Generalized Initialization of the Duplex Construction

Christoph Dobraunig[1] and Bart Mennink[2]

[1] Intel Labs, Hillsboro, USA
christoph.dobraunig@intel.com
[2] Radboud University, Nijmegen, The Netherlands
b.mennink@cs.ru.nl

Abstract. The duplex construction is already well analyzed with many papers proving its security in the random permutation model. However, so far, the first phase of the duplex, where the state is initialized with a secret key and an initialization vector (IV), is typically analyzed in a worst case manner. More detailed, it is always assumed that the adversary is allowed to choose the IV on its will. In this paper, we analyze how the security changes if restrictions on the choice of the IV are imposed, varying from the global nonce case over the random IV case to the IV on key case. The last one, in particular, is the duplex analogue of the use of a nonce masked with a secret in AES-GCM in TLS 1.3. We apply our findings to duplex-based encryption and authenticated encryption, and discuss the practical applications of our results.

Keywords: symmetric cryptography, duplex construction, initialization vector, nonce

1 Introduction

The duplex construction of Bertoni et al. [8], the sibling of the sponge construction [7], lends itself to efficiently fulfill many cryptographic tasks including encryption and authenticated encryption. The impact the duplex has on symmetric cryptography has recently been impressively showcased in the NIST lightweight standardization process, where 5 (including the winner) [1,2,9,12,16] out of 10 algorithms follow the duplex construction or are duplex-inspired. Those schemes are nonce-based authenticated encryption schemes with the property that their security relies just on the uniqueness of the nonce, but not on how it is chosen. The same level of nonce formalism is also followed by the various security proofs of the various different duplex constructions we have seen in literature [8, 11, 17, 21, 22]: the proofs in large parts assume an attacker-favorable choice of nonce, or, as called more generally in the duplex, the initialization vector (IV).

In this work, we will restrict our focus to the very general duplex construction of Daemen et al. [11], but with the rephasing suggested by Mennink [21]. (We also refer to the excellent work of Mennink [21] for a detailed treatment of the duplex in full generality (in his Section 3.1) and the role and importance of rephasing (in his Section 3.4).) In a nutshell, the duplex consists of an initialization interface, where a b-bit state is initialized using a key K and an initialization vector IV. Then an arbitrary amount of duplexing calls can be made. Each duplexing call first evaluates a cryptographic b-bit permutation p on the state, then squeezes its r outermost bits of the state (as digest), and it absorbs a b-bit message. There is an additional input, namely a flag, that indicates whether the r outermost bits of the message will be added to the outer part of the state, or will overwrite that part. A detailed version of this duplex construction — or, in fact, a construction that generalizes the initialization function as we will discuss shortly — is given in Section 3.1.

Daemen et al. [11] derived a general security bound of this duplex construction, which gives an indication of the degree at which the construction is indistinguishable from random depending on a fine-grained set of adversarial resources. Dobraunig and Mennink [17] considered this construction in the leakage resilience setting. A crucial observation in their security analyses is that, intuitively, security of the scheme is guaranteed as long as the intermediate inner parts stay unknown to the adversary and never collide. Different keys or different IVs will give a different initialization, and thus (likely) different and unpredictable inner parts after evaluation of the permutation. However, current duplex results assume that the attacker has full control over the IV, and the influence of initialization calls on the security bound only appears in two terms of the entire security bound:

$$\frac{\mu \cdot N}{2^k} + \frac{\binom{\mu}{2}}{2^k}\,, \tag{1}$$

where k is the key size, μ the number of users/keys, and N is the number of permutation queries that the adversary can make. The terms of (1) are the last two terms of (5c) of the bound of Theorem 1, where in general $Q_i \leq \mu$. The first term of (1) comes from the event that the adversary can "guess a key", i.e., it makes a permutation query (N attempts) for one of the μ k-bit keys. The second term comes from "unlucky key collisions", as they would lead to colliding inner parts for different users.[3]

However, as said, these bounds assume that the attacker has fairly full control over the choice of IV and they are merely dominated by the adversarial power in guessing one of the μ keys. However, in practice, restrictions on the IV are quite conceivable. A typical example of this is the idea of masking the IV with a secret, similar to AES-GCM in TLS 1.3 [23]. In this paper, we investigate how the security bound of the duplex construction changes if we impose restrictions on the IV.

[3] We remark that there is a third term influenced by initialization calls, namely the first term of (5c) of Theorem 1. This term is not relevant for the introductory discussion of our work, but is taken into account in the technical analysis; see also Section 3.5.

1.1 Improved Bounds for Different IV Usages

The two terms capturing "guessing the key" and "unlucky key collisions" of (1) are tight in the general case, where the attacker is allowed to repeat IVs under different keys. However, if this is not the case, these two terms change and can be significantly improved in some use cases.

To investigate the exact role of the initialization in the duplex, we first consider a generalization of it in Section 3.1. In this generalization, the initialization does not anymore consist of just a concatenation of the key with the IV, but instead, we consider initialization as a concatenation of two functions that initialize both parts of the state on input of the key array and an (always unique) index. The generalization seems subtle but is strictly necessary to capture applications that rely on random IVs.

Now, using our generalized duplex construction, we consider five different practical cases to initialize the state of the duplex.

- The first case we consider is the global IV in Section 4.1. This principle is just the description of the idea of a globally unique key identifier presented by, e.g., Daemen et al. [9]: the IV is unique over all μ users. Intuitively (but technically the story is a little bit more involved), this ensures that $\mu = 1$ and we get
$$\frac{N}{2^k} + \frac{0}{2^k}\,;$$
- For the second case, we consider randomly chosen IVs in Section 4.2. If it can be ensured that the IV is chosen randomly outside the influence of an attacker, the collision of IVs for a single user or between different users becomes probabilistic. On the other hand, contrary to the attacker-favorable case where all parties may start using as IV an encoding of 0 and count upwards, a randomly chosen IV does improve the bound, although it becomes a bit more technical;
- For the third case, we assume that the involved parties agree on a random offset from which they then count upwards in Section 4.3. Compared to the previous case, this eliminates IV collisions of a single user, thus improving the bound further;
- As a fourth scenario, we consider that the IV is masked with additional key material in Section 4.4. In contrast to all previous scenarios, the actual choice of the IV does not matter in this case, as long as it is unique. This is akin to what is done in TLS 1.3 for AES-GCM [23], which is analyzed in [5]. We show that this is also possible for the duplex, with a potential strong uplift in the security bound, because for the duplex, this has a similar (albeit a bit weaker) effect than just extending the size of the key;
- Finally, we show that the security can further be improved, by combining the ideas of a masked IV (fourth case) with a global IV (first case) in Section 4.5.

For all five cases, the precise details/conditions and improved security bounds are given in Section 4. We apply the improved bounds to duplex-based stream encryption in Section 5 and authenticated encryption in Section 6, and we discuss the practical meaning and limitations of the results using a typical parameter set inspired by the NIST Lightweight Cryptography winner Ascon [15, 16] in Section 7.

1.2 Outline

We describe basic preliminaries in Section 2. The duplex construction, including our generalization with respect to the initialization interface, is given in Section 3. This section also includes the duplex security model and a copy of the existing security result (adapted to our generalization) of Daemen et al. [11]. We derive improved security bounds for the five types of initialization in Section 4. These improved bounds are then applied to stream encryption in Section 5 and authenticated encryption in Section 6. Afterwards, we discuss practical implication of our results in Section 7 and conclude in Section 8.

2 Preliminaries

We consider 0 to be a member of $\mathbb{N}$. For a value $n \in \mathbb{N}$, the set of bit strings of length n is defined as $\{0,1\}^n$. We denote $\{0,1\}^* = \cup_{n\in\mathbb{N}}\{0,1\}^n$, and we denote the set of infinitely long strings by $\{0,1\}^\infty$. We define by $\text{perm}(n)$ the set of all permutations p on $\{0,1\}^n$. For a bit string $X \in \{0,1\}^*$, the length of X is denoted as $|X|$, and for an additional bit string $Y \in \{0,1\}^*$, the concatenation of X and Y is denoted as $X\|Y$ and their bitwise exclusive or (XOR) as $X \oplus Y$ truncated to $\min\{|X|, |Y|\}$ bits. For a finite set $\mathcal{S}$, we denote by $S \xleftarrow{\$} \mathcal{S}$ the uniformly random drawing of an element S from a finite set $\mathcal{S}$.

