MacaKey V2: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid

Charlotte Lefevre[1] and Mario Marhuenda Beltrán[1]

Radboud University, Nijmegen, The Netherlands
m.marhuenda@cs.ru.nl, charlotte.lefevre@ru.nl

Abstract. The keyed sponge construction has benefited from various efficiency advancements over time, most notably leading to the possibility to absorb over the entire state, as in the full-state keyed sponge. However, squeezing has always remained limited to blocks smaller than the permutation size, as security is determined by the capacity c, the size of the non-squeezed state. In this work, we present `MacaKey`, an improved version of the full-state keyed sponge that not only absorbs over the entire state but also squeezes *over the entire state*. The scheme combines ideas of the full-state keyed sponge with those of the summation-truncation hybrid of Gunsing and Mennink. We demonstrate that, with no sacrifice in generic security and with only using c bits of extra storage, `MacaKey` can significantly boost performance, particularly in scenarios requiring large amounts of output. For example, using the 320-bit Ascon permutation with a 256-bit capacity, `MacaKey` outputs five times as many bits as the full-state keyed sponge.

1 Introduction

The sponge construction is a permutation-based approach for cryptographic hashing. It was first introduced by Bertoni et al. [BDPV07], and it quickly gained popularity, in particular in light of the SHA-3 competition [NIS12], which was won by the Keccak sponge function [BDPV13]. More recently, the winner of the NIST lightweight cryptography competition `Ascon` [SMC+25, DEMS21, NIS19] is also sponge-based. The sponge construction operates on a state of size b bits, which is itself split into $b = r + c$, where r is the rate, and c the capacity. The leftmost r bits of the state are referred to as the *outer part*, while the rightmost c bits are called the *inner part*. To process a message, the sponge first applies an injective padding and then divides the message into blocks of r bits. Each block is added to the outer part of the state, followed by the application of a permutation. After all blocks have been processed, the digest is extracted in r-bit chunks. Shortly after its introduction, the sponge was proven to be indifferentiable [MRH04, CDMP05] from a random oracle [BDPV08]. Roughly, this result means that the sponge "behaves" like a random oracle in single-stage games [RSS11] up to $2^{c/2}$ queries. Later work proposed extensions of the sponge construction to increase the squeezing-phase rate r', as in `PHOTON` [GPP11]. A comparable indifferentiability bound was proven for these extensions, essentially

when r' does not exceed $r + c/2$ [NO14]. These bounds are tight, so that any attempt to reduce c would directly weaken the security guarantees.

However, if the sponge is used in keyed applications, the secrecy of the state allows for further optimization strategies [BDPV07,BDPV11b,BPV12,CDH+12, ADMV15]. Ultimately, this led to the full-state keyed sponge (`FSKS`) [MRV15], which absorbs over the entire state, effectively reducing the capacity during the absorption phase to 0. This construction is proven secure up to min $\left(2^c/\mathcal{M}, 2^k\right)$ permutation calls, where $\mathcal{M}$ denotes the online complexity [MRV15]. Notably, this bound can be improved by applying domain separation between the absorbing and the squeezing phases. This approach can be found in the construction underlying `Ascon-PRF` [DEMS24], with generic security proven in [Men23, Corollary 1]. From now on, we will use r and c to denote the rate and capacity during the squeezing phase.

Despite all of these improvements, the output rate of standard keyed sponges is limited to r bits at a time, with maximal generic security up to min $\left(2^c, 2^k\right)$ permutation queries (e.g., the aforementioned construction underlying `Ascon-PRF`). The only exceptions we know of are functions that include a feed-forward at the end of their evaluation, such as `Muffler` [BBN22] and `Kirby` [LBD23], or blind the state with a key, such as `Chaskey` [MMH+14]. However, those designs are limited to a fixed output length of b bits, requiring an additional mode of operation on top to extend their output length.

