Research Article

Alexander Moch*

# Provable security against generic attacks on stream ciphers

https://doi.org/10.1515/jmc-2022-0033
received November 24, 2022; accepted May 08, 2023

**Abstract:** Recent lightweight hardware-based stream cipher designs keep an external non-volatile internal state that is *not* part of the cipher's hardware module. The purpose of these so-called small-state ciphers is to keep the size of the hardware and the power consumption low. We propose a random oracle model for stream ciphers. This will allow us to analyse the recent small-state stream cipher designs' resistance against generic attacks and, in particular, time-memory-data tradeoff attacks. We analyse the conventional construction underlying stream ciphers like Grain and Trivium, constructions continuously using the external non-volatile secret key during keystream generation like Sprout, Plantlet, Fruit, and Atom, constructions continuously using the external non-volatile IV, and constructions using a combination of the IV and the key like DRACO. We show the tightness of all bounds by first presenting the time-memory-data tradeoff attacks on the respective constructions, establishing the upper bound on security, and then presenting the proof of security to establish the lower bound on security. In this work, we extend the theoretical work done by Hamann et al. who introduced the DRACO stream cipher at FSE 2023. We use the same random oracle model as the aforementioned work and apply it to the earlier work by Hamann et al. presented at SAC 2019, which showed security for two of the four constructions we consider in this work. Our model is equivalent but allows for a much simpler proof of security. Furthermore, we provide a proof of security for stream ciphers continuously using the secret key during keystream generation, giving upper and lower bounds for all four generic stream cipher constructions proposed so far.

**Keywords:** symmetric-key cryptography, lightweight cryptography, stream ciphers, provable security, TMDTO attacks

**MSC 2020:** 68P25, 94A55, 94A60

## 1 Introduction

The symmetric encryption of a plaintext is usually performed using block ciphers or stream ciphers. The most commonly used block cipher, AES, operates on blocks of 128 bits in size. The inputs to a block cipher are a plaintext block and a secret key that has been communicated between the legitimate parties using a key exchange protocol. If the plaintext consists of more than one block, a mode of operation defines how to repeatedly apply the block cipher to the plaintext.

Stream ciphers, on the other hand, generate a stream of key bits from a secret key and a public variable called the initial value, initialization vector, or just nonce. This pseudorandom keystream is then combined, usually via the bitwise exclusive-or operation, with the plaintext to produce the ciphertext.

Counter-based stream cipher designs produce a block of keystream from a counter value. As an example, one may use the counter mode of operation for block ciphers to encrypt the counter and use the block cipher's

* **Corresponding author: Alexander Moch,** Lehrstuhl für Theoretische Informatik, Universität Mannheim, 68131 Mannheim, Germany, e-mail: moch@uni-mannheim.de

Open Access. © 2023 the author(s), published by De Gruyter. This work is licensed under the Creative Commons Attribution 4.0 International License.

output as a keystream block, increment the counter, and repeat the process. Stateful stream ciphers, on the other hand, maintain an internal state from which the key bits are generated. That internal state is updated before the next key bits are output.

Hardware-based stream ciphers are typically based on feedback shift registers (FSRs) that maintain and update an internal state [1–6]. In this work, we are not concerned with FSRs themselves and refer the interested reader to [7,8]. One objective in designing secure hardware-based stream ciphers is to create an efficient design. In particular, it is important to keep the register size small and have a low power demand.

The goal of an adversary is to recover some unknown plaintext bits. For stream ciphers, this is often done by recovering an internal state, as the internal state ultimately determines the keystream. A known internal state allows one to generate further key bits and thus decrypt ciphertext and recover plaintext. A different adversarial goal is to distinguish the cipher from a truly random bitstream. Proofs of security often show that an adversary cannot distinguish a cipher's output from random, as *indistinguishability* implies resistance against recovery of plaintext bits.

Stream ciphers are vulnerable to an attack type called time-memory-data tradeoff (TMDTO) attacks, introduced by Babbage [9] and Golić [10]. These attacks exploit the birthday paradox to recover an internal state in time $2^{\ell_s/2}$, where $\ell_s$ denotes the internal state length. The TMDTO attacks introduced by Babbage and Golić are generic, meaning they need not exploit any internals of the cipher. Thus, any stream cipher design must have an internal state length of at least twice the desired security level.

Conventional designs keep the entire internal state within the cipher's hardware module. We refer to these as ciphers with a large state and small key (LSSK). Examples of this are the eSTREAM hardware portfolio members Trivium [2] and Grain [1]. More recent designs reduce the register size within the cipher's hardware module by accessing an externally available resource that will *not* be updated during internal state updates. This externally available resource may be the secret key, the IV, or a combination of both stored in EEPROM. We therefore make the distinction between a volatile and a non-volatile internal state. We also note that the non-volatile state may also be kept inside the cipher's hardware module. This will obviously not decrease the register size within the cipher's hardware module but may decrease the power consumption as those register bits need not be updated.

Sprout [5], Plantlet [6], Fruit [3], and Atom [4] are stream ciphers that continuously use the non-volatile secret key during keystream generation. We refer to these as CKEY. The work [11] describes a distinguishing attack on ciphers using the non-volatile secret key during state updates, which has a complexity of $2^{\ell_v/2}$, where $\ell_v$ denotes the size of the *volatile* internal state only. The relevance of this distinguishing attack in practice may be debated, as no internal state recovery or key recovery attack is known. Instead of using the secret key as a non-volatile external resource, one may use the IV [12], a construction we call CIV. These ciphers use an additional parameter $\ell_p$ called the packet length, which specifies how many keystream bits may be generated per key-IV pair. It has been shown that these ciphers achieve a security level of $\ell_v - \log(\ell_p)$ bits, i.e., any successful generic attack has a complexity of at least $2^{\ell_v}/\ell_p$ [13]. The DRACO stream cipher [14] uses the non-volatile IV with a key prefix of length at least $\log(\ell_p)$ to compensate for the factor $\ell_p^{-1}$. DRACO achieves a security level against generic attacks of $\ell_v$ bits, i.e., any successful generic attack has a complexity of at least $2^{\ell_v}$. These cipher constructions are referred to as CIVK.

In this work, we propose a random oracle model (ROM) for stream ciphers. A ROM identifies building blocks within a cryptographic algorithm and replaces those with idealized, i.e., random, primitives. An adversary will get oracle access to these building blocks as well as the construction function in either of two worlds. In the real world, the construction function makes use of the underlying idealized building blocks. In the ideal world, the construction function is an idealized version of the cryptographic algorithm under consideration. In the case of stream ciphers, the idealized version would output a perfectly random keystream as opposed to a pseudorandom one. The adversary's task is to distinguish these two worlds. We will present upper bounds on the adversary's success probability for all the stream cipher constructions that have been proposed so far.

Similar work was done for block ciphers initiated by Even and Mansour [15]. The Even–Mansour construction and its iterated variant can be seen as an abstraction of block ciphers, and in particular, of AES. The round function of the respective block cipher is replaced by an ideal variant, namely, a random permutation. Before every round and after the last round, the secret round key is added. A computationally unbounded, i.e.,

information-theoretic, distinguisher may then query the construction and the underlying permutations. The distinguisher then has to decide whether it was interacting with the Even-Mansour construction or a truly random permutation. In this model, for $r$ rounds, a lower bound of security of $\Theta\left(2^{\frac{rn}{r+1}}\right)$ has been shown [16], where $n$ denotes the block length. There is a huge body of work done on the iterated Even-Mansour cipher and its iterated variant [15–21].

The model is a strong one, as the internals of a cryptographic algorithm are clearly not random. Therefore, proofs in this model will *not* imply the absence of attacks against a concrete scheme but rather show that the interaction of these building blocks is secure. This is particularly interesting for stream ciphers, as TMDTO attacks are generic and need not exploit the internals of the stream cipher. While these proofs do not prove any concrete scheme to be secure, they still provide a solid foundation to base a stream cipher on, as opposed to ad-hoc designs without such proofs. From a different perspective, these types of proofs show that any successful attack *must* exploit the internals of the respective construction. Hence, a generic TMDTO attack cannot have a complexity lower than the bounds we provide in this work. We discuss why proofs in the ROM are valid more thoroughly in Section 1.2.

Hamann et al. [14] present a ROM and proof of security for stream ciphers continuously using the IV and a prefix of the secret key. The present work extends [14] and applies the model to the constructions considered in ref. [13], i.e., the conventional construction and stream ciphers continuously using the IV, as well as to stream ciphers continuously using the secret key, for which, to the best of our knowledge, a proof of security has not yet been presented. We use an equivalent model to that of ref. [13] and, by making use of the H-coefficients technique [22], our proofs of security are significantly simpler than those presented in ref. [13].

## 1.1 Distinguishing attacks

A distinguisher is an adversary, more generally an algorithm, that is given access to an oracle and outputs a decision bit. It is allowed to ask the oracle for encryptions and possibly decryptions of some chosen messages. The decision bit corresponds to a challenge to be solved by the distinguisher. In the IND-CPA and IND-CCA games, a probabilistic polynomial-time distinguisher submits two plaintexts $m_1$ and $m_2$ to the oracle and receives the encryption $c$ of either $m_1$ or $m_2$. Its task is then to decide whether $c$ corresponds to $m_1$ or $m_2$. Before and after receiving $c$, the adversary may query the oracle an arbitrary number of times.

A different approach is that of indistinguishability from random. The goal is to design a cryptographic algorithm with outputs that are indistinguishable from random noise. In this case, the distinguisher gets access to one of two oracles $O_{\text{real}}$ or $O_{\text{ideal}}$ and has to decide which oracle it is interacting with. $O_{\text{real}}$ corresponds to the cryptographic algorithm to be analysed, and $O_{\text{ideal}}$ an ideal counterpart. In the case of a block cipher, the ideal counterpart would be a random permutation. In the case of a stream cipher, the ideal counterpart would be a random bitstream. The distinguishing advantage of a distinguisher $\mathcal{D}$ with access to $O_{\text{real}}$ or $O_{\text{ideal}}$ is measured by:

$$\Delta_{\mathcal{D}}(O_{\text{real}}, O_{\text{ideal}}) = |\Pr[\mathcal{D}^{O_{\text{real}}} \Rightarrow 1] - \Pr[\mathcal{D}^{O_{\text{ideal}}} \Rightarrow 1]|.$$

Typically, an information-theoretic distinguisher is considered, i.e., its computational resources are unbounded and the only measure of complexity is the number of queries to the respective oracle. In our work, we will derive an upper bound on the distinguishing advantage and thus a lower bound on the stream cipher's security. This will show that the stream cipher cannot be distinguished from a truly random bitstream with a complexity lower than the bounds given. The security against distinguishing attacks also implies the hardness of internal state recovery attacks and key recovery attacks.

## 1.2 Random oracle model

Proofs in the ROM were made popular by the work *Random Oracles are Practical* by Bellare and Rogaway [23]. The idea behind the ROM is to identify primitives, e.g., the round function in AES as a pseudorandom

permutation or hash functions in cryptographic protocols as pseudorandom functions, in a cryptosystem and replace these primitives with an idealized version, i.e., a random permutation or a random function. All parties, good and bad alike, get access to the idealized primitives through an oracle. The oracle will receive a query to that primitive, sample an answer uniformly at random, and return the answer.

The cryptosystem is then proved to be secure in this model. Finally, when instantiating the cryptosystem, the random oracles are replaced with real-life primitives. In practice, instantiations of the cryptosystem's primitives are significantly simpler than their random counterparts. Bellare and Rogaway "stress that the proof is in the ROM and the last step is heuristic in nature" [23].

The soundness of the ROM has been debated. A work by Canetti et al. [24] shows that it is possible to design cryptosystems that are secure in the ROM but for which no secure instantiation exists. Yet, Canetti et al. [24] also note that:

> [The random oracle model] may be useful as a test-bed (or as a sanity check).
>
> After all, a security proof in the Random Oracle Model eliminates a broad class of potential attacks (i.e., the ones that would work also in the Random Oracle Model), and in many cases it seems that attacks of this type are usually the ones that are easier to find.
>
> For now, however, I view the random oracle methodology as a very useful "engineering tool" for devising schemes. As a practical matter, I would much rather see today's standards built around schemes that are proven secure in the Random Oracle Model, than around schemes for which no such proofs exist. If nothing else, it makes finding attacks on such schemes a whole lot harder.

Therefore, we see proofs in the ROM as a useful groundwork and a solid foundation to base stream cipher designs upon.

## 1.3 Our contribution

We propose a ROM for the four stream cipher designs mentioned earlier. We then derive an upper bound on a distinguisher's advantage and a lower bound on the security of the scheme, and thus show the absence of generic attacks with lower complexity than the given bounds. In particular, we confirm the lower bound of half the internal state of conventional stream cipher designs and increased security for the enhanced state stream cipher that incorporates the initial value during keystream generation. We note that designs continuously using the secret key during keystream generation offer no benefit over the conventional design with regard to security against distinguishing attacks. Furthermore, we present matching TMDTO attacks on each of the constructions presented in this work, thus showing that the presented bounds are tight.