For $X \in \{0,1\}^n$ and for $m \leq n$, the function $\text{left}_m(X)$ outputs the m leftmost bits of X and $\text{right}_m(X)$ the m rightmost bits of X. For $Y \in \{1,\ldots,2^n\}$, we denote by $\text{encode}_n[Y]$ the encoding of Y as an n-bit string.

2.1 Distinguisher

A distinguisher D is an algorithm. It is given access to a list of oracles, either O or P, which we denote as D^{O} or D^{P}. It can make queries to its list of oracles, and in the end it outputs a decision bit $b \in \{0,1\}$. We denote by

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

the advantage that distinguisher D has in distinguishing the lists of oracles O from P. In our work, distinguishers have unbounded computational power, and we will measure their success probabilities solely by the number of queries made (see also Section 3.3).

2.2 Multicollision Limit Function

We will use the notion of multicollision limit functions from Daemen et al. [11], which considers a balls-into-bins experiment tailored to sponge constructions.

Definition 1 (multicollision limit function (mulf)). *Let $Q, c, r \in \mathbb{N}$. Consider the experiment of throwing Q 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 (mulf) $\nu_{r,c}^{Q}$ as the smallest natural number x that satisfies*

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

The mulf appears a bit artificial, but can be easily bounded. In particular, Daemen et al. [11, Section 6.5] demonstrated that the value x targeted in Definition 1 satisfies

$$\frac{2^b e^{-\lambda} \lambda^x}{(x-\lambda)x!} \leq 1\,. \tag{2}$$

where $\lambda = Q/2^r$. We will not dive into detail on how this bound is derived, but will highlight the core idea as we will use it later on. This bound is obtained by observing that for any particular bin, the number of balls in that bin follows the binomial distribution with $n = Q$ trials and probability and $p = 1/2^r$, and for large enough values it is upper bounded by a Poisson distribution with mean $\lambda = np = Q/2^r$. We refer to [11, Section 6.5] for the detailed reasoning behind (2) and to Mennink [21, Section 4.2] for a more detailed discussion of the mulf in general.

3 Duplex Construction and Security

We will consider the description of the duplex construction of Daemen et al. [11] and Dobraunig and Mennink [17]. We will, however, adopt the description of Mennink [21], who described the duplex in the more convenient permute-squeeze-absorb phasing (a detailed discussion on the sponge phasing is given in [21, Section 3.4]). However, we will not consider this duplex construction as is: we will consider a generalization to be able to properly and rigorously study the initialization of the duplex. Our generalized duplex construction is given in Section 3.1. We describe the security model in Section 3.2 and an explanation on how to parametrize distinguishers in Section 3.3. These two sections are based on [21, Sections 3.2, 3.3, and 4.1], but generalized to reflect the generalization in the initialization phase. The security result of Daemen et al. [11] for the generalized duplex construction *for the default (baseline) initialization* is given in Section 3.4. Finally, in Section 3.5 we dive into the security proof of Daemen et al., isolate the parts in the security proof where the initialization plays a role, and re-describe them for our generalized initialization phase.

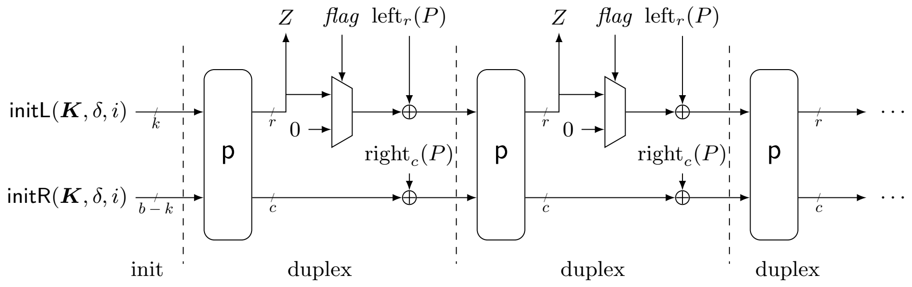

Fig. 1: The duplexing interface of KD. The sole difference with the duplexing interface of Mennink [21] is in the initialization.

3.1 Construction

Let $b, c, r, k, l, \mu \in \mathbb{N}$ such that $c + r = b$ and $k \leq b$. We describe our generalized version of the keyed duplex construction KD in Algorithm 1. It operates on 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 $\mathsf{p} \in \text{perm}(b)$. The construction internally maintains a b-bit state S, and has two interfaces: KD.init and KD.duplex. Typical interface calls are depicted in Figure 1.

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

```
Interface: KD.init
Input: $(\delta, i) \in \{1, \ldots, \mu\} \times \{1, \ldots, 2^l\}$
Output: $\varnothing$
  $S \leftarrow \mathsf{initL}(\boldsymbol{K}, \delta, i) \,\|\, \mathsf{initR}(\boldsymbol{K}, \delta, i)$
  return $\varnothing$

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

Duplexing Interface. The duplexing interface KD.duplex is identical to that of Mennink [21]. The duplexing interface is phased in the permute-squeeze-absorb fashion: first the underlying permutation p is applied on the state S. Then, it outputs an r-bit block $Z \in \{0,1\}^r$ off the internal state S. Finally, it absorbs

$P \in \{0,1\}^b$, where the flag *flag* $\in \{true, false\}$ indicates whether the outer r bits of P are XORed to the outer part of the state (if *flag* $=$ *false*) or if they overwrite the outer part of the state (if *flag* $=$ *true*).

Initialization Interface. The initialization interface is structurally different from that of Mennink [21], both in the actual inputs that it receives as in the way it processes these inputs. This is done to make it possible to rigorously define and study different initialization approaches. In detail, the initialization interface gets two inputs (apart from the implicit key array $\boldsymbol{K}$), a key index $\delta \in \{1, \ldots, \mu\}$ and an index $i \in \{1, \ldots, 2^l\}$ in such a way that (δ, i) is always unique. The value i may be a global counter, a counter per δ, or anything else, and this does not yet matter for the specification. Then, $\mathsf{KD.init}$ initializes the state as

$$S \leftarrow \mathsf{initL}(\boldsymbol{K}, \delta, i) \parallel \mathsf{initR}(\boldsymbol{K}, \delta, i)$$

for certain initialization functions

$$\begin{aligned} \mathsf{initL} &: (\{0,1\}^k)^\mu \times \{1, \ldots, \mu\} \times \{1, \ldots, 2^l\} \mapsto \{0,1\}^k \,, \\ \mathsf{initR} &: (\{0,1\}^k)^\mu \times \{1, \ldots, \mu\} \times \{1, \ldots, 2^l\} \mapsto \{0,1\}^{b-k} \,. \end{aligned}$$

It outputs nothing.

Comparison With Original Duplex. We remark that this initialization is different from that of Mennink [21] in the fact that his initialization (just like that of [11, 17]) got as input the key index δ and an initialization vector $IV \in \{0,1\}^{b-k}$, and initialized the state as $S \leftarrow \boldsymbol{K}[\delta] \parallel IV$. It can be seen to be covered by restricting the domain of i to $\{1, \ldots, 2^{b-k}\}$ and by taking initialization functions

$$\mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \,, \tag{3a}$$

$$\mathsf{initR}(\boldsymbol{K}, \delta, i) = \mathrm{encode}_{b-k}[i] \,. \tag{3b}$$

We will refer to this case as the *baseline case.*

3.2 Security Model

Daemen et al. [11] described the ideal extendable input function (IXIF) as ideal equivalent for the keyed duplex. We will also consider this function, but adapted to the superficial changes implemented by Mennink [21] and to our generalized initialization of Section 3.1. The function is described in Algorithm 2.

The IXIF has the same interface as the keyed duplex. However, it is not based on a key array $\boldsymbol{K}$ and primitive p, but rather on a random oracle $\mathsf{ro} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$ 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 [4]. For $P \in \{0,1\}^*$,

$\mathsf{ro}(P, r)$ outputs the first r bits of $\mathsf{ro}_\infty(P)$. The IXIF additional maintains a path *path*. In this path, it stores all data input by the user. It is initialized by $\mathrm{encode}_{\log_2 \mu}[\delta] \parallel \mathrm{encode}_l[i]$, 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 $\mathsf{IXIF}[\mathsf{ro}]$

```
Interface: IXIF.init
Input: $(\delta, i) \in \{1, \ldots, \mu\} \times \{1, \ldots, 2^l\}$
Output: $\varnothing$
  $path \leftarrow \mathrm{encode}_{\log_2 \mu}[\delta] \parallel \mathrm{encode}_l[i]$
  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 $Z$
```

The security of the duplex construction is defined as the distance between KD and IXIF. More formally, let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random transformation, $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ a random array of keys, and ro be a random oracle. One considers a distinguisher D 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:

$$\mathbf{Adv}_{\mathsf{KD}}(\mathsf{D}) = \Delta_{\mathsf{D}}\left(\mathsf{KD}[\mathsf{p}]_{\boldsymbol{K}}, \mathsf{p}, \mathsf{p}^{-1} \; ; \; \mathsf{IXIF}[\mathsf{ro}], \mathsf{p}, \mathsf{p}^{-1}\right) . \tag{4}$$

Without loss of generality, the distinguisher always makes at least one duplexing call after each initialization call.

3.3 Parameterization of Distinguishers

The three main measures to quantify the resources of D are the number of queries it can make to its oracles:

- Q: the number of distinct initialization queries;
- M: the number of distinct duplexing queries;
- N: the number of distinct primitive queries.

The number of initialization calls are further refined in the maximum number of calls per δ and per i, and per (δ, i):

- Q_δ: the maximum number of initialization queries for a single δ;
- Q_i: the maximum number of initialization calls for a single i;

– $Q_{\delta,i}$: the maximum number of initialization calls for a single δ, i.

We remark that earlier works [11, 17] used Q_{IV} as the maximum number of initialization calls for a single IV (or more broadly seen, for a single inner part). This notation is deprecated, as we will consider different inner parts that depend only on δ, on i, on (δ, i), or on neither of them.

Finally, different evaluations of the duplex can be the same up to a common prefix, and common subpaths can actually benefit the distinguisher. To measure the degree in which it could help the distinguisher, we will define a path *path* that keeps track of 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 $(\delta, i) \mapsto \varnothing$, the associated path is defined as $\mathit{path} = \mathrm{encode}_{\log_2 \mu}[\delta] \parallel \mathrm{encode}_l[i]$. For each duplexing call $(\mathit{flag}, P) \mapsto Z$, the value $[\mathit{flag}] \cdot (Z \| 0^{b-r}) \oplus P$ is appended to the path of the previous construction query. 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. Using this terminology, we define additional measures:

– 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 $\mathit{flag} = \mathit{true}$.

3.4 Security of Baseline Case

As the current duplex construction of Section 3.1 has a more general initialization than that of Daemen et al. [11], we cannot translate their security result to the current duplex construction. Nevertheless, the duplex construction of Daemen et al. is in fact the baseline case, i.e., our duplex construction but with the initialization functions of (3). For this baseline case, we can thus carry over the result of Daemen et al. [11]:

Theorem 1 (security of duplex construction [11]). *Let $b, c, r, k, l, \mu \in \mathbb{N}$, 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 quantified as in Section 3.3 and with $M + N \leq 0.1 \cdot 2^c$,*

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

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

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

3.5 Role of Initialization Vector

The bound of Theorem 1 is rather involved, but the part that matters for our analysis is only (5c). In detail, these three terms are related to key guessing or key hitting problems that may occur in a security game and that involve the initialization of the duplex. These three terms correspond to three bad events in the analysis of Daemen et al., namely (in this order) [11, (22)], [11, (20)], and [11, (23)]. We will restate those (and exactly those) bad events, but updated[4] for the initialization interface of our duplex of Section 3.1:

[11, (22)] There exists an initialization call (δ, i) to $\mathsf{KD.init}$ and a duplex evaluation (s, t) of p within $\mathsf{KD.duplex}$ such that $\mathsf{initL}(\boldsymbol{K}, \delta, i) \parallel \mathsf{initR}(\boldsymbol{K}, \delta, i) = s \oplus 0^r \| \kappa$, where κ is a random c-bit dummy key;

[11, (20)] There exists an initialization call (δ, i) to $\mathsf{KD.init}$ and a primitive evaluation (x, y) of p such that $\mathsf{initL}(\boldsymbol{K}, \delta, i) \parallel \mathsf{initR}(\boldsymbol{K}, \delta, i) = x$;

[11, (23)] There exist two distinct initialization calls $(\delta, i), (\delta', i')$ to $\mathsf{KD.init}$ such that $\mathsf{initL}(\boldsymbol{K}, \delta, i) \parallel \mathsf{initR}(\boldsymbol{K}, \delta, i) = \mathsf{initL}(\boldsymbol{K}, \delta', i') \parallel \mathsf{initR}(\boldsymbol{K}, \delta', i')$.

We recall from Section 3.3 that the distinguisher makes at most Q initialization calls, where at most Q_i are made for a single i, at most $M - Q$ duplex calls, and at most N primitive calls. Using this, we can easily obtain that **[11, (20)]** happens with probability at most $\frac{Q_i N}{2^k}$ and **[11, (23)]** with probability at most $\frac{\binom{\mu}{2}}{2^k}$. Event **[11, (22)]**, finally, occurs with probability at most $\frac{Q(M-L-Q)}{2^{\min\{c+k,b\}}}$, where the presence of $-L$ in the numerator is for technical reasons that are irrelevant for the remainder of the work.

4 Improvements Under Specific *IV* Generation

We will discuss improvements of the security bound of Theorem 1, and in particular the terms in equation (5c), in case of various types of IV conventions, as outlined in Table 1. Here, RIV is an independently drawn random IV each evaluation and RIV_δ is a random IV independently drawn per user.

4.1 Global *IV*

A global IV, in this context, means that different users never employ the same IV. In Table 1, this is formally defined by using an encoding function for initR that encodes *both* the $\delta \in \{1, \ldots, \mu\}$ and the $i \in \{1, \ldots, 2^l\}$, where we require that $\log_2 \mu + l \leq b - k$. As KD is never initialized twice for the same (δ, i), the rightmost $b - k$ bits of the initialization will always be distinct.

This use case is in fact the easiest one to consider, as it allows to derive quite strong improved bounds on the relevant bad events of [11] as outlined in Section 3.5:

[4] The update is fairly straightforward, merely replacing $\boldsymbol{K}[\delta] \parallel IV$ with $\mathsf{initL}(\boldsymbol{K}, \delta, i) \parallel \mathsf{initR}(\boldsymbol{K}, \delta, i)$.

Table 1: Different types of IV generation cases considered in this work, and a reference to their security analyses. Here, RIV stands for "random IV" each evaluation, and RIV_δ is a unique random IV per user (independently distributed for each user). They are of length n bits, and 0-padded.

case	$\mathsf{initL}(\boldsymbol{K}, \delta, i)$	$\mathsf{initR}(\boldsymbol{K}, \delta, i)$	restriction	reference
baseline	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[i]$	$l \leq b-k$	Section 3.4
global IV	$\boldsymbol{K}[\delta]$	$\mathrm{encode}_{b-k}[(\delta, i)]$	$\log_2 \mu + l \leq b-k$	Section 4.1
random IV	$\boldsymbol{K}[\delta]$	$RIV \| 0^{b-k-n}$	—	Section 4.2
quasi-random IV	$\boldsymbol{K}[\delta]$	$(RIV_\delta \oplus \mathrm{encode}_n[i]) \| 0^{b-k-n}$	i counter, $l \leq n$	Section 4.3
IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	0^{b-k}	$l \leq k$	Section 4.4
global IV on key	$\boldsymbol{K}[\delta] \oplus \mathrm{encode}_k[i]$	$\mathrm{encode}_{b-k}[\delta]$	$l \leq k$, $\log_2 \mu \leq b-k$	Section 4.5

ad [**11, (22)**] The analysis of this event remains mostly unchanged. The reason is that the dummy key κ "blinds" the rightmost c bits of $\mathsf{initR}(\boldsymbol{K}, \delta, i) = \mathrm{encode}_{b-k}[(\delta, i)]$ anyway. A small improvement may be possible in case the blinding is incomplete, i.e., if $|\mathsf{initR}(\boldsymbol{K}, \delta, i)| = b-k > c$, but the improvement is negligible as, even though $\mathsf{initR}(\boldsymbol{K}, \delta, i)$ is distinct for each input (δ, i), they may collide on their left $b-k-c$ bits and we cannot rely on their uniqueness;

ad [**11, (20)**] Consider any primitive evaluation (x, y). In case of a global IV, $\mathsf{initR}(\boldsymbol{K}, \delta, i) = \mathrm{encode}_{b-k}[(\delta, i)]$ is distinct for each input (δ, i). This means that there is only 1 initialization call (δ, i) that satisfies $\mathrm{right}_{b-k}(x) = \mathsf{initR}(\boldsymbol{K}, \delta, i)$. For this initialization call, we have $\mathrm{left}_k(x) = \mathsf{initL}(\boldsymbol{K}, \delta, i)$ with probability $1/2^k$. (For inverse queries, the probability is strictly smaller as also $\mathrm{right}_{b-k}(x) = \mathsf{initR}(\boldsymbol{K}, \delta, i)$ needs to hold, and it holds only with a probability at most 1.) Summing over all initialization calls, we obtain that this bad event occurs with probability at most $N/2^k$;

ad [**11, (23)**] In the case of a global IV, we will have $\mathsf{initR}(\boldsymbol{K}, \delta, i) \neq \mathsf{initR}(\boldsymbol{K}, \delta', i')$ for any two distinct initialization calls, and hence this bad event occurs with probability 0.

In the case of a global IV, we can thus replace (5c) of Theorem 1 by:

$$\frac{Q(M-L-Q)}{2^{\min\{c+k,b\}}} + \frac{N}{2^k} \,. \tag{6}$$

4.2 Random IV

In the case of random IV generation, we will consider a setting where each user will always select the right part of the state, $\mathsf{initR}(\boldsymbol{K}, \delta, i)$ uniformly randomly from $\{0,1\}^n$ with $n \leq b-k$, denoted as RIV in Table 1.

The analysis is a bit different to that of Section 4.1. To wit, for the case of a global IV, event [**11, (20)**] was very similar to the original analysis of [11], and for event [**11, (23)**] it sufficed to observe that [11] made the assumption that the IV is always chosen in favor of the attacker. Now, in the case of random IVs, the situation changes in that (i) IVs may repeat, and (ii) there is no clear bound on the maximum occurrence of a single IV. The latter issue is particularly problematic as we cannot claim a clear upper bound on the maximum number of initialization calls for a single inner part (formally known as Q_{IV} (cf., Section 3.3)). Instead, we will have to employ a probabilistic argument using the mulf (Definition 1). In detail, we will obtain the following improved bounds on the three relevant bad events of [11] as outlined in Section 3.5:

ad [11, (22)] This event remains mostly unchanged, for the same reason as in Section 4.1;

ad [11, (20)] Consider any primitive evaluation (x, y). As the RIVs are chosen uniformly at random, we can use the mulf $\nu_{n,k}^{Q}$ on the maximum multicollision on RIV (note that the RIV is of size $n \leq b-k$ bits). More detailed, assume the highest occurrence of an inner part of RIV is ν, then there are at most ν initialization calls (δ, i) that satisfy $\text{right}_{b-k}(x) = \mathsf{initR}(\boldsymbol{K}, \delta, i)$. For those specific initialization calls, we have $\text{left}_k(x) = \mathsf{initL}(\boldsymbol{K}, \delta, i)$ with probability $\nu/2^k$. (As before, for inverse queries, the probability is strictly smaller.) Summing over all initialization calls, we obtain that this bad event occurs with probability at most $\nu N/2^k$. We then select the term ν so that the sum of this term and the probability of a ν-collision is small, and this minimum is achieved for the mulf $\nu_{n,k}^{Q}$:

$$\min_{\nu} \frac{\nu N}{2^k} + \mathbf{Pr}\left(\nu > \nu_{n,k}^{Q}\right) \leq \frac{\nu_{n,k}^{Q} \cdot (N+1)}{2^k}\,;$$

ad [11, (23)] In the original proof, the authors simply assumed that the IV was always chosen in favor of the attacker, and the bad event simplified to a simple key collision in $\mathsf{initL}(\boldsymbol{K}, \delta, i)$. Now, not only the k-bit keys $\mathsf{initL}(\boldsymbol{K}, \delta, i) = K[\delta]$ and $\mathsf{initL}(\boldsymbol{K}, \delta', i') = K[\delta']$ are randomly distributed but also the n-bit IVs $\mathsf{initR}(\boldsymbol{K}, \delta, i) = RIV \| 0^{b-k-n}$ and $\mathsf{initR}(\boldsymbol{K}, \delta', i') = RIV' \| 0^{b-k-n}$ (here, the accent is put in RIV' to make the distinction clear). This leads to a more complex analysis of **[11, (23)]**, and we have to distinguish depending on whether $\delta = \delta'$ or not:

- $\delta = \delta'$: in this case, $K[\delta] = K[\delta']$ by default, but $RIV = RIV'$ with probability $1/2^n$;
- $\delta \neq \delta'$: in this case $K[\delta] \| RIV = K[\delta'] \| RIV'$ with probability at most $1/2^{k+n}$.[5]

There are at most $\mu\binom{Q_\delta}{2}$ tuples $\{(\delta, i), (\delta', i')\}$ of the former category and at most $\binom{Q}{2}$ tuples of the second category. The bad event is thus set with probability at most $\mu\binom{Q_\delta}{2}/2^n + \binom{Q}{2}/2^{k+n}$.

In the case of a random IV, we can thus replace (5c) of Theorem 1 by:

$$\frac{Q(M-L-Q)}{2^{\min\{c+k,b\}}} + \frac{\nu_{b-k,k}^{Q} \cdot (N+1)}{2^k} + \frac{\mu\binom{Q_\delta}{2}}{2^n} + \frac{\binom{Q}{2}}{2^{k+n}}\,. \tag{7}$$

[5] This could be improved by conditioning on which keys in $\boldsymbol{K}$ actually collide, but the gain in following this avenue is negligible as this is not the main term anyway.

4.3 Quasi-Random IV

The case of a quasi-random IV is a subtle combination of the global IV and a random IV. In detail, we consider a setting where each user δ uses a counter starting from a random offset RIV_δ. For example, in a simplified case of two users Alice and Bob, Alice will be assigned a random initialization vector $RIV_A \xleftarrow{\$} \{0,1\}^n$ and will use $\{RIV_A \oplus \mathrm{encode}_n[1], RIV_A \oplus \mathrm{encode}_n[2]), \dots\}$ padded with 0^{b-k-n}, whereas Bob will be assigned a random initialization vector $RIV_B \xleftarrow{\$} \{0,1\}^n$ and will use $\{RIV_B \oplus \mathrm{encode}_n[1], RIV_B \oplus \mathrm{encode}_n[2], \dots\}$ padded with 0^{b-k-n}.

The security analysis, for this case, becomes much more subtle. Indeed, the initialization vectors (RIV_A and RIV_B in above use case) are random, but the following IVs have no randomness *given* the initial ones. To resolve the issue, we need to define a variant of the mulf of Definition 1, namely one that considers collisions in *sets*.

Definition 2 (sequence multicollision limit function (smulf)). *Let Q_i, Q_δ, c, $r \in \mathbb{N}$. Consider the experiment of throwing Q_i balls uniformly at random in 2^r bins and for each of the randomly thrown balls, throwing a ball in the Q_δ subsequent bins, and let ν be the maximum number of balls in a single bin. We define the sequence multicollision limit function (smulf) $\bar{\nu}_{r,c}^{Q_i,Q_\delta}$ as the smallest natural number x that satisfies*

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

Intuitively, the smulf on parameters Q_i, Q_δ is at most the mulf on parameter Q_iQ_δ. As a matter of fact, it can be argued that the value x targeted in Definition 2 also satisfies (2), but with adjusted λ. The reason is almost identical to that of [11, Section 6.5]. In detail, they observe that $\mathbf{Pr}\left(\nu > x\right) \le 2^r \mathbf{Pr}\left(X > x\right)$, where $\mathbf{Pr}\left(X > x\right)$ is the probability that any particular bin has more than x balls, then they observe that the number of balls in a particular bin is binomially distributed with $n = Q$ trials and success probability $p = 1/2^r$, and finally they observe that for sufficiently large parameters this is upper bounded by a Poisson distribution with mean $\lambda = np = Q/2^r$. In the case of the smulf of Definition 2, the same story applies with the difference that the number of balls in a particular bin is now binomially distributed with $n = Q_i$ trials and success probability $p = Q_\delta/2^r$, which can be upper bounded by a Poisson distribution with mean $\lambda = np = Q_iQ_\delta/2^r$.

Using the definition of the smulf, can derive the following improved bounds on the three relevant bad events of [11] as outlined in Section 3.5:

ad [11, (22)] This event remains mostly unchanged, for the same reason as in Section 4.1;

ad [11, (20)] The first part of the analysis is identical to that of Section 4.2. Consider any primitive evaluation (x, y). Denoting the highest occurrence of an inner part of $RIV_\delta \oplus \mathrm{encode}_n[i]$ by ν, then there are at most ν initialization calls (δ, i) that satisfy $\mathrm{right}_{b-k}(x) = \mathsf{initR}(\boldsymbol{K}, \delta, i)$, and the bad event occurs with probability at most $\nu N/2^k$. We then select the term ν so that the sum of this term and the probability of a ν-collision is small, and this minimum is achieved for the smulf $\bar{\nu}_{n,k}^{Q_i,Q_\delta}$ on sets defined by RIV_δ for $\delta \in \{1, \ldots, \mu\}$:

$$\min_\nu \frac{\nu N}{2^k} + \mathbf{Pr}\left(\nu > \bar{\nu}_{n,k}^{Q_i,Q_\delta}\right) \leq \frac{\bar{\nu}_{n,k}^{Q_i,Q_\delta} \cdot (N+1)}{2^k}\,;$$

ad [11, (23)] The analysis of Section 4.2 mostly carries over, a difficulty occurs in that we have to investigate the probability that two sets (with specific distributions) collide or not. Note that the k-bit keys $\mathsf{initL}(\boldsymbol{K}, \delta, i) = K[\delta]$ and $\mathsf{initL}(\boldsymbol{K}, \delta', i') = K[\delta']$ are randomly distributed but the n-bit IVs satisfy $\mathsf{initR}(\boldsymbol{K}, \delta, i) = (RIV_\delta \oplus \text{encode}_n[i]) \| 0^{b-k-n}$ and $\mathsf{initR}(\boldsymbol{K}, \delta', i') = (RIV_{\delta'} \oplus \text{encode}_n[i']) \| 0^{b-k-n}$ for counters $i, i' \leq Q_\delta$. We again distinguish depending on whether $\delta = \delta'$ or not:

- $\delta = \delta'$: in this case, $K[\delta] = K[\delta']$ by default, but $RIV_\delta = RIV_{\delta'}$ and $\text{encode}_n[i] \neq \text{encode}_n[i']$, so the initializations collide with probability 0;
- $\delta \neq \delta'$: in this case $K[\delta] = K[\delta']$ with probability $1/2^k$. For $RIV_\delta \oplus \text{encode}_n[i] = RIV_{\delta'} \oplus \text{encode}_n[i']$, we can observe that any pair $(\delta, i), (\delta', i')$ fixes $\text{encode}_n[i]$ and $\text{encode}_n[i']$ and thus collides with probability $1/2^n$. We can tighten this by only focusing on δ and δ'. Per δ, there are at most Q_δ consecutive values i. Thus, the sets

$$\{RIV_\delta \oplus \text{encode}_n[1], RIV_\delta \oplus \text{encode}_n[2], \ldots\}\,,$$
$$\{RIV_{\delta'} \oplus \text{encode}_n[1], RIV_{\delta'} \oplus \text{encode}_n[2], \ldots\}$$

overlap with probability at most $(2Q_\delta - 1)/2^n$.

There are at most $\binom{\mu}{2}$ different choices $\{\delta, \delta'\}$ for bounding the second category. The bad event thus occurs with probability at most $\binom{\mu}{2}(2Q_\delta - 1)/2^{k+n}$.

In the case of a quasi-random IV, we can thus replace (5c) of Theorem 1 by:

$$\frac{Q(M-L-Q)}{2^{\min\{c+k,b\}}} + \frac{\bar{\nu}_{n,k}^{Q_i,Q_\delta} \cdot (N+1)}{2^k} + \frac{\binom{\mu}{2}(2Q_\delta - 1)}{2^{k+n}}\,. \tag{8}$$

4.4 *IV* on Key

The case of a IV added to the key is structurally different from previous ones. In detail, we will consider a setting where an IV is added to a user's key $\boldsymbol{K}[\delta]$: $\mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$. The right part of the initialization stays blank.

The analysis is a bit different to the previous ones, but not necessarily harder, simply as it leads to a similar query trade-off that can already be observed in the Even-Mansour construction [19]. We will see that event **[11, (20)]** introduces a multiplicative term between initialization and primitive queries. Event **[11, (23)]** corresponds to two different initializations for which $\boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ and $\boldsymbol{K}[\delta'] \oplus \text{encode}_k[i']$ collide (looking ahead, in the case of Section 4.5 the latter case is avoided by encoding the user index in the right part of the initial state). In detail, we will obtain the following bounds on the three relevant bad events of [11] as outlined in Section 3.5:

ad [11, (22)] This event remains mostly unchanged, for the same reason as in Section 4.1;

ad [11, (20)] Consider any primitive evaluation (x, y), without loss of generality satisfying $\text{right}_{b-k}(x) = 0^{b-k}$. For any initialization query, the probability that $\text{left}_k(x) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ is the same as the probability that $\text{left}_k(x) \oplus \text{encode}_k[i] = \boldsymbol{K}[\delta]$, which entirely depends on the randomness of the key and is $1/2^k$. (As before, for inverse queries, the probability is strictly smaller.) Summing over all primitive queries and all initialization queries, the bad event is set with probability at most $QN/2^k$;

ad [11, (23)] For any two initialization calls, the $(b-k)$-bit outer parts $\mathsf{initR}(\boldsymbol{K}, \delta, i)$ always equal 0^{b-k}. For any two initialization queries, we have to consider collisions between $\mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ and $\mathsf{initL}(\boldsymbol{K}, \delta', i') = \boldsymbol{K}[\delta'] \oplus \text{encode}_k[i']$. Again, the probability that $\boldsymbol{K}[\delta] \oplus \text{encode}_k[i] = \boldsymbol{K}[\delta'] \oplus \text{encode}_k[i']$ is the same to the probability that $\boldsymbol{K}[\delta] \oplus \boldsymbol{K}[\delta'] = \text{encode}_k[i] \oplus \text{encode}_k[i']$. This probability $1/2^k$ due to $\boldsymbol{K}[\delta]$ and $\boldsymbol{K}[\delta']$ being randomly chosen. The bad event is thus set with probability at most $\binom{Q}{2}/2^k$.

In the case of a IV on the key, we can thus replace (5c) of Theorem 1 by:

$$\frac{Q(M - L - Q)}{2^{\min\{c+k, b\}}} + \frac{QN}{2^k} + \frac{\binom{Q}{2}}{2^k}\,. \tag{9}$$

4.5 Global *IV* on Key

In this section, we extend the case of Section 4.4 to a global IV on the key, where "global" refers to the fact that the inner part of the initialization is an encoding of the user: $\mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ as before but $\mathsf{initR}(\boldsymbol{K}, \delta, i) = \text{encode}_{b-k}[\delta]$.

By encoding the user index into the outer part, key collisions among different users do not matter anymore for the security, only colliding IV's for a fixed user. This affects the analysis of both **[11, (20)]** and **[11, (23)]**. In detail, we will obtain the following improved bounds on the three relevant bad events of [11] as outlined in Section 3.5:

ad [11, (22)] This event remains mostly unchanged, for the same reason as in Section 4.1;

ad [11, (20)] Consider any primitive evaluation (x, y). Let δ be such that $\text{right}_{b-k}(x) = \text{encode}_{b-k}[\delta]$. By assumption, there are at most Q_δ initialization queries for this particular δ. For any of those queries, the probability that $\text{left}_k(x) = \mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ is $1/2^k$. (As before, for inverse queries, the probability is strictly smaller.) Summing over all primitive queries and all Q_δ initialization queries, the bad event is set with probability at most $Q_\delta N/2^k$;

ad [11, (23)] Clearly, if $\delta \neq \delta'$, then $\mathsf{initR}(\boldsymbol{K}, \delta, i) \neq \mathsf{initR}(\boldsymbol{K}, \delta', i')$ and the bad event cannot be set. On the other hand, if $\delta = \delta'$, the right part of the initial states are equal, and the left parts $\mathsf{initL}(\boldsymbol{K}, \delta, i) = \boldsymbol{K}[\delta] \oplus \text{encode}_k[i]$ and $\mathsf{initL}(\boldsymbol{K}, \delta', i') = \boldsymbol{K}[\delta'] \oplus \text{encode}_k[i']$ collide with probability $1/2^k$. There are at most $\mu\binom{Q_\delta}{2}$ tuples $\{(\delta, i), (\delta', i')\}$ such that $\delta = \delta'$. Summing over all these queries, the bad event is set with probability at most $\mu\binom{Q_\delta}{2}/2^k$.

In the case of a global IV on the key, we can thus replace (5c) of Theorem 1 by:

$$\frac{Q(M-L-Q)}{2^{\min\{c+k,b\}}} + \frac{Q_\delta N}{2^k} + \frac{\mu\binom{Q_\delta}{2}}{2^k} \,. \tag{10}$$

5 Stream Encryption

We will consider one of the most elementary use case of the duplex, namely (sequential) stream encryption, following Mennink [21, Section 7]. The construction and its security model are outlined in Section 5.1, and we discuss its security under different types of initialization in Section 5.2.

5.1 Construction and Security Model

Consider the stream cipher $\mathsf{SC} : \{0,1\}^k \times \{1,\ldots,2^l\} \times \mathbb{N} \to \{0,1\}^*$, that gets as input a k-bit key K, an index value $i \in \{1,\ldots,2^l\}$, and a requested output length ℓ, and that outputs a key stream S of length ℓ bits. It is defined using the duplex as follows:

– Initialize the keyed duplex of Algorithm 1 with permutation p and key array $\boldsymbol{K} = (K)$;
– Evaluate $\mathsf{KD.init}(1, i)$;
– Evaluate $\mathsf{KD.duplex}(\mathit{false}, 0^b)$ for exactly $\lceil \ell/r \rceil$ times, concatenate their outputs, and truncate this string to ℓ bits to obtain S.

The scheme is depicted in the multi-user setting in Figure 2. We note that this is a very natural way of duplex-based stream generation; a variant of it (with a significantly more involved initialization to suit side-channel resilience) can be observed in ISAP v2 [12–14] and in Asakey [18].

We will consider its security as indistinguishability from a random function in the multi-user setting. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation. Let $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ be a random array of keys and $(\$_j)_{j=1}^{\mu}$ be functions that for each input $i \in \{1,\ldots,2^l\}$ define a random string of infinite length and on input of a tuple (i,ℓ) return the first ℓ bits of the string related to input i. Let case $\in$ $\{$baseline, global, random, quasirandom, onkey, globalonkey$\}$ describe the type of initialization, corresponding to the six cases outlined in Table 1.

We define the multi-user security of SC under initialization type case as

$$\mathbf{Adv}_{\mathsf{SC}}^{\mu\text{-prf-case}} = \Delta_{\mathsf{D}}\left((\mathsf{SC}[\mathsf{p}]_{\boldsymbol{K}_j})_{j=1}^{\mu}, \mathsf{p}^{\pm} \;;\; (\$_j)_{j=1}^{\mu}, \mathsf{p}^{\pm}\right) \,. \tag{11}$$

5.2 Security Under Different Initializations

We will consider a distinguisher that can make Q initialization queries (i.e., Q queries to its construction oracle), M duplexing queries (i.e., the Q queries are of total length M duplexing calls), and N primitive queries, in accordance with

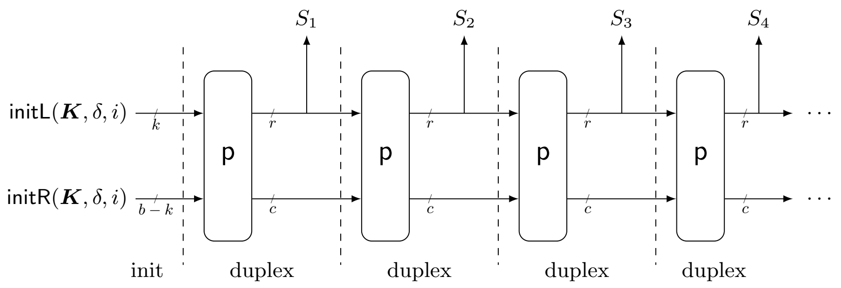

Fig. 2: Stream cipher SC in the multi-user setting. The function gets as input a key array $\boldsymbol{K}$, key index δ, and index i. It outputs keystream blocks $(S_1, S_2, \ldots)$. The actual number of output blocks is determined by an additional input parameter ℓ. The sole difference with the sequential keystream generation construction of Mennink [21] is in the initialization.

Section 3.3. For the refined values of Q, we have that $Q_\delta \leq \min\{2^l, Q\}$, $Q_i \leq \mu$, and $Q_{\delta,i} = 1$. Finally, just like in [21, Section 7], all queries start with a new i (so $L = 0$) and all duplexing calls are for *flag* = *false* (so $\Omega = 0$).

We obtain the following general bound over all cases:

$$\mathbf{Adv}_{\mathsf{SC}}^{\mu\text{-prf-case}}(\mathsf{D}) \leq \frac{2\nu_{r,c}^{2M}(N+1)}{2^c} + \frac{(M-Q)Q}{2^b - Q} + \frac{M(M-1)}{2^b} \tag{12a}$$

$$+ \frac{Q(M-Q)}{2^{\min\{c+k,b\}}} + \Xi^{\text{case}}\,. \tag{12b}$$

The first part (12a) is the same for all different initializations and corresponds to (5a) and (5b). Part (12b) corresponds to (5c), which is actually improved for the specific cases:

$$\Xi^{\text{case}} = \begin{cases} \frac{\mu N}{2^k} + \frac{\binom{\mu}{2}}{2^k} & (\text{case} = \text{baseline})\,, \\ \frac{N}{2^k} & (\text{case} = \text{global})\,, \\ \frac{\nu_{n,k}^{Q}\cdot(N+1)}{2^k} + \frac{\mu\binom{\min\{2^l,Q\}}{2}}{2^n} + \frac{\binom{Q}{2}}{2^{k+n}} & (\text{case} = \text{random})\,, \\ \frac{\bar{\nu}_{n,k}^{\mu,\min\{2^l,Q\}}\cdot(N+1)}{2^k} + \frac{\binom{\mu}{2}(2\min\{2^l,Q\}-1)}{2^{k+n}} & (\text{case} = \text{quasirandom})\,, \\ \frac{QN}{2^k} + \frac{\binom{Q}{2}}{2^k} & (\text{case} = \text{onkey})\,, \\ \frac{\min\{2^l,Q\}N}{2^k} + \frac{\mu\binom{\min\{2^l,Q\}}{2}}{2^k} & (\text{case} = \text{globalonkey})\,, \end{cases} \tag{13}$$

which are based on (5c), (6), (7), (8), (9), and (10), respectively.

6 Authenticated Encryption

The main raison d'être of the duplex construction is authenticated encryption. We will consider the MonkeySpongeWrap construction as described by Mennink [21, Section 9], which generalizes the original SpongeWrap construction [8]. However, we do so including our generalized initialization. The construction and its security model are outlined in Section 6.1, and we discuss its security under different types of initialization in Section 6.2.

6.1 Construction and Security Model

Consider the authenticated encryption scheme $\mathsf{AE} : \{0,1\}^k \times \{1,\dots,2^l\} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^t$, that gets as input a k-bit key K, an index value $i \in \{1,\dots,2^l\}$, arbitrary length associated data A, and arbitrary length message P, and that outputs a ciphertext C of size $|P|$ bits and a tag T of size t bits. It is defined using the duplex as follows:

- Initialize the keyed duplex of Algorithm 1 with permutation p and key array $\boldsymbol{K} = (K)$;
- Evaluate $\mathsf{KD.init}(1, i)$;
- Perform 10^*-padding on A to obtain r-bit associated data blocks $(A_1, \dots, A_v)$, and for each block, evaluate $\mathsf{KD.duplex}(\mathit{false}, A_i \| 0 \| 0^{c-1})$ and discard the output;
- Perform 10^*-padding on M to obtain r-bit plaintext blocks $(P_1, \dots, P_w)$, and, for each block, evaluate $\mathsf{KD.duplex}(\mathit{false}, P_i \| 1 \| 0^{c-1})$ and XOR the output with P_i to obtain C_i;
- Evaluate $\mathsf{KD.duplex}(\mathit{false}, 0^b)$ for exactly $\lceil t/r \rceil$ times, concatenate their outputs, and truncate this string to t bits to obtain T.

The scheme is depicted in the multi-user setting in Figure 2. This variant of authenticated encryption can be observed in various NIST candidates, most notably Xoodyak [9, 10] and Gimli [6]. We refer to [21, Section 9] for a more detailed algorithmic description as well as a discussion of the inverse AE^{-1}.

We will consider its security as indistinguishability from ideal in the multi-user setting. Here, we consider the ideal setting as the scheme that upon encryption, always outputs random strings, and upon decryption always outputs the failure symbol $\perp$, assuming that the distinguisher never relays an encryption output to the decryption oracle. Let $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$ be a random permutation. Let $\boldsymbol{K} \xleftarrow{\$} (\{0,1\}^k)^\mu$ be a random array of keys and $(\$_j)_{j=1}^{\mu}$ be functions that for each input $i \in \{1,\dots,2^l\}$ define a random string of infinite length and on input of a tuple (i, A, P) return the first $|P| + t$ bits of the string related to input i. Let $\perp$ be the function that always returns the failure symbol $\perp$. Let case $\in$ {baseline, global, random, quasirandom, onkey, globalonkey} describe the type of initialization, corresponding to the six cases outlined in Table 1.

We define the multi-user security of AE under initialization type case as

$$\mathbf{Adv}_{\mathsf{AE}}^{\mu\text{-ae-case}} = \Delta_{\mathsf{D}} \left((\mathsf{AE}[\mathsf{p}]_{\boldsymbol{K}_j}, \mathsf{AE}[\mathsf{p}]_{\boldsymbol{K}_j}^{-1})_{j=1}^{\mu}, \mathsf{p}^{\pm} \; ; \; (\$_j, \perp)_{j=1}^{\mu}, \mathsf{p}^{\pm} \right) . \qquad (14)$$

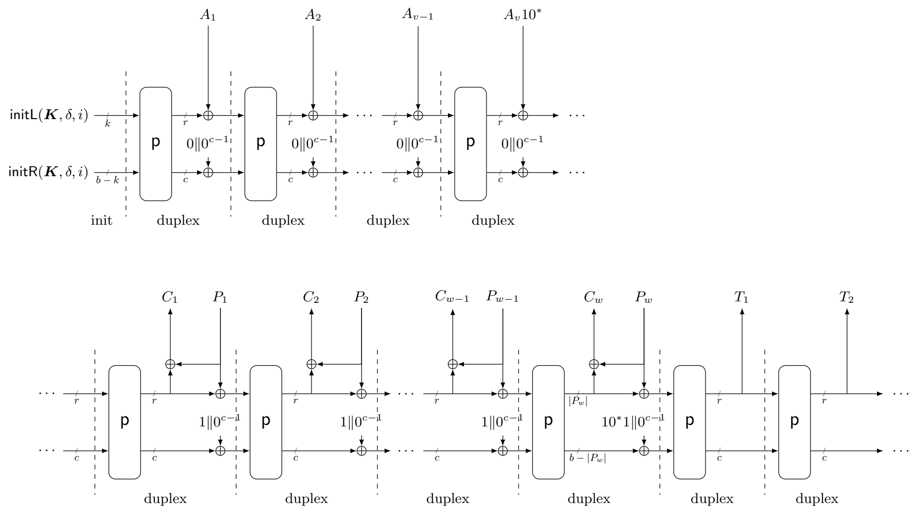

Fig. 3: Authenticated encryption scheme AE in the multi-user setting. The function gets as input a key array $\boldsymbol{K}$, key index δ, index i, associated data blocks $(A_1, A_2, \ldots, A_v)$, and plaintext blocks $(P_1, P_2, \ldots, P_w)$ (the last blocks of associated data and plaintext may be partial). It outputs a ciphertext $C = (C_1, C_2, \ldots, C_w)$ of size $|P|$ bits and tag blocks $(T_1, T_2, \ldots)$ truncated to t bits (the last blocks of ciphertext and tag may be partial). The sole difference with the MonkeySpongeWrap construction of Mennink [21] is in the initialization.

Distinguisher D is not allowed to repeat an index for encryption queries but it may do so for decryption queries. In the (quasi-)random IV case we assume that the oracle maintains a table to re-use earlier RIV or RIV_δ in case of repeated indices. It is not allowed to relay an encryption output to the decryption oracle.

6.2 Security Under Different Initializations

We will consider a distinguisher that can make Q initialization queries (i.e., Q queries to its construction oracle), split into Q_e encryption and Q_d decryption queries, M duplexing queries (i.e., the Q queries are of total length M duplexing calls), again split into M_e encryption and M_d decryption queries, and N primitive queries, in accordance with Section 3.3. For the refined values of Q, we have that $Q_\delta \leq \min\{2^l, Q\}$, $Q_i \leq \mu$, and $Q_{\delta,i} = 1$. Finally, just like in [21, Section 9], all encryption queries start with a new i, but decryption queries may repeat i (so $L \leq Q_d$) and all duplexing encryption calls are for $\mathit{flag} = \mathit{false}$ but duplexing decryption calls may be for $\mathit{flag} = \mathit{true}$ (so $\Omega \leq M_d - 2Q_d$).

We obtain the following general bound over all cases:

$$\mathbf{Adv}_{\mathsf{AE}}^{\mu\text{-ae-case}}(\mathsf{D}) \le \frac{2\nu_{r,c}^{2M}(N+1)}{2^c} + \frac{(M-Q)Q}{2^b - Q} + \frac{M(M-1)}{2^b} \tag{15a}$$

$$+ \frac{M_d N + \binom{M_d}{2}}{2^c} + \frac{Q_d}{2^t} \tag{15b}$$

$$+ \frac{Q(M-Q)}{2^{\min\{c+k,b\}}} + \varXi^{\text{case}} . \tag{15c}$$

The first part (15a) is identical to what we saw for stream encryption in (12a) as derived from (5a) and (5b). The second part (15b) has an additional fraction coming from (5a) due to the fact that now $L + \varOmega$ may be as high as $M_d - Q_d$, and an additional term $Q_d/2^t$ corresponding to random tag guesses (refer to [21, Theorem 7]). The third part (15c) corresponds to (5c) and is identical to (12b). As such, the term $\varXi^{\text{case}}$ is actually the same as for encryption, i.e., as in (13).

7 Practical Implications

Given the improved bounds of Section 4 and its generic application to duplex-based stream encryption in Section 5 and authenticated encryption in Section 6, we next discuss the practical implications (and limitations) of the different strategies to choose the IV/nonce. We will perform this discussion using a typical parameters set, namely $b = 320$, $r = 64$, $c = 256$, $k = 128$, and a 128-bit IV. This parameter set is analogue to the NIST Lightweight Cryptography winner Ascon-128 [15, 16].

Note that, in practice, we can assume that the entity that performs the encryption chooses the IV outside the influence of a potential adversary. However, for decryption, an adversary can potentially manipulate the transmitted IV.

7.1 Baseline

For the baseline version, we assume that an attacker can manipulate the IV anyway. In this case, we get as a bound following Section 5.2 (similar for authenticated encryption following Section 6.2):

$$\frac{2\nu_{r,c}^{2M}(N+1)}{2^{256}} + \frac{(M-Q)Q}{2^{320} - Q} + \frac{M(M-1)}{2^{320}} + \frac{Q(M-Q)}{2^{320}} + \varXi^{\text{baseline}} , \tag{16}$$

with

$$\varXi^{\text{baseline}} = \frac{\mu N}{2^{128}} + \frac{\binom{\mu}{2}}{2^{128}} .$$

For practical settings, $\varXi^{\text{baseline}}$ likely dominates the bound.

7.2 Global *IV*

If a global IV is used, the bound is independent of the number of users and we get (16) but instead with

$$\Xi^{\text{global}} = \frac{N}{2^{128}} .$$

However, in practice, the question is how to ensure the use of a global IV. Ensuring a global IV with just 128 bits seems to be unrealistic. One way would be to allocate a bit more space and separate the IV into a unique identifier per key and an actual nonce per transmission akin to [9]. Assuming the unique identifier is randomly chosen like the key during the key setup, it also does not have to be transmitted. To be sure that the identifier is unique, it is probably wise to use a 256-bit value. Combined with a 128-bit key and a 128-bit actual nonce, relying on a global IV seems to be a viable choice for permutations with $b \geq 512$ bits.

7.3 Random *IV*

Considering a random IV, we get (16) but instead with

$$\Xi^{\text{random}} = \frac{\nu_{128,128}^{Q} \cdot (N+1)}{2^{128}} + \frac{\mu\binom{\min\{2^l,Q\}}{2}}{2^{128}} + \frac{\binom{Q}{2}}{2^{256}} .$$

Here, we must ensure the decryption party has authentic access to the random IV chosen by the encrypting party. This is in practice often not the case for single-pass (authenticated) encryption schemes. In some two-pass schemes like encrypt-then-MAC [3, 20], the authenticity of the random IV can be verified before decryption starts as it is done for, e.g., ISAP v2 [12–14]. However, in this case the MAC verification cannot rely on the randomness of the IV for security reasons.

7.4 Quasi-Random *IV*

With the quasi-random IV we get (16) but instead with

$$\Xi^{\text{quasirandom}} = \frac{\bar{\nu}_{128,128}^{\mu,\min\{2^l,Q\}} \cdot (N+1)}{2^{128}} + \frac{\binom{\mu}{2}(2\min\{2^l,Q\}-1)}{2^{256}} .$$

In practice, such a scheme could be realized by setting up the random starting point of the IV during the setup of the keys and then counting upwards. In essence, this then works similarly to the method described in Section 7.2. However, it can be easily realized with smaller permutations.

7.5 *IV* on Key

With the IV on key scheme, we want to essentially picture what happens in TLS 1.3 for AES-GCM [23] which is analyzed in [5]. In principle, we have additional key material that masks the IV. When doing this in a duplex-based scheme, one can potentially profit more from having this additional key material compared to AES-GCM since it effectively extends the key. In our example, we would move from a 128-bit key and a 128-bit IV to a 128-bit IV on 256-bit key scheme, and we get (16) but instead with

$$\Xi^{\text{onkey}} = \frac{QN}{2^{256}} + \frac{\binom{Q}{2}}{2^{256}} .$$

One may wonder why to define the IV usage this way and not just concatenate a larger key with the IV. First, having an additional key to mask the IV can be agreed on protocol level without changing the underlying scheme. Second, for small permutations, like in our 320-bit example, there must be some overlap between a 256-bit key and a 128-bit IV.

7.6 Global *IV* on Key

This is the extension of Section 7.5, which in practice would require additional space to fit a unique partial IV. This would then lead to (16) but instead with

$$\Xi^{\text{globalonkey}} = \frac{\min\{2^l, Q\}N}{2^{256}} + \frac{\mu\binom{\min\{2^l,Q\}}{2}}{2^{256}} .$$

8 Conclusion

In this paper, we have shown that different ways to initialize the state of a duplex can lead to a significant improvement in the security bound considering multi-user setting. What is even more interesting is that many approaches we discuss, like masking the IV with an additional key, can be retrofitted on protocol level without changing the specification of the underlying algorithm. However, one still has to consider that the proofs are done in the random permutation model. Concretely, this means that for an actual instantiation of the duplex, assuming that it uses a permutation designed for 128-bit security, using the IV on key method (of Section 4.4) with a 256-bit key might not necessarily result in 256-bit security. Overall, care must be taken when instantiating a permutation-based cryptographic construction with any actual permutation.

Acknowledgements. We want to thank the authors of [11,17] 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. Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., Yasuda, K.: Photon-beetle authenticated encryption and hash family. Finalist of NIST lightweight cryptography standardization process (2021)
2. Beierle, C., Biryukov, A., dos Santos, L.C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q.: Lightweight AEAD and hashing using the sparkle permutation family. IACR Trans. Symmetric Cryptol. 2020(S1), 208–261 (2020), https://doi.org/10.13154/tosc.v2020.iS1.208-261
3. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000), https://doi.org/10.1007/3-540-44448-3_41
4. 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
5. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. 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. 247–276. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_10
6. Bernstein, D.J., Kölbl, S., Lucks, S., Maat Costa Massolino, P., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli. Second Round Submission to NIST Lightweight Cryptography (2019)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
8. 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
9. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. IACR Trans. Symmetric Cryptol. 2020(S1), 60–87 (2020), https://doi.org/10.13154/tosc.v2020.iS1.60-87
10. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. Final Round Submission to NIST Lightweight Cryptography (2021)
11. 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

12. 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
13. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP v2. Final Round Submission to NIST Lightweight Cryptography (2021)
14. 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
15. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Winning Submission to NIST Lightweight Cryptography (2021)
16. 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
17. 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
18. 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
19. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. J. Cryptol. 10(3), 151–162 (1997), https://doi.org/10.1007/s001459900025
20. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is ssl?). In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 310–331. Springer (2001), https://doi.org/10.1007/3-540-44647-8_19
21. Mennink, B.: Understanding the Duplex and Its Security. IACR Trans. Symmetric Cryptol. 2023(2) (2023), to appear
22. 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
23. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018), https://www.rfc-editor.org/info/rfc8446