1.1 Contributions

In this work, we present a novel permutation-based variable-input-length and variable-output-length keyed construction, called `MacaKey`. Our design draws inspiration from fixed-length PRF constructions, particularly the summation-truncation hybrid by Gunsing and Mennink [GM20]. The core observation in their approach is that, when one considers the truncation of a secret permutation with output size of $a < b$ bits per call, one can pairwise group the evaluations and XOR the truncated parts together, allowing to output $(b - a)$ extra bits *without sacrificing security*. The authors also suggest that this method can be extended further by grouping more permutation evaluations, similar to the `CENC` construction [Iwa06].

Building on this idea, `MacaKey` integrates this summation-truncation technique into the squeezing part of `FSKS`. More precisely, the inner parts during squeezing are summed pairwise in a `CENC` fashion before being output (see Fig. 1 for an illustration). This enhancement increases the throughput to b bits per permutation call during the squeezing phase, which is larger than all previously known permutation-based variable-output-length PRFs. We provide a security bound for `MacaKey`, proving the adversary's distinguishing advantage in the ideal permutation model and in the multi-user setting. We prove generic security of `MacaKey` up to $\approx \min\left(2^c, 2^b/\mathcal{M}, 2^k\right)$ permutation calls with a tight bound.

On Earlier Versions of MacaKey. An earlier version of this work used the same permutation during the absorbing and squeezing phase. This version was at-

tacked by Chakraborty and Dhar [CD25], showing that the original scheme was insecure. The root cause is that the internal state immediately before the first squeeze was used as a mask on outputs. This lets an adversary reabsorb previously output blocks in a way that can force collisions between an intermediate state during absorption and a state during squeezing. On the proof side, the flaw comes from the distance between `RS` and `RO` (cf., Section 4.3), where it was missed that the set of absorbing and squeezing states could be non-disjoint, and the intermediate states released in the `RO` world ignored that aspect. As a result, the bad event **IColl** could trivially be set in the `RS` world. The issue can be corrected by enforcing a strong separation between the absorbing and squeezing phases. In this work, we use two different permutations during the absorbing and squeezing phases. The rationale for this choice is discussed in more detail in Section 3.1.

1.2 Comparison to Existing Permutation-Based PRFs

The security bound of `MacaKey` is of the same order as that of `FSKS` with domain separation. Concretely, this makes `MacaKey` more efficient than `FSKS` with no security cost and small overhead. Whereas this improvement has a limited impact if the output length is required to be short anyway (e.g., a tag), it becomes much more significant for applications like key-stream generation. Consider, for example, the size of the permutation underlying `Ascon` [DEMS21], i.e., $b = 320$, and take $c = 256$. Compared to `FSKS`, the throughput of `MacaKey` with those parameter sizes is increased by 400%. Moreover, compared to the variant of the sponge that absorbs r bits at a time and squeezes with a rate of $r + c/2$ bits [GPP11, NO14] (where the key is concatenated with the message before being input into the hash function), the improvement is still significant. Even in this context, `MacaKey` still provides a 66% improvement in throughput during the squeezing phase, all while supporting full-state absorption.

While our main motivation was to improve over `FSKS`, it is also relevant to compare `MacaKey` against modes of operation instantiated with the Even-Mansour construction [EM97] (`EM`). In the context of keystream generation, natural comparisons are with OFB or CTR modes [Dwo01] instantiated with `EM` (referred to as `OFB-EM` and `CTR-EM`, respectively). Although `EM`-based constructions can output the entire state, they fundamentally differ in how they handle key material and security bounds. Indeed, `EM` blinds both the input and the output with the key, and guarantees birthday-bound security *in the key size*, or in other words security is guaranteed up to approximately $2^k/\mathcal{M}$ permutation calls. In contrast, `MacaKey` (and keyed sponges in general) injects the key only at the first permutation call, and achieves linear security in the key size and birthday bound security in the capacity. For example, in a single-user scenario with a maximal online complexity $\mathcal{M}$ of 2^{64}, achieving 128-bit security with `EM` would require keys as large as 192 bits. In contrast, `MacaKey` maintains the desired security level using the same permutation size, but with a key size of 128 bits. Moreover, similarly to sponge-based PRFs, `MacaKey` natively absorbs the entire input before beginning the squeezing phase, so every bit of output depends on

the full message. This is not the case of the aforementioned EM-based modes, where an additional mode of operation is needed.

