1$ **do**
    $M_i \leftarrow C_i \oplus Z$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_i, F_{\mathrm{AM}}, A'_i), r)$
  if $0 < a' < m$ **or** $0 < a', a^*$ **then**
    $M_{a'} \leftarrow C_{a'} \oplus \mathsf{left}_{|C_{a'}|}(Z)$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_{a'}, F_{\mathrm{AM}|}, A'_{a'}), r)$
  else if $0 < m = a'$ **and** $a^* = 0$ **then**
    $M_{a'} \leftarrow C_{a'} \oplus \mathsf{left}_{|C_{a'}|}(Z)$
    $T' \leftarrow D.\mathrm{dpx}(Q(M_{a'}, \bar{F}_{\mathrm{AM}}, A'_{a'}), r)$
    $F \leftarrow \bar{F}_{\mathrm{AM}}$
  for $i \leftarrow a' + 1$ **to** $m - 1$ **do**
    $M_i \leftarrow C_i \oplus Z$
    $Z \leftarrow D.\mathrm{dpx}(Q(M_i, F_{\mathrm{M}}, \varepsilon), r)$
  if $a' < m$ **then**
    $M_m \leftarrow C_m \oplus \mathsf{left}_{|C_m|}(Z)$
    $T' \leftarrow D.\mathrm{dpx}(Q(M_m, \bar{F}_{\mathrm{M}}, \varepsilon), r)$
    $F \leftarrow \bar{F}_{\mathrm{M}}$
  for $i \leftarrow 1$ **to** $a^* - 1$ **do**
    $(L, R) \leftarrow \mathsf{lsplit}(A^*_i, r)$
    $D.\mathrm{dpx}(Q(L, F_{\mathrm{A}}, R), 0)$
  if $a^* > 0$ **then**
    $(L, R) \leftarrow \mathsf{lsplit}(A^*_{a^*}, r)$
    $T' \leftarrow D.\mathrm{dpx}(Q(L, \bar{F}_{\mathrm{A}}, R), r)$
    $F \leftarrow \bar{F}_{\mathrm{A}}$
  while $|T'| < \tau$ **do**
    $T' \leftarrow T' \parallel D.\mathrm{dpx}(Q(\varepsilon, F, \varepsilon), r)$
  $Z \leftarrow D.\mathrm{dpx}(Q(\varepsilon, F_{\mathrm{N}}, \varepsilon), r)$
  $M \leftarrow M_1 \parallel \ldots \parallel M_m$
  if $T = \mathsf{left}_\tau(T')$ **then**
    return M
  else
    return $\perp$

7.3 Security of FSW

The security of FSW is relatively easy to analyze, thanks to the result from Sect. 6.

Lemma 5. *Let* $W = \mathrm{FSW}[p, r, k, n, \tau]$ *be an instance of* FSW *as described in Sect. 7.2. Denote any query to* W.initialize *and a list of subsequent queries to* W.wrap *by* $(N, (A_1, M_1), \ldots, (A_n, M_n))$*. Then,* FSW *injectively maps this sequence to a sequence of corresponding* FDS *duplexing queries* $(Q_1, \ldots, Q_d)$*.*

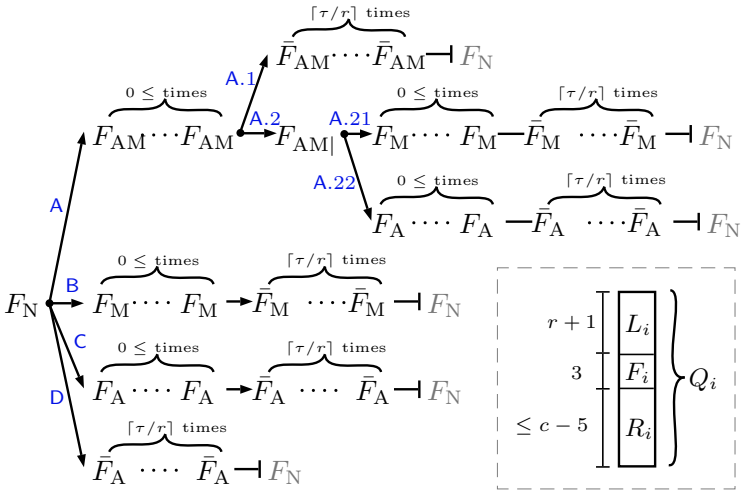

Fig. 4: The tree of all possible frame bits sequences for a single AD-message pair (top-left). The composition of an FDS query Q_i (bottom-right).

Proof. We prove the injectivity of the mapping by showing how it can be inverted. We refer to the mapping Q of (3) to argue that every Q_i can be split into three strings L_i, F_i, R_i with $|L_i| = r + 1$, $|F_i| = 3$ and $|R_i| \leq c - 5$ just as depicted in Fig. 4. The main trick is to use the frame bits used in FSW to determine boundaries of wrapping queries and their logical parts. We will refer to the FDS queries as "frames".

We can recover the AD-message pairs (in the following just "pair") from $\mathbf{Q} = (Q_1, \ldots, Q_d)$ in a left-to-right fashion. Any pair (A, M) is encoded in a subsequence of $\mathbf{Q}$ that starts by a frame with frame bits F_N and ends by a frame just before the next frame with frame bits F_N. Depending on the lengths of A and M, the pattern of frame bits between these boundary frames can differ as depicted in Fig. 4.

If both A and M are non-empty, we follow the edge marked as A. If there is the same number of r-bit blocks in M as there is of $c - 5$ bit blocks in A, then

we follow the path A.1. Otherwise we follow the path A.2 and then A.21 if there were fewer blocks in A than in M and the path A.22 if there were in turn more blocks in A than in M.

If $M \neq A = \varepsilon$, then we follow the path B; if $A \neq M = \varepsilon$ we follow the path C. In a special case, where both $A = M = \varepsilon$, we follow path D. We can see, that every possible case of lengths of M and A in terms of blocks yields a distinct pattern of frame bit sequences.

Having identified which path in Fig. 4 we are following, we can recover A and M. Every frame Q_i with $F_i \in \{F_{\mathrm{AM}}, F_{\mathrm{AM|}}\}$ holds a padded block of M in L_i and an unpadded block of A in R_i. If $F_i = F_{\mathrm{M}}$, then there is a padded block of M in L_i and $R_i = \varepsilon$. If $F_i = F_{\mathrm{A}}$, then there is a padded block of A in L_i and another unpadded block of A in R_i. The frames with $F_i \in \{\bar{F}_{\mathrm{AM}}, \bar{F}_{\mathrm{M}}, \bar{F}_{\mathrm{A}}\}$ are used to produce the tag and are thus treated specially. The first frame with $\bar{F}_\chi$ holds data blocks and the following ones do not. If $\chi = \mathrm{AM}$, then there is a padded block of M in L_i and an unpadded block of A in R_i. If $\chi = \mathrm{M}$, then there is only a padded block of M in L_i. If $\chi = \mathrm{A}$ and we are not on path D then there is a padded block of A in L_i and a following unpadded block of A in R_i. If we are on path D then none of the frames holds any data, since both A and M are empty.

Once we extract all the blocks of A and M, we concatenate them all in the order in which they were extracted to obtain A and M. We note that the nonce is contained in the very first frame with $F_1 = F_{\mathrm{N}}$ as $L_1 = \mathrm{pad}_r(N) \,\|\, 0$. □

Theorem 3. *Let $b, r, c, k, n, \tau > 0$ be such that $b = r + c$, $k \leq c$ and $n < r$. Let* FSW *be the scheme of Sect. 7.2. Then,*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{FSW}}(q, \ell, \mu, N) \leq \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c} + \frac{\mu N}{2^k},$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{FSW}}(q, \ell, \mu, N) \leq \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c} + \frac{\mu N}{2^k} + \frac{1}{2^\tau}.$$

Proof. We start by defining the *RO*FSW—an idealized FSW that internally uses the RO^r_{FDS} instead of FDS (and thus does not use p at all). By Thm. 2 we have that

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{FSW}}(q, \ell, \mu, N) \leq \mathbf{Adv}^{\mathrm{priv}}_{\mathit{RO}\mathrm{FSW}}(q, \ell, \mu) + \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c} + \frac{\mu N}{2^k},$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{FSW}}(q, \ell, \mu, N) \leq \mathbf{Adv}^{\mathrm{auth}}_{\mathit{RO}\mathrm{FSW}}(q, \ell, \mu) + \frac{(q\ell)^2}{2^b} + \frac{(q\ell)^2}{2^c} + \frac{\mu N}{2^k}.$$

By Lem. 5, we know that a unique sequence of a nonce and AD-message pairs yields unique sequence of RO^r_{FDS} queries. We have that $\mathbf{Adv}^{\mathrm{priv}}_{\mathit{RO}\mathrm{FSW}}(q, \ell, \mu) = 0$. This is because the nonce requirement implies that every *RO*FSW.wrap(A, M) query is either processed using RO^r_{FDS} with a unique internal state, or the previous query processed with RO^r_{FDS} with the same state was (A, M).

In order to forge, the adversary must produce a sequence, which ends by a triplet (A, C, T) that did not appear in the experiment before and that passes the

authentication check. Uniqueness of the triplet implies that either C or A or T is different from a triplet that appeared in the experiment. If only T is modified, then it cannot be correct. In any other case, the tag is compared to outputs of RO^r_{FDS} evaluated with a fresh combination of internal state and inputs. We have $\mathbf{Adv}^{\mathrm{auth}}_{RO\mathrm{FSW}}(q, \ell, \mu) \leq 2^{-\tau}$. □

8 Discussion

Related-Key Security. Our treatment of the security of the full-state constructions is in the traditional model where the adversary has no control over selection of the secret keys or relations among different keys. If one considers the stronger model of related-key attack security then care must be taken in utilizing these schemes. Indeed, if an adversary has access to two instances $F_1 = \mathrm{FKS}^p_{K_1}$ and $F_2 = \mathrm{FKS}^p_{K_2}$, and it knows the relation $\Delta = K_1 \oplus K_2$, then it can make the outputs of F_1 and F_2 collide trivially by asking two b-bit queries $F_1(M)$ and $F_2(M \oplus \Delta)$.

Although it is outside the scope of this paper to treat related-key security thoroughly, we informally propose some easy solutions to prevent trivial related-key attacks like the one mentioned before. We start by noticing that the inner-keyed Sponge construction [2] is not susceptible to this problem, as the secret key and the adversarial data blocks never overlap; hence, a simple way of thwarting such trivial related-key attacks is to always prepend the input data with a block of b zeroes. Thus the adversary can no longer xor an arbitrary value directly to the key prior to the application of the permutation. If the original adversarial resources were (q, ℓ, μ, N), we can without any further argumentation use the bound with the resources $(q, \ell + 1, \mu, N)$ for this new construction.

Another possibility would be to slightly modify the constructions and partition the input data into an r-bit starting block and b-bit blocks afterward. The initial block would be xored to the outer r bits of the initial state. Our security analysis would carry over to this construction with minimal modifications.

Generalized Security Model. The security analyses of FKS and FDS cover those of the original Sponge and Duplex constructions as special cases. Beyond that, for the security analysis of FDS itself, we have generalized the security model of the original Duplex construction from Bertoni et al. [9, 10]. In more detail, Bertoni et al. assume that the Duplex is initialized only once and then the adversary can only do duplexing queries, while we allow the adversary to force FDS to be reinitialized up to q times and then allow it to duplex up to ℓ blocks after each initialization.

This generalized setting seems more closely matching the use of the Duplex construction in several AE schemes which do not require sessions and new session keys, where one would initialize the Duplex (or FDS) construction for every query. This is well demonstrated by the example of FSW. More precisely, the way we design and analyze the security of FSW allows for a very versatile use. FSW can be used to secure AD-message pairs in a single *session* [12], i.e. using

a single initialize call during the lifetime of the key or alternatively every AD-message pair can be preceded by an initialize call with a unique nonce. In fact, FSW can be used for anything between these two extremes; for example, a setting where every AD-message pair is processed with a unique nonce, but can get fragmented into smaller sub-pairs. The security analysis of FSW covers each of these use cases.

On the Keying of the Sponge. As we have claimed in the introduction, the difference in the security of the outer-keyed and inner-keyed Sponges vanishes in presence of the full state absorption. On one hand, using a key of more than c bits does not increase the security level, as the extra bits cannot be used by the low-entropy Even-Mansour construction. On the other hand, absorbing several b-bit blocks of the key only results into a derived key of effective length of c bits. We remark that both the outer- and inner-keyed Sponges can be seen as special cases of FKS, by using more restrictive padding rules that only place the message blocks in the outer part of the state.

Boosting Sponge-based AE. Out of 57 CAESAR candidates, 10 are using a Sponge-based design. The method we used to enhance SpongeWrap can be straightforwardly adjusted to boost the performance of five of these 10 schemes: Keyak, Ketje, STRIBOB, CBEAM and ICEPOLE [3]. This is because all the said designs are using frame bits for domain separation. The other designs cannot benefit from our modifications, either due to a domain separation method relying on intangibility of the inner part of the state (NORX) or due to producing tag from the inner part of the state (Ascon, Primates) [3].

Acknowledgments. This work was partially supported by Microsoft Research under MRL Contract No. 2014-006 (DP1061305). This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007). Bart Mennink is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO).

References

1. Abed, F., Forler, C., Lucks, S.: Classification of the CAESAR candidates. IACR Cryptology ePrint Archive 2014, 792 (2014), http://eprint.iacr.org/2014/792
2. Andreeva, E., Daemen, J., Mennink, B., Van Assche, G.: Security of Keyed Sponge Constructions Using a Modular Proof Approach. In: Leander, G. (ed.) FSE 2015. LNCS (To appear), Springer (2015)
3. Bernstein, D.J.: Cryptographic competitions: CAESAR. http://competitions.cr.yp.to
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge Functions. ECRYPT Hash Workshop 2007, http://csrc.nist.gov/groups/ST/hash/documents/JoanDaemen.pdf
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak Specifications. NIST SHA-3 Submission (2008), http://keccak.noekeon.org/

6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer (2008)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge-Based Pseudo-Random Number Generators. In: Mangard, S., Standaert, F. (eds.) CHES 2010. LNCS, vol. 6225, pp. 33–47. Springer (2010)
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. IACR Cryptology ePrint Archive 2011, 499 (2011), http://eprint.iacr.org/2011/499
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Security of the Keyed Sponge Construction. In: Symmetric Key Encryption Workshop 2011 (2011)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2012)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-Based Encryption, Authentication and Authenticated Encryption. In: Workshop Records of DIAC 2012. pp. 159–170 (2012)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Keyak v1 (Mar 2014), http://competitions.cr.yp.to/round1/keyakv1.pdf
13. Chang, D., Dworkin, M., Hong, S., Kelsey, J., Nandi, M.: A Keyed Sponge Construction with Pseudorandomness in the Standard Model. NIST SHA-3 2012 Workshop, http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/documents/papers/CHANG_paper.pdf
14. Chen, S., Steinberger, J.P.: Tight Security Bounds for Key-Alternating Ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer (2014)
15. Even, S., Mansour, Y.: A Construction of a Cipher From a Single Pseudorandom Permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT '91. LNCS, vol. 739, pp. 210–224. Springer (1993)
16. Even, S., Mansour, Y.: A Construction of a Cipher from a Single Pseudorandom Permutation. J. Cryptology 10(3), 151–162 (1997)
17. Gaži, P., Pietrzak, K., Tessaro, S.: The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS (To appear), Springer (2015)
18. Jovanovic, P., Luykx, A., Mennink, B.: Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014 - Part I. LNCS, vol. 8873, pp. 85–104. Springer (2014)
19. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer (2004)
20. Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In: Joux, A., Youssef, A.M. (eds.) SAC 2014. LNCS, vol. 8781, pp. 306–323. Springer (2014)
21. Patarin, J.: The "Coefficients H" Technique. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer (2009)
22. Perlner, R.: Extendable-Output Functions (XOFs). NIST SHA-3 2014 Workshop, http://csrc.nist.gov/groups/ST/hash/sha-3/Aug2014/documents/perlner_XOFs.pdf

23. Reyhanitabar, R., Vaudenay, S., Vizár, D.: Boosting OMD for Almost Free Authentication of Associated Data. In: Leander, G. (ed.) FSE 2015. LNCS (To appear), Springer (2015)
24. Rivest, R.L., Schuldt, J.C.N.: Spritz – a Spongy RC4-like Stream Cipher and Hash Function (2014), `https://people.csail.mit.edu/rivest/pubs/RS14.pdf`
25. Sasaki, Y., Yasuda, K.: How to Incorporate Associated Data in Sponge-Based Authenticated Encryption. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 353–370. Springer (2015)