This work enhances the previous works [13,14]. Hamann et al. [13] provide lower bounds for LSSK and CIV. In their model, the adversary receives a block of keystream per query. This is similar to TMDTO attacks, where one typically searches for collisions between keystream blocks. This makes the analysis more cumbersome, as one has to consider overlapping blocks. In our model, the adversary receives individual bits per query, which more closely models the *stream* of bits. Both models are equivalent. Furthermore, we utilize the H-coefficients technique [22], which allows us to significantly simplify the proof of security for LSSK and CIV.

We show the same bound for CIVK as [14], yet we distinguish the key length $\ell_k$ and the volatile state length $\ell_v$, whereas in ref. [14], it is assumed that $\ell_k = \ell_v$. Furthermore, we present a proof of security for CKEY.

We focus on the theoretical foundations of an enhanced state and do not argue about the hardware implications. In particular, we consider this work to be an extension of the theoretical part of ref. [14]. One interesting aspect to note is that the enhanced state stream ciphers show that not all bits of the internal state need to be updated. Our proofs consider the state to be split into a non-volatile and a volatile part. There is no requirement, however, that the non-volatile internal state may not be kept inside the hardware module. In fact, DRACO [14] considers a variant where the IV and the key prefix are also kept in the hardware module and shows a reduction of power consumption by 34% compared to Grain-128a [25].

### 1.4 Structure of this work

In Section 2, we give an overview of stateful stream ciphers and present the enhanced state stream ciphers. In Section 3, we present the known TMDTO attacks on the four constructions. In Section 4, we present the random oracle model and the proof technique. Sections 5–9 present the proofs of security. Section 10 concludes this article.

## 2 Enhanced state stream ciphers

Stream ciphers are inspired by the one-time-pad encryption scheme. The one-time-pad uses a secret stream of bits drawn uniformly at random that is added, usually via the exclusive-or operation, to the plaintext message. This also implies that the secret keystream must have at least the length of the plaintext message to be sent, making the key exchange rather cumbersome. Further complicating matters, the keystream must not be used more than once.

Stream ciphers generate a pseudorandom keystream from a small seed, typically a combination of a small key and an initial value (also called initialization vector or nonce). The pseudorandom keystream is then added to the plaintext to produce the ciphertext. The small key is communicated between the legitimate parties via a key exchange protocol. The initial value need not be kept secret and is typically publicly known, but it must not be repeated, as this would lead to identical keystreams.

### 2.1 Counter-based and stateful designs

One typically distinguishes two types of stream ciphers: counter based and stateful designs. The block cipher mode of operation counter mode (CTR) is an example of a counter-based stream cipher design. A counter is encrypted using a block cipher to produce a block of keystream. The counter is then incremented and encrypted to produce the next block of keystream, and so on.

Stateful stream cipher designs maintain an internal state that is updated throughout keystream generation. The output feedback mode (OFB) and cipher feedback mode (CFB) may be viewed as such designs, as the keystream (OFB) or ciphertext (CFB) is used as an internal state “updated” by a block cipher encryption.

### 2.2 Hardware-oriented designs

Hardware-oriented stream cipher designs typically employ FSRs. In each update, the FSRs shift their content by one bit (possibly more), where the bit leaving the register is returned as output, and the empty position is filled using the feedback function that computes the new bit depending on the current register contents.

There are linear feedback shift registers (LFSRs) and nonlinear feedback shift registers (NFSRs). The difference lies in whether the feedback function is linear. For LFSRs, there are efficient means to determine whether a feedback function produces a maximal period LFSR. However, it is not known how to do this for NFSRs. The advantage of NFSRs, however, is that they are cryptographically stronger due to their nonlinearity. For small NFSRs, one can empirically verify the period, and by combining several of them, a sufficient state size can be attained. To further strengthen a stream cipher design, instead of just outputting the leaving bit, the output bit and the content of an FSR are typically processed by a filter or output function that produces the keystream bits.

## 2.3 Abstracting a stream cipher

In this work, we will focus on stream ciphers based on FSRs and provide an abstraction to analyse their security against generic attacks, in particular TMDTO attacks. Furthermore, contemporary stream cipher designs split their internal state into a volatile part and a non-volatile part that may or may not be held in the cipher's hardware module [3–6,14]. We identify the following parts that make up an FSR-based stream cipher:

- **Internal state:** A stream cipher maintains an internal state that may be divided into a volatile part and a non-volatile part.
- **Loading function:** The secret key, the IV, and possibly some constants are loaded into the registers (respectively, internal state) before any keystream is generated. We will typically ignore loading and start with the loading state.
- **Initialization:** An initialization function computes the initial state, i.e., the state from which the first bit is output, from the loading state. This is also referred to as *mixing*.
- **State update:** An update function updates one internal state to the next internal state.
- **Output:** An output function computes an output bit from an internal state.

Later, in our random oracle model, *initialization* and *output* will be replaced with ideal counterparts.

## 2.4 Description of the ciphers

We denote by $Q_{\mathrm{nv}}$ the non-volatile internal state space and by $Q_{\mathrm{v}}$ the volatile internal state space. We denote an internal state by $\langle a \mid b \rangle$, where the left side $a \in Q_{\mathrm{nv}}$ denotes the non-volatile internal state and the right side $b \in Q_{\mathrm{v}}$ denotes the volatile internal state. We chose this notation to make it easier to distinguish internal states from arbitrary tuples. Also, if, for example, the non-volatile state $a$ consists of two parts $a_1$ and $a_2$, we will denote this state by $\langle a_1, a_2 \mid b \rangle$. We hide constants in the notation of the internal states as they are mostly irrelevant to the analysis. In particular, they will only increase the state size and thus, in our model, not negatively affect security. The key space is denoted by $\mathcal{K}$ and the IV space is denoted by $\mathcal{IV}$. The lengths of the non-volatile internal state space, the volatile internal state space, the secret key, and the IV are denoted by $\ell_{\mathrm{nv}}$, $\ell_{\mathrm{v}}$, $\ell_{\mathrm{k}}$, and $\ell_{\mathrm{IV}}$ respectively. The following description will define the keystream generation for the four constructions.

- **Packet length:** LSSK and CKEY do not define a packet length. CIV and CIVK, on the other hand, define a packet length $\ell_{\mathrm{p}}$ as an additional parameter that limits the number of output bits per key-IV pair. For each key-IV pair $(k, x) \in \mathcal{K} \times \mathcal{IV}$, CIV and CIVK can output up to $\ell_{\mathrm{p}}$ keystream bits, after which a new IV must be exchanged. For simplicity, we define the packet length for LSSK and CKEY to be the total number of keystream bits that can be generated from a key-IV pair, i.e., $\ell_{\mathrm{p}} = 2^{\ell_{\mathrm{s}}}$.
- **Enhanced state:** Conventional stream cipher designs according to LSSK only use a volatile internal state, i.e., the entire internal state is contained in the cipher's hardware module. For CKEY, CIV, and CIVK, the internal state is divided into a volatile part that is updated during state updates and a non-volatile part that is *not* updated during state updates. The non-volatile state of the enhanced state ciphers is composed as follows:

  **CKEY**: The non-volatile internal state consists of the secret key $k$ only.
  **CIV**: The non-volatile internal state consists of the initial value $x$ only.
  **CIVK**: Same as CIV, but the non-volatile internal state further contains a key prefix $k^{\mathrm{pre}}$ of length at least $\log(\ell_{\mathrm{p}})$.

- **Loading function:** The loading function load takes a key-IV pair $(k, x) \in \mathcal{K} \times \mathcal{IV}$ as input and loads it into the registers to produce the *loading state*. We will omit the loading process for the most part of our analysis and start the keystream generation process with the loading state, but including it in our notation is useful.

$$\textbf{LSSK}: \text{load} : \mathcal{K} \times \mathcal{IV} \to Q_{\text{v}}, (k, x) \mapsto |x, k\rangle,$$
$$\textbf{CKEY}: \text{load} : \mathcal{K} \times \mathcal{IV} \to Q_{\text{nv}} \times Q_{\text{v}}, (k, x) \mapsto \langle k \mid x\rangle,$$
$$\textbf{CIV}: \text{load} : \mathcal{K} \times \mathcal{IV} \to Q_{\text{nv}} \times Q_{\text{v}}, (k, x) \mapsto \langle x \mid k\rangle,$$
$$\textbf{CIVK}: \text{load} : \mathcal{K} \times \mathcal{IV} \to Q_{\text{nv}} \times Q_{\text{v}}, (k, x) \mapsto \langle x, k^{\text{pre}} \mid k\rangle.$$

– **Mixing function:** The loading state is used as input to the mixing function $p$. Its task is to provide the initial state with enough confusion and diffusion for further operation and corresponds to clocking the cipher without producing output bits.

$$\textbf{LSSK}: p : Q_{\text{v}} \to Q_{\text{v}}, |x, k\rangle \mapsto |y\rangle,$$
$$\textbf{CKEY}: p : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle k \mid x\rangle \mapsto \langle k \mid y\rangle,$$
$$\textbf{CIV}: p : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle x \mid k\rangle \mapsto \langle x \mid y\rangle,$$
$$\textbf{CIVK}: p : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle x, k^{\text{pre}} \mid k\rangle \mapsto \langle x, k^{\text{pre}} \mid y\rangle.$$

– **State update:** The state update function $\pi$ updates a (volatile) internal state to the next (volatile) internal state.

$$\textbf{LSSK}: \pi : Q_{\text{v}} \to Q_{\text{v}}, |y\rangle \mapsto |y'\rangle,$$
$$\textbf{CKEY}: \pi : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle k \mid y\rangle \mapsto \langle k \mid y'\rangle,$$
$$\textbf{CIV}: \pi : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle x \mid y\rangle \mapsto \langle x \mid y'\rangle,$$
$$\textbf{CIVK}: \pi : Q_{\text{nv}} \times Q_{\text{v}} \to Q_{\text{nv}} \times Q_{\text{v}}, \langle x, k^{\text{pre}} \mid y\rangle \mapsto \langle x, k^{\text{pre}} \mid y'\rangle.$$

$r$ successive invocations of the state update function $\pi$ on an internal state $\langle a \mid b\rangle$ are denoted by $\pi^r(\langle a \mid b\rangle)$, e.g., for three successive invocations we write:

$$\pi^3(\langle a \mid b\rangle) = \pi(\pi(\pi(\langle a \mid b\rangle))).$$

It is needed that the period of the state update function $\pi$ is larger than or equal to $\ell_{\text{p}}$ for the entire internal state space. This means that for any internal state $\langle a \mid b\rangle$, the set $\{\langle a \mid b\rangle, \pi^1(\langle a \mid b\rangle), \ldots, \pi^{\ell_{\text{p}}-1}(\langle a \mid b\rangle)\}$ contains $\ell_{\text{p}}$ distinct elements.

– **Output function:** The output function $f$ maps an internal state $\langle a \mid b\rangle$ to an output bit $z \in \{0, 1\}$.

$$\textbf{LSSK}: f : Q_{\text{v}} \to \{0, 1\}, |y\rangle \mapsto z,$$
$$\textbf{CKEY}: f : Q_{\text{nv}} \times Q_{\text{v}} \to \{0, 1\}, \langle k \mid y\rangle \mapsto z,$$
$$\textbf{CIV}: f : Q_{\text{nv}} \times Q_{\text{v}} \to \{0, 1\}, \langle x \mid y\rangle \mapsto z,$$
$$\textbf{CIVK}: f : Q_{\text{nv}} \times Q_{\text{v}} \to \{0, 1\}, \langle x, k^{\text{pre}} \mid y\rangle \mapsto z.$$

– **Keystream generation:** Let $(k, x)$ be an arbitrary key-IV pair. By using the functions defined earlier, we can define the construction function $e$ that corresponds to the keystream generation using the aforementioned constructions:

$$e : \mathcal{K} \times \mathcal{IV} \times \{0, \ldots, \ell_{\text{p}} - 1\} \to \{0, 1\}, (k, x, r) \mapsto f(\pi^r(p(\text{load}(k, x)))).$$

We consider individual output bits, and $e$ outputs the $r$th keystream bit. The entire keystream packet of length $\ell_{\text{p}}$ for a key-IV pair $(k, x)$ looks as follows:

$$e(k, x, 0) \,||\, e(k, x, 1) \,||\, \cdots \,||\, e(k, x, \ell_{\text{p}} - 1).$$

Note that $\ell_{\text{p}} = 2^{\ell_s}$ for LSSK and CKEY.