Finally, other EM-based constructions, such as SoEM [CLM19], take a fundamentally different approach by aiming to achieve security beyond the birthday bound in the permutation size. However, this comes at the cost of significantly increasing the number of permutation calls.

We summarize the aforementioned comparison in Table 1. Here, we define the throughput as

$$\frac{\#\text{bits squeezed per permutation call}}{\text{permutation size}}.$$

The comparison includes permutation-based variants with variable-length squeezing: the construction FSKS, the aforementioned improved squeezing rate variant of PHOTON, modes of operation instantiated with EM, and our construction. Note that we excluded Kirby [LBD23], Muffler [BBN22], and Chaskey [MMH+14] due to their limited output length.

1.3 Applications

The strength of MacaKey lies in its ability to produce very large output streams with only a few calls to the underlying permutation. This makes it a versatile building block in applications that need long pseudorandom data; for example, as a keystream generator, as a KDF that must derive multiple keys from a single master secret, or as the core of an authenticated-encryption scheme. In each case, MacaKey's large-output capability reduces the number of permutation invocations, improving performance in high-throughput environments.

1.4 Outline

The remainder of this paper is organized as follows. Section 2 introduces the notation, the security model and useful techniques for the proof. Section 3 describes MacaKey and provides a security bound, whose proof is given in Section 4. Finally, Section 5 concludes.

2 Preliminaries

2.1 Notation

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two sets, we denote their disjoint union by $\boldsymbol{A} \sqcup \boldsymbol{B}$. Moreover, if $\boldsymbol{S}$ is a finite set, we write $x \xleftarrow{\$} \boldsymbol{S}$ to denote that x is sampled uniformly from $\boldsymbol{S}$. We use $\emptyset$ to denote the empty set, and, abusing notation, we also use $\emptyset$ to denote the empty string.

[1] This bound has not been formally derived, but by analogy with the outer-keyed sponge construction [ADMV15,NY16,Men18], we conjecture the same sort of bound.

Design	Throughput	Security	Reference
FSKS	$\frac{r}{b}$	$\frac{\mathcal{MN}}{2^c} + \frac{\mathcal{N}}{2^k}$	[DMV17]
PHOTON	$\frac{r+c/2}{b}$	at best $\frac{\mathcal{MN}}{2^c} + \frac{\mathcal{N}}{2^k}$ [1]	[GPP11, NO14]
OFB-EM, CTR-EM	1	$\frac{\mathcal{MN}}{2^k}$	[EM97]
MacaKey	1	$\frac{\mathcal{N}}{2^k} + \frac{\mathcal{MN}}{2^b} + \frac{\mathcal{N}}{2^c}$	[This work]

Table 1: Comparison of MacaKey with previous constructions. The security bounds are expressed with the big-O notation (logarithmic factors are ignored) and in the single-user setting.

Let $n \in \mathbb{N}$ such that $n \geq 1$. We will abbreviate the set $\{1, \ldots, n\}$ as $[n]$. We denote by $\{0,1\}^n$ the set of binary strings of size n and we denote $\boldsymbol{P}(n)$ as the set of all permutations over the set $\{0,1\}^n$. Given $k < n$, we denote by $(n)_k$ the falling factorial of n of depth k, i.e., $\prod_{i=0}^{k-1}(n-i)$.

Also, we denote $\{0,1\}^* = \bigcup_{i=1}^{\infty}\{0,1\}^i$ and $\{0,1\}^\infty = \prod_{i=1}^{\infty}\{0,1\}^i$. I.e., $\{0,1\}^*$ is the set of all *finite* strings of any length, and $\{0,1\}^\infty$ is the set of all *infinite* strings. Given a string $s = (s_1, s_2, \ldots, s_n) \in \{0,1\}^*$, $n_1, n_2 \in \mathbb{N}$ such that $1 \leq n_1 \leq n_2 \leq n$, we denote $s[n_1 : n_2] = (s_{n_1}, \ldots, s_{n_2})$. Additionally, we denote $\mathbf{outer}_n(s) = s[1:n]$ and $\mathbf{inner}_n(s) = s[|s| - n + 1 : n]$. We denote by $\mathtt{pad}_b$ an injective function $\{0,1\}^* \to \left(\{0,1\}^b\right)^*$ such that the last block of every padded message is not zero. I.e., $\mathtt{pad}_b$ splits a given string in blocks of length b, in an easily reversible way. Let $\mathtt{RO}^\infty : \{0,1\}^* \to \{0,1\}^\infty$ denote a random oracle in the sense of Bellare and Rogaway [BR93]. From $\mathtt{RO}^\infty$ we build $\mathtt{RO} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$, which, on input (M, n), returns $\mathtt{RO}^\infty(M)[1:n]$. We will abuse notation and refer to RO as a random oracle.

2.2 Distinguishing Advantage

In this work we analyze the security of our constructions according to the *distinguishability framework*. More precisely, we consider an information-theoretic adversary $\mathcal{A}$ that has access to one or more oracles X; this is denoted by $\mathcal{A}^X$, and the notation $\mathcal{A}^X \Rightarrow 1$ means that $\mathcal{A}$, after interaction with X, returns 1. It is a classical fact (for a simple proof see [CS14]) that in the information-theoretic setting, adversaries can be assumed to be deterministic without loss of generality. Consider a deterministic information-theoretic adversary $\mathcal{A}$ whose goal is to distinguish two oracles X and Y. The distinguishing advantage of $\mathcal{A}$ is defined as:

$$\Delta_{\mathcal{A}}(X, Y) := \left|\mathbf{Pr}\left[\mathcal{A}^X \Rightarrow 1\right] - \mathbf{Pr}\left[\mathcal{A}^Y \Rightarrow 1\right]\right|.$$

Here, X and Y are randomized algorithms; the randomization depends on the specific scenario and for now is left implicit. The interaction with any of the two systems X or Y is summarized in a transcript τ. We denote by D_X the probability distribution of transcripts when interacting with X, and similarly, D_Y the distribution of transcripts when interacting with Y. A transcript τ is called

attainable if $\mathbf{Pr}[D_Y = \tau] > 0$, meaning that it can occur during interaction with Y. We denote by $\mathcal{T}$ the set of all attainable transcripts.

2.3 PRF Security

Let us restrict the definition in the binary setting. Let $k \in \mathbb{N}$. Consider a construction $\mathtt{F}$, parametrized further by a primitive $\mathcal{P}$, and keyed from the space $\{0,1\}^k$. $\mathtt{F}^{\mathcal{P}}$ takes as input a message $M \in \{0,1\}^*$, the digest output length $n \in \mathbb{N}$, and returns a binary string of length n, so that

$$\begin{aligned} \mathtt{F}^{\mathcal{P}} : \{0,1\}^k \times \{0,1\}^* \times \mathbb{N} &\to \{0,1\}^* \\ (K, M, n) &\to Z \in \{0,1\}^n . \end{aligned}$$

Moreover, let $\mu \in \mathbb{N}$ denote the number of users. In the multi-user setting, each user's key is sampled independently and uniformly at random. For $m \in [\mu]$, denote by $\mathtt{F}^{\mathcal{P}}_{K_m}$ the specific instance of $\mathtt{F}^{\mathcal{P}}$ of user m. The multi-user security of $\mathtt{F}$ against an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathtt{F}}(\mathcal{A}) := \Delta_{\mathcal{A}}\left(\left(\mathtt{F}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P} \; ; \; \left(\mathtt{RO}_m\right)_{m=1}^{\mu}, \mathcal{P} \right) ,$$

where $\mathtt{RO}_1, \ldots, \mathtt{RO}_\mu$ are random oracles, and $\mathcal{P}$ is sampled uniformly at random. Without loss of generality, we can assume that the adversary makes at least one construction query for every user, since otherwise we can consider the multi-user setting with a smaller number of users.

2.4 H-Coefficient Technