Understanding the Duplex and Its Security

Bart Mennink

Radboud University, Nijmegen, The Netherlands

Abstract. At SAC 2011, Bertoni et al. introduced the keyed duplex construction as a tool to build permutation based authenticated encryption schemes. The construction was generalized to full-state absorption by Mennink et al. (ASIACRYPT 2015). Daemen et al. (ASIACRYPT 2017) generalized it further to cover much more use cases, and proved security of this general construction, and Dobraunig and Mennink (ASIACRYPT 2019) derived a leakage resilience security bound for this construction. Due to its generality, the full-state keyed duplex construction that we know today has plethora applications, but the flip side of the coin is that the general construction is hard to grasp and the corresponding security bounds are very complex. Consequently, the state-of-the-art results on the full-state keyed duplex construction are not used to the fullest. In this work, we revisit the history of the duplex construction, give a comprehensive discussion of its possibilities and limitations, and demonstrate how the two security bounds (of Daemen et al. and Dobraunig and Mennink) can be interpreted in particular applications of the duplex.

Keywords: sponge · duplex · permutation · applications · MAC · authenticated encryption.

1 Introduction

Since the introduction of the sponge hash construction by Bertoni et al. [13] in 2007, permutation based cryptography immensely gained in popularity. The sponge hash function construction operates on a b-bit state, split into an inner part of c bits, called the "capacity", and an outer part of r bits, called the "rate". To hash some data P, the data is first injectively padded into r-bit blocks, which are then absorbed to the outer part block by block, interleaved by a b-bit cryptographic permutation p. After the last block, a squeezing phase starts, where the outer part of the state is output iteratively, until a sufficient amount of digest bits are squeezed. As such, the sponge construction is not just a hash function, but rather a hash function with variable output lengths (later called an extendable output function, or XOF).

As plain hash function construction, the sponge immediately faced rapid adoption around the NIST SHA-3 competition [66]: multiple candidates were inspired by the sponge methodology, and the eventual winner Keccak that is now standardized as SHA-3 [65] internally uses the actual sponge construction. In part due to its minimalist and simple design, the sponge construction is also a popular approach for lightweight hashing [5, 21, 50]. In the ongoing NIST

Lightweight Cryptography competition [67], 5 out of the 10 finalists [6, 7, 31, 34, 37] support hashing using the sponge or a sponge-like construction.

Bertoni et al. [14] proved in 2008 that the sponge construction is hard to differentiate from a random oracle, in the indifferentiability framework of Maurer et al. [56] and Coron et al. [30], provided that the permutation is assumed to be perfectly random and the adversary cannot make more than around $2^{c/2}$ permutation evaluations. This is a powerful result: it implies that the sponge construction behaves like a random oracle and can replace it as such in many applications, provided at most $2^{c/2}$ permutation evaluations are made. It also implies that the "conventional" hash function attacks like finding collisions, preimages, and second preimages for the sponge construction are not easier than for the random oracle, up to $2^{c/2}$ evaluations [4, Appendix A]. (However, refer to Lefevre and Mennink [54] for an improved security bound on the preimage resistance of the sponge construction.) Another implication of the sponge function indifferentiability result is that the construction can be easily used for keyed applications, such as keystream generation and MAC computation [17], reseedable pseudorandom sequence generation [15, 45], and authenticated encryption [16, 19]. For example, one can easily build a MAC construction out of the sponge by simply concatenating key K and plaintext P, as in $K\|P$, and feeding it to the hash construction [18]. (This construction later became known as the "outer-keyed sponge construction.") Security of this, and other, constructions follows from the hash function indifferentiability result, provided the number of permutation evaluations does not exceed $2^{c/2}$.

However, despite its power, it was quickly acknowledged that the indifferentiability bound also has its limitation: the $2^{c/2}$ bound is tight for unkeyed use cases, i.e., for plain cryptographic hashing, but for keyed use cases a higher level of security could be achieved. This is because in keyed applications, one must make a distinction between permutation evaluations known to the adversary and permutation evaluations unknown to the adversary. This fact has led to an impressive amount of research on the generic security of keyed versions of the sponge constructions, all for slightly differing constructions and with slightly differing bounds. Chang et al. [24] suggested an alternative to simply hashing $K\|P$, namely one where the initial state of the sponge contains the key in its inner part. Andreeva et al. [3] generalized and improved the analyses of both constructions, since then called the outer-keyed and inner-keyed sponge, and also analyzed security in the multi-user setting. Naito and Yasuda [63] developed an improved security analysis of these constructions. Finally, whereas these two constructions stayed reasonably close to the original sponge hash function design, it was quickly acknowledged that, due to the secrecy of the sponge state after key injection, one can absorb data over the entire b-bit state, and therewith maximize the absorption rate. The idea of full-state absorption was first suggested in the donkeySponge MAC construction [19]. Security analyses for fixed output length were given by Gaži et al. [44], and for variable output length and a general description of the full-state keyed sponge by Mennink et al. [61].

These constructions, the outer-keyed, inner-keyed, and full-state keyed sponge, are mostly relevant for message authentication and stream encryption. Authenticated encryption, on the other hand, is typically done via the keyed duplex construction. The keyed duplex dates back to a work of Bertoni et al. [16] from 2011, and is a stateful sponge-inspired construction that consists of an initialization interface and a duplexing interface. The initialization interface resets the state, and the duplexing interface can be seen as a 1-round sponge on the current state: it absorbs data, permutes the state, and squeezes data. By subsequent application of the duplexing interface for different types of inputs and outputs, one can build an authenticated encryption scheme, like the SpongeWrap construction of Bertoni et al. [16]. Also this duplex construction has undergone various improvements. Mennink et al. [61] introduced the full-state keyed duplex, where data is absorbed over the entire state (but squeezing only happens from the outer part). Daemen et al. [32] generalized the construction even further, allowing arbitrary length keys, covering multi-user security by design, and most importantly, allowing the user to not only absorb data but also *overwrite* the outer part of the state with data. This additional feature made the scheme much more complex, but was needed to make the general full-state duplex construction more broadly applicable (e.g., to bound the security in the case of release of unverified plaintext [2], see also Remark 5). Finally, Dobraunig and Mennink [40] considered a slight generalization of aforementioned full-state duplex, and analyzed security of the construction in the leakage resilience setting. Dobraunig and Mennink applied their result to the encryption ideas of Taha and Schaumont [73] and ISAP v2 [34, 36]. This result was later combined with the leakage resilience of the suffix keyed sponge construction [41] to obtain a security proof of the entire ISAP v2 construction under leakage [35].

Although the gradual generalization of the full-state duplex construction has its advantages, namely that it has the potential to apply to many applications, it also had a disadvantageous side-effect: the security bounds became harder to grasp and it is not immediately clear how to *interpret* the security results. For example, the security bound of Daemen et al. [32] was defined in terms of 6 adversarial complexity parameters, and the one of Dobraunig and Mennink [40] even in 7 parameters, some of which were rather obscure and required deep understanding of the construction to judge whether they were relevant or not in a specific use case. This has led to the fact that we have not seen many applications of the security bounds of Daemen et al. [32] and of Dobraunig and Mennink [40], despite that the full-state keyed duplex has been present, in disguise, in many applications over the years.

In a nutshell, we can conclude that the two general full-state duplex results of Daemen et al. [32] and of Dobraunig and Mennink [40], although they are very powerful, found very minimal applications. This seems to have two main reasons: (i) the general full-state duplex is too general and it is not immediately clear how it can be used in practical applications, and (ii) the security bounds are extremely complex and it is not clear when — and if so, how — the security bounds simplify for specific applications. In this work, we aim to give a comprehensive idea of the power of the versatile duplex construction and of the two bounds of Daemen et al. and Dobraunig and Mennink.

1.1 Understanding the Duplex Construction

As first part of the work, we have a fresh look at the duplex construction as we know it today, in particular the two very comparable constructions of Daemen et al. [32] and of Dobraunig and Mennink [40]. The constructions are, in fact, almost identical, barring two differences: (i) the construction of Dobraunig and Mennink allowed for use cases that rotated the initialized state, and (ii) the two schemes adopted a different type of "phasing", referring to what operations constitute a single duplex round.

The first difference makes the construction of Dobraunig and Mennink *slightly* more general, and we therefore adopt their scheme over that of Daemen et al., noting that the security result of Daemen et al. can be easily generalized to rotated initialized states.

The second difference between the constructions of Daemen et al. and Dobraunig and Mennink, regarding the phasing, sounds rather innocent. For example, Daemen et al. considered a single duplexing call to consist of a sequential operation of permute-squeeze-absorb, whereas Dobraunig and Mennink considered a single duplexing call to consist of a sequential operation of squeeze-absorb-permute. While the choice of phasing does not really change the security of the scheme, it is just about how to *interpret* the duplex, different interpretations might be useful for different applications of the duplex. In our attempt to fully understand the phasing of the duplex, and which choice of phasing might be best suited for most applications, we took a fresh look at how the duplex has actually grown from its original introduction in 2011 [16] to the currently known full-fledged construction, and observe that *all results so far* [16, 32, 40, 61] used different phasing. Even more surprising, by looking ahead at applications of the duplex that we consider in this work, we conclude that the most suitable phasing is *yet another one*: in this work, a duplexing call consists of a sequential application of permute-squeeze-absorb, and the initialization simply initializes the state (with no permutation call, unlike in the phasing of Daemen et al.).

In conclusion, the eventual construction that we adopt is the one of Dobraunig and Mennink, i.e., with rotatable initial state, but with our new rephasing. The construction is given in detail in Section 3.1. We also describe the ideal duplex in Section 3.2 and the duplex security model in Section 3.3, mostly following Dobraunig and Mennink [40] mutatis mutandis. An in-depth discussion on the phasing of the duplex, including a table summarizing all earlier and current phasing, is given in Section 3.4. In Section 4, we summarize, in full generality, the security results of Daemen et al. [32] and of Dobraunig and Mennink [40] in terms of our generalized construction of Section 3.1.

1.2 Understanding the Duplex Bound

The next step is to understand *how* the duplex can be used in applications. We demonstrate its actual power along the discussion of multiple different constructions, all of them either existing ones, direct derivatives of existing ones, or rather natural constructions in the first place. The constructions are described in detail, and for each of these constructions, we demonstrate the power of the duplex bounds of Daemen et al. [32] and of Dobraunig and Mennink [40] by mapping these bounds to simplified and tangible security bounds and implications.

In detail, in this work we look at the following constructions in detail:

- Truncated permutation in Section 5;
- Parallel keystream generation in Section 6;
- Sequential keystream generation in Section 7;
- Message authentication, and more specifically the full-state keyed sponge construction and an application to the Ascon-PRF [38] construction, in Section 8;
- Authenticated encryption, and more specifically a modernization of the SpongeWrap [16] construction, called MonkeySpongeWrap, in Section 9.

For some of these constructions, such as the keystream generation constructions, the security bounds are not very surprising but nevertheless never written out in detail. For others, including Ascon-PRF [38] and the authenticated encryption construction of Section 9, our new findings have immediate applications. Of particular interest is the modernization of SpongeWrap, noting that the original SpongeWrap construction as introduced and proven secure by Bertoni et al. [16] was too restrictive. Instead, many designers rather resorted to using the SpongeWrap construction inside the MonkeyDuplex of Bertoni et al. [19], often with some adjustments. This construction, however, was never formally analyzed and the original SpongeWrap proof did not carry over. The MonkeySpongeWrap construction that we describe and analyze in Section 9 more closely captures the applications, and has some other simplifications compared to the original SpongeWrap construction as well.

1.3 Outline

The remainder of the work starts with preliminaries, mostly discussing conventional PRF and AE security models, in Section 2. The duplex construction, including a discussion of its rationale, its ideal counterpart, and the duplex security model, in Section 3. The known security bounds of Daemen et al. [32] and of Dobraunig and Mennink [40] are discussed in full generality in Section 4. The various use cases of the duplex construction are given in Sections 5-9. The work is concluded in Section 10.

2 Preliminaries

Throughout, $\mathbb{N}$ includes 0. Let $n \in \mathbb{N}$ and $m \in \mathbb{N} \cup \{*\}$. The set of n-bit strings is denoted $\{0,1\}^n$. The set of arbitrarily length strings is denoted $\{0,1\}^*$, and the set of infinitely long strings is denoted $\{0,1\}^\infty$. We define by $\mathrm{func}(m,n)$ the set of all functions $\mathsf{f} : \{0,1\}^m \to \{0,1\}^n$, by $\mathrm{func}(n)$ the set of all functions $\mathsf{f} : \{0,1\}^n \to \{0,1\}^n$, and by $\mathrm{perm}(n)$ the set of all permutations $\mathsf{p} : \{0,1\}^n \to \{0,1\}^n$. By $X \leftarrow Y$ we denote the assignment of the value Y to X. For $X \in \{0,1\}^n$ and $Y \in \{0,1\}^m$, we denote by $X \oplus Y$ the bitwise exclusive OR of the first $\min\{m,n\}$ bits of X and Y.

For a finite set $\mathcal{X}$, we denote by $X \xleftarrow{\$} \mathcal{X}$ the uniformly random drawing of an element X from a finite set $\mathcal{X}$. We will slightly abuse this notation for sets with infinite inputs, and by $\mathsf{f} \xleftarrow{\$} \mathrm{func}(*,n)$ we denote a function f that for each new input generates a uniformly random string of length n bits.

For $P \in \{0,1\}^* \backslash \{\varnothing\}$, we denote by $\mathrm{pad}_n(P)$ the function that transforms P into blocks $P_1,\ldots,P_\ell$, where $|P_1|,\ldots,|P_{\ell-1}| = n$ and $0 < |P_\ell| \leq n$. For $P \in \{0,1\}^*$, we denote by $\mathrm{pad}_n^{10^*}(P) = \mathrm{pad}_n(P10^{-|P|-1 \bmod n})$, i.e., the function that first appends a 1 and a sufficient number of 0s to P, so that the last block will be of length n bits. For $X \in \{0,1\}^n$ and for $m \leq n$, we denote by $\mathrm{left}_m(X)$ the m leftmost bits of X and by $\mathrm{right}_m(X)$ the m rightmost bits of X. For $Y \in \{0,\ldots,2^n-1\}$, we denote by $\langle Y \rangle_n$ the encoding of Y as an n-bit string. We denote by $\mathrm{rot}_m(X)$ the right-rotation of X by m bits.

2.1 Distinguisher

A distinguisher D is an algorithm that is given access to one or more oracles O, denoted D^{O}, and that outputs a bit $b \in \{0,1\}$ after interaction with O. If O and P are oracles, we denote by

$$\Delta_{\mathsf{D}}\left(\mathsf{O}\ ;\ \mathsf{P}\right) = \left|\mathbf{Pr}\left(\mathsf{D}^{\mathsf{O}} \to 1\right) - \mathbf{Pr}\left(\mathsf{D}^{\mathsf{P}} \to 1\right)\right|$$

the advantage of a distinguisher D in distinguishing O from P. In our work, we will only be concerned with information-theoretic distinguishers: these have unbounded computational power, and their success probabilities are solely measured by the number of queries made to the oracles.

2.2 PRF Security

Let $b,k,t \in \mathbb{N}$ and $m \in \mathbb{N} \cup \{*\}$. Consider a function F that gets as input a key $K \in \{0,1\}^k$ and a plaintext $P \in \{0,1\}^m$, outputs a value $T \in \{0,1\}^t$, and that is instantiated using a b-bit permutation $\mathsf{p} \in \mathrm{perm}(b)$. We will consider multi-user security of F, where a distinguisher can query up to $\mu \geq 1$ versions of the scheme. The multi-user pseudorandom function (PRF) security of F against a distinguisher D is defined as

$$\mathbf{Adv}_{\mathsf{F}}^{\mu\text{-prf}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\left(\mathsf{F}[\mathsf{p}]_{K[j]}\right)_{j=1}^{\mu}, \mathsf{p}^{\pm}\ ;\ \left(\mathsf{R}_j^{\mathrm{prf}}\right)_{j=1}^{\mu}, \mathsf{p}^{\pm}\right), \qquad (1)$$

where $\boldsymbol{K} = (K[1],\ldots,K[\mu]) \xleftarrow{\$} (\{0,1\}^k)^\mu$, $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$, $^{\pm}$ indicates that the distinguisher can make forward and inverse queries, and $\mathsf{R}_1^{\mathrm{prf}},\ldots,\mathsf{R}_\mu^{\mathrm{prf}} \xleftarrow{\$} \mathrm{func}(m,t)$.

For PRF security, the distinguisher is usually bounded by the number of queries q it makes to the μ constructions ($\mathsf{F}[\mathsf{p}]_{K[j]}$ or $\mathsf{R}_j^{\mathrm{prf}}$ for $j = 1, \ldots, \mu$), the total construction query length in blocks σ, and the number of queries N to the primitive.

2.3 AE Security

Let $b, k, n, t \in \mathbb{N}$. An authenticated encryption scheme AE consists of an encryption function ENC and a decryption function DEC. The encryption function ENC gets as input a key $K \in \{0,1\}^k$, a nonce $NIV \in \{0,1\}^n$ (typically, one uses N, but that parameter will be used to bound the adversarial resources), associated data $A \in \{0,1\}^*$, and a plaintext $P \in \{0,1\}^*$, and it outputs a ciphertext $C \in \{0,1\}^{|P|}$ and a tag $T \in \{0,1\}^t$. The decryption function DEC gets as input a key $K \in \{0,1\}^k$, a nonce $NIV \in \{0,1\}^n$, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^t$, and it outputs either a plaintext $P \in \{0,1\}^{|C|}$ if the tag is correct, or a dedicated $\perp$-sign if the tag is incorrect. The two functions should satisfy

$$\mathsf{DEC}(K, NIV, A, \mathsf{ENC}(K, NIV, A, P)) = P\,.$$

We will restrict our focus to authenticated encryption instantiated using a b-bit permutation $\mathsf{p} \in \mathrm{perm}(b)$. As before, we will consider multi-user security of AE, where a distinguisher can query up to $\mu \geq 1$ versions of the scheme. The multi-user authenticated encryption (AE) security of AE against a distinguisher D is defined as

$$\mathbf{Adv}_{\mathsf{AE}}^{\mu\text{-ae}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\left(\mathsf{ENC}[\mathsf{p}]_{K[j]}, \mathsf{DEC}[\mathsf{p}]_{K[j]}\right)_{j=1}^{\mu}, \mathsf{p}^{\pm} \; ; \; \left(\mathsf{R}_j^{\mathrm{ae}}, \perp\right)_{j=1}^{\mu}, \mathsf{p}^{\pm} \right)\,, \tag{2}$$

where $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \xleftarrow{\$} (\{0,1\}^k)^\mu$, $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$, and the functions $\mathsf{R}_1^{\mathrm{ae}}, \ldots, \mathsf{R}_\mu^{\mathrm{ae}}$ that for each new input (NIV, A, P) return a random string of size $|P| + t$ bits. The function $\perp$ returns the $\perp$-sign for each query.

For AE security, we will only consider nonce-respecting distinguishers. These distinguishers never make, for any $j \in \{1, \ldots, \mu\}$, two encryption queries to $\mathsf{ENC}_{K[j]}$ for the same nonce NIV. These distinguishers are also not allowed to query, for any $j \in \{1, \ldots, \mu\}$, their decryption oracle ($\mathsf{DEC}_{K[j]}$ in the real world and $\perp$ in the ideal world) on input of the output of an earlier encryption query ($\mathsf{ENC}_{K[j]}$ in the real world and $\mathsf{R}_j^{\mathrm{ae}}$ in the ideal world). The distinguisher is usually bounded by the number of queries q it makes to the μ constructions $((\mathsf{ENC}[\mathsf{p}]_{K[j]}, \mathsf{DEC}[\mathsf{p}]_{K[j]})$ or $(\mathsf{R}_j^{\mathrm{ae}}, \perp)$ for $j = 1, \ldots, \mu)$, the total construction query length in blocks σ, and the number of queries N to the primitive.

3 Duplex Construction and Security Model

We describe the general duplex construction of Daemen et al. [32], but we do so in the notation of Dobraunig and Mennink [40]. This notation allows us to easier discuss the multiple use cases in Sections 5-9 in a consistent and clean way without conflicting notation. We remark that the main change in notation between the original work of Daemen et al. and the newer work of Dobraunig and Mennink is in the parametrization of the bit strings that are processed in/by the duplex construction; the security parameters and state sizes are identical in the works of Daemen et al., Dobraunig and Mennink, and this work. As a rule of thumb, calligraphic letters denote sets, sans serif letters denote functions, capital letters denote state values, and small letters denote state sizes. A few exceptions apply when the complexity of the distinguisher is defined in Section 3.3.

The general keyed duplex construction that we consider will be the one of Dobraunig and Mennink [40] but with (yet another) rephasing. It is given in Section 3.1. We describe the ideal equivalent of it in Section 3.2, and define the duplex security model in Section 3.3. In Section 3.4 we discuss the history of duplex phasing and put the phasing of our description of Section 3.1 into context. Sections 3.1-3.3 are largely copied from Dobraunig and Mennink [40], but differ in the rephasing and are amended with clearly indicated remarks important for further understanding of this work.

3.1 Duplex Construction

Let $b, c, r, k, \mu, \alpha \in \mathbb{N}$, with $c+r=b$, $k \leq b$, and $\alpha \leq b-k$. We describe the keyed duplex construction KD in Algorithm 1. The keyed duplex construction gets as input a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$ consisting of μ keys, and it is instantiated using a b-bit permutation $\mathsf{p} \in \mathrm{perm}(b)$. The construction internally maintains a b-bit state S, and has two interfaces: $\mathsf{KD.init}$ and $\mathsf{KD.duplex}$.

Algorithm 1 Keyed duplex construction $\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}$

```
Interface: $\mathsf{KD.init}$
Input: $(\delta, IV) \in \{1, \ldots, \mu\} \times \mathcal{IV}$
Output: $\varnothing$
    $S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$
    return $\varnothing$

Interface: $\mathsf{KD.duplex}$
Input: $(\mathit{flag}, P) \in \{\mathit{true}, \mathit{false}\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
    $S \leftarrow \mathsf{p}(S)$
    $Z \leftarrow \mathrm{left}_r(S)$
    $S \leftarrow S \oplus [\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$        ▷ if flag, overwrite outer part
    return $Z$
```

The initialization interface gets as input a key index $\delta \in \{1, \ldots, \mu\}$ and an initialization vector $IV \in \mathcal{IV} \subseteq \{0,1\}^{b-k}$, and initializes the state with the δ-th key and the initialization vector IV as $S \leftarrow \mathrm{rot}_\alpha(\boldsymbol{K}[\delta] \parallel IV)$. It outputs nothing.

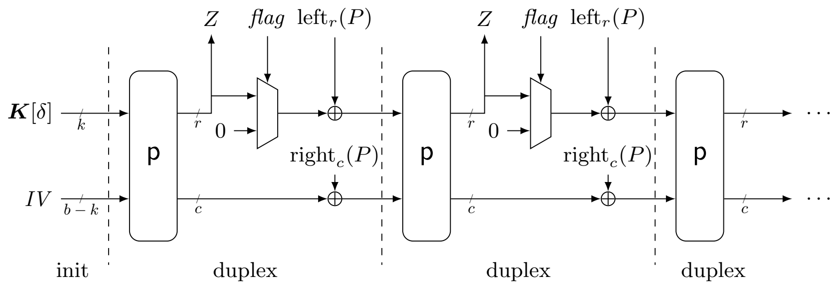


Fig. 1: The duplexing interface of KD.

The duplexing interface gets as input a flag $flag \in \{true, false\}$ and a new data block $P \in \{0,1\}^b$. The interface first applies an evaluation of the underlying permutation p on the state S. Then, it outputs an r-bit block $Z \in \{0,1\}^r$ off the internal state S, and transforms the state using the new data block P. The flag $flag$ describes how absorption is done on the r leftmost bits of the state that are squeezed: those r bits are either overwritten (if $flag = true$) or XORed with r bits of the input block P (if $flag = false$). See also Figure 1, where the duplex is depicted for key offset $\alpha = 0$.

Remark 1. The sole difference between our construction and that of Dobraunig and Mennink [40] is in the rephasing of the duplex interface. In a nutshell, in our description one duplexing call consists of a primitive evaluation, a squeeze, and an absorb (in this order), whereas in Dobraunig and Mennink one duplexing call consisted of a squeeze, and absorb, and a permutation evaluation. As a matter of fact, the phasing in Algorithm 1 is the same as the phasing of Daemen et al. [32], with the difference that Daemen et al. basically "integrated" one duplexing call *in* the initialization call. We separated those. We will discuss the history duplex phasing in more detail in Section 3.4. As we will explain in Section 4.3, the rephasing does not change the security results.

Besides the phasing, our description of the duplex technically differs from that of Daemen et al. in the fact that the original duplex construction of Daemen et al. did not rotate the input, i.e., it matches Algorithm 1 for $\alpha = 0$. Our construction is thus a strict generalization, but it is easy to incorporate this change in the security analysis, and once we describe the main security bound in Section 4.3, it will be described in generality for any α. The use of rotating the initial state may be unclear at first sight, but a typical example is given in [40, Figure 4, Section 7.1]. This example consists of two duplex constructions evaluated one after the other, where the first one puts the key at the top but the second one uses the output of the first one as inner part and de facto puts the key at the bottom (in this specific case, $\alpha = 1$). In general, including rotation in the description simply allows the duplex to be applicable to designs that put the key in the inner part instead of the outer part.

Remark 2. The usage of the flag input *flag* is new since Daemen et al. [32] and also included in the work of Dobraunig and Mennink [40]. It is confusing at first sight, but does nothing else than distinguishing between a case where the outer part of the input block is added to the state and where it replaces the state. In Figure 2, we give a simplified depiction of a duplexing call for $\mathit{flag} = \mathit{false}$ or $\mathit{flag} = \mathit{true}$.

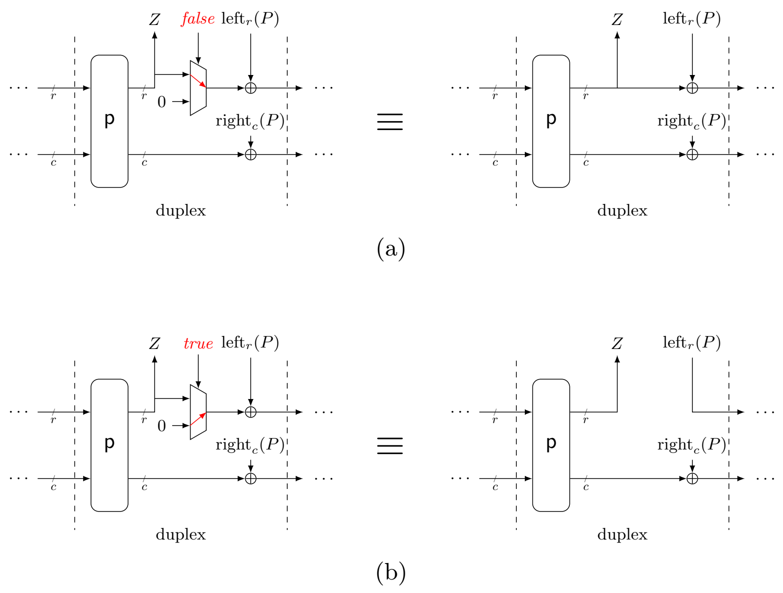


Fig. 2: A duplexing call with $\mathit{flag} = \mathit{false}$ (a) and with $\mathit{flag} = \mathit{true}$ (b).

Although a duplexing call gets as input a flag $\mathit{flag} \in \{\mathit{true}, \mathit{false}\}$ and a new data block $P \in \{0,1\}^b$, there is a difference between the two in that a user (typically) has more freedom in choosing the latter. Indeed, the data block may for example consist of plaintext bits, whereas the flag is determined by how the duplexing call occurs in the cryptographic scheme. To be more precise, there exist cryptographic schemes built on top of the duplex construction that *sometimes* allow the attacker to overwrite the outer part of the state; a typical example for this is SpongeWrap (see Section 9).

On the other hand, intuition suggests that one should try to limit the amount of duplexing calls for $\mathit{flag} = \mathit{true}$, the reason being that in these duplexing calls an attacker has much more freedom in choosing the value that is entered into the next permutation call. Not coincidentally, the existing security bounds of the duplex construction in Section 4 rely on a parameter that keeps track of the amount of duplexing calls made for $\mathit{flag} = \mathit{true}$.

3.2 Ideal Duplex

Daemen et al. [32] described the ideal extendable input function (IXIF) as ideal equivalent for the keyed duplex. We will also consider this function, modulo syntactical changes based on the changes we made on the keyed duplex in Section 3.1. The function is described in Algorithm 2.

The IXIF has the same interface as the keyed duplex, but instead of being based on a key array $\boldsymbol{K} \in (\{0,1\}^k)^\mu$ and being built on primitive $\mathsf{p} \in \text{perm}(b)$, it is built on a random oracle $\mathsf{ro} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^\infty$, that is defined as follows. Let $\mathsf{ro}_\infty : \{0,1\}^* \to \{0,1\}^\infty$ be a random oracle in the sense of Bellare and Rogaway [11]. For $P \in \{0,1\}^*$, $\mathsf{ro}(P, r)$ outputs the first r bits of $\mathsf{ro}(P)$. The IXIF maintains a path *path*, in which it unambiguously stores all data input by the user. It is initialized by $\text{encode}[\delta] \parallel IV$ for some suitable injective encoding function $\text{encode} : \{1,\ldots,\mu\} \to \{0,1\}^k$, and upon each duplexing call, the new plaintext block is appended to the path. Duplexing output is generated by evaluating the random oracle on *path*.

Algorithm 2 Ideal extendable input function IXIF[ro]

```
Interface: IXIF.init
Input: $(\delta, IV) \in \{1,\ldots,\mu\} \times \mathcal{IV}$
Output: $\varnothing$
  $path \leftarrow \text{encode}[\delta] \parallel IV$
  return $\varnothing$

Interface: IXIF.duplex
Input: $(flag, P) \in \{true, false\} \times \{0,1\}^b$
Output: $Z \in \{0,1\}^r$
  $Z \leftarrow \mathsf{ro}(path, r)$
  $path \leftarrow path \parallel ([flag] \cdot (Z \| 0^{b-r}) \oplus P)$        ▷ if $flag$, overwrite outer part
  return $Z$
```

3.3 Security Model

Let $b, c, r, k, \mu, \alpha \in \mathbb{N}$, with $c + r = b$, $k \leq b$, and $\alpha \leq b - k$. Let $\mathsf{p} \xleftarrow{\$} \text{perm}(b)$ be a random permutation, ro be a random oracle, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. In the black-box security model, one considers a distinguisher that has access to either $(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^\pm)$ in the real world or $(\mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^\pm)$ in the ideal world, where "$^\pm$" stands for the fact that the distinguisher has bi-directional query access:

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}} \left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^\pm \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^\pm \right) . \tag{3}$$

In our analyses, the distinguisher is obliged to always make at least one duplexing call after an initialization call (this condition is required to make sure that the results of [32, 40] still hold after our duplex rephasing).

Remark 3. The analysis of Daemen et al. [32] considered a slight relaxation of above model, namely where the keys are not uniformly randomly generated, but rather randomly according to some distribution $\mathcal{D}$. As this slight generalization hinders a conceptually simple discussion of the applications, we have dropped this generalization and restrict our focus to uniformly randomly generated keys.

The analysis of Dobraunig and Mennink [40] significantly differs from above model in that leakage is taken into account. In detail, in their model, each evaluation of p within $\mathsf{KD}[\mathsf{p}]$ might leak λ bits of information about its in- and/or output. Dobraunig and Mennink also show how a *specific* use of the keyed duplex construction, namely one along the same lines as a proposal of Taha and Schaumont [73] and as ISAP v2 [36], results in leakage resilient encryption. General constructions, e.g., the ones we discuss in Sections 5-9, do not necessarily yield security in the leaky setting.