## 2.5 Discussion of the packet length

The eSTREAM hardware portfolio's three ciphers are all designed to handle potentially very large keystream sequences per key-IV pair. Grain v1 [1], MICKEY 2.0 [26], and Trivium [2] all use 80-bit keys, and they have different IV lengths of 64 bits, up to 80 bits, and 80 bits, respectively. The authors of Grain v1 do not give an explicit limit on the number of keystream bits that should be generated for each key-IV pair. MICKEY 2.0 limits

the amount of keystream bits to $2^{40}$ per key-IV pair, and Trivium has a limit of $2^{64}$ keystream bits per key-IV pair. Much smaller packet sizes are used by transmission standards like A5/1, Bluetooth, TLS, and IEEE 802.11 for wireless local area networks.

- Per key-IV pair, **A5/1** can only generate 228 keystream bits. The session key is 64 bits long, and the IV corresponds to 22 bits of the publicly known frame number.
- For the so-called basic rate, **Bluetooth** packets can only include a maximum of 2790 bits. The Bluetooth cipher $E_0$ takes a 128-bit session key and uses 26 bits of the master's clock as the packet-specific IV. The master's clock is assumed to be publicly known.
- **Wireless LAN** communication is standardized in the IEEE 802.11 technical standards. At most 11454 bytes (i.e., $<2^{17}$ bits) are encrypted under the same key-IV pair using CCMP as specified by the currently active IEEE 802.11-2020 standard [27].
- Because **SSL/TLS** is the foundation of HTTPS, it is crucial for protecting the World Wide Web. In the most recent version, TLS 1.3 [28], the maximum amount of data encrypted under the same key-IV pair is $2^{14} + 2^8$ bytes (i.e., $2^{17} + 2^{11} < 2^{18}$ bits), as long as RC4, which is now forbidden for all TLS versions by RFC 7465 [29], is not used.

We view the packet length as a valid additional parameter in stream cipher design since all of the popular transmission standards mentioned earlier impose a limit on the number of keystream bits generated per key-IV pair. We also want to emphasize that, once the $\ell_\mathrm{p}$ keystream bit limit is reached, only the IV needs to be changed. *Rekeying* is not necessary. As we argue in the following subsection, a full $2^{\ell_\mathrm{k}}$ bits can be generated per key if the IV length and the packet length are properly selected.

## 2.6 Discussion of the state lengths

It is up to the designer of the corresponding cipher to specify the lengths of the volatile and non-volatile. The proofs of the four constructions will show how the lengths affect the security. However, we would like to provide a brief discussion in this subsection.

As the attack by Babbage [9] and Golić [10] is still applicable without modification, the combined total state length $\ell_s = \ell_\mathrm{nv} + \ell_\mathrm{v}$ must still be twice as large as the desired security level.

It is important to maintain a balance between the IV length $\ell_\mathrm{IV}$ and the packet length $\ell_\mathrm{p}$ such that $2^{\ell_\mathrm{IV}} \cdot \ell_\mathrm{p} \geq 2^{\ell_\mathrm{k}}$. If this condition is not met, the codebook would be smaller than $2^{\ell_\mathrm{k}}$, which would allow for a trivial attack with a lower complexity than exhaustive search.

Usually, the size of the key prefix, $k^\mathrm{pre}$, is chosen to be at least $\log(\ell_\mathrm{p})$. This compensates for the factor $\log(\ell_\mathrm{p})$ in the bound of CIV, as a guess of the key prefix is correct with a probability of $\ell_\mathrm{p}^{-1}$. Minor modifications to the key prefix's size will not significantly impact the security level.

Constants that may be added for a variety of reasons will not be taken into account in our notation or analysis. They will not have a negative effect on the security level in our analysis because they will only increase the size of the state. In our proofs, we will also distinguish the key length $\ell_\mathrm{k}$ and the IV length $\ell_\mathrm{IV}$ from the state lengths $\ell_s$, $\ell_\mathrm{v}$, and $\ell_\mathrm{nv}$. The difference between them may only be due to constants.

## 2.7 Security requirements

Throughout this work, we consider indistinguishability from a random bitstream as our goal. This implies the hardness of *state recovery* and *key recovery*.

If the adversary has acquired an internal state, it allows them to generate further keystream bits and distinguish these bits from randomly generated bits. If the key is known, the adversary can use the initialization and state updates to acquire any internal state and thus obtain any keystream bits.

Stream ciphers like Trivium [2] have an efficiently invertible initialization algorithm. In the case of internal state recovery, this allows for the recovery of the secret session key.

# 3 TMDTO attacks

Stream ciphers are vulnerable to a type of attack called TMDTO attacks, as described by Babbage [9] and Golić [10]. TMDTO attacks consider three cost dimensions: time $T$, memory $M$, and data $D$. Data $D$ refers to the amount of data, usually observed keystream, that the adversary has obtained. Memory $M$ measures the memory consumption of the attack, and time $T$ measures the required computation time. Often, time $T$ is divided into an online phase with $T_{\text{on}}$ and an offline phase $T_{\text{off}}$. The offline phase is also known as precomputation and involves computing some auxiliary data structures before carrying out the actual attack in the online phase. TMDTO attacks typically specify a relation between the cost dimensions that allows for a successful attack. In other words, an adversary may spend more on one resource and, in return, less on another. Hence, these attacks are referred to as *tradeoff* attacks.

The TMDTO attacks by Babbage and Golić are generic in the sense that they do not need to exploit the internal structure of the stream cipher. In these attacks, an adversary generates $T$ keystream prefixes of length $\ell_s$ from randomly chosen internal states and observes $D$ keystream blocks of length $\ell_s$. A collision between these two is highly likely if $T \cdot D = 2^{\ell_s}$, and it is often caused by a collision of the underlying internal states. This collision allows for the recovery of the internal state used to generate the keystream. The relation $T \cdot D = 2^{\ell_s}$ has its optimal point at $T = D = 2^{\ell_s/2}$, i.e., the birthday bound, named after the birthday paradox.

An attack with a tradeoff $T \cdot M^2 \cdot D^2 = 2^{2 \cdot \ell_s}$ and $T_{\text{off}} = 2^{\ell_s}/D$ was developed by Biryukov and Shamir [30]. They combined the attacks of Babbage and Golić with Hellman's attack on block ciphers [31]. The GSM cipher A5/1 was targeted by an attack from Biryukov, Shamir, and Wagner using a method known as BSW-sampling [32], which Biryukov and Shamir [30] also discuss. The tradeoff curve $T \cdot M^2 \cdot D^2 = 2^{2 \cdot \ell_s}$ and the relation $T_{\text{off}} = 2^{\ell_s}/D$ remain the same, even though BSW-sampling allows for the relaxation of the restriction $T \geq D^2$ in the aforementioned attack. Therefore, even the use of BSW-sampling does not result in an attack with overall complexity lower than $2^{\ell_s/2}$, considering precomputation as part of the overall attack complexity. In addition, because BSW-sampling is highly cipher-specific (see ref. [30] for further details), the corresponding TMDTO attacks are not entirely generic.

Instead of sampling from the space of internal states, Hong and Sarkar [33] consider the TMDTO case of sampling pairs of keys and IVs. The overall complexity of this approach (including precomputation) in a single-key scenario, as analysed by us, is at least as high as that of an exhaustive key search. A lower overall complexity can only be achieved in scenarios with multiple keys, where the attacker's objective is to discover *one* of these keys. Therefore, neither the results of [30] nor [33] conflict with our security bounds.

In conventional stream ciphers, a recovered internal state allows for the generation of further keystream bits and thus the decryption of additional ciphertext. Furthermore, the state update, respectively the mixing function, is often bijective. From a recovered internal state, this allows for the recovery of the loading state and thus the secret key $k$.

In this section, we will present TMDTO attacks on all the constructions considered in this work. For simplicity, we focus on the cost dimensions time $T$ and data $D$. These dimensions correspond to the queries made by an adversary, where $T$ corresponds to queries made to the underlying primitives, and $D$ corresponds to queries made to the construction function. Hence, focusing on $T$ and $D$ only will suffice to demonstrate that our bounds are tight.

## 3.1 The conventional TMDTO attack

The conventional TMDTO attack is due to Babbage [9] and Golić [10]. It is targeted against conventional stream ciphers according to LSSK. Its goal is to recover an internal state from some observed keystream. An adversary generates several internal states uniformly at random and computes a corresponding keystream prefix of at least length $\ell_v$. A collision of this keystream prefix with the observed keystream is highly likely due to a collision in the corresponding internal states. We will now explain the basic attack.

(1) Generate $a$ internal states $s_i$ uniformly at random and compute a keystream prefix $Z_i$ of length $\ell_v$ for every $s_i$. Store all $(s_i, Z_i)$ in an efficiently searchable data structure, indexed by $Z_i$.

(2) Obtain $b$ keystream prefixes $Y_j$ of length slightly larger than $\ell_{\mathrm{v}}$. This may be done by sliding a window of length $\ell_{\mathrm{v}}$ over a $b$-bit keystream.
(3) Search for a collision between $Z_i$ and $Y_j$. Obtain $s_i$.
(4) Verify if $s_i$ is the corresponding internal state by generating some more keystream bits and comparing them to the observed ones.

If $a \cdot b = 2^{\ell_{\mathrm{v}}}$, a collision is highly likely to occur due to the birthday paradox. For $a = b = 2^{\ell_{\mathrm{v}}/2}$, one obtains the optimal point with regard to overall complexity, and hence, the security of the cipher is capped by a level of $\ell_{\mathrm{v}/2}$ bits.

## 3.2 TMDTO attacks against CKEY

Mounting the attack in its original form over the entire (volatile + non-volatile) state would entail guessing the key and is therefore moot. However, there exists a distinguishing attack that works in time $2^{\ell_{\mathrm{v}}/2}$. Contrary to the conventional TMDTO attack, it requires a chosen-IV attacker. The idea is that within a session, the IV changes but the key does not. In other words, one can provoke a collision within the volatile internal state for two IVs, as the non-volatile state remains fixed throughout the session.
(1) Generate an initial value $x'$ and obtain a keystream prefix $Z'$ of length $a$. Let $\ell' > \ell_{\mathrm{v}}$. Slide an $\ell'$-bit window over $Z'$ to obtain approximately $a$ keystream blocks $z_i'$. Store $z_i'$ in an efficiently searchable data structure $\mathcal{Z}$.
(2) For $b$ initial values $x_i$, obtain a keystream prefix $Z_i$ of length $\ell'$. Search for a collision in $\mathcal{Z}$. If a collision is found for $a \cdot b = 2^{\ell_{\mathrm{v}}}$, we are in the real world.

As the key remains fixed through the session, an internal state collision occurs if the volatile internal states collide. Due to the birthday paradox, this occurs with high probability if $a \cdot b = 2^{\ell_{\mathrm{v}}}$. The optimal point is $a = b = 2^{\ell_{\mathrm{v}}/2}$. Note that this distinguishing attack only works for a chosen-IV attacker and does not allow for the recovery of an internal state. In a world with truly random outputs, one would obtain a keystream collision of length $\ell'$ after generating and observing $a = b = 2^{\ell'/2} > 2^{\ell_{\mathrm{v}}/2}$ blocks.

## 3.3 TMDTO attacks against CIV

For CIV, an attacker can perform the original TMDTO attack on the entire internal state. This will yield the tradeoff curve $D \cdot T = 2^{\ell_s} = 2^{\ell_{\mathrm{v}}+\ell_{\mathrm{nv}}}$, i.e. the optimal point is at $D = T = 2^{\ell_s/2}$. Also, note that the maximum amount of data that can be obtained is $\ell_{\mathrm{p}} \cdot 2^{\ell_{\mathrm{IV}}}$. The alternative is to exploit the structure of the cipher and obtain an attack with better complexity:
(1) Obtain $a$ keystream *packets* $p_i$ of length $\ell_{\mathrm{p}}$. Each packet contains approximately $\ell_{\mathrm{p}}$ windows $W_i^j$ of length $\ell_s$. Store $W_i^j$ in an efficiently searchable data structure. Note that the data complexity is $D = a \cdot \ell_{\mathrm{p}}$.
(2) For each packet $p_i$ with initial value $x_i$, generate $b$ random volatile internal states $s_j$ and compute their corresponding keystream prefix $Z_j^i$ of length $\ell_s$. Search for a collision between $Z_j^i$ and $W_i^j$. Note that the time complexity is $T = a \cdot b$.

A collision in the volatile internal state is likely if $D \cdot b = a \cdot b \cdot \ell_{\mathrm{p}} = 2^{\ell_{\mathrm{v}}}$. In particular, we obtain that the time complexity of this attack is $T = 2^{\ell_{\mathrm{v}}}/\ell_{\mathrm{p}}$. Note that $T$ does not depend on the amount of keystream packets observed.

### 3.4 TMDTO attacks against CIVK

Similar to CIV, one can apply both attack types to CIVK. The first one remains unchanged. The second one needs to account for the key prefix and guess the corresponding key prefix in step 2.

(1) Obtain $a$ keystream *packets* $p_i$ of length $\ell_p$. Each packet contains approximately $\ell_p$ windows $W_i^j$ of length $\ell_s$. Store $W_i^j$ in an efficiently searchable data structure. Note that the data complexity is $D = a \cdot \ell_p$.

(2) For each packet $p_i$ with initial value $x_i$, generate $b$ pairs of key prefixes *and* random volatile internal states $(k^{\text{pre}}, s_j)$ and compute their corresponding keystream prefix $Z_j^i$ of length $\ell_s$. Search for a collision between $Z_j^i$ and $W_i^j$. Note that the time complexity is $T = a \cdot b$.

A collision in the volatile internal state is likely if $D \cdot b = a \cdot b \cdot \ell_p = 2^{\ell_v}$. Further, one needs to consider that the key prefix guesses are correct with a probability of $\ell_p^{-1}$, i.e., $a \cdot b \cdot \ell_p \cdot \ell_p^{-1} = 2^{\ell_v}$. This yields a total time complexity of the attack of $T = 2^{\ell_v}$. Note that $T$ does not depend on the amount of keystream packets observed.

**Remark.** Note that for CIV and CIVK, assuming a chosen-IV attacker, all $a \cdot b = 2^{\ell_v}$ keystream packets can be precomputed and need to be stored in an efficiently searchable data structure, i.e., a binary tree. This will require a time complexity of $2^{\ell_v}$ in the offline phase and a space (memory) complexity of $2^{\ell_v}$. The online phase, when observing a keystream packet, will then have a time complexity of $\log_2(2^{\ell_v}) = \ell_v$ to search for the collision on the previously computed keystream packets. While the online time complexity is significantly reduced, there is an excessive amount of precomputation time necessary, and the space complexity in the online phase is $2^{\ell_v}$. Even if a time complexity of $2^{\ell_v}$ were not excessive, a space complexity of $2^{\ell_v}$ may very well be.

## 4 Proof preliminaries

In the following, we will present the preliminaries, notation, and ROM necessary for the proof of security.

In this section, we will stick with the generic $\langle a \mid b \rangle$ notation, where $a$ describes the non-volatile part of the cipher and $b$ describes the volatile part of the cipher. This will suffice for describing the model and keep the notation simple in this section. In the proofs of the respective constructions, we will give detailed information about the internal state, queries, and transcripts.

### 4.1 Random oracle model

An adversary will be interacting with a set of three oracles in one of two worlds: the real world or the ideal world. There will be an oracle $P$ for the mixing function, an oracle $F$ for the output function, and an oracle $E$ for the construction function.[1] $E$ is defined differently in either world. The adversary can query the oracles with the inputs of the respective functions, and it will receive answers from the oracle. These query-answer pairs are collected by the adversary in a transcript $\tau$. Finally, the adversary has to output a decision bit.

The $P$- and $F$-oracles will answer their queries using ideal randomized primitives in either world. The $P$-oracle will use a random permutation $\mathbf{P}$ (for CIV and CIVK: multiple random permutations as described later), and the $F$-oracle will use a random function $\mathbf{F}$. In the real world, the $E$-oracle uses $\mathbf{P}$ and $\mathbf{F}$ as underlying building blocks. In the ideal world, the $E$-oracle will have access to another independent random function $\mathbf{E}$. By assuming the underlying building blocks to be ideal, one can abstract from possible weaknesses in the

1 $\pi$ is publicly known and hence no oracle is needed.

mixing function and the output function that an implementation may have and show that the scheme, the interaction of those building blocks, is secure. This will *not* prove an instantiation to be secure but provide a plausible justification for the structure of a cipher built upon the respective construction.

In the ideal world, **E**, corresponding to the encryption function, will sample the output bits uniformly at random from $\{0, 1\}$. We will show that the adversary cannot distinguish the ideal world from the real world in this scenario. In particular, this will show that the keystream generated by the "real" encryption function is indistinguishable from a truly random bitstream.

### 4.1.1 The distinguishing game

In the beginning of the adversary's interaction with the oracles, a key $k \xleftarrow{\$} \mathcal{K}$ will be sampled uniformly at random from the key space $\mathcal{K}$. Next, the adversary poses its questions to the $P$-, $F$-, and $E$-oracles with the additional limit of at most $\ell_{\mathrm{p}}$ $E$-queries per IV $x$.[2] All of the query-answer pairs will be collected in the corresponding $\tau_P$, $\tau_F$, and $\tau_E$ transcripts. When the adversary is finished with its interaction with the oracles, it is given the secret key $k$, as well as the transcript $\tau_a$, which contains the intermediate values corresponding to internal states generated during an $E$-query and will be defined more explicitly later. Based on the transcript $\tau = (\tau_P, \tau_E, \tau_F, \tau_a, k)$, the adversary has to make its decision whether it was interacting with the real world or the ideal world and output a decision bit. If the adversary's guess is correct, it wins the game. Our task is to upper bound the adversary's success probability.

### 4.1.2 Oracle queries

The adversary will be given access to the $P$-, $F$-, and $E$-oracles that correspond to the mixing function, output function, and construction function, respectively. For clarity, in the beginning of the proof of each construction, we provide a table that describes the structure of the inputs and outputs of the respective queries, as chosen by the adversary. We will then explain how the oracles are implemented. Note that we index the query variables with the query type, i.e., $\langle a_P \mid b_P \rangle$ denotes a $P$-query where $a_P$ and $b_P$ are chosen by the adversary.

(1) **P-oracle:** The $P$-oracle that corresponds to the mixing function will be implemented using random permutations. Note that the mixing function $p$ keeps the non-volatile internal state unchanged and only the volatile internal state gets permuted. Hence, for every $a \in \mathcal{Q}_{\mathrm{nv}}$, the volatile internal state $b \in \mathcal{Q}_{\mathrm{v}}$ is permuted using an independent random permutation $\mathbf{P}_a : \mathcal{Q}_{\mathrm{v}} \to \mathcal{Q}_{\mathrm{v}}$. For each $\mathbf{P}_a$, we will use lazy sampling: as the first $P$-query $\langle a_P \mid b_P \rangle$ arrives, the oracle will sample the answer uniformly at random from all $2^{\ell_{\mathrm{v}}}$ possible answers. As the second $P$-query arrives, the oracle will sample the answer uniformly at random from all $2^{\ell_{\mathrm{v}}} - 1$ remaining possible answers, and so on. The permutation $\mathbf{P}$ is defined as follows:

$$\mathbf{P} : \mathcal{Q}_{\mathrm{nv}} \times \mathcal{Q}_{\mathrm{v}} \to \mathcal{Q}_{\mathrm{nv}} \times \mathcal{Q}_{\mathrm{v}}, \quad \langle a_P \mid b_P \rangle \mapsto \langle a_P \mid \mathbf{P}_{a_P}(b_P) \rangle.$$

The adversary may query the $P$-oracle in either the forward or the backward direction. That is, the adversary gets oracle access to $\mathbf{P}$ as well as $\mathbf{P}^{-1}$.

(2) **F-oracle:** The $F$-oracle that corresponds to the output function will be implemented using a random function $\mathbf{F} : \mathcal{Q}_{\mathrm{nv}} \times \mathcal{Q}_{\mathrm{v}} \to \{0, 1\}$. Only queries in the forward direction are allowed. We again use lazy sampling; as an $F$-query arrives, the output is sampled uniformly at random from $\{0, 1\}$.

(3) **E-oracle:** The $E$-oracle is defined differently in the real world and in the ideal world: In the real world, it is defined similarly to the construction function $e$ and implicitly uses the secret key $k$, the random permutation $\mathbf{P}$, the state update function $\pi$, and the random function $\mathbf{F}$:

$$\mathbf{E} : \mathcal{IV} \times \{0, \ldots, \ell_{\mathrm{p}} - 1\} \to \{0, 1\}, \quad (x_E, r_E) \mapsto \mathbf{F}(\pi^{r_E}(\mathbf{P}(\mathrm{load}(k, x_E))))$$

2 Note that the key remains fixed throughout the game, so only the IV is changed.

In the ideal world, $\mathbf{E} : \mathcal{IV} \times \{0, \ldots, \ell_p - 1\} \to \{0,1\}$ is a random function with outputs sampled uniformly at random from $\{0,1\}$.

We will assume that the adversary does not make redundant queries, i.e. the adversary will not query for a value that has been queried before. As redundant queries yield no additional information, this assumption does not lower the adversary's distinguishing advantage.

### 4.1.3 Transcripts

All of the adversary's queries to the oracles and the corresponding answers will be collected in a transcript $\tau$. In particular, we will keep separate transcripts $\tau_P$, $\tau_F$, and $\tau_E$ for each of the corresponding oracles $P$, $F$, and $E$, respectively, defined as follows:

$$\tau_P \coloneqq \{(a_P, b_P, y_P) \mid \langle a_P \mid b_P \rangle \text{ is a } P\text{-query and } \langle a_P \mid y_P \rangle \text{ its answer.}\}$$
$$\tau_F \coloneqq \{(a_F, b_F, z_F) \mid \langle a_F \mid b_F \rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}$$
$$\tau_E \coloneqq \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$

The transcripts mentioned earlier are visible to the adversary as it makes its queries to the oracles. Once the adversary's interaction with the oracles is finished, it will additionally be given the secret key $k$ and the transcript $\tau_\alpha$ defined as follows:

$$\tau_\alpha \coloneqq \{(x_E, r_E, \alpha_E, \alpha_E^r) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^r) \text{ its } \alpha\text{-values.}\}$$

The $\alpha$-values for each $E$-query $(x_E, r_E)$ are defined as follows:

$$\alpha_E \coloneqq P(\mathrm{load}(k, x_E)) \text{ and } \alpha_E^{r_E} \coloneqq \pi^{r_E}(P(\mathrm{load}(k, x_E))).$$

The $\alpha$-values correspond to the internal states that are generated during an $E$-query. $\alpha_E$ represents the initial state, and $\alpha_E^{r_E}$ represents the internal state that is the input to the output function $\mathbf{F}$ after $r_E$ state updates on the initial state. We will also sample these values in the ideal world, even though the construction function $E$ does not depend on these.

The full transcript can be written as a 5-tuple $\tau = (\tau_P, \tau_E, \tau_F, \tau_\alpha, k)$.

### 4.1.4 H-coefficient technique

To prove the security of the four constructions, we will use the H-coefficients technique. In this section, we wish to give an overview of the H-coefficients technique [22]. This overview is based on the popular tutorial by Chen and Steinberger [16].

The setting is that of an information-theoretic adversary $\mathbf{A}$ that asks $q$ queries to one of two sets of oracles. These two sets of oracles correspond to two "worlds": the real world and the ideal world. The task of the distinguisher is to determine with which world it is interacting, i.e. to distinguish.

Typically, the oracles correspond to the building blocks of the cryptosystem under consideration. Further, there is an oracle for the construction function that is often the only difference between the two worlds. In the real world, the construction function is composed of the oracles of the underlying building blocks and mimics their interaction in the cryptosystem. In the ideal world, it is independent of the other oracles and corresponds to the primitive we want to show our cryptosystem to be indistinguishable from. In the case of block ciphers, this would be a random permutation; for stream ciphers, a random bitstream; for MACs, a random function.

The interaction of the distinguisher with the oracles is recorded in a transcript $\tau$. The transcript $\tau$ contains all the query-answer pairs of the interaction with the oracle. Without loss of generality, the adversary $\mathbf{A}$ is assumed to be deterministic and makes its final decision as a deterministic function of the transcript obtained.

The probability of obtaining a transcript is different in the two worlds. Let $\Theta_{\text{real}}$ and $\Theta_{\text{ideal}}$ denote the distribution of transcripts in the real and the ideal world, respectively. We call a transcript $\tau$ *attainable* if

$\Pr[\Theta_{\text{ideal}} = \tau] > 0$. Let $\mathcal{T}$ denote the set of all attainable transcripts, and let $\mathcal{T}^+$ denote all the transcripts $\tau \in \mathcal{T}$ where $\Pr[\Theta_{\text{ideal}} = \tau] > \Pr[\Theta_{\text{real}} = \tau]$. **A**'s distinguishing advantage is upper bounded by the statistical distance:

$$\begin{aligned}\Delta_{\mathbf{A}}(\Theta_{\text{ideal}}, \Theta_{\text{real}}) &\leq \frac{1}{2}\sum_{\tau\in\mathcal{T}} |\Pr[\Theta_{\text{ideal}} = \tau] - \Pr[\Theta_{\text{real}} = \tau]| \\ &= \sum_{\tau\in\mathcal{T}^+} (\Pr[\Theta_{\text{ideal}} = \tau] - \Pr[\Theta_{\text{real}} = \tau]) \\ &= \sum_{\tau\in\mathcal{T}^+} \Pr[\Theta_{\text{ideal}} = \tau]\cdot\left(1 - \frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]}\right) \\ &= \sum_{\tau\in\mathcal{T}} \Pr[\Theta_{\text{ideal}} = \tau]\left(1 - \min\left(1, \frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]}\right)\right).\end{aligned}$$

Some transcripts are better than others. The ratio $\Pr[\Theta_{\text{real}} = \tau]/\Pr[\Theta_{\text{ideal}} = \tau]$ may be small (bad) for some transcripts and close to 1 (good) for other transcripts. A typical proof partitions the set of attainable transcripts $\mathcal{T}$ into a set of *good* transcripts $\mathcal{T}_{\text{good}}$ and a set of *bad* transcripts $\mathcal{T}_{\text{bad}}$, i.e. $\mathcal{T} = \mathcal{T}_{\text{good}} \sqcup \mathcal{T}_{\text{bad}}$. A central idea of this method is that most transcripts are equally likely in both worlds, i.e. their ratio is close to 1, and the bad ones only occur with negligible probability. We will define values $\delta$ and $\varepsilon$ to upper bound the probability of a bad transcript occurring and to lower bound the ratio of the good transcripts, respectively. Define $\delta \in [0, 1]$ such that

$$\sum_{\tau\in\mathcal{T}_{\text{bad}}} \Pr[\Theta_{\text{real}} = \tau] \leq \delta$$

and define $\varepsilon \in [0, 1]$ such that for all $\tau \in \mathcal{T}_{\text{good}}$:

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \geq 1 - \varepsilon.$$

We then find **A**'s distinguishing advantage to be upper bounded by:

$$\Delta_{\mathbf{A}}(\Theta_{\text{ideal}}, \Theta_{\text{real}}) \leq \sum_{\tau\in\mathcal{T}_{\text{bad}}} \Pr[\Theta_{\text{ideal}} = \tau] + \sum_{\tau\in\mathcal{T}_{\text{good}}} \Pr[\Theta_{\text{ideal}} = \tau]\cdot\varepsilon \leq \delta + \varepsilon.$$

Lemma 4.1 summarizes the above and states the fundamental lemma of the H-coefficients technique.

**Lemma 4.1.** (H-coefficients technique) *Assume the set of attainable transcripts can be partitioned into two disjoint sets $\mathcal{T}_{\text{good}}$ and $\mathcal{T}_{\text{bad}}$. Further, assume that there exist $\delta, \varepsilon \in [0, 1]$ such that for any transcript $\tau \in \mathcal{T}_{\text{good}}$, it holds that*:

$$\Pr[\Theta_{\text{ideal}} \in \mathcal{T}_{\text{bad}}] \leq \delta \quad \textit{and} \quad \frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \geq 1 - \varepsilon.$$

*Then for all adversaries* **A**, *we have that the distinguishing advantage satisfies*

$$\Delta_{\mathbf{A}}(\Theta_{\text{ideal}}, \Theta_{\text{real}}) \leq \delta + \varepsilon.$$

**Remark.** Note that we did not touch upon the topic of *compatibility* of transcripts as done by Chen and Steinberger [16]. To clarify the concept, imagine an adversary that has queried a randomly drawn permutation $P$ through an oracle with input $a$ and received $b = P(a)$ as a reply. All permutations $P'$ with $P'(a) \neq b$ are certainly in the probability space for this experiment, but they are incompatible with the transcript and in particular with the query-answer pair $(a, b)$. Any permutation $P'$ with $P'(a) = b$ is a possible candidate for the random draw $P$.

Chen and Steinberger explained compatibility more thoroughly to elaborate on how to compute the ratio $\Pr[\Theta_{\text{real}} = \tau]/\Pr[\Theta_{\text{ideal}} = \tau]$ for the good transcripts. We skipped it as the good transcripts have a really nice structure in our model, and thus we do not need the concept of compatibility in Section 9.

Patarin [22] considers a keyed function $G_k$ where the adversary has observed some $a_i$ and $b_i$ with $G_k(a_i) = b_i$. $H$ is a coefficient that denotes the number of keys that are compatible with $G_k(a_i) = b_i$ for all $i \in \{1, \ldots, m\}$. Hence, the name "H-coefficients technique."

### 4.1.5 The adversarial strategy

From an attacker's point of view, the goal is to recover an internal state by obtaining a collision between an observed keystream and a generated keystream. If the keystreams collide, there is a high probability that they were generated from the same internal state. In particular, the same internal state always generates the same keystream in the real world.

In our model, the adversary does *not* have to rely on observing collisions in the keystream as it is provided the $\alpha$-values at the end of the interaction. The $\alpha$-values correspond to the internal states of the respective output bit. Hence, the adversary's goal will be to obtain a collision between the $\alpha$-values and the inputs of the $F$-queries.

If there occurs a collision between an $\alpha$-value (corresponding to an $E$-query) and an $F$-query, the corresponding outputs of the $E$-query and the $F$-query will always be identical *in the real world*, as $\mathbf{E}$ internally uses $\mathbf{P}$, $\pi$, and $\mathbf{F}$. *In the ideal world*, however, $\mathbf{E}$ is another random function independent of $\mathbf{P}$, $\pi$, and $\mathbf{F}$, and hence, colliding $\alpha$-values and $F$-query inputs only lead to the same output bit with a probability of $\frac{1}{2}$. This will be used as a distinguishing, i.e., bad, event. In general, there are two ways to obtain such a collision: (1) guessing an internal state corresponding to the $\alpha$-values and (2) guessing the secret key $k$ and deriving the $\alpha$-value from a $P$-query.

Further, for LSSK and CKEY, there may occur internal state collisions between two distinct (i.e. with different IVs) $E$-queries. In the ideal world, the output bits may be different even though the $\alpha$-values collide. For CIV and CIVK, an $\alpha$-value collision between two distinct $E$-queries may not occur as distinct IVs imply distinct $\alpha$-values.

### 4.1.6 Structure of the analysis

Using the H-coefficients technique, the bounds in all of our proofs will ultimately only be determined by $\delta$, as we will show that $\Pr[\Theta_{\text{real}} = \tau] = \Pr[\Theta_{\text{ideal}} = \tau]$, and thus, $\varepsilon = 0$. Hence, for a good transcript, the real and the ideal world are indistinguishable, so the security bounds will consist only of the bad events' probability. In Sections 5–8, we will analyse and upper bound the probability of a bad event in the ideal world for each of the four constructions. Once this has been done, the analysis of the good transcripts is identical for all constructions and is presented in Section 9.

We chose this structure because the analyses in Sections 5–8 are quite similar, so it makes sense to group them together. In addition, when analysing the good transcripts, it is necessary to know which specific bad events have been excluded in order to perform a proper analysis and obtain a good bound for $\varepsilon$. If a bad event were missed, it would appear in the good transcripts and lead to a poor bound for $\varepsilon$.

### 4.1.7 Query notation

When writing a query, we will use indexed letters like $x_P$, $k_P$, and $y_P$ to denote the adversary's choice. These need not be equal to an actual IV, key, or internal state. For example, $k_P$ denotes a *key guess* in a $P$-query by the adversary and needs not be equal to the actual secret key $k$.

# 5 Analysis of LSSK

We begin with the analysis of the conventional LSSK construction. All parts of the internal state are volatile, i.e., the non-volatile part of the cipher is empty, respectively non-existent. Therefore, the state length $\ell_s$ is equal to the volatile state length $\ell_v$, i.e., $\ell_s = \ell_v$. Further, LSSK does not define a packet length, and thus up to $2^{\ell_s}$ construction queries are possible per key-IV pair $(k, x)$. The inputs and outputs of the queries to the respective oracles can be found in Table 1. For an $E$-query $(x_E, r_E)$, the $\alpha$-values of LSSK are defined as follows:

$$\alpha_E \coloneqq P(x_E, k) \quad \text{and} \quad \alpha_E^{r_E} \coloneqq \pi^{r_E}(\alpha_E).$$

The four transcripts are defined in analogy to the oracle queries and $\alpha$-values:

$$\begin{aligned}
\tau_P &\coloneqq \{(x_P, k_P, y_P) \mid (x_P, k_P) \text{ is a } P\text{-query and } y_P \text{ its answer.}\}\\
\tau_F &\coloneqq \{(y_F, z_F) \mid y_F \text{ is an } F\text{-query and } z_F \text{ its answer.}\}\\
\tau_E &\coloneqq \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}\\
\tau_\alpha &\coloneqq \{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\}
\end{aligned}$$

From Lemma 4.1 (H-coefficients technique), Lemma 5.2 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for LSSK:

**Corollary 5.1.** *For the* LSSK *construction as defined in Section* 2.4, *where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Section* 4.1.2, *we have that for all adversaries* **A** *asking at most q queries, it holds that*:

$$\underset{\mathbf{A}}{\Delta}(O_{\text{real}}, O_{\text{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{2q^2}{2^{\ell_v} - q}.$$

## 5.1 Overview of the bad events

We will define three distinguishing events for LSSK. The first bad event covers the adversary guessing an internal state as input to the $F$-oracle that collides with an $\alpha$-value (internal state) of an $E$-query. The second bad event is similar to the first one in that the adversary is trying to create the same type of collision. This time, however, the adversary guesses the key and tries to obtain the colliding internal state through a $P$-query (if the key guess is correct). The third bad event differs. In LSSK, there may be $\alpha$-value collisions between two $E$-queries that allow for distinguishing.

In the real world, these collisions would imply identical outputs. In the ideal world, however, $E$ is independent of $F$, and thus, the collisions need not have identical outputs as a consequence. This is used for distinguishing.

## 5.2 Bad events

According to Section 4, we will define our first bad event. This event covers collisions between construction queries and output function queries. This type of collision corresponds to a correct internal state guess.

**Table 1:** Inputs and outputs of oracle queries to LSSK

| Query type | Input | Output |
|---|---|---|
| $P$ | $(x_P, k_P)$ | $y_P$ |
| $P^{-1}$ | $y_P$ | $(x_P, k_P)$ |
| $F$ | $y_F$ | $z_F \in \{0, 1\}$ |
| $E$ | $(x_E, r_E)$ | $z_E \in \{0, 1\}$ |

$\mathrm{bad}_1$: There exists an $E$-query $(x_E, r_E)$ and an $F$-query $y_F$ such that

$$\alpha_E^{r_E} = y_F.$$

Note that $P(x_E, k)$, i.e., the $\alpha$-value, is sampled at the end of the interaction.

The second bad event covers correct key guesses. Here, a correct key guess allows the adversary to obtain a valid internal state for a given IV $x_P$ after asking the corresponding $P$-query.

$\mathrm{bad}_2$: There exist a $P$-query $(x_P, k_P)$, an $E$-query $(x_P, r_E)$, and an $F$-query $\pi^{r_E}(P(x_P, k_P))$ such that

$$k_P = k.$$

In case of the LSSK construction, we also need to consider internal state collisions between two $E$-queries. Generally, the internal states of two different $E$-queries can collide. In the ideal world, however, the outputs can differ as $E$ is modelled as a random function.

$\mathrm{bad}_3$: There exist two $E$-queries $(x_E, r_E)$ and $(x'_E, r'_E)$, where $x_E \neq x'_E$, such that

$$\alpha_E^{r_E} = \alpha_E^{r'_E}.$$

**Remark.** Note that to successfully distinguish, the adversary further needs to observe $E(x_E, r_E) \neq F(y_F)$ for $\mathrm{bad}_1$, $E(x_P, r_E) \neq F(\pi^{r_E}(P(x_P, k_P)))$ for $\mathrm{bad}_2$, and $E(x_E, r_E) \neq E(x'_E, r'_E)$ for $\mathrm{bad}_3$. We omit these to simplify the analysis of the good transcripts in Section 9.

## 5.3 Bounding the bad events

In this subsection, we will present an upper bound on the bad events introduced in Section 5.2. In particular, we will show the following lemma:

**Lemma 5.2.** *The probability of a bad event occurring for* LSSK *is upper bounded by*:

$$\Pr[\Theta_{\mathrm{ideal}} \in \mathcal{T}_{\mathrm{bad}}] \leq \frac{q}{2^{\ell_{\mathrm{k}}}} + \frac{2q^2}{2^{\ell_{\mathrm{v}}} - q}.$$

**Proof.** Since we are in the ideal world, the answers to the $E$-, $F$-, and $P$-queries are independent of the secret key $k$. For simplicity, we will sample the secret key $k$ after the adversary's interaction with the oracles. Also, we will sample the values $\alpha_E$ for each query after the adversary's interaction with the oracles.

– $\mathrm{bad}_2$: We will first consider the event $\mathrm{bad}_2$. There are at most $q$ $P$-queries with at most $q$ distinct $k_P$. The secret key $k$ is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_{\mathrm{k}}}$. The probability of a collision with the secret key $k$ can therefore be upper bounded by $q/2^{\ell_{\mathrm{k}}}$, i.e. we obtain:

$$\Pr[\mathrm{bad}_2] \leq \frac{q}{2^{\ell_{\mathrm{k}}}}.$$

– $\mathrm{bad}_1$: We will now consider the event $\mathrm{bad}_1$. Since we already bounded the probability for $\mathrm{bad}_2$, we will now consider $\Pr[\mathrm{bad}_1 | \neg\, \mathrm{bad}_2]$. We decided to sample $\alpha_E$ after the adversary's interaction with the oracles. Further, since we conditioned on $\neg\mathrm{bad}_2$, the corresponding values $\alpha_E^r$ for all $E$-queries $(x_E, r)$ have not yet been sampled. The amount of $E$- and $F$-queries, respectively, can trivially be upper bounded by $q$. Hence, the amount of pairs can be upper bounded by $q^2$. As we sample the $\alpha_E$ values at the end of the interaction uniformly at random without replacement from $\{0,1\}^{\ell_{\mathrm{v}}}$, we can upper bound the collision probability by $(2^{\ell_{\mathrm{v}}} - q)^{-1}$ for each pair. Thus, we obtain:

$$\Pr[\text{bad}_1] \leq \frac{q^2}{2^{\ell_v} - q}.$$

– $\text{bad}_3$: The total amount of distinct $E$-queries can be upper bounded by $q$. Fix some $E$-query $(x_E, r_E)$ and consider the value $a_E^{r_E}$. Now consider a second $E$-query $(x'_E, r'_E)$, where the corresponding value $a_E^{r'_E}$ has not yet been sampled. The probability that $P(x'_E, k)$ maps to $\pi^{-r'_E}(a_E^{r_E})$ is upper bounded by $(2^{\ell_v} - q)^{-1}$. Over all $q^2$ pairs of $E$-queries, we obtain:

$$\Pr[\text{bad}_3] \leq \frac{q^2}{2^{\ell_v} - q}.$$

Lemma 5.2 follows from the union bound of the above individual bad events. □

# 6 Analysis of CKEY

Of the enhanced state ciphers, we will consider the CKEY construction first. Here, the non-volatile part of the cipher consists only of the secret key $k$. Accordingly, the volatile loading state consists of the IV $x$ only. CKEY does not define a packet length, and thus up to $2^{\ell_v}$ construction queries are possible per key-IV pair $(k, x)$. The inputs and outputs of the queries to the respective oracles can be found in Table 2. For an $E$-query $(x_E, r_E)$, the $a$-values of CKEY are defined as follows:

$$a_E \coloneqq P\langle k \mid x_E\rangle \quad \text{and} \quad a_E^{r_E} \coloneqq \pi^{r_E}(a_E).$$

The four transcripts are defined in analogy to the oracle queries and $a$-values:

$$\begin{aligned}
\tau_P &\coloneqq \{(x_P, k_P, y_P) \mid \langle k_P \mid x_P\rangle \text{ is a } P\text{-query and } \langle k_P \mid y_P\rangle \text{ its answer.}\}\\
\tau_F &\coloneqq \{(k_F, y_F, z_F) \mid \langle k_F \mid y_F\rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}\\
\tau_E &\coloneqq \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}\\
\tau_a &\coloneqq \{(x_E, r_E, a_E, a_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (a_E, a_E^{r_E}) \text{ its } a\text{-values.}\}
\end{aligned}$$

From Lemma 4.1 (H-coefficients technique), Lemma 6.2 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CKEY:

**Corollary 6.1.** *For the* CKEY *construction as defined in Section* 2.4, *where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Section* 4.1.2, *we have that for all adversaries* **A** *asking at most q queries, it holds that*:

$$\underset{\mathbf{A}}{\Delta}(O_{\text{real}}, O_{\text{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{q^2}{2^{\ell_v} - q}.$$

## 6.1 Overview of the bad events

We will define two distinguishing events for CKEY. Essentially, the distinguishing events are the same as for LSSK. One should note that for guessing an internal state, the adversary would have to guess the volatile

**Table 2:** Inputs and outputs of oracle queries to CKEY

| Query type | Input | Output |
|---|---|---|
| $P$ | $\langle k_P \mid x_P\rangle$ | $\langle k_P \mid y_P\rangle$ |
| $P^{-1}$ | $\langle k_P \mid y_P\rangle$ | $\langle k_P \mid x_P\rangle$ |
| $F$ | $\langle k_F \mid y_F\rangle$ | $z_F \in \{0,1\}$ |
| $E$ | $(x_E, r_E)$ | $z_E \in \{0,1\}$ |

internal state as well as the non-volatile key correctly. Therefore, it makes more sense for an adversary to guess the key and obtain the colliding internal state through a $P$-query. Hence, we only consider these as one bad event, namely, correct key guesses. As is the case for LSSK, there may be $\alpha$-value collisions between two $E$-queries that allow for distinguishing.

In the real world, these collisions would imply identical outputs. In the ideal world, however, $E$ is independent of $F$, and thus, the collisions need not have identical outputs as a consequence. This is used for distinguishing.

## 6.2 Bad events

The first distinguishing event covers a collision between a construction query and an output function query. This corresponds to an adversary correctly guessing the internal state.

$\mathrm{bad}_1$: There exists an $E$-query $(x_E, r_E)$ and an $F$-query $\langle k_F \mid y_F \rangle$ such that:

$$a_E^{r_E} = \langle k_F \mid y_F \rangle.$$

Note that in case of the CKEY construction, a prerequisite to get a proper internal state collision, the attacker needs to guess the key right. Here, we will cover all key guesses, i.e. also through $P$-queries, in the relaxed variant of $\mathrm{bad}_1$:

$\mathrm{bad}_2$: There exists a $P$-query $\langle k_P \mid x_P \rangle$ or an $F$-query $\langle k_F \mid y_F \rangle$ such that:

$$k_P = k \quad \text{or} \quad k_F = k.$$

**Remark.** Ultimately, the only way for the adversary to distinguish is to observe different outputs for identical internal states. This can only happen in the ideal world. In the case of the CKEY construction, it is, in fact, easier for the adversary to guess the key and then derive a corresponding internal state using a $P$-query than to guess an internal state for an $F$-query, as this also involves guessing the key. We, therefore, ignore $\mathrm{bad}_1$ as it is implied by $\mathrm{bad}_2$.

In the case of the CKEY construction, we also need to consider internal state collisions between two $E$-queries. Colliding internal states between two different $E$-queries with different outputs give an adversary a tool to distinguish.

$\mathrm{bad}_3$: There exist two $E$-queries $(x_E, r_E)$ and $(x'_E, r'_E)$, where $x_E \neq x'_E$ such that:

$$a_E^{r_E} = a_E^{r'_E}.$$

**Remark.** Note that to successfully distinguish, the adversary further needs to observe $E(x_E, r_E) \neq F\langle k_F \mid y_F \rangle$ for $\mathrm{bad}_1$, and $E(x_E, r_E) \neq E(x'_E, r'_E)$ for $\mathrm{bad}_3$. We omit these to simplify the analysis of the good transcripts in Section 9.

## 6.3 Bounding the bad events

In this subsection, we will present an upper bound on the bad events introduced in Section 6.2. In particular, we will show the following lemma:

**Lemma 6.2.** *The probability of a bad event occurring for* CKEY *is upper bounded by*:

$$\Pr[\Theta_{\mathrm{ideal}} \in \mathcal{T}_{\mathrm{bad}}] \leq \frac{q}{2^{\ell_k}} + \frac{q^2}{2^{\ell_v} - q}.$$

**Proof.** Since we are in the ideal world, the answers to the $E$-, $F$-, and $P$-queries are independent of the secret key $k$. For simplicity, we will sample the secret key $k$ after the adversary's interaction with the oracles. Also, we will sample the values $a_E$ for each query after the adversary's interaction with the oracles.

– $\text{bad}_2$: We will first consider the event $\text{bad}_2$. There are at most $q$ queries with at most $q$ distinct $k_P$ or $k_F$. The secret key $k$ is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_k}$. The probability of a collision with the secret key $k$ can therefore be upper bounded by:

$$\Pr[\text{bad}_2] \leq \frac{q}{2^{\ell_k}}.$$

– $\text{bad}_3$: The total amount of distinct $E$-queries can be upper bounded by $q$. Fix some $E$-query $(x_E, r_E)$ and consider the value $a_E^{r_E}$. Now consider a second $E$-query $(x_E', r_E')$ where the corresponding value $a_E^{r_E'}$ has not yet been sampled. The probability that $P\langle k \mid x_E'\rangle$ maps to $\pi^{-r_E'}(a_E^{r_E})$ is upper bounded by $(2^{\ell_v} - q)^{-1}$. Over all $q^2$ pairs of $E$-queries, we obtain:

$$\Pr[\text{bad}_3] \leq \frac{q^2}{2^{\ell_v} - q}.$$

Lemma 6.2 follows from the union bound of the above individual bad events. □

# 7 Analysis of CIV

CIV uses the IV in the non-volatile part of the cipher, whereas the key is contained in the volatile part of the loading state. A packet length is defined, and up to $\ell_p$ bits may be output per key-IV pair $(k, x)$. The inputs and outputs of the queries to the respective oracles can be found in Table 3. The $a$-values for each $E$-query $(x_E, r_E)$ are defined as follows:

$$a_E \coloneqq P\langle x_E \mid k\rangle \quad \text{and} \quad a_E^{r_E} \coloneqq \pi^{r_E}(P\langle x_E \mid k\rangle).$$

The four transcripts are defined in analogy to the oracle queries and $a$-values:

$$\tau_P \coloneqq \{(x_P, k_P, y_P) \mid \langle x_P \mid k_P\rangle \text{ is a } P\text{-query and } \langle x_P \mid y_P\rangle \text{ its answer.}\}$$
$$\tau_F \coloneqq \{(x_F, y_F, z_F) \mid \langle x_F \mid y_F\rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}$$
$$\tau_E \coloneqq \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$
$$\tau_a \coloneqq \{(x_E, r_E, a_E, a_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (a_E, a_E^{r_E}) \text{ its } a\text{-values.}\}$$

From Lemma 4.1 (H-coefficients technique), Lemma 7.2 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CIV:

**Corollary 7.1.** *For the* CIV *construction as defined in Section* 2.4, *where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Section* 4.1.2, *and up to* $\ell_p$ *adversarial queries per IV* $x \in \mathcal{IV}$, *we have that for all adversaries* $\mathbf{A}$ *asking at most q queries, it holds that*:

$$\Delta_{\mathbf{A}}(O_{\text{real}}, O_{\text{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{\ell_p \cdot q}{2^{\ell_v} - q}.$$

**Table 3:** Inputs and outputs of the oracle queries to CIV

| Query type | Input | Output |
|---|---|---|
| $P$ | $\langle x_P \mid k_P\rangle$ | $\langle x_P \mid y_P\rangle$ |
| $P^{-1}$ | $\langle x_P \mid y_P\rangle$ | $\langle x_P \mid k_P\rangle$ |
| $F$ | $\langle x_F \mid y_F\rangle$ | $z_F \in \{0,1\}$ |
| $E$ | $(x_E, r_E)$ | $z_E \in \{0,1\}$ |

## 7.1 Overview of the bad events

We will define two distinguishing events for CIV. Essentially, the distinguishing events are identical to the first two of LSSK. One distinguishing event corresponds to guessing an internal state to provoke a collision, and the other distinguishing event corresponds to guessing the secret key to then derive an internal state collision.

Internal state collisions between two $E$-queries cannot occur. As the IV $x$ is part of the non-volatile internal state, a collision between queries with different IVs is not possible. Collisions of internal states from queries with the same IV are not possible as the state update function $\pi$ is required to have a period larger than $\ell_{\mathrm{p}}$.

In the real world, these collisions would imply identical outputs. In the ideal world, however, $E$ is independent of $F$, and thus, the collisions need not have identical outputs as a consequence. This is used for distinguishing.

## 7.2 Bad events

We have to identify bad events that will trivially allow to distinguish the real world from the ideal world. There is only one way for the adversary to distinguish the real world from the ideal world: the adversary needs to recover some internal state. If the outputs of the corresponding $E$- and $F$-queries differ, then we are in the ideal world.

$\mathrm{bad}_1$: There exists an $E$-query $(x_E, r_E)$ and an $F$-query $\langle x_F \mid y_F \rangle$ such that:

$$\alpha_E^{r_E} = \langle x_F \mid y_F \rangle.$$

Note that $P\langle x_E \mid k \rangle$ is sampled at the end of the interaction.

We also define a bad event that covers key guesses. The adversary guesses a key, performs a $P$-query with its guess, and uses the then obtained internal state to produce a collision between an $E$- and an $F$-query.

$\mathrm{bad}_2$: There exists a $P$-query $\langle x_P \mid k_P \rangle$, an $E$-query $(x_P, r_E)$, and an $F$-query $\pi^{r_E}(P\langle x_P \mid k_P \rangle)$ such that:

$$k_P = k.$$

**Remark.** Note that $\mathrm{bad}_2$ represents a special case of $\mathrm{bad}_1$. In particular, we have that:

$$P\langle x_P \mid k_P \rangle = \alpha_E \quad \text{and} \quad \pi^{r_E}(P\langle x_P \mid k_P \rangle) = \alpha_E^{r_E}.$$

Ultimately, the correct key guess is used to acquire a valid internal state. That internal state is then used to check whether the corresponding $E$- and $F$-queries differ, which then allows to distinguish. Yet, for simplicity, we consider these as separate bad events. We will first bound the probability of $\mathrm{bad}_2$ and then derive a bound for $\mathrm{bad}_1$ conditioned on $\mathrm{bad}_2$ *not* occurring.

Also note that to successfully distinguish, the adversary further needs to observe $E(x_E, r_E) \neq F\langle x_F \mid y_F \rangle$ for $\mathrm{bad}_1$, and $E(x_P, r_E) \neq F\big(\pi^{r_E}(P\langle x_P \mid k_P \rangle)\big)$ for $\mathrm{bad}_2$. We omit these to simplify the analysis of the good transcripts in Section 9.

## 7.3 Bounding the bad events

In this subsection, we will present an upper bound on the bad events introduced in Section 7.2. In particular, we will show the following lemma:

**Lemma 7.2.** *The probability of a bad event occurring for* CIV *is upper bounded by*:

$$\Pr[\Theta_{\mathrm{ideal}} \in \mathcal{T}_{\mathrm{bad}}] \leq \frac{q}{2^{\ell_{\mathrm{k}}}} + \frac{\ell_{\mathrm{p}} \cdot q}{2^{\ell_{\mathrm{v}}} - q}.$$

**Proof.** Since we are in the ideal world, the answers to the $E$-, $F$-, and $P$-queries are independent of the secret key $k$. For simplicity, we will sample the secret key $k$ after the adversary's interaction with the oracles. Also, we will sample the values $\alpha_E$ for each query after the adversary's interaction with the oracles.

– $\mathrm{bad}_2$: We will first consider the event $\mathrm{bad}_2$. There are at most $q$ $P$-queries with at most $q$ distinct $k_P$. The secret key $k$ is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_\mathrm{k}}$. The probability of a collision with the secret key $k$ can therefore be upper bounded by $q/2^{\ell_\mathrm{k}}$. We obtain:

$$\Pr[\mathrm{bad}_2] \leq \frac{q}{2^{\ell_\mathrm{k}}}.$$

– $\mathrm{bad}_1$: We will now consider the event $\mathrm{bad}_1$. Since we already bounded the probability for $\mathrm{bad}_2$, we will now consider $\Pr[\mathrm{bad}_1 | \neg\, \mathrm{bad}_2]$. We decided to sample $\alpha_E$ after the adversary's interaction with the oracles. Further, since we conditioned on $\neg\mathrm{bad}_2$, the corresponding $\alpha$-values $P\langle x_E \mid k\rangle$ for all $E$-queries $(x_E, r_E)$ have not yet been sampled.
The amount of $F$-queries can trivially be upper bounded by $q$. Fix any $F$-query $\langle x_F \mid y_F\rangle$. Since the amount of $E$-queries is bounded to $\ell_\mathrm{p}$ queries per IV $x_E$, there are at most $\ell_\mathrm{p}$ $\alpha$-values to collide with.
Fix some $E$-query $(x_E, r_E)$ where $x_E = x_F$. Consider the internal state $\rho = \pi^{-r_E}\langle x_F \mid y_F\rangle$. The $\alpha$-value $P\langle x_E \mid k\rangle$ is sampled after the $F$-query. As there are at most $q$ queries, we obtain for a fixed $E$-query and a fixed $F$-query:

$$\Pr[P\langle x_E, k^{\mathrm{pre}} \mid k\rangle = \rho] \leq (2^{\ell_\mathrm{v}} - q)^{-1}.$$

Over at most $q$ $F$-queries and up to $\ell_\mathrm{p}$ matching $E$-queries per $F$-query, we obtain:

$$\Pr[\mathrm{bad}_1 | \neg\, \mathrm{bad}_2] \leq \frac{\ell_\mathrm{p} \cdot q}{2^{\ell_\mathrm{v}} - q}.$$

Lemma 7.2 follows from the union bound of the above individual bad events. □

# 8 Analysis of CIVK

CIVK uses the IV as well as a key prefix in the non-volatile part of the cipher, whereas the key is contained in the volatile part of the loading state. A packet length is defined, and up to $\ell_\mathrm{p}$ bits may be output per key-IV pair $(k, x)$. The inputs and outputs of the queries to the respective oracles can be found in Table 4. The $\alpha$-values for each $E$-query $(x_E, r_E)$ are defined as follows:

$$\alpha_E \coloneqq P\langle x_E, k^{\mathrm{pre}} \mid k\rangle \quad \text{and} \quad \alpha_E^{r_E} \coloneqq \pi^{r_E}(P\langle x_E, k^{\mathrm{pre}} \mid k\rangle).$$

The four transcripts are defined in analogy to the oracle queries and $\alpha$-values:

$$\tau_P \coloneqq \{(x_P, k_P^{\mathrm{pre}}, k_P, y_P) \mid \langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle \text{ is a } P\text{-query and } \langle x_P, k_P^{\mathrm{pre}} \mid y_P\rangle \text{ its answer.}\}$$
$$\tau_F \coloneqq \{(x_F, k_F^{\mathrm{pre}}, y_F, z_F) \mid \langle x_F, k_F^{\mathrm{pre}} \mid y_F\rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}$$
$$\tau_E \coloneqq \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$
$$\tau_\alpha \coloneqq \{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\}$$

**Table 4:** Inputs and outputs of the oracle queries to CIVK

| Query type | Input | Output |
|---|---|---|
| $P$ | $\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle$ | $\langle x_P, k_P^{\mathrm{pre}} \mid y_P\rangle$ |
| $P^{-1}$ | $\langle x_P, k_P^{\mathrm{pre}} \mid y_P\rangle$ | $\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle$ |
| $F$ | $\langle x_F, k_F^{\mathrm{pre}} \mid y_F\rangle$ | $z_F \in \{0,1\}$ |
| $E$ | $(x_E, r_E)$ | $z_E \in \{0,1\}$ |

From Lemma 4.1 (H-coefficients technique), Lemma 8.2 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CIVK:

**Corollary 8.1.** *For the* CIVK *construction as defined in Section* 2.4, *where the mixing function p and the output function f are replaced with their ideal counterparts, as defined in Section* 4.1.2, *and up to* $\ell_{\mathrm{p}}$ *adversarial queries per IV* $x \in \mathcal{IV}$, *we have that for all adversaries* $\mathbf{A}$ *asking at most q queries, it holds that*:

$$\Delta_{\mathbf{A}}(O_{\mathrm{real}}, O_{\mathrm{ideal}}) \leq \frac{q}{2^{\ell_{\mathrm{k}}}} + \frac{q}{2^{\ell_{\mathrm{v}}} - q}.$$

**Remark.** Note that there is a minor glitch in the proof of CIVK in [14]. The bad events in [14] additionally contain the condition that the outputs of the corresponding $E$- and $F$-queries differ. This means that the $a$-values and the inputs to the $F$-queries may collide in the good transcripts if the corresponding outputs of the $E$- and $F$-queries are identical. The consequence is that for the analysis of the good transcripts, one cannot say that $\mathcal{A} \cap \mathcal{F} = \emptyset$. This glitch also occurred in the submitted version of this work.

By removing the condition in the bad events that the outputs of the $E$- and $F$-queries must differ, one can say that $\mathcal{A} \cap \mathcal{F} = \emptyset$. That makes the bounds in [14] worse by a factor of 2. As asymptotic security is considered, this factor is negligible. Corollary 8.1 presents the corrected bound.

## 8.1 Overview of the bad events

Note that the IV $x$ is chosen by the adversary. There are two strategies for the adversary to obtain a collision between the $a$-values and an $F$-query input:

(1) Guess the key prefix $k^{\mathrm{pre}}$ as well as a volatile internal state $y$ correctly and ask the corresponding $F$-query $F\langle x, k^{\mathrm{pre}} \mid y\rangle$.
(2) Guess the key $k$ correctly, ask the corresponding $P$-query $P\langle x, k^{\mathrm{pre}} \mid k\rangle$ to obtain $\langle x, k^{\mathrm{pre}} \mid y\rangle$, and ask the corresponding $F$-query $F\langle x, k^{\mathrm{pre}} \mid y\rangle$.

If the outputs of the $F$-query and the output of the $E$-query corresponding to the colliding $a$-value differ, the adversary surely is in the ideal world. We will introduce two bad events that correspond to the two strategies mentioned earlier.

**Remark.** In either world, it is possible to choose $k_P^{\mathrm{pre}}$ and $k_P$ such that $k_P^{\mathrm{pre}}$ is *not* a prefix of $k_P$. As the mixing function is bijective, the corresponding internal state $\langle x_P, k_P^{\mathrm{pre}} \mid y_P\rangle$ will be invalid, i.e. it will not occur during an actual encryption using CIVK. These states cannot be used to obtain a distinguishing event. We will ignore queries of this type as they yield no advantage to the adversary.

## 8.2 Bad events

The two bad events are similar to the ones for the CIV construction presented in Section 7.2. The main difference now is that we have to account for the key prefix. The first bad event covers an internal state collision.

$\mathrm{bad}_1$: There exists an $E$-query $(x_E, r_E)$ and an $F$-query $\langle x_F, k_F^{\mathrm{pre}} \mid y_F\rangle$ such that

$$a_E^{r_E} = \langle x_F, k^{\mathrm{pre}} \mid y_F\rangle.$$

The second bad event covers key guesses from which the adversary can obtain an internal state.

$\mathrm{bad}_2$: There exists a $P$-query $\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle$, an $E$-query $(x_P, r_E)$, and an $F$-query $\pi^{r_E}(P\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle)$ such that:

$$(k_P^{\mathrm{pre}}, k_P) = (k^{\mathrm{pre}}, k).$$

**Remark.** Note that $\mathrm{bad}_2$ represents a special case of $\mathrm{bad}_1$. In particular, we have that:

$$P\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle = \alpha_E \quad \text{and} \quad \pi^{r_E}(P\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle) = \alpha_E^{r_E}.$$

Considering these as separate bad events will simplify our analysis.

Also note that to successfully distinguish, the adversary further needs to observe $E(x_E, r_E) \neq F\langle x_F, k^{\mathrm{pre}} \mid y_F\rangle$ for $\mathrm{bad}_1$, and $E(x_P, r_E) \neq F(\pi^{r_E}(P\langle x_P, k_P^{\mathrm{pre}} \mid k_P\rangle))$ for $\mathrm{bad}_2$. We omit these to simplify the analysis of the good transcripts in Section 9.

## 8.3 Bounding the bad events

In this subsection, we will present an upper bound on the bad events introduced in Section 8.2. In particular, we will show the following lemma:

**Lemma 8.2.** *The probability of a bad event occurring for* CIVK *is upper bounded by*:

$$\Pr[\Theta_{\mathrm{ideal}} \in \mathcal{T}_{\mathrm{bad}}] \leq \frac{q}{2^{\ell_{\mathrm{k}}}} + \frac{q}{2^{\ell_{\mathrm{v}}} - q}.$$

**Proof.** Since we are in the ideal world, the answers to the $E$-, $F$-, and $P$-queries are independent of the secret key $(k^{\mathrm{pre}}, k)$. For simplicity, we will sample the secret key $(k^{\mathrm{pre}}, k)$ after the adversary's interaction with the oracles. Also, we will sample the values $\alpha_E$ for each query after the adversary's interaction with the oracles.

– $\mathrm{bad}_2$: We will first consider the event $\mathrm{bad}_2$. There are at most $q$ $P$-queries with at most $q$ distinct $(k_P^{\mathrm{pre}}, k_P)$. The secret key $(k^{\mathrm{pre}}, k)$ is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_{\mathrm{k}}}$. The probability of a collision with the secret key $(k^{\mathrm{pre}}, k)$ can therefore be upper bounded by $q/2^{\ell_{\mathrm{k}}}$. We obtain:

$$\Pr[\mathrm{bad}_2] \leq \frac{q}{2^{\ell_{\mathrm{k}}}}.$$

– $\mathrm{bad}_1$: We will now consider the event $\mathrm{bad}_1$. Since we already bounded the probability for $\mathrm{bad}_2$, we will now consider $\Pr[\mathrm{bad}_1 \mid \neg\, \mathrm{bad}_2]$. We decided to sample $\alpha_E$ after the adversary's interaction with the oracles.
By conditioning on $\neg\mathrm{bad}_2$, we know that for all $x \in \mathcal{IV}$ no value $P\langle x, k^{\mathrm{pre}} \mid k\rangle$ has yet been sampled. This applies since there was no adversarial $P$-query with the correct key. This implies, as we sample the $\alpha$-values after the adversary's interaction with the oracle, no $\alpha$-value has yet been sampled.
To obtain a collision between an $F$-query and an $\alpha$-value, three conditions need to apply:
(1) The key prefix $k^{\mathrm{pre}}$ needs to be guessed correctly.
(2) The $F$-query and the $E$-query, respectively the $\alpha$-value, need to utilize the same IV $x$.
(3) A collision in the volatile state needs to occur.

We will individually bound the three conditions above. Fix any F-query, $F\langle x_F, k_F^{\mathrm{pre}} \mid y_F\rangle$.

(1) As we sample the key $k$ after the adversary's interaction with the oracles, a key prefix collision occurs with a probability of: $\Pr[k_F^{\mathrm{pre}} = k^{\mathrm{pre}}] = \ell_{\mathrm{p}}^{-1}$.

(2) There are at most $\ell_{\mathrm{p}}$ E-queries, respectively $\alpha$-values, to collide with, as the $E$-queries are limited to $\ell_{\mathrm{p}}$ queries per IV $x \in \mathcal{IV}$.

(3) Fix some $E$-query $(x_E, r_E)$, where $x_E = x_F$. Consider the internal state $\rho = \pi^{-r_E}\langle x_F, k^{\mathrm{pre}} | y_F\rangle$. The value $\alpha_E = P\langle x_E, k^{\mathrm{pre}} \mid k\rangle$ is sampled after the $F$-query. As there are at most $q$ queries, we obtain:

$$\Pr[P\langle x_E, k^{\text{pre}} \mid k\rangle = \rho] \leq \left(2^{\ell_v} - q\right)^{-1}.$$

Note that there could be $P$-queries utilizing $x_E$ and $k^{\text{pre}}$ with $k_P \neq k$, and therefore, we need to upper bound the amount of queries above by $q$. We obtain that the probability of a single fixed $F$-query to collide with an $\alpha$-value is upper bounded by:

$$\Pr[k_F^{\text{pre}} = k^{\text{pre}}] \cdot \ell_p \cdot \Pr[P\langle x_E, k^{\text{pre}} \mid k\rangle = \rho] = \frac{1}{\ell_p} \cdot \frac{\ell_p}{2^{\ell_v} - q} = \frac{1}{2^{\ell_v} - q}.$$

Summing up over at most $q$ $F$-queries, we obtain:

$$\Pr[\text{bad}_1 | \neg\, \text{bad}_2] \leq \frac{q}{2^{\ell_v} - q}.$$

Lemma 8.2 follows from the union bound of the above individual bad events. □

# 9 Good transcripts

We have upper bounded probability of the occurrence of a distinguishing event in the ideal world in the previous four sections. In this section, we will show that in the absence of a distinguishing event, the ideal construction is indistinguishable from the real construction:

**Theorem 9.1.** *For any good transcript $\tau \in \mathcal{T}_{\text{good}}$, it holds that*

$$\Pr[\Theta_{\text{real}} = \tau] = \Pr[\Theta_{\text{ideal}} = \tau].$$

**Proof.** In either world, the permutation $\mathbf{P}$ and the secret key $k$ are sampled uniformly at random, and therefore, they are trivially indistinguishable.

As the random function $\mathbf{F}$ is not sampled by $E$-queries in the ideal world, we still need to argue about the answers to the $E$- and the $F$-queries. We define $\mathcal{A}$ to be the *multiset* of all $\alpha^r$-values, and $\mathcal{F}$ as the set of all $F$-query inputs that are contained in the transcript $\tau$. Formally, $\mathcal{A}$ is defined as:

$$\mathcal{A} \coloneqq \{\pi^r(P(\text{load}(x, k)))|(x, r) \text{is an } E\text{-query}\}.$$

Remember that, in the ideal world, the output of $E$ is sampled independently of the $\alpha$-values. Consider two $E$-queries $(x, r) \neq (x', r')$ with colliding $\alpha^r$-values. $E$-queries with colliding $\alpha^r$-values correspond to elements in $\mathcal{A}$ with a multiplicity greater than 1 and thus:

$$1 = \Pr_{\text{real}}[E(x, r) = z | E(x', r') = z] \neq \Pr_{\text{ideal}}[E(x, r) = z | E(x', r') = z] = \frac{1}{2}.$$

We will now argue why, for good transcripts, each element in $\mathcal{A}$ has a multiplicity of 1, and thus, $\mathcal{A}$ is a set. For LSSK and CKEY, all values in $\mathcal{A}$ have a multiplicity of 1, as we excluded collisions between two $E$-queries with the bad event $\text{bad}_3$ in Section 5.2 for LSSK and in Section 6.2 for CKEY. For CIV and CIVK, all values in $\mathcal{A}$ have a multiplicity of 1, as all states within a packet need to be distinct due to the requirement on the packet length to have a period of at least $\ell_p$. In addition, as the entire IV is part of the non-volatile internal state, states cannot collide for two different IVs.[3]

As no collisions occur in the $\alpha^r$-values, we can say that for all $E$-queries $(x, r)$ and their corresponding $\alpha^r$-values $\alpha^r \in \mathcal{A}$, and for all $z \in \{0, 1\}$, we have:

$$\Pr_{\text{real}}[F(\alpha^r) = z] = \Pr_{\text{ideal}}[E(x, r) = z].$$

Note that, in the real world, $E(x, r) = F(\alpha^r)$.

**3** For CIV and CIVK the multiplicity of 1 will also hold true in the bad transcripts.

Collisions between $E$- and $F$-queries do not occur in the good transcripts, as these have been excluded in the bad events $\text{bad}_1$ and $\text{bad}_2$. Hence, we know that $\mathcal{A} \cap \mathcal{F} = \emptyset$. In the ideal world, the random function $\mathbf{F}$ is only evaluated on $F$-queries. In the real world, $E$-queries also evaluate $\mathbf{F}$. Since $\mathcal{A} \cap \mathcal{F} = \emptyset$, we know that for all $y \in \mathcal{F}$, the values $\mathbf{F}(y)$ are "fresh" in the real world and have not been sampled through an $E$-query. Obviously, the same applies in the ideal world as $E$-queries do not sample the random function $\mathbf{F}$.

In particular, all random draws from $\mathcal{A} \cup \mathcal{F}$ are independent and will output 0 or 1 with equal probability in either world. We can conclude that for all $E$-queries $(x, r) \in \tau_E$ and their corresponding $a^r$-values $a^r \in \mathcal{A}$, for all $y \in \mathcal{F}$, and for all $z, z' \in \{0, 1\}$:

$$\Pr_{\text{real}}[F(a^r) = z, F(y) = z'] = \Pr_{\text{ideal}}[E(x, r) = z, F(y) = z'].$$

Again, note that, in the real world, $E(x, r) = F(a^r)$. □

# 10 Conclusion

The presented bounds serve as a reference for the development of future stream cipher designs. In particular, one can see that if a high resistance against distinguishing attacks is desired, the LSSK and CKEY constructions will both need a larger volatile internal state. On the other hand, despite having a small volatile internal state, CIV and CIVK can still offer significant protection against distinguishing attacks.

**Acknowledgement:** The author would like to thank the anonymous reviewers for their valuable comments.

**Conflict of interest**: The author declares no conflicts of interest.

# References

[1] Hell M, Johansson T, Meier W. Grain - A Stream Cipher for Constrained Environments; 2006. eSTREAM: the ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.

[2] Canniére CD, Preneel B.. Trivium - Specifications; 2005. eSTREAM: the ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[3] Amin Ghafari V, Hu H. Fruit-80: A secure ultra-lightweight stream cipher for constrained environments. Entropy. 2018;20(3):180.

[4] Banik S, Caforio A, Isobe T, Liu F, Meier W, Sakamoto K, et al. Atom: a stream cipher with double key filter. IACR Trans Symmetric Cryptol. 2021(1):5–36.

[5] Armknecht F, Mikhalev V. On lightweight stream ciphers with shorter internal states. In: FSE 2015. Berlin, Heidelberg: Springer; 2015. p. 451–70.

[6] Mikhalev V, Armknecht F, Mü ller C. On ciphers that continuously access the non-volatile key. IACR ToSC. 2016;2016(2):52–79. https://dblp.org/rec/journals/tosc/MikhalevAM16.html.

[7] Aumasson JP. Serious cryptography: a practical introduction to modern encryption. San Francisco, CA 94103 USA: No Starch Press; 2017.

[8] Klein A. Stream ciphers. London: Springer; 2013.

[9] Babbage SH. Improved "exhaustive search" attacks on stream ciphers. In: European Convertion on Security and Detection 1995; 1995. p. 161–6.

[10] Golić JD. On the security of nonlinear filter generators. In: Gollmann D, editor. FSE 1996. Berlin, Heidelberg: Springer; 1996. p. 173–88. doi: 10.1007/3-540-60865-6_52.

[11] Hamann M, Krause M, Meier W, Zhang B. Design and analysis of small-state grain-like stream ciphers. Cryptography Commun. 2018;10(5):803–34.

[12] Hamann M, Krause M, Meier W. A note on stream ciphers that continuously use the IV. IACR Cryptol ePrint Archive. 2017;2017:1172.

[13] Hamann M, Krause M, Moch A. Tight security bounds for generic stream cipher constructions. In: SAC 2019. Cham: Springer; 2019. p. 335–64.

[14] Hamann M, Moch A, Krause M, Mikhalev V. The DRACO stream cipher: a power-efficient small-state stream cipher with full provable security against TMDTO attacks. IACR Trans Symmetric Cryptol. 2022;2022(2):1–42. https://tosc.iacr.org/index.php/ToSC/article/view/9712.

[15] Even S, Mansour Y. A construction of a cipher from a single pseudorandom permutation. J Cryptol. 1997;10(3):151–61.
[16] Chen S, Steinberger J. Tight security bounds for key-alternating ciphers. In: EUROCRYPT 2014. Berlin, Heidelberg: Springer; 2014. p. 327–50.
[17] Chen S, Lampe R, Lee J, Seurin Y, Steinberger J. Minimizing the two-round even-Mansour cipher. In: CRYPTO 2014. Berlin, Heidelberg: Springer; 2014. p. 39–56.
[18] Dunkelman O, Keller N, Shamir A. Minimalism in cryptography: the even-Mansour scheme revisited. In: EUROCRYPT 2012. Berlin, Heidelberg: Springer; 2012. p. 336–54.
[19] Lampe R, Patarin J, Seurin Y. An asymptotically tight security analysis of the iterated even-Mansour cipher. In: ASIACRYPT 2012. Berlin, Heidelberg: Springer; 2012. p. 278–95.
[20] Bogdanov A, Knudsen LR, Leander G, Standaert FX, Steinberger J, Tischhauser E. Key-alternating ciphers in a provable setting: encryption using a small number of public permutations. In: EUROCRYPT 2012. Berlin, Heidelberg: Springer; 2012. p. 45–62.
[21] Andreeva E, Bogdanov A, Dodis Y, Mennink B, Steinberger JP. On the indifferentiability of key-alternating ciphers. In: CRYPTO 2013. Berlin, Heidelberg: Springer; 2013. p. 531–50.
[22] Patarin J. The “coefficients H” technique. In: SAC 2008. Springer; 2008. p. 328–45.
[23] Bellare M, Rogaway P. Random oracles are practical: a paradigm for designing efficient protocols. In: Denning DE, Pyle R, Ganesan R, Sandhu RS, Ashby V, editors.CCS ’93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3–5, 1993. ACM; 1993. p. 62–73. doi: 10.1145/168588.168596.
[24] Canetti R, Goldreich O, Halevi S. The random oracle methodology, revisited. J ACM (JACM). 2004;51(4):557–94.
[25] Ågren M, Hell M, Johansson T, Meier W.. Grain-128a: a new version of grain-128 with optional authentication. IJWMC. 2011 Dec;5(1):48–59. doi: 10.1504/IJWMC.2011.044106.
[26] Babbage S, Dodd M. The stream cipher MICKEY 2.0; 2006. eSTREAM: the ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf.
[27] Institute of Electrical and Electronics Engineers. IEEE Standard for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 80211-2020 (Revision of IEEE Std 80211-2016). 2021:1–4379.
[28] Rescorla E. The transport layer security (TLS) protocol version 1.3. RFC Editor; 2018. RFC 8446. https://rfc-editor.org/rfc/rfc8446.txt.
[29] Popov A. Prohibiting RC4 cipher suites. IETF; 2015. RFC 7465 (Proposed Standard). http://www.ietf.org/rfc/rfc7465.txt.
[30] Biryukov A, Shamir A. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto T, editor. ASIACRYPT 2000. Berlin, Heidelberg: Springer; 2000. p. 1–13. doi: http://dx.doi.org/10.1007/3-540-44448-3_1.
[31] Hellman M. A cryptanalytic time-memory trade-off. IEEE Trans Inform Theory. 1980 Jul;26(4):401–6.
[32] Biryukov A, Shamir A, Wagner D. Real time cryptanalysis of A5/1 on a PC. In: FSE 2000. Berlin, Heidelberg: Springer; 2001. p. 1–18. doi: http://dx.doi.org/10.1007/3-540-44706-7_1.
[33] Hong J, Sarkar P. New applications of time memory data tradeoffs. In: Roy B, editor. ASIACRYPT 2005. Berlin, Heidelberg: Springer; 2005. p. 353–72.