3.4 Understanding Duplex Phasing

One typical evaluation of a (long) duplex-based construction consists of (i) an absorption of the key and initialization vector, followed by an arbitrarily long loop of (ii) a primitive evaluation, (iii) a squeeze, and (iv) an absorption of the data. In our description of Algorithm 1, step (i) is covered by an initialization call whereas (ii-iv) are covered by a duplexing call. This seems to be the most logical choice, but it is not that trivial. As a matter of fact, this choice of phasing differs from *all earlier duplex security results* [16, 32, 40, 61]. In this section, we put our choice in perspective. A visualization of the phasing approaches over time is given in Table 1.

The earliest introduction of the duplex, by Bertoni et al. in 2011 [16], considered one duplex round to consist of an absorption, a primitive evaluation, and a squeeze (in this order). Mennink et al. in 2015 [61], who introduced the full-state duplex, followed the same approach. Here, strictly seen in initialization the absorption was separated into key absorption (inner part) and data absorption (outer part); these two parts of absorption are merged in this discussion and in Table 1 we consider the first duplexing call to be an initialization call as this interpretation makes most sense. The absorb-permute-squeeze approach of Bertoni et al. and Mennink et al. makes sense historically, noting that one duplex round can be seen as a "mini-sponge" and, indeed, Bertoni et al. argued security of the duplex based on the indifferentiability of the sponge construction [14].

In 2017, Daemen et al. [32] noticed, however, that this approach was too restrictive in getting a good security bound. Indeed, the absorb-permute-squeeze approach technically allows the attacker to always *overwrite* the outer part of the state to a certain value. This is because the duplex security proof, by default, considers blockwise attackers: the attacker can make a duplex evaluation and use the resulting squeeze to construct the data block of the next duplexing call. In practical applications, it is *sometimes* possible for an attacker to overwrite the outer part of the state, and it really depends on the use case. For example, if a duplex construction is used for authenticated encryption and authenticated decryption is performed with release of unverified plaintext [2], the attacker has the possibility to overwrite the outer part of the state in decryption (see also Remark 5 in Section 9).

Daemen et al. resolved the issue by rephasing the duplex to permute-squeeze-absorb, with the subtle difference that absorption not only takes a data block but also flag *flag* that indicates whether the outer part is overwritten or not. They, in addition, integrated one integral duplexing call *in* the initialization, making the initialization consist of absorb-permute-squeeze-absorb, where the first absorption consisted of key and initialization vector. The phasing of Daemen et al. is also the phasing that we used in our description in Algorithm 1, with the difference that we removed the duplexing call from the initialization call. See also Table 1. This change has as main advantage that the primitive evaluations *only* occur in duplexing calls and not in initialization calls, making the analysis and results conceptually cleaner.

Dobraunig and Mennink [40], took a different avenue. In their phasing, one duplexing call consisted of squeeze-absorb-permute (with the initialization consisting of absorb-permute). Their approach was taken because they analyzed the duplex construction under leakage and it made most sense to include the "next" permutation evaluation in the duplexing call. In practical instantiations, however, one would not make the "last" permutation call of a (long) duplex-based construction, and adopting their approach in our work would lead to complications in the discussion of the use cases in Sections 5-9.

Table 1: Depiction of earlier and current phasing choice of the duplex construction. Here, A stands for absorb, P for permutation evaluation, and S for squeeze. The first absorb includes the key. We remark that there are differences how the absorption is implemented in the five variants.

	A	P	S	A	P	S	A	P	S	A	···
BDPV11 [16]	init			duplex			duplex			···	
MRV15 [61]	init			duplex			duplex			···	
DMV17 [32]	init				duplex			duplex			···
DM19 [40]	init		duplex			duplex			···		
now (Alg. 1)	init	duplex			duplex			duplex			···

4 Security of Duplex Construction

In Section 4.1 we summarize the notation used to describe the adversarial resources. This section is again largely copied from Dobraunig and Mennink [40], but updated to the current setting. Then, in Section 4.2, we will discuss a mathematical function used in the duplex security bounds, namely the multicollision limit function. Finally, the known (black-box) security bounds on the keyed duplex, derived from Daemen et al. [32] and Dobraunig and Mennink [40], are given in Section 4.3.

4.1 Distinguisher's Resources

We consider an information-theoretic distinguisher D that has access to either the real world $(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}^{\pm})$ or the ideal world $(\mathsf{IXIF}[\mathsf{ro}], \mathsf{p}^{\pm})$, where p is some permutation. Two basic measures to quantify the distinguisher's resources are its online complexity M and offline complexity N:

- M: the number of distinct construction (duplexing) calls;
- N: the number of distinct primitive queries.

For each construction call, we define a path *path* that "registers" the data that got absorbed in the duplex up to the point that the cryptographic primitive (p in the real world and ro in the ideal world) is evaluated. For an initialization call $(\delta, IV) \mapsto \varnothing$, the associated path is defined as $path = \text{encode}[\delta] \parallel IV$. For each duplexing call $(flag, P) \mapsto Z$, the value $[flag] \cdot (Z \| 0^{b-r}) \oplus P$ is appended to the path of the previous construction query. Not surprisingly, the definition matches the actual definition of *path* in the $\mathsf{IXIF}[\mathsf{ro}]$ construction of Algorithm 2, but defining the same thing for the real world will allow us to better reason about the security of the keyed duplex. Note that the value *path* contains no information that is secret to the distinguisher. In order to reason about duplexing calls, we will also define a *subpath* of a *path*, which is the path leading to the particular duplexing call. In other words, for a path *path*, its *subpath* is simply *path* with the last b bits removed.

In order to derive a detailed and versatile security bound, that in particular well-specifies how leakage influences the bound, we further parameterize the distinguisher as follows. For initialization calls:

- Q: the number of initialization calls;
- Q_{IV}: the maximum number of initialization calls for a single IV.

For duplexing calls:

- L: the number of duplexing calls with repeated subpath, i.e., M minus the number of distinct subpaths;
- Ω: the number of duplexing queries with $flag = true$.

Note that these parameters can all be described as a function of the duplexing calls and the related *path*'s, and the distinguisher can compute these values based on the queries it made so far. The parametrization of the distinguisher is as that of Daemen et al. [32]. The parameters L and Ω are, as in [32], used to upper bound the number of duplexing calls for which the distinguisher may have *set* the r leftmost bits of the input to the permutation in the duplexing call to a certain value of its choice. This brings us to the last parameter:

- ν_{fix}: the maximum number of duplexing calls for which the adversary has set the outer part to a single value $\text{left}_r(T)$.

Note that $\nu_{\mathsf{fix}} \leq L + \Omega$, but it may be much smaller in specific use cases of the duplex, for example, if overwrites only happen for unique values. It also plays a role in the application on $\mathsf{Ascon\text{-}PRF}$ in Section 8.3.

4.2 Multicollision Limit Function

We will use the notion of multicollision limit functions from Daemen et al. [32], which considers a balls-into-bins experiment tailored to sponge constructions.

Definition 1 (multicollision limit function). *Let $M, b, c, r \in \mathbb{N}$ with $c + r = b$. Consider the experiment of throwing M balls uniformly at random in 2^r bins, and let ν be the maximum number of balls in a single bin. We define the multicollision limit function $\nu_{r,c}^{M}$ as the smallest natural number x that satisfies*

$$\mathbf{Pr}\left(\nu > x\right) \leq \frac{x}{2^c}\,.$$

The definition of multicollision limit functions looks a bit artificial, but is inspired by the applications where it is used. In sponge or duplex proofs, we often need an upper bound on the maximum multicollision in the outer part. Call this maximum ν. This value ν then appears in the final security bound in a term of the form $\frac{\nu \cdot N}{2^c}$. However, we can be unlucky: there may be a multicollision of size larger than ν. However, if we choose ν to be equal to $\nu_{r,c}^{M}$, by Definition 1 this happens with probability at most $\frac{\nu}{2^c}$. This fraction is negligible compared to $\frac{\nu \cdot N}{2^c}$.

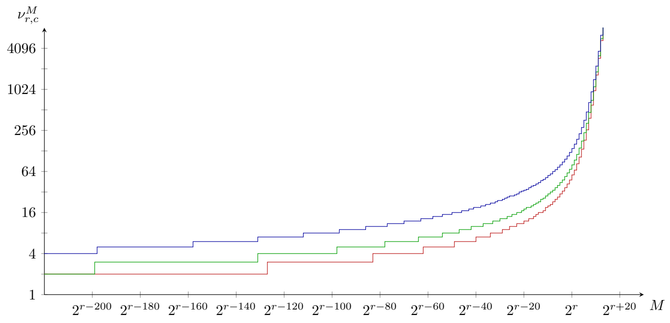
The multicollision limit function can be clearly estimated. In particular, the targeted value x satisfies [32, Section 6.5]

$$\frac{2^b e^{-M/2^r} (M/2^r)^x}{(x - M/2^r) x!} \leq 1\,. \tag{4}$$

It turns out that this bound behaves differently depending on the value $M/2^r$, and Daemen et al. also gave a technical interpretation of the upper bounds suggested for $\nu_{r,c}^{M}$ for specific values of $M/2^r$. We will describe these in more accessible terminology below, and also exemplify those for a running example of $(b, c, r) = (400, 272, 128)$.

- If $M/2^r < 1$, an appropriate choice for $\nu_{r,c}^{M}$ is the smallest integer x such that $M/2^r \leq 2^{-b/x}$, i.e., $x = \left\lceil \frac{b}{r - \log_2(M)} \right\rceil$. For our running example, if $M = 2^{64}$, one would get $\nu_{128,272}^{2^{64}} \leq 7$, and if $M = 2^{88}$, one would get $\nu_{128,272}^{2^{88}} \leq 10$;
- If $M/2^r = 1$, an appropriate choice for $\nu_{r,c}^{M}$ is the smallest integer x such that $x \geq \frac{\ln(2) b}{\ln(x) - 1}$. For our running example, if $M = 2^{128}$, one would get $\nu_{128,272}^{2^{128}} \leq 82$;
- If $M/2^r > 1$, the bound (4) becomes less controllable. However, if M is a multiple of 2^r, $\nu_{r,c}^{M}$ could be upper bounded by $\frac{M}{2^r} + \nu_{r,c}^{2^r} \cdot \left\lceil \frac{M}{2^r} \right\rceil$. For our running example, if $M = 2^{132}$, one would get $\nu_{128,272}^{2^{132}} \leq 16 + 82 \cdot 16 = 1328$.

We remark that above example values on $\nu_{128,272}^{M}$ are based on the simplified bounds; (4) would give a tighter value. A depiction of (4) for $b = 256$, for our running example of $b = 400$, and for $b = 800$, but with a more general choice of (c, r), is given in Figure 3. A naive (non-optimized) Python script for computing the value x such that (4) holds, is given in Appendix A.



$M/2^r$	$\nu^M_{r,256-r}$	$\nu^M_{r,400-r}$	$\nu^M_{r,800-r}$
2^{-256}	—	2	4
2^{-128}	2	4	7
2^{-64}	4	7	12
2^{-32}	8	12	23
2^{-16}	14	21	40
2^{-8}	23	34	64
2^{0}	57	80	139
2^{8}	601	707	944
2^{16}	70205	71484	74119
2^{19}	537313	540887	548194

Fig. 3: Plot of the multicollision limit function for $b = 256$ (red, lowest), $b = 400$ (green, middle), and $b = 800$ (blue, highest), based on (4), where $\nu^M_{r,c}$ is computed as function of $M/2^r$. Example values are given in the table.

4.3 Main Result

We can now state the *existing* security results on the keyed duplex construction in the model of (3). In fact, we have two bounds that are derived using two different proof techniques. The first bound, Theorem 1 is the original result of Daemen et al. [32], extended to cover arbitrary initial state rotation α. The security bound still holds under the rephasing (cf., Remark 1), due to our requirement that the distinguisher always makes at least one duplexing call after each initialization call.

Theorem 1 (security of duplex construction [32]). *Let $b, c, r, k, \mu, \alpha \in \mathbb{N}$, with $c+r = b$, $k \leq b$, and $\alpha \leq b-k$. Let $\mathsf{p} \xleftarrow{\$} \text{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D quantified as in Section 4.1 and with $M + N \leq 0.1 \cdot 2^c$ (cf., Remark 4),*

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{(L+\Omega)N}{2^c} + \frac{2\nu^{2(M-L)}_{r,c}(N+1)}{2^c} + \frac{\binom{L+\Omega+1}{2}}{2^c} \tag{5a}$$

$$+ \frac{(M-L-Q)Q}{2^b - Q} + \frac{M(M-L-1)}{2^b} \tag{5b}$$

$$+ \frac{Q(M-L-Q)}{2^{\min\{c+k,\max\{b-\alpha,c\}\}}} + \frac{Q_{IV}N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \tag{5c}$$

The second bound, Theorem 2, is the leakage resilience result of Dobraunig and Mennink [40], restricted to the setting of zero leakage ($\lambda = 0$). Also for this result, the rephasing (cf., Remark 1) does not change the bound, noting that the security analysis of [41] relies on a proper categorization of the construction evaluations into their *type* (*init*, *full*, or *fix*, to be precise) and that the number of evaluations per *type* does not change after rephasing. Also here, we rely on the condition that the distinguisher always makes at least one duplexing call after each initialization call.

Theorem 2 (security of duplex construction [40]). *Let $b, c, r, k, \mu, \alpha \in \mathbb{N}$, with $c + r = b$, $k \leq b$, and $\alpha \leq b - k$. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D quantified as in Section 4.1,*[1]

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{\nu_{\mathsf{fix}} N}{2^c} + \frac{2\nu_{r,c}^M (N+1)}{2^c} + \frac{\nu_{r,c}^M (L + \Omega) + \frac{\max\{\nu_{\mathsf{fix}}-1,0\}}{2}(L+\Omega)}{2^c} \tag{6a}$$

$$+ \frac{\binom{M-L-Q}{2} + (M-L-Q)(L+\Omega)}{2^b} + \frac{\binom{M+N}{2} + \binom{N}{2}}{2^b} \tag{6b}$$

$$+ \frac{Q(M-Q)}{2^{\min\{c+k, \max\{b-\alpha, c\}\}}} + \frac{Q_{IV} N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \tag{6c}$$

The bounds of Theorems 1 and 2 share many similarities, but also expose subtle differences. These differences come from the different proof techniques. For example:

- Dobraunig and Mennink started their proof with a RP-to-RF-switch. This switch contributed to the second fraction of (6b). The advantage of this switch is that it significantly simplified further analysis of the scheme (and also led to easier-to-parse terms), but on the downside, the bound contains a term of the order $O(N^2/2^b)$, which is rather restrictive in the case of small b;
- Dobraunig and Mennink introduced a term ν_{fix} to resolve a minor lossiness in the bound, noting that although there are typically $L + \Omega$ evaluations where the distinguisher may have *set* the r leftmost bits to a value of its choice, the actual number is in practice often smaller than $L + \Omega$, namely ν_{fix}. The usage of ν_{fix} will become apparent in the application to Ascon-PRF in Section 8.3.

In upcoming Sections 5-9, we will discuss various applications of the duplex construction and of Theorems 1 and 2. In the security analyses of these applications, we will also consider distinguishers that are bound with certain resources. One of them is the number of distinct primitive queries, which always happens to be the same as the number of distinct primitive queries in the duplex security proofs. We will henceforth stick to the parameter N, just like in Section 4.1. For the *online* complexity of the distinguisher in below applications, we will explicitly use different parameters than in Section 4.1, namely q and σ, to make sure the transition of the bounds of Theorems 1 and 2 to the security of the applications is clear and transparent. See also the models in Section 2.

Remark 4. The multicollision limit function of Definition 1 is for uniform throws with replacement whereas Daemen et al. actually apply it to uniform throws without replacement. For the setting considered in this work (their actual result

[1] The max in the third fraction of (6a) was missing in the original bound, but should obviously be included.

is slightly more general), they proved that the multicollision limit function for the case of $M' \leq M$ throws without replacement is at most that of $2M'$ throws with replacement, provided $M + N \leq 0.1 \cdot 2^c$ holds. It may be possible to avoid this side condition by deriving a direct multicollision bound in case of drawing without replacement, or by resorting to the multicollision approach of Choi et al. [27]. However, both approaches will make the overall security bound harder to interpret and the $0.1 \cdot 2^c$ limit is irrelevant in most cases anyway.

5 Use Case 1: Truncated Permutation

According to the well-known PRP-PRF switch [9,12,25,51,52], an n-bit PRP behaves like a PRF up to approximately $2^{n/2}$ evaluations. As this bound could be problematic for small values of n, there has been performed a significant amount of research to designing a PRF from a PRP with beyond-birthday bound security. These studies have given rise to various schemes, like the sum of (secret) permutations [8, 10, 28, 33, 55, 60, 68–70], EDM [29, 33, 59], EDMD [59], truncation [8, 20, 46–48, 51, 58, 72], and the summation-truncation hybrid [49].

In recent years, advances have been made in understanding how to turn a *public* random permutation into a PRF, constructions of which were often inspired by the above. For example, Chen et al. [26] considered a sum of externally keyed public permutations, and Dutta et al. [43] a permutation-based variant of EDM. In this section, we will highlight the public permutation based variant of truncation. The construction is described in Section 5.1 and its security is analyzed in Section 5.2.

5.1 Construction

Let $b, c, r, k \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. The truncated permutation construction $\mathsf{TP} : \{0,1\}^k \times \{0,1\}^{b-k} \to \{0,1\}^r$ is defined as

$$\mathsf{TP}[\mathsf{p}](K, X) = \mathrm{left}_r(\mathsf{p}(K \| X)) \,. \tag{7}$$

It can be described in terms of a duplex construction as in Algorithm 3, and as depicted in Figure 4. In case we consider multiple instances of the scheme, the key input in Algorithm 3 will be replaced by a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to $\mathsf{KD.init}$ will be the index of the instance that is evaluated. We admit that this description is slightly odd, where first r bits of the permutation output are squeezed and then the entire state is truncated, but this is simply done to make the similarity with the duplex construction clear.

5.2 Security

As the truncated permutation construction TP of (7) can be described in terms of a duplex, we can reduce the PRF security of TP to the security of the duplex construction, and rely on the results of Section 4.3. Note that we obtain two different results, one based on Theorem 1 and one based on Theorem 2. It depends

Algorithm 3 Truncated permutation $\mathsf{TP}[\mathsf{p}]$

```
Input: $(K, X) \in \{0,1\}^k \times \{0,1\}^{b-k}$
Output: $Y \in \{0,1\}^r$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $\mathsf{KD.init}(1, X)$
  $Y \leftarrow \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $Y$
```

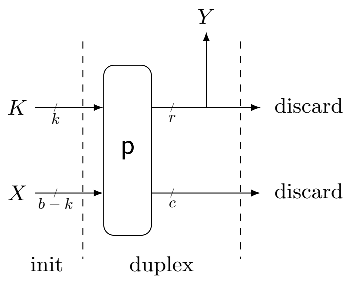


Fig. 4: Truncated permutation TP. The function gets as input a key K and an input value X. It outputs Y.

on the actual values of b, c, r, k which of the two bounds is better. In general, the former bound (based on Theorem 1) is better, except if the bound is still $\ll 1$ for $q + N$ exceeding $0.1 \cdot 2^c$, because in that case, this bound is inapplicable.

Theorem 3 (PRF security of truncated permutation). *Let $b, c, r, k, \mu \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D making at most q construction queries and N primitive queries, we have the following results:*

(i) Provided $q + N \leq 0.1 \cdot 2^c$,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{TP}}(\mathsf{D}) \leq \frac{2\nu^{2q}_{r,c}(N+1)}{2^c} + \frac{2\binom{q}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,; \tag{8}$$

(ii) In general,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{TP}}(\mathsf{D}) \leq \frac{2\nu^{q}_{r,c}(N+1)}{2^c} + \frac{\binom{q+N}{2} + \binom{N}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \tag{9}$$

Intuitively, provided that μ and $\nu^{2q}_{r,c}$ are small enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$ and $q \ll 2^{b/2}$.

Proof (Proof of Theorem 3). We will first discuss how TP fits in the description of the duplex construction of Section 3.1, then we discuss what this means for the actual power of the distinguisher, i.e., the distinguisher's resources of Section 4.1, and finally we derive our bound.

Starting with mapping TP to the description of the duplex construction, note that we also here consider multi-user security. In other words, we consider security of TP under μ keys $K[1], \ldots, K[\mu] \in \{0,1\}^k$, which we store in key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$ fed to the duplex construction. In addition, $\mathcal{IV} = \{0,1\}^{b-k}$, i.e., $\mathcal{IV}$ simply determines the domain of the truncated permutation construction. Now, an evaluation of TP for key $K[\delta] \in \{0,1\}^k$ and input $X \in \{0,1\}^{b-k}$ corresponds to calling the initialization interface $\mathsf{KD.init}$ on input $(\delta, X) \in \{1, \ldots, \mu\} \times \mathcal{IV}$, where the state is initialized as $K[\delta] \| X$ (hence $\alpha = 0$), subsequently calling the duplexing interface $\mathsf{KD.duplex}$ on no input and outputting the leftmost r bits. See also Algorithm 3.

The TP distinguisher D can make q construction queries and N primitive queries. Each construction query is made for different (δ, X) and corresponds to exactly one initialization and one duplexing call. This particularly means that the parameters M and Q of the distinguisher's resources equal q. Another parameter of the distinguisher's resources to consider is Q_{IV}, the maximum number of initialization calls for a single IV, i.e., for different X. Note that all distinguisher's construction queries must be for different input (δ, X). This means that Q_{IV} is at most μ, as each value X can be queried alongside at most μ different keys. Finally, as each construction query de facto starts with a different (δ, X) each path is distinct, and also no duplexing calls are made for *flag* $=$ *true* (looking at the construction, the flag simply does not matter). We conclude that $L = \Omega = 0$ and thus $\nu_{\mathsf{fix}} = 0$.

To summarize, the distinguisher's resources of Section 4.1 satisfy:

parameter in Section 4.1	parameter in current proof
M	q
N	N
Q	q
Q_{IV}	$\leq \mu$
L	0
Ω	0
ν_{fix}	0

If we plug these values into the bound (5) of Theorem 1, we obtain the following result, provided $q + N \leq 0.1 \cdot 2^c$:

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{2\nu_{r,c}^{2q}(N+1)}{2^c} + \frac{2\binom{q}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} \,. \tag{10}$$

If we plug these values into the bound (6) of Theorem 2, we obtain the following result:

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq \frac{2\nu_{r,c}^{q}(N+1)}{2^c} + \frac{\binom{q+N}{2} + \binom{N}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} \,. \tag{11}$$

The bound (10) is in general better. The subtle differences between the two bounds arise from the fact that Dobraunig and Mennink started their proof with a RP-to-RF-switch. This added the second term in (11) but lead to a simpler first term. See also the remark regarding this RP-to-RF-switch at the end of Section 4.3. However, (10) only holds provided $q + N \leq 0.1 \cdot 2^c$. This means that (11) is still meaningful as it may stay well below 1 once $q + N$ goes beyond this bound. We will thus continue with both bounds.

What remains is to translate these bounds into the multi-user PRF security of TP. This can be done by a simply triangle inequality, noting that the formal description $\mathsf{TP}[\mathsf{p}]$ of Algorithm 3 *is* in fact a description $\mathsf{TP}[\mathsf{KD}[\mathsf{p}]]$, where the key array $\boldsymbol{K}$ input to TP is directly fed into KD. Thus, denoting $\mathbf{R}^{\mathrm{prf}} = (\mathsf{R}_1^{\mathrm{prf}}, \ldots, \mathsf{R}_\mu^{\mathrm{prf}}) \xleftarrow{\$} \mathrm{func}(b-k, r)^\mu$,

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{TP}}^{\mu\text{-prf}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{TP}[\mathsf{KD}[\mathsf{p}]]_{\boldsymbol{K}}, \mathsf{p}^{\pm} \;;\; \mathbf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&= \Delta_{\mathsf{D}}\left(\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{p}^{\pm} \;;\; \mathbf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) \\
&\leq \Delta_{\mathsf{D}}\left(\mathsf{TP}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{p}^{\pm} \;;\; \mathsf{TP}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) && (12) \\
&+ \Delta_{\mathsf{D}}\left(\mathsf{TP}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm} \;;\; \mathbf{R}^{\mathrm{prf}}, \mathsf{p}^{\pm}\right) . && (13)
\end{aligned}$$

The distance of (12) is the security of KD as bounded in (10) or (11), and the distance of (13) equals 0 as both oracles output uniform random and independent strings for each input. □

6 Use Case 2: Parallel Keystream Generation

The truncated permutation construction TP of Section 5 behaves like a PRF but only outputs r bits. However, in practice, b is much larger than k and the input value X can be quite large. It is thus possible to include a counter in X, and basically use TP in counter mode, yielding a parallel keystream generation construction. The construction is described in Section 6.1 and its security is analyzed in Section 6.2.

6.1 Construction

Let $b, c, r, k, a \in \mathbb{N}$, with $c + r = b$ and $k + a \leq b$. We consider the following duplex-based parallel keystream generation construction:

$$\begin{aligned}
\mathsf{P\text{-}SC} : \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0, \ldots, r2^a\} &\to \{0,1\}^\infty \\
(K, NIV, \ell) &\mapsto S\,.
\end{aligned} \quad (14)$$

The construction gets as input a key K, a nonce NIV, and a requested keystream length ℓ, and outputs a keystream S of length ℓ bits. It simply evaluates TP of (7) in counter mode, on input of $(K, NIV \| \langle i-1 \rangle_a)$ for $i = 1, \ldots, \lceil \ell/r \rceil$. The construction is described in Algorithm 4, and for one counter value i it is depicted in Figure 5. In case we consider multiple instances of the scheme, the key input in Algorithm 4 will be replaced by a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to $\mathsf{KD.init}$ will be the index of the instance that is evaluated. We admit that the description in Figure 5 of the construction is a bit odd, in the sense that K gets an own arrow while NIV and $\langle i \rangle_a$ have to share an arrow. The reason for this depiction is to show the similarity with TP of Section 5.

Algorithm 4 Duplex-based parallel keystream generation $\mathsf{P\text{-}SC}[\mathsf{p}]$

```
Input: $(K, NIV, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k-a} \times \{0, \ldots, r2^a\}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $S \leftarrow \varnothing$
  for $i = 1, \ldots, \lceil \ell/r \rceil$ do
      $\mathsf{KD.init}(1, NIV \| \langle i-1 \rangle_a)$
      $S \leftarrow S \,\|\, \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $\mathrm{left}_\ell(S)$
```

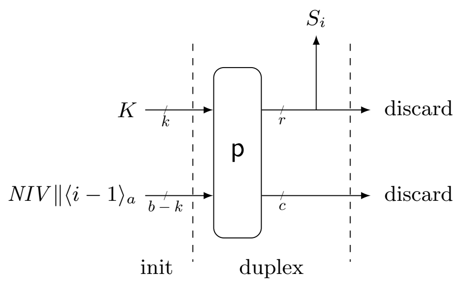


Fig. 5: One evaluation of duplex-based parallel keystream generation $\mathsf{P\text{-}SC}$. The function gets as input a key K, a nonce NIV, and a counter value $i \in \{1, \ldots, 2^a\}$. It outputs a keystream block S_i.

6.2 Security

PRF security of $\mathsf{P\text{-}SC}$ immediately follows from the fact that counter mode based on a fixed-output-length random function is perfectly secure (as long as the input to the underlying random function is never repeated), and from the fact that TP behaves like a fixed-output-length random function, as proven in Theorem 3. As before, we obtain two different results.

Theorem 4 (PRF security of duplex-based parallel keystream generation). *Let $b, c, r, k, a, \mu \in \mathbb{N}$, with $c + r = b$ and $k + a \leq b$. Let* $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ *be a random permutation, and* $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ *a random array of keys. For any distinguisher* D *making at most q construction queries, of total length at most σ permutation calls, and N primitive queries, we have the following results:*

(i) Provided $\sigma + N \leq 0.1 \cdot 2^c$,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{P\text{-}SC}}(\mathsf{D}) \leq \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,; \tag{15}$$

(ii) In general,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{P\text{-}SC}}(\mathsf{D}) \leq \frac{2\nu^{\sigma}_{r,c}(N+1)}{2^c} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \tag{16}$$

Just like the case of Theorem 3, provided that μ and $\nu^{2\sigma}_{r,c}$ are small enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$ and $\sigma \ll 2^{b/2}$.

Proof (Proof of Theorem 4). The function $\mathsf{P\text{-}SC}$ can we written in terms of TP of (5) as

$$\begin{aligned}&\mathsf{P\text{-}SC}(K, \mathit{NIV}, \ell) = \\ &\quad \mathsf{TP}(K, \mathit{NIV} \| \langle 0 \rangle_a) \;\|\; \mathsf{TP}(K, \mathit{NIV} \| \langle 1 \rangle_a) \;\|\; \cdots \;\|\; \mathsf{TP}(K, \mathit{NIV} \| \langle \lceil \ell/r \rceil - 1 \rangle_a)\,,\end{aligned}$$

truncated to ℓ bits.

As such, security of $\mathsf{P\text{-}SC}$ against a distinguisher D making at most q construction queries, of total length at most σ permutation calls, follows immediately from the security of TP against a distinguisher D' making at most σ construction queries (both D and D' have primitive complexity N):

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{P\text{-}SC}}(\mathsf{D}) \leq \mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{TP}}(\mathsf{D}')\,.$$

The result now immediately follows from Theorem 3, with q replaced by σ. □

7 Use Case 3: Sequential Keystream Generation

The parallel keystream generation of Section 6 is clean and simple, but it also has a disadvantage that the key is evaluated multiple times. This could be a problem if the scheme is evaluated in a leaky environment. In addition, the sponge and duplex are mostly designed to be evaluated in a sequential direction. We will now consider what the duplex results of Section 4.3 could imply for the naive and most logical way of building stream encryption from the duplex construction. The construction is described in Section 7.1 and its security is analyzed in Section 7.2.

7.1 Construction

Let $b, c, r, k \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. We consider the following duplex-based sequential keystream generation construction:

$$\begin{aligned} \mathsf{S\text{-}SC} : \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N} &\to \{0,1\}^\infty \\ (K, NIV, \ell) &\mapsto S\,. \end{aligned} \tag{17}$$

The construction gets as input a key K, a nonce NIV, and a requested keystream length ℓ, and outputs a keystream S of length ℓ bits. It simply initializes a duplex with state $K \| NIV$, and then it makes duplexing calls that do not absorb any data but simply squeeze r bits at a time. The construction is described in Algorithm 5, and is depicted in Figure 6. In case we consider multiple instances of the scheme, the key input in Algorithm 5 will be replaced by a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to KD.init will be the index of the instance that is evaluated.

Algorithm 5 Duplex-based sequential keystream generation $\mathsf{S\text{-}SC}[\mathsf{p}]$

```
Input: $(K, NIV, \ell) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \mathbb{N}$
Output: $S \in \{0,1\}^\ell$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $S \leftarrow \varnothing$
  $\mathsf{KD.init}(1, NIV)$
  for $i = 1, \ldots, \lceil \ell/r \rceil$ do
      $S \leftarrow S \,\|\, \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $\mathrm{left}_\ell(S)$
```

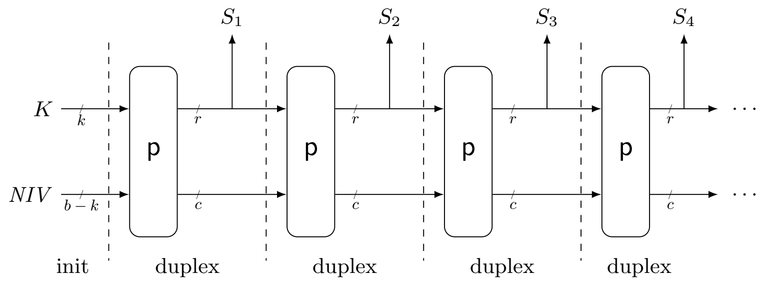


Fig. 6: Duplex-based sequential keystream generation S-SC. The function gets as input a key K and a nonce NIV. It outputs keystream blocks $(S_1, S_2, \ldots)$. The actual number of output blocks is determined by an additional input parameter ℓ.

7.2 Security

The S-SC construction described above is, in fact, the most logical way of keystream generation in the sponge. It can for example be recognized in the encryption of ISAP v2 [34–36] and in Asakey [42], though with a different initialization of the state (rather than simply $K\|NIV$) in order to guarantee improved strength against side-channel attacks.

The S-SC is, in fact, not just a duplex based construction, it is a specific case of the full-state keyed sponge, and hence one can rely on the result of Mennink et al. [61]. However, below, we derive a more accurate bound using the results of Section 4.3. As before, we obtain two different results.

Theorem 5 (PRF security of duplex-based sequential keystream generation). *Let $b, c, r, k, \mu \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D making at most q construction queries, of total length at most σ permutation calls, and N primitive queries, we have the following results:*

(i) Provided $\sigma + N \leq 0.1 \cdot 2^c$,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{S\text{-}SC}}(\mathsf{D}) \leq{}& \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,;\end{aligned} \tag{18}$$

(ii) In general,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{S\text{-}SC}}(\mathsf{D}) \leq{}& \frac{2\nu^{\sigma}_{r,c}(N+1)}{2^c} + \frac{\binom{\sigma-q}{2}}{2^b} + \frac{\binom{\sigma+N}{2}+\binom{N}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{19}$$

Just like the case of Theorem 4, provided that μ and $\nu^{2\sigma}_{r,c}$ are small enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$ and $\sigma \ll \min\{2^{(c+k)/2}, 2^{b/2}\}$.

Proof (Proof of Theorem 5). As in Theorem 3, we will first discuss how S-SC fits in the description of the duplex construction of Section 3.1, then we discuss what this means for the actual power of the distinguisher, i.e., the distinguisher's resources of Section 4.1, and finally we derive our bound.

The S-SC construction in fact looks very much like TP. Each initialization is made for a different state: $K[\delta]\|NIV$ in S-SC and $K[\delta]\|X$ in TP. The main difference is that TP makes 1 duplexing call $\mathsf{KD.duplex}(\mathit{false}, 0^b)$ per initialization call, whereas S-SC makes $\lceil \ell/r \rceil$ of them. Concretely, this means that we again have a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$ fed to the duplex construction. In addition, $\mathcal{IV} = \{0,1\}^{b-k}$, i.e., $\mathcal{IV}$ simply determines the set of nonces of S-SC. As before, the state is initialized as $K[\delta]\|NIV$ (hence $\alpha = 0$).

The S-SC distinguisher D can make q construction queries, of total length at most σ duplexing calls, and N primitive queries. Each construction query is made for different (δ, NIV) and corresponds to exactly one initialization and $\lceil \ell/r \rceil$ duplexing calls (where ℓ is the input parameter to $\mathsf{S\text{-}SC}$). The total amount of duplexing calls is at most σ. This particularly means that the parameters M and Q of the distinguisher's resources equal σ and q, respectively. Another parameter of the distinguisher's resources to consider is Q_{IV}, the maximum number of initialization calls for a single IV, i.e., in current case, for different NIV. As in the proof of Theorem 3, this value is bounded by μ, as each value NIV can be queried alongside at most μ different keys. (Note that one can imagine applications with user-dependent NIV, in which case $Q_{IV} = 1$.) Finally, as each construction query de facto starts with a different (δ, NIV) each path is distinct, and also no duplexing calls are made for $\mathit{flag} = \mathit{true}$. We conclude that $L = \Omega = 0$ and thus $\nu_{\mathsf{fix}} = 0$.

To summarize, the distinguisher's resources of Section 4.1 satisfy:

parameter in Section 4.1	parameter in current proof
M	σ
N	N
Q	q
Q_{IV}	$\leq \mu$
L	0
Ω	0
ν_{fix}	0

If we plug these values into the bound (5) of Theorem 1, we obtain the following result, provided $\sigma + N \leq 0.1 \cdot 2^c$:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq\ & \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(\sigma - q)q}{2^b - q} + \frac{2\binom{\sigma}{2}}{2^b} \\ & + \frac{q(\sigma - q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{20}$$

If we plug these values into the bound (6) of Theorem 2, we obtain the following result:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq\ & \frac{2\nu_{r,c}^{\sigma}(N+1)}{2^c} + \frac{\binom{\sigma-q}{2}}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} \\ & + \frac{q(\sigma - q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{21}$$

As in the proof of Theorem 3, the bound (20) is better in general, but (21) is better for certain parameter settings. However, it is obvious that in this case, the differences are a bit subtler than in the proof of Theorem 3 (though still minor). The final step of translating these bounds into the multi-user PRF security of $\mathsf{S\text{-}SC}$ is identical to the reasoning in the proof of Theorem 3, and henceforth omitted. □

8 Use Case 4: Message Authentication

As mentioned in Section 7.2, the S-SC construction is not just a duplex based construction, but rather a specific case of the full-state keyed sponge. In this section, we dive in more detail into the full-state keyed sponge construction, and consider what the results of Section 4.3 imply for this construction. The construction is described in Section 8.1 and its security is analyzed in Section 8.2.

Note that the full-state keyed sponge construction can be found in many applications. For example, the DonkeySponge of Bertoni et al. [19] is de facto a full-state keyed sponge, be it with round-reduced permutations for which it is unreasonable to assume perfect randomness. Chaskey [62] is also a variant of the full-state keyed sponge construction. Ascon-PRF [38] is a special case where the state is also initialized with the key, but then one only does outer-part absorption. However, something special holds for the Ascon-PRF construction, as we will discuss in Section 8.3.

8.1 Construction

Let $b, c, r, k, t \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathcal{IV} \subseteq \{0,1\}^{b-k}$ be a set of initialization vectors. The full-state keyed sponge construction is defined as follows:

$$\begin{aligned} \mathsf{FSKS} : \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^* &\to \{0,1\}^t \\ (K, IV, P) &\mapsto T \,. \end{aligned} \tag{22}$$

The construction gets as input a key K, an initialization vector IV (which may be repeated), and a plaintext P, and outputs a tag T of length t bits. It simply initializes a duplex with state $K \| IV$, then it absorbs plaintext blocks b bits at a time, and finally it squeezes tag blocks r bits at a time. The construction is described in Algorithm 6 and is depicted in Figure 7. In case we consider multiple instances of the scheme, the key input in Algorithm 6 will be replaced by a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to KD.init will be the index of the instance that is evaluated.

8.2 Security

Security of a variant of FSKS, where a full b-bit plaintext block was already absorbed at initialization, already follows from the result of Mennink et al. [61], who derived the following bound:

$$\mathbf{Adv}_{\mathsf{FSKS}}^{\mu\text{-prf}}(\mathsf{D}) \leq \frac{2(q\ell)^2}{2^b} + \frac{2q^2\ell}{2^c} + \frac{\rho N}{2^k} \,,$$

where the distinguisher can make q construction queries, each of maximum length ℓ, and N primitive queries, and where ρ is the "multiplicity" (named μ in [61], but renamed to avoid parameter collision), which roughly upper bounds the maximum amount of construction query blocks for a certain fixed outer part.

Algorithm 6 Full-state keyed sponge $\mathsf{FSKS}[\mathsf{p}]$

```
Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_b^{10^*}(P)$
  $T \leftarrow \varnothing$
  $\mathsf{KD.init}(1, IV)$
  for $i = 1, \ldots, w$ do
    $\mathsf{KD.duplex}(\mathit{false}, P_i)$                    ▷ discard output
  for $i = 1, \ldots, \lceil t/r \rceil$ do
    $T \leftarrow T \parallel \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $\text{left}_t(T)$
```

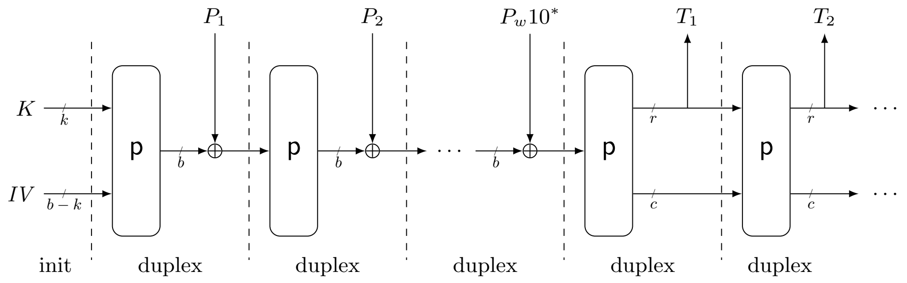


Fig. 7: Full-state keyed sponge FSKS. The function gets as input a key K and a plaintext P. The plaintext gets padded into w blocks as $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_b(P)$. It outputs tag blocks $(T_1, T_2, \ldots)$ truncated to t bits.

However, as mentioned in [32], this multiplicity term ρ should have been left implicit in the proof. Below, we derive a new, more advanced, security bound based on the results of Section 4.3. As before, we obtain two different results.

Theorem 6 (PRF security of full-state keyed sponge). *Let $b, c, r, k, t, \mu \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathsf{p} \xleftarrow{\$} \text{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D making at most q construction queries, of total length at most σ permutation calls, and N primitive queries, we have the following results:*

(i) Provided $\sigma + N \leq 0.1 \cdot 2^c$,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{FSKS}}(\mathsf{D}) \leq \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(q-1)N + \binom{q}{2}}{2^c} + \frac{(\sigma - q)q}{2^b - q} + \frac{2\binom{\sigma}{2}}{2^b} + \frac{q(\sigma - q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} \; ; \tag{23}$$

(ii) In general,

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{FSKS}}(\mathsf{D}) \le \frac{2\nu^{\sigma}_{r,c}(N+q)}{2^c} + \frac{(q-1)N + \binom{q-1}{2}}{2^c} + \frac{\binom{\sigma-q}{2} + (\sigma-q)q}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} + \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \tag{24}$$

The significant difference between this result and that of Theorem 5 is the term $\frac{(q-1)N+\binom{q}{2}}{2^c}$. This term gives an additional constraint compared to those of Theorem 4. Intuitively, provided that μ and $\nu^{2\sigma}_{r,c}$ are small enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$, $\sigma \ll \min\{2^{(c+k)/2}, 2^{b/2}\}$, and $q \cdot N \ll 2^c$.

Proof (Proof of Theorem 6). As in Theorem 3, we will first discuss how FSKS fits in the description of the duplex construction of Section 3.1, then we discuss what this means for the actual power of the distinguisher, i.e., the distinguisher's resources of Section 4.1, and finally we derive our bound.

The FSKS construction is, from a security perspective, quite different from the constructions of previous sections (TP, $\mathsf{P\text{-}SC}$, and $\mathsf{S\text{-}SC}$). The reason is (i) that there is no unique nonce/input that makes sure the state is always initialized to a different value, and (ii) that the distinguisher can freely choose the plaintexts input to the absorption part. In detail, each initialization simply sets the state to $K[\delta]\|IV$, where δ is the key index and IV an initial value that can be reused. In other words, we again have a key array $\boldsymbol{K} = (K[1], \dots, K[\mu]) \in (\{0,1\}^k)^\mu$ fed to the duplex construction, and we have $\alpha = 0$ and $\mathcal{IV}$ as defined.

The FSKS distinguisher D can make q construction queries, of total length at most σ duplexing calls (counting both the absorption and the squeezing phase), and N primitive queries. This particularly means that the parameters M and Q of the distinguisher's resources equal σ and q, respectively. Another parameter of the distinguisher's resources to consider is Q_{IV}, the maximum number of initialization calls for a single $IV \in \mathcal{IV}$. As in the proof of Theorem 3, this value is bounded by μ, as any IV can be queried alongside at most μ different keys.

So far, the quantification of the distinguisher's resources in the duplex setting is similar to that of Theorem 3 and Theorem 5. However, for the more advanced adversarial parameters, the quantification is more involved. The easiest one, Ω, which counts the number of duplexing calls for *flag* $=$ *true*, satisfies $\Omega = 0$ as before (see Algorithm 6). The value L, which counts the number of duplexing calls with repeated subpath, could be quite large. Indeed, suppose the distinguisher makes a query $P_1\|P_2\|P_3 10^*$ (padding included) and a query $P_1\|P_2\|P_3' 10^*$ where $P_3 \neq P_3'$, then the absorption of these two plaintexts consists of 4 distinct duplexing calls (noting that P_1 and P_2 are identical in both queries, but $P_3 \neq P_3'$), but the duplexing calls for P_3 and P_3' have the same subpath.[2] However, it is important to observe that two duplexing calls can have the same subpath *only*

[2] In practice, the distinguisher can actually *use* this to set the outer part of an absorption call to a value of its choice. To see this, ignore the IV and ignore padding for a moment, and assume that the distinguisher makes an evaluation of FSKS on input

if they are at the same distance to their corresponding initialization calls, as the length of a subpath is initialized at an initialization call and increases per duplexing call. In addition, once two subpaths are different, any extension of these subpaths will never be the same. This means that, per initialization call, the distinguisher can end up with a repeated subpath for a duplexing call *at most once*. Concretely, we obtain that $L \leq q-1$. Finally, regarding ν_{fix}, in the current example the distinguisher actually *can* set all L outer parts to a single value, and hence $\nu_{\mathsf{fix}} = L \leq q-1$.

To summarize, the distinguisher's resources of Section 4.1 satisfy:

parameter in Section 4.1	parameter in current proof
M	σ
N	N
Q	q
Q_{IV}	$\leq \mu$
L	$\leq q-1$
Ω	0
ν_{fix}	$\leq q-1$

If we plug these values into the bound (5) of Theorem 1, and simplify the bound at some points for readability, we obtain the following result, provided $\sigma + N \leq 0.1 \cdot 2^c$:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq{}& \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(q-1)N + \binom{q}{2}}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{25}$$

If we plug these values into the bound (6) of Theorem 2, and again simplify the bound at some points for readability, we obtain the following result:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq{}& \frac{2\nu_{r,c}^{\sigma}(N+q)}{2^c} + \frac{(q-1)N + \binom{q-1}{2}}{2^c} + \frac{\binom{\sigma-q}{2} + (\sigma-q)q}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{26}$$

Due to the presence of a fraction of order $O((qN+q^2)/2^c)$, one never exceeds the condition $\sigma + N \leq 0.1 \cdot 2^c$, and the bound (25) is better. However, in Section 8.3 we will discuss an extension of current result where this fraction happens to drop, in both (25) and (26). Therefore, we will continue with both bounds. The final step of translating these bounds into the multi-user PRF security of FSKS is identical to the reasoning in the proof of Theorem 3, and henceforth omitted. □

of a b-bit plaintext P to obtain an r-bit tag T. Then, it makes an evaluation of FSKS on input of $P\|T\|0^c$. The state of this duplex evaluation after the second duplexing call equals 0^r on its leftmost bits. This shows why the parameter L (and a similar issue occurs for the parameter Ω) is relevant for the security of the construction.

8.3 Application to Ascon-PRF

As mentioned in footnote 2 in the proof of Theorem 6, the distinguisher can actually use reappearing paths to actually set the outer r bits of a certain state to a certain value. The value $L + \Omega$ counts the number of cases this occurs. However, looking more closely at the example in this footnote, here the distinguisher exploits a subpath from which two distinct duplexing calls are made: one squeezing call and one absorbing call. It turns out that if reappearing paths only happen among absorbing calls, the situation is less worrisome. A good example of this is the Ascon-PRF construction [38]. Even though Ascon-PRF is an actual PRF function instantiated with the actual Ascon permutation [39], we will restrict our focus to the mode only (i.e., for arbitrary parameters and in the ideal permutation model).

Let $b, c, r, k, t \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathcal{IV} \subseteq \{0,1\}^{b-k}$ be a set of initialization vectors. The Ascon-PRF construction is defined as follows:

$$\begin{aligned} \mathsf{Ascon\text{-}PRF} : \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^* &\to \{0,1\}^t \\ (K, IV, P) &\mapsto T\,. \end{aligned} \tag{27}$$

Just like FSKS, the construction gets as input a key K, an initialization vector IV (which may be repeated), and a plaintext P, and outputs a tag T of length t bits. The main difference is that it has a special domain separator bit before squeezing. The construction is described in Algorithm 7 and is depicted in Figure 8. In case we consider multiple instances of the scheme, the key input in Algorithm 7 will be replaced by a key array $\boldsymbol{K} = (K[1], \dots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to KD.init will be the index of the instance that is evaluated.

Algorithm 7 Ascon-PRF[p]

```
Input: $(K, IV, P) \in \{0,1\}^k \times \mathcal{IV} \times \{0,1\}^*$
Output: $T \in \{0,1\}^t$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $(P_1, P_2, \dots, P_w) \leftarrow \mathrm{pad}_r^{10^*}(P)$
  $T \leftarrow \varnothing$
  $\mathsf{KD.init}(1, IV)$
  for $i = 1, \dots, w-1$ do
      $\mathsf{KD.duplex}(\mathit{false}, P_i)$                ▷ discard output
  $\mathsf{KD.duplex}(\mathit{false}, P_w \| 0^{c-1}1)$
  for $i = 1, \dots, \lceil t/r \rceil$ do
      $T \leftarrow T \| \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $\mathrm{left}_t(T)$
```

One could see the Ascon-PRF construction as a special case of FSKS, where the r-bit blocks $P_1, P_2, \dots, P_{w-1}$ can be appended with 0^c. However, the domain separation at the last plaintext block plays a prominent role: it makes sure that

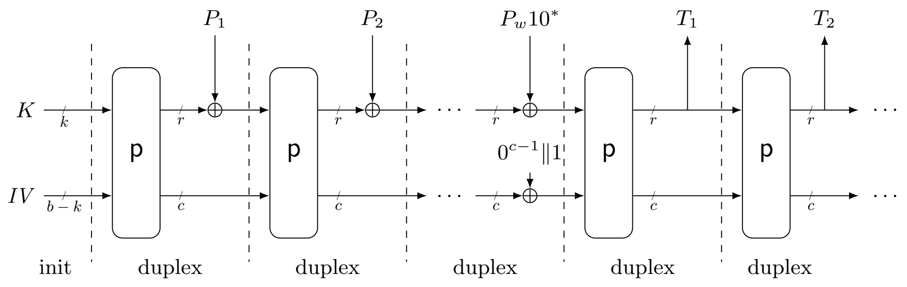


Fig. 8: The construction underlying Ascon-PRF. The function gets as input a key K, initial value IV, and a plaintext P. The plaintext gets padded into w blocks as $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_r(P)$. It outputs tag blocks $(T_1, T_2, \ldots)$ truncated to t bits.

there will never be two different evaluations of Ascon-PRF with reappearing subpaths, ending up in the squeezing phase at one evaluation and in the absorption phase at the other evaluation. In other words, the trick of footnote 2 does not work.

Nevertheless, the analysis of Daemen et al. [32] does not seem to accommodate this. This is caused by the fact that *in* the proof of Daemen et al., there is a certain lossy bounding. In detail, in the proof of Daemen et al., the following is stated: "Denote by S the size of the subset of [the] occasions for which the adversary can (in the worst case) force the outer part of [a state] to be a value of its choice. Note that $S \leq \binom{L+\Omega+1}{2}$." However, as the distinguisher can only force the outer part of a state to a value of its choice if it first learns it (through a squeezing call), and this never happens in Ascon-PRF, one can use a tighter bound of $S = 0$ here. Concretely, this means that of the bound (23) of Theorem 6, the term

$$\frac{\binom{L+\Omega+1}{2}}{2^c} = \frac{\binom{q}{2}}{2^c}$$

vanishes.

The bounding appears in disguise later on in the proof as well: "Therefore, if [we] take $T_{\text{fw}} = L + \Omega + \nu_{r,c}^{2(M-L)} \ldots$" If we drop the $L + \Omega$ here (for the same reason), it happens to be the case that

$$\frac{(L+\Omega)N}{2^c} = \frac{(q-1)N}{2^c}$$

vanishes from (23) as well.

The same improvement can be observed for (24) of Theorem 6, be it for different reasons. In the current case, we have $\nu_{\mathsf{fix}} = 0$, as the maximum number of duplexing calls for which the adversary can set the outer part to a single value

$\mathrm{left}_r(T)$ is 0. This means that in (24) the fraction

$$\frac{(q-1)N + \binom{q-1}{2}}{2^c}$$

never appears in the first place.

In summary, one obtains the following improved bounds for the Ascon-PRF mode, compared to (23) and (24) of Theorem 6.

Corollary 1 (PRF security of Ascon-PRF). *Let $b, c, r, k, t, \mu \in \mathbb{N}$, with $c+r = b$ and $k \leq b$. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D making at most q construction queries, of total length at most σ permutation calls, and N primitive queries, we have the following results:*

(i) Provided $\sigma + N \leq 0.1 \cdot 2^c$,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{Ascon\text{-}PRF}}(\mathsf{D}) \leq{}& \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma-q)q}{2^b-q} + \frac{2\binom{\sigma}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,;\end{aligned} \tag{28}$$

(ii) In general,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-prf}}_{\mathsf{Ascon\text{-}PRF}}(\mathsf{D}) \leq{}& \frac{2\nu^{\sigma}_{r,c}(N+q)}{2^c} + \frac{\binom{\sigma-q}{2} + (\sigma-q)q}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,.\end{aligned} \tag{29}$$

The term $\frac{(q-1)N+\binom{q}{2}}{2^c}$ that appeared in Theorem 6 has disappeared again. We thus end up with the conditions as originally stated for Theorem 5: provided that μ and $\nu^{2\sigma}_{r,c}$ are small enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$ and $\sigma \ll \min\{2^{(c+k)/2}, 2^{b/2}\}$.

9 Use Case 5: Authenticated Encryption

The main advantage of the duplex over the ordinary keyed sponge is that the duplex, unlike the keyed sponge, is very well-suited to design an authenticated encryption scheme. Already in the original introduction of the duplex construction, the designers proposed a mode for authenticated encryption called SpongeWrap [16].

At a very high level, the encryption of SpongeWrap took as input a key K, associated data A, and plaintext P, all of which were padded into $(r-2)$-bit blocks. To each block, a 0/1-bit was appended to assure domain separation.

Then, each block was appended with a 1 (we will get back to this later). Subsequently, each of the key, associated data, and plaintext blocks were processed one-by-one, where for the plaintext blocks, corresponding ciphertext blocks were derived from the state. Finally, a tag of required length was generated. This original SpongeWrap construction is depicted in Figure 9.

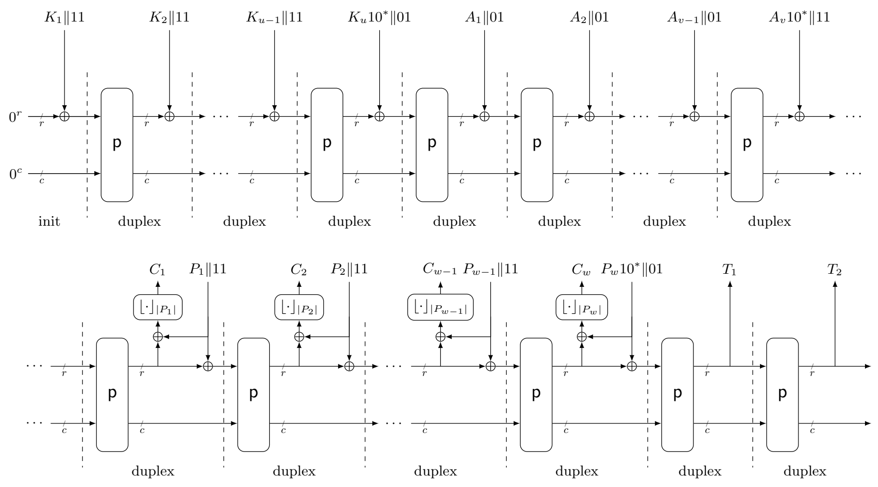


Fig. 9: Encryption of SpongeWrap. The function gets as input a key K, associated data A, and a plaintext P. The key gets padded into μ blocks as $(K_1, K_2, \ldots, K_u) \leftarrow \text{pad}_{r-2}(K)$, the associated data into v blocks as $(A_1, A_2, \ldots, A_v) \leftarrow \text{pad}_{r-2}(A)$, and the plaintext into w blocks as $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_{r-2}(P)$. It outputs a ciphertext $C = C_1\|C_2\|\cdots\|C_w$ of size $|P|$ and tag blocks $(T_1, T_2, \ldots)$ truncated to t bits.

The construction has formed an inspiration of many SpongeWrap based authenticated encryption schemes. However, none of them took the literal SpongeWrap construction as described in [16], but made adaptations to improve the efficiency or simplicity of the construction.

As a matter of fact, looking back at this original SpongeWrap construction from 2011, there are a few quite notable properties. These properties may look odd at first sight, but are, in retrospect (in particular in light of Section 3.4), perfectly understandable:

(1) As mentioned above, *each* key, associated data, and plaintext block is padded with a 1. The reason for this is that the original security proof of the duplex was reduced to the indifferentiability of the sponge hash function [14]. Stated differently, looking at Figure 9, the absorption of $K_1\|1\|1$ and the subsequent permutation call can be seen as a plain sponge hash function evaluation on input of $K_1\|1$, noting that the sponge does a 10^*-padding.

Next, the evaluation in Figure 9 up to the permutation after the absorption of $K_2\|1\|1$ can be seen as a plain sponge hash function evaluation on input of $K_1\|1\|1\|K_2\|1$, again noting that the sponge does a 10^*-padding. The exact same reasoning continues until the last absorbed block. Concretely, one can conclude that this consistent 1-padding to each block was an artifact of the proof technique adopted in [16], and this proof technique has become deprecated in light of follow-up work [32, 40, 61];

(2) The domain separator bits seem slightly off. In detail, domain separator bit 1 is used for all key blocks except for the last one, domain separator bit 0 is used for all associated data blocks except for the last one, and domain separator bit 1 is used for all plaintext blocks except for the last one. This seems odd, but seems to be a mere consequence of the phasing adopted in the original proof of Bertoni et al. [16]. In detail, Bertoni et al. followed the absorb-permute-squeeze approach (see Section 3.4), and the domain separator bit then corresponds to the "role" of the upcoming permutation call rather than the "type" of absorbed plaintext block. In light of the current permute-squeeze-absorb approach, a slight readjustment of the domain separator bits makes sense;

(3) Strictly seen, as the currently known duplex construction allows to absorb over the entire state (only squeezing happens with the outer r bits) the domain separation bits can spill over into the capacity. We note that it is also possible to absorb associated data in parallel to the message encryption, as in the construction of Sasaki and Yasuda [71] or in the full-state SpongeWrap [61], but it results in a more complicated description of the construction;

(4) The original SpongeWrap does not explicitly include a nonce, but rather expects the associated data to be different each time. We can make the nonce input explicit;

(5) The key is not used to initialize the state, but is (like associated data and plaintext) absorbed $r-2$ bits at a time. As pointed out in point (1) above, this design decision seems to be inspired from the reduction to the security of SpongeWrap to the indifferentiability of the sponge hash function. More generally, we have seen this in the outer-keyed sponge [3] as well. Although Mennink [57] demonstrated that this does not significantly degrade security, simply *initializing* the state with the key, and optionally also the nonce of point (4), conceptually simplifies the design. This idea is not new. As a matter of fact, in [19], Bertoni et al. described the MonkeyDuplex construction, which basically consists of the original duplex construction, but with the state initialized as $K\|NIV$ (exactly as in S-SC of Section 7). The duplexing part in this construction could, for example, be the associated data, plaintext/ciphertext, and tag portion of SpongeWrap, as suggested by the authors.

In this section, we will consider a generalization of the SpongeWrap construction that is based on the original SpongeWrap (Figure 9) but with the changes proposed in above five points taken into account. (Admittedly, the change of step (5) also makes it easier to apply our results of Section 4.3, as they do not natively support multi-round key absorption.) Given that our generalization is basically the $\mathsf{MonkeyDuplex}$ construction [19] with the duplexing part replaced by the associated data, plaintext/ciphertext, and tag portion of $\mathsf{SpongeWrap}$, we will refer to the scheme as $\mathsf{MonkeySpongeWrap}$.[3] The construction is described in Section 9.1 and its security is analyzed in Section 9.2.

9.1 Construction

Let $b, c, r, k, t \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. The generalized $\mathsf{SpongeWrap}$ authenticated encryption mode $\mathsf{MonkeySpongeWrap}$ consists of an encryption and a decryption algorithm:

$$\begin{aligned} \mathsf{ENC}: \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^* &\to \{0,1\}^\infty \times \{0,1\}^t \\ (K, NIV, A, P) &\mapsto (C, T)\,, \end{aligned} \tag{30}$$

$$\begin{aligned} \mathsf{DEC}: \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\to \{0,1\}^\infty \cup \{\perp\} \\ (K, NIV, A, C, T) &\mapsto P \text{ or } \perp\,. \end{aligned} \tag{31}$$

The encryption construction gets as input a key K, a nonce NIV, associated data A, and a plaintext P, and outputs a ciphertext C of length $|P|$ bits and a tag T of length t bits. It initializes a duplex with state $K \| NIV$, then it absorbs associated data blocks r bits at a time, then it does duplexing calls that both squeeze r bits (keystream to be added to the corresponding r-bit plaintext block) and absorb r bits (the actual plaintext block), and finally it squeezes tag blocks r bits at a time. The decryption construction gets as input a key K, a nonce NIV, associated data A, a ciphertext C, and a tag T, and outputs a plaintext P of length $|C|$ bits or $\perp$ if authentication failed. It differs from the encryption construction in two ways. The first difference is that the decryption of ciphertext blocks into plaintext blocks is done using duplexing calls that *overwrite* the outer r bits of the state. The second difference is that decryption does not output a tag, but instead it computes a new tag $T^\star$, which it subsequently compares with input T, and if the values match, the plaintext P is output. The encryption and decryption constructions are described in Algorithm 8 and are depicted in Figure 10 (encryption) and Figure 11 (decryption). In case we consider multiple instances of the scheme, the key input in Algorithm 8 will be replaced by a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$, and the first input to $\mathsf{KD.init}$ will be the index of the instance that is evaluated.

9.2 Security

The original $\mathsf{SpongeWrap}$ (Figure 9) was analyzed by Bertoni et al. [16] in its original introduction. However, the construction was subsequently rarely used as

[3] Credits of this name go to Richie Frame who coined this terminology before, at StackExchange in 2015, https://crypto.stackexchange.com/questions/31051/norx-duplex-padding.

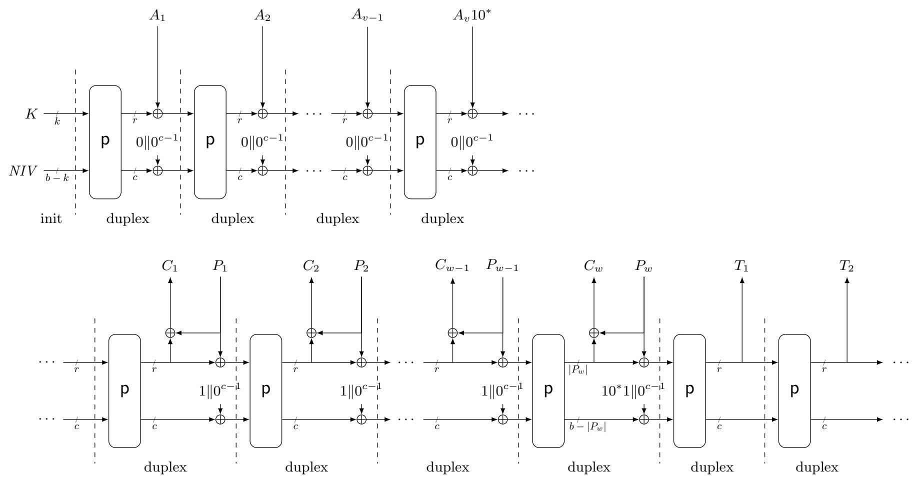


Fig. 10: Encryption of MonkeySpongeWrap. The function gets as input a key K, a nonce NIV, associated data A, and a plaintext P. The associated data gets padded into v blocks as $(A_1, A_2, \ldots, A_v) \leftarrow \text{pad}_r(A)$ and the plaintext into w blocks as $(P_1, P_2, \ldots, P_w) \leftarrow \text{pad}_r(P)$. It outputs a ciphertext $C = C_1\|C_2\|\cdots\|C_w$ of size $|P|$ and tag blocks $(T_1, T_2, \ldots)$ truncated to t bits.

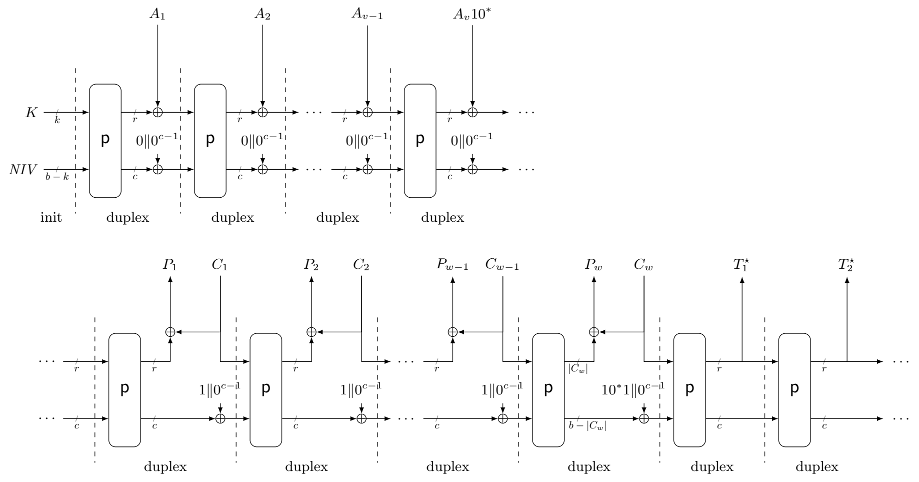


Fig. 11: Decryption of MonkeySpongeWrap. The function gets as input a key K, a nonce NIV, associated data A, a ciphertext C, and a tag T. The associated data gets padded into v blocks as $(A_1, A_2, \ldots, A_v) \leftarrow \text{pad}_r(A)$ and the ciphertext into w blocks as $(C_1, C_2, \ldots, C_w) \leftarrow \text{pad}_r(C)$. It subsequently computes tag blocks $(T_1^\star, T_2^\star, \ldots)$ truncated to t bits. If $T_i^\star = T_i$ for all $i = 1, 2, \ldots$, it outputs a plaintext $P = P_1\|P_2\|\cdots\|P_w$ of size $|C|$, otherwise it outputs $\perp$.

Algorithm 8 MonkeySpongeWrap[p]

```
Interface: ENC
Input: $(K, NIV, A, P) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^*$
Output: $(C, T) \in \{0,1\}^{|P|} \times \{0,1\}^t$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $(A_1, A_2, \ldots, A_v) \leftarrow \mathrm{pad}_r^{10^*}(A)$
  $(P_1, P_2, \ldots, P_w) \leftarrow \mathrm{pad}_r^{10^*}(P)$
  $C \leftarrow \varnothing$
  $T \leftarrow \varnothing$
  $\mathsf{KD.init}(1, NIV)$
  for $i = 1, \ldots, v$ do
      $\mathsf{KD.duplex}(\mathit{false}, A_i \| 0 \| 0^{c-1})$                ▷ discard output
  for $i = 1, \ldots, w$ do
      $C \leftarrow C \,\|\, \mathsf{KD.duplex}(\mathit{false}, P_i \| 1 \| 0^{c-1}) \oplus P_i$
  for $i = 1, \ldots, \lceil t/r \rceil$ do
      $T \leftarrow T \,\|\, \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $(\mathrm{left}_{|P|}(C), \mathrm{left}_t(T))$

Interface: DEC
Input: $(K, NIV, A, C, T) \in \{0,1\}^k \times \{0,1\}^{b-k} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t$
Output: $P \in \{0,1\}^{|C|}$ or $\perp$
Underlying keyed duplex: $\mathsf{KD}[\mathsf{p}]_{(K)}$
  $(A_1, A_2, \ldots, A_v) \leftarrow \mathrm{pad}_r^{10^*}(A)$
  $(C_1, C_2, \ldots, C_w) \leftarrow \mathrm{pad}_r^{10^*}(C)$
  $P \leftarrow \varnothing$
  $T^\star \leftarrow \varnothing$
  $\mathsf{KD.init}(1, NIV)$
  for $i = 1, \ldots, v$ do
      $\mathsf{KD.duplex}(\mathit{false}, A_i \| 0 \| 0^{c-1})$                ▷ discard output
  for $i = 1, \ldots, w$ do
      $P \leftarrow P \,\|\, \mathsf{KD.duplex}(\mathit{true}, C_i \| 1 \| 0^{c-1}) \oplus C_i$
  for $i = 1, \ldots, \lceil t/r \rceil$ do
      $T^\star \leftarrow T^\star \,\|\, \mathsf{KD.duplex}(\mathit{false}, 0^b)$
  return $\mathrm{left}_t(T) = \mathrm{left}_t(T^\star)$ ? $\mathrm{left}_{|C|}(P) : \perp$
```

such, but people rather resorted to "SpongeWrap-like" (or rather "MonkeySpongeWrap-like") constructions. These all differed subtly, resulting in various different proofs.

For example, security proofs for dedicated SpongeWrap-like constructions have been given by Andreeva et al. [1] (for APE), Jovanovic et al. [53] (for NORX), Sasaki and Yasuda [71] (for a variant with more efficient associated data absorption), Chakraborti et al. [22] (for Beetle), Dobraunig et al. [35] (for ISAP v2), and Chakraborty et al. [23] (for a generalized Beetle construction). Mennink et al. [61] introduced the full-state SpongeWrap, where the associated data is absorbed along with the message encryption, noting that the duplex allows for full-state absorption but only r-bit squeezing.

Beyond these results, various SpongeWrap-like constructions have been introduced whose security was claimed to follow from the original SpongeWrap result of Bertoni et al. [16]. These schemes, however, often had subtle differences compared to the original SpongeWrap, most importantly in the initialization with the key and nonce. Our generalized construction MonkeySpongeWrap, as such, is more broadly applicable. Below, we derive a security bound for the more general MonkeySpongeWrap, based on the results of Section 4.3. Afterwards, in Remark 5, we elaborate on the issues that come with release of unverified plaintext in MonkeySpongeWrap. As before, we obtain two different results.

Theorem 7 (AE security of MonkeySpongeWrap). *Let $b, c, r, k, t, \mu \in \mathbb{N}$, with $c + r = b$ and $k \leq b$. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation, and $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys. For any distinguisher D making at most q_e encryption construction queries, of total length at most σ_e permutation calls, q_d decryption construction queries, of total length at most σ_d permutation calls, writing $\sigma = \sigma_e + \sigma_d$ and $q = q_e + q_d$ for brevity, and N primitive queries, we have the following results:*

(i) Provided $\sigma_e + \sigma_d + N \leq 0.1 \cdot 2^c$,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) \leq{}& \frac{2\nu^{2\sigma}_{r,c}(N+1)}{2^c} + \frac{(\sigma_d + q_e - 1)N + \binom{\sigma_d+q_e}{2}}{2^c} \\ &+ \frac{(\sigma - q)q}{2^b - q} + \frac{2\binom{\sigma}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} + \frac{q_d}{2^t}\,;\end{aligned} \tag{32}$$

(ii) In general,

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-ae}}_{\mathsf{MonkeySpongeWrap}}(\mathsf{D}) \leq{}& \frac{2\nu^{\sigma}_{r,c}(N+\sigma_d+q_e)}{2^c} + \frac{(\sigma_d + q_e - 1)N + \binom{\sigma_d+q_e-1}{2}}{2^c} \\ &+ \frac{\binom{\sigma-q}{2} + (\sigma-q)(\sigma_d+q_e-1)}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} + \frac{q_d}{2^t}\,.\end{aligned} \tag{33}$$

With respect to implications, the bound is comparable to that of Theorem 6: the difference is that in the significant term, q is replaced by $\sigma_d + q_e$. Intuitively, provided that μ and $\nu^{2\sigma}_{r,c}$ are small enough and t large enough, the result implies security as long as $N \ll \min\{2^c, 2^k\}$, $\sigma \ll \min\{2^{(c+k)/2}, 2^{b/2}\}$, and $(\sigma_d+q_e)\cdot N \ll 2^c$. It is important to note that this last, most significant, restriction is on the number of q_e encryption *initialization calls* and σ_d decryption *duplexing calls*.

Proof (Proof of Theorem 7). As in Theorem 3, we will first discuss how MonkeySpongeWrap fits in the description of the duplex construction of Section 3.1, then we discuss what this means for the actual power of the distinguisher, i.e., the distinguisher's resources of Section 4.1, and finally we derive our bound.

The MonkeySpongeWrap construction is structurally different from the constructions of previous sections (TP, P-SC, S-SC, and FSKS), most importantly as the distinguisher has access to an encryption *and* a decryption interface. For encryption queries, we require nonce uniqueness which means that the state is always initialized to a different values. For decryption queries, however, the distinguisher may repeat nonces. Nevertheless, in either way, we have a key array $\boldsymbol{K} = (K[1], \ldots, K[\mu]) \in (\{0,1\}^k)^\mu$ fed to the duplex construction. In addition, $\mathcal{IV} = \{0,1\}^{b-k}$, i.e., $\mathcal{IV}$ simply determines the set of nonces of MonkeySpongeWrap. As before, the state is initialized as $K[\delta] \| NIV$ (hence $\alpha = 0$).

The MonkeySpongeWrap distinguisher D can make q_e encryption queries, of total length at most σ_e duplexing calls (counting both the absorption and the squeezing phase), q_d decryption construction queries, of total length at most σ_d permutation calls, and N primitive queries. This particularly means that the parameters M and Q of the distinguisher's resources equal $\sigma_e + \sigma_d$ and $q_e + q_d$, respectively. Another parameter of the distinguisher's resources to consider is Q_{IV}, the maximum number of initialization calls for a single $IV \in \mathcal{IV}$. As in the proof of Theorem 3, this value is bounded by μ, as any IV can be queried alongside at most μ different keys.

So far, the quantification of the distinguisher's resources in the duplex setting is similar to that of Theorem 3 and Theorem 5. However, for the more advanced adversarial parameters, the quantification is more involved. It makes sense to distinguish between encryption queries and decryption queries. Each encryption query is made for a different nonce and thus starts with a (δ, NIV) different from all earlier encryption queries. However, the nonce could have appeared in an earlier decryption query, and hence it could be that a subpath is repeated. This means that the encryption queries contribute at most q_e to L. As in encryption queries, no duplexing calls are made for $\mathit{flag} = \mathit{true}$, we conclude that they contribute 0 to Ω. Now, for decryption queries, the distinguisher can repeat nonces. As before, this means that decryption queries contribute at most q_d to L. In decryption queries, the duplexing calls corresponding to the deciphering of the ciphertext are made for $\mathit{flag} = \mathit{true}$, and there can be at most $\sigma_d - 2q_d$ such calls, where the subtraction of $2q_d$ comes from the fact that each decryption call contains at least two duplexing calls for $\mathit{flag} = \mathit{false}$. (Refer to Remark 5 below for some clarifications regarding these bounds.) We conclude that $L \le q_e + q_d - 1$ (noting that there must always be a first query that sets the first subpath), $\Omega \le \sigma_d - 2q_d$ and thus $\nu_{\mathsf{fix}} \le \sigma_d - 2q_d + q_e + q_d - 1 = \sigma_d - q_d + q_e - 1$.

To summarize, the distinguisher's resources of Section 4.1 satisfy:

parameter in Section 4.1	parameter in current proof
M	$\sigma_e + \sigma_d$
N	N
Q	$q_e + q_d$
Q_{IV}	$\leq \mu$
L	$\leq q_e + q_d - 1$
Ω	$\leq \sigma_d - 2q_d$
ν_{fix}	$\leq \sigma_d - q_d + q_e - 1$

Write $\sigma = \sigma_e + \sigma_d$ and $q = q_e + q_d$ for brevity. If we plug these values into the bound (5) of Theorem 1, and simplify the bound at some points for readability, we obtain the following result, provided $\sigma + N \leq 0.1 \cdot 2^c$:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq{}& \frac{2\nu_{r,c}^{2\sigma}(N+1)}{2^c} + \frac{(\sigma_d + q_e - 1)N + \binom{\sigma_d+q_e}{2}}{2^c} \\ &+ \frac{(\sigma-q)q}{2^b - q} + \frac{2\binom{\sigma}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \end{aligned} \tag{34}$$

If we plug these values into the bound (6) of Theorem 2, and again simplify the bound at some points for readability, we obtain the following result:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) \leq{}& \frac{2\nu_{r,c}^{\sigma}(N+\sigma_d+q_e)}{2^c} + \frac{(\sigma_d + q_e - 1)N + \binom{\sigma_d+q_e-1}{2}}{2^c} \\ &+ \frac{\binom{\sigma-q}{2} + (\sigma-q)(\sigma_d+q_e-1)}{2^b} + \frac{\binom{\sigma+N}{2} + \binom{N}{2}}{2^b} \\ &+ \frac{q(\sigma-q)}{2^{\min\{c+k,b\}}} + \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,. \end{aligned} \tag{35}$$

Note that, typically, $\sigma_d \ll \sigma_e$. This means that, as before, the bound (34) is in general better, but only holds provided $q + N \leq 0.1 \cdot 2^c$. This means that (35) is still meaningful as it may stay well below 1 once $q + N$ goes beyond this bound. We will thus continue with both bounds.

What remains is to translate these bounds into the multi-user AE security of MonkeySpongeWrap. The reasoning is similar as in the proof of Theorem 3, but differs slightly in the decryption functionality of MonkeySpongeWrap. We can note that the formal description MonkeySpongeWrap[p] of Algorithm 3 *is* in fact a description MonkeySpongeWrap[KD[p]], where the key array $\boldsymbol{K}$ input to TP is directly fed into KD. Thus, denoting $\mathbf{R}^{\mathrm{ae}} = (\mathsf{R}_1^{\mathrm{ae}}, \ldots, \mathsf{R}_\mu^{\mathrm{ae}})$ as a list of μ random functions as defined in Section 2.3, and $\bot$ as a list of μ $\bot$-symbols,

$$\begin{aligned}
\mathbf{Adv}^{\mu\text{-ae}}_{\mathsf{AE}}(\mathsf{D}) &= \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{KD}[\mathsf{p}]]_{\boldsymbol{K}}, \mathsf{DEC}[\mathsf{KD}[\mathsf{p}]]_{\boldsymbol{K}}, \mathsf{p}^{\pm}\ ;\ \mathbf{R}^{\text{ae}}, \bot, \mathsf{p}^{\pm}\right) \\
&= \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{DEC}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{p}^{\pm}\ ;\ \mathbf{R}^{\text{ae}}, \bot, \mathsf{p}^{\pm}\right) \\
&\leq \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{DEC}[\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}], \mathsf{p}^{\pm}\ ;\ \mathsf{ENC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{DEC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\right) \quad (36) \\
&\quad + \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{DEC}[\mathsf{IXIF}[\mathsf{ro}]], \mathsf{p}^{\pm}\ ;\ \mathsf{ENC}[\mathsf{IXIF}[\mathsf{ro}]], \bot, \mathsf{p}^{\pm}\right) \quad (37) \\
&\quad + \Delta_{\mathsf{D}}\left(\mathsf{ENC}[\mathsf{IXIF}[\mathsf{ro}]], \bot, \mathsf{p}^{\pm}\ ;\ \mathbf{R}^{\text{ae}}, \bot, \mathsf{p}^{\pm}\right)\ . \quad (38)
\end{aligned}$$

The distance of (36) is the security of KD as bounded in (34) or (35), the distance of (37) is at most $q_d/2^t$ as distinguishing both world can only be done by a random tag guess, and the distance of (38) equals 0 as both oracles output uniform random and independent strings for each input. □

Remark 5. In the proof of Theorem 7, it is mentioned that any encryption query may have a colliding subpath with an earlier decryption query, and hence that q_e decryption queries contribute at most q_e to L. Typically, however, these queries with colliding subpaths are not expected to help the distinguisher as it likely has not seen the output of the corresponding decryption query. Likewise for decryption queries, the distinguisher can enforce a repeated subpath (contributing at most q_d to L), and the distinguisher can even set up to $\sigma_d - 2q_d$ outer values of its choice. Again, here we are overly generous to the distinguisher in that it never sees the output of the decryption, unless with negligible probability, and it is unlikely the distinguisher can deduce any information from those queries. In a strict sense, the counting performed here is comparable to what would be done in a setting where unverified plaintexts are released [2]. Concretely, if the distinguisher makes a decryption query, the permutation calls are defined internally and collisions may occur, and this is the reason the bounds on L and Ω are defined as such. (We do not claim that any security of MonkeySpongeWrap under release of unverified plaintext can be concluded from Theorem 7.)

10 Conclusion

The potential of the general full-state keyed duplex is huge, as it can be (and actually, *is*) used to describe a wide range of permutation based symmetric cryptographic schemes. However, the explicit use of the security bounds of the keyed duplex has been left behind, in part due to the generality of the construction as well as the generality of the security analysis. In this paper, we aimed to give a comprehensive overview of (i) how the duplex is defined in general, (ii) why it is defined as such, (iii) how the general security bounds look like, (iv) why they look like this, and finally, (v) how we can actually use these bounds by ways of simple yet practical applications.

The applications presented in Sections 5-9 are only the tip of the iceberg. Other potential applications not covered in these sections include reseedable pseudorandom sequence generation [15, 45], password-based key derivation [64], Beetle-style authenticated encryption [22], and more. However, the same mechanics as in Sections 5-9 can be used to apply the bounds of Section 4 to these types of construction.

Having said that, the results in this work also deserve a word of caution, namely that all results in this work only hold in the ideal permutation model. This means that the underlying permutation is assumed to be perfectly random. If instantiated with an actual permutation like the permutation of Keccak [65], of Ascon [39], or of PHOTON [50], the security may be lower. At the very least, our bounds guarantee security under the assumption that the adversary does not take any internal primitives of the permutation into account; breaking the scheme faster than the proven bounds requires the adversary to dive into the permutation and exploit certain properties there.

Acknowledgements. We want to thank the authors of [32] and [40] for the many insightful discussions. Bart Mennink is supported by the Netherlands Organisation for Scientific Research (NWO) under grant VI.Vidi.203.099.

References

1. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 168–186. Springer (2014), https://doi.org/10.1007/978-3-662-46706-0_9
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 105–125. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_6
3. Andreeva, E., Daemen, J., Mennink, B., Van Assche, G.: Security of Keyed Sponge Constructions Using a Modular Proof Approach. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 364–384. Springer (2015), https://doi.org/10.1007/978-3-662-48116-5_18
4. Andreeva, E., Mennink, B., Preneel, B.: Security Reductions of the Second Round SHA-3 Candidates. In: Burmester, M., Tsudik, G., Magliveras, S.S., Ilic, I. (eds.) Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6531, pp. 39–53. Springer (2010), https://doi.org/10.1007/978-3-642-18178-8_5
5. Aumasson, J., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A Lightweight Hash. In: Mangard, S., Standaert, F. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA,

August 17-20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225, pp. 1–15. Springer (2010), https://doi.org/10.1007/978-3-642-15031-9_1

6. Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., Yasuda, K.: PHOTON-Beetle. Final Round Submission to NIST Lightweight Cryptography (2019)
7. Beierle, C., Biryukov, A., Cardoso dos Santos, L., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., Moradi, A., Shahmirzadi, A.: Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family. Final Round Submission to NIST Lightweight Cryptography (2019)
8. Bellare, M., Impagliazzo, R.: A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to PRP to PRF conversion. Cryptology ePrint Archive, Report 1999/024 (1999), http://eprint.iacr.org/1999/024
9. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 341–358. Springer (1994), https://doi.org/10.1007/3-540-48658-5_32
10. Bellare, M., Krovetz, T., Rogaway, P.: Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-invertible. In: Nyberg, K. (ed.) Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. Lecture Notes in Computer Science, vol. 1403, pp. 266–280. Springer (1998), https://doi.org/10.1007/BFb0054132
11. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM (1993), https://doi.org/10.1145/168588.168596
12. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006), https://doi.org/10.1007/11761679_25
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
14. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer (2008), https://doi.org/10.1007/978-3-540-78967-3_11
15. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge-Based Pseudo-Random Number Generators. In: Mangard, S., Standaert, F. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225, pp. 33–47. Springer (2010), https://doi.org/10.1007/978-3-642-15031-9_3

16. 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.) Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011), https://doi.org/10.1007/978-3-642-28496-0_19
17. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (January 2011)
18. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the security of the keyed sponge construction. Symmetric Key Encryption Workshop (February 2011)
19. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-based encryption, authentication and authenticated encryption. Directions in Authenticated Ciphers (July 2012)
20. Bhattacharya, S., Nandi, M.: A note on the chi-square method: A tool for proving cryptographic security. Cryptogr. Commun. 10(5), 935–957 (2018), https://doi.org/10.1007/s12095-017-0276-z
21. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: spongent: A Lightweight Hash Function. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6917, pp. 312–325. Springer (2011), https://doi.org/10.1007/978-3-642-23951-9_21
22. Chakraborti, A., Datta, N., Nandi, M., Yasuda, K.: Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(2), 218–241 (2018), https://doi.org/10.13154/tches.v2018.i2.218-241
23. Chakraborty, B., Jha, A., Nandi, M.: On the Security of Sponge-type Authenticated Encryption Modes. IACR Trans. Symmetric Cryptol. 2020(2), 93–119 (2020), https://doi.org/10.13154/tosc.v2020.i2.93-119
24. Chang, D., Dworkin, M., Hong, S., Kelsey, J., Nandi, M.: A keyed sponge construction with pseudorandomness in the standard model. NIST SHA–3 Workshop (March 2012)
25. Chang, D., Nandi, M.: A Short Proof of the PRP/PRF Switching Lemma. Cryptology ePrint Archive, Report 2008/078 (2008), http://eprint.iacr.org/2008/078
26. Chen, Y.L., Lambooij, E., Mennink, B.: How to Build Pseudorandom Functions from Public Random Permutations. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 266–293. Springer (2019), https://doi.org/10.1007/978-3-030-26948-7_10
27. Choi, W., Lee, B., Lee, J.: Indifferentiability of Truncated Random Permutations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 175–195. Springer (2019), https://doi.org/10.1007/978-3-030-34578-5_7
28. Cogliati, B., Lampe, R., Patarin, J.: The Indistinguishability of the XOR of k Permutations. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 285–302. Springer (2014), https://doi.org/10.1007/978-3-662-46706-0_15

29. Cogliati, B., Seurin, Y.: EWCDM: An Efficient, Beyond-Birthday Secure, Nonce-Misuse Resistant MAC. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 121–149. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_5
30. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 430–448. Springer (2005), https://doi.org/10.1007/11535218_26
31. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. Final Round Submission to NIST Lightweight Cryptography (2019)
32. Daemen, J., Mennink, B., Van Assche, G.: Full-State Keyed Duplex with Built-In Multi-user Support. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10625, pp. 606–637. Springer (2017), https://doi.org/10.1007/978-3-319-70697-9_21
33. Dai, W., Hoang, V.T., Tessaro, S.: Information-Theoretic Indistinguishability via the Chi-Squared Method. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 497–523. Springer (2017), https://doi.org/10.1007/978-3-319-63697-9_17
34. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP v2. Final Round Submission to NIST Lightweight Cryptography (2019)
35. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: Isap v2.0. IACR Trans. Symmetric Cryptol. 2020(S1), 390–416 (2020), https://doi.org/10.13154/tosc.v2020.iS1.390-416
36. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - Towards Side-Channel Secure Authenticated Encryption. IACR Trans. Symmetric Cryptol. 2017(1), 80–105 (2017), https://doi.org/10.13154/tosc.v2017.i1.80-105
37. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Winning Submission to NIST Lightweight Cryptography (2019)
38. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon PRF, MAC, and Short-Input MAC. Cryptology ePrint Archive, Report 2021/1574 (2021), http://eprint.iacr.org/2021/1574
39. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight Authenticated Encryption and Hashing. J. Cryptol. 34(3), 33 (2021), https://doi.org/10.1007/s00145-021-09398-9
40. Dobraunig, C., Mennink, B.: Leakage Resilience of the Duplex Construction. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 225–255. Springer (2019), https://doi.org/10.1007/978-3-030-34618-8_8

41. Dobraunig, C., Mennink, B.: Security of the Suffix Keyed Sponge. IACR Trans. Symmetric Cryptol. 2019(4), 223–248 (2019), https://doi.org/10.13154/tosc.v2019.i4.223-248
42. Dobraunig, C., Mennink, B., Primas, R.: Leakage and Tamper Resilient Permutation-Based Cryptography. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022. pp. 859–873. ACM (2022), https://doi.org/10.1145/3548606.3560635
43. Dutta, A., Nandi, M., Talnikar, S.: Permutation Based EDM: An Inverse Free BBB Secure PRF. IACR Trans. Symmetric Cryptol. 2021(2), 31–70 (2021), https://doi.org/10.46586/tosc.v2021.i2.31-70
44. 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.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 368–387. Springer (2015), https://doi.org/10.1007/978-3-662-47989-6_18
45. Gazi, P., Tessaro, S.: Provably Robust Sponge-Based PRNGs and KDFs. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 87–116. Springer (2016), https://doi.org/10.1007/978-3-662-49890-3_4
46. Gilboa, S., Gueron, S.: Distinguishing a truncated random permutation from a random function. Cryptology ePrint Archive, Report 2015/773 (2015), http://eprint.iacr.org/2015/773
47. Gilboa, S., Gueron, S.: The Advantage of Truncated Permutations. CoRR abs/1610.02518 (2016), http://arxiv.org/abs/1610.02518
48. Gilboa, S., Gueron, S., Morris, B.: How Many Queries are Needed to Distinguish a Truncated Random Permutation from a Random Function? J. Cryptol. 31(1), 162–171 (2018), https://doi.org/10.1007/s00145-017-9253-0
49. Gunsing, A., Mennink, B.: The Summation-Truncation Hybrid: Reusing Discarded Bits for Free. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 187–217. Springer (2020), https://doi.org/10.1007/978-3-030-56784-2_7
50. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011), https://doi.org/10.1007/978-3-642-22792-9_13
51. Hall, C., Wagner, D.A., Kelsey, J., Schneier, B.: Building PRFs from PRPs. In: Krawczyk, H. (ed.) Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1462, pp. 370–389. Springer (1998), https://doi.org/10.1007/BFb0055742
52. Impagliazzo, R., Rudich, S.: Limits on the Provable Consequences of One-way Permutations. In: Goldwasser, S. (ed.) Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA,

August 21-25, 1988, Proceedings. Lecture Notes in Computer Science, vol. 403, pp. 8–26. Springer (1988), https://doi.org/10.1007/0-387-34799-2_2

53. Jovanovic, P., Luykx, A., Mennink, B.: Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 85–104. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_5
54. Lefevre, C., Mennink, B.: Tight Preimage Resistance of the Sponge Construction. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 185–204. Springer (2022), https://doi.org/10.1007/978-3-031-15985-5_7
55. Lucks, S.: The Sum of PRPs Is a Secure PRF. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 470–484. Springer (2000), https://doi.org/10.1007/3-540-45539-6_34
56. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2951, pp. 21–39. Springer (2004), https://doi.org/10.1007/978-3-540-24638-1_2
57. Mennink, B.: Key Prediction Security of Keyed Sponges. IACR Trans. Symmetric Cryptol. 2018(4), 128–149 (2018), https://doi.org/10.13154/tosc.v2018.i4.128-149
58. Mennink, B.: Linking Stam's Bounds with Generalized Truncation. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11405, pp. 313–329. Springer (2019), https://doi.org/10.1007/978-3-030-12612-4_16
59. Mennink, B., Neves, S.: Encrypted Davies-Meyer and Its Dual: Towards Optimal Security Using Mirror Theory. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 556–583. Springer (2017), https://doi.org/10.1007/978-3-319-63697-9_19
60. Mennink, B., Preneel, B.: On the XOR of Multiple Random Permutations. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9092, pp. 619–634. Springer (2015), https://doi.org/10.1007/978-3-319-28166-7_30
61. Mennink, B., Reyhanitabar, R., Vizár, D.: Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland,

New Zealand, November 29 - December 3, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 465–489. Springer (2015), https://doi.org/10.1007/978-3-662-48800-3_19

62. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8781, pp. 306–323. Springer (2014), https://doi.org/10.1007/978-3-319-13051-4_19
63. Naito, Y., Yasuda, K.: New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 3–22. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_1
64. National Institute of Standards and Technology: NIST Special Publication 800-132: Recommendation for Password-Based Key Derivation: Part 1: Storage Applications (December 2010), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
65. National Institute of Standards and Technology: FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (August 2015), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
66. NIST: SHA-3 Project (February 2007), https://csrc.nist.gov/projects/hash-functions/sha-3-project
67. NIST: Lightweight Cryptography (February 2019), https://csrc.nist.gov/Projects/Lightweight-Cryptography
68. Patarin, J.: A Proof of Security in O(2n) for the Xor of Two Random Permutations. In: Safavi-Naini, R. (ed.) Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings. Lecture Notes in Computer Science, vol. 5155, pp. 232–248. Springer (2008), https://doi.org/10.1007/978-3-540-85093-9_22
69. Patarin, J.: Introduction to Mirror Theory: Analysis of Systems of Linear Equalities and Linear Non Equalities for Cryptography. Cryptology ePrint Archive, Report 2010/287 (2010), http://eprint.iacr.org/2010/287
70. Patarin, J.: Security in $O(2^n)$ for the Xor of Two Random Permutations – Proof with the standard H technique–. Cryptology ePrint Archive, Report 2013/368 (2013), http://eprint.iacr.org/2013/368
71. Sasaki, Y., Yasuda, K.: How to Incorporate Associated Data in Sponge-Based Authenticated Encryption. In: Nyberg, K. (ed.) Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9048, pp. 353–370. Springer (2015), https://doi.org/10.1007/978-3-319-16715-2_19
72. Stam, A.J.: Distance between sampling with and without replacement. Statistica Neerlandica 32(2), 81–91 (1978), https://dx.doi.org/10.1111/j.1467-9574.1978.tb01387.x
73. Taha, M.M.I., Schaumont, P.: Side-channel countermeasure for SHA-3 at almost-zero area overhead. In: 2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014. pp. 93–96. IEEE Computer Society (2014), https://doi.org/10.1109/HST.2014.6855576

A Python Script for Solving (4)

```
import sys
import math

def FindMinX(b,L):
    # Stores the minimum
    MinX = sys.maxsize;

    # Finds the minimum
    x = 0;
    while MinX == sys.maxsize:
        x = x+1;
        if x > 2**L:
            if ( b*math.log(2)-2**(L) + x*math.log(2**(L))
                        <= math.log(x-2**(L))
                        + math.log(math.factorial(x)) ):
                MinX = x;

    # Returns the minimum
    return MinX;

if __name__ == "__main__":
    b = 400;
    for L in range(-b,22):
        print("M/2^r = 2^",L,", minimal x = ",FindMinX(b,L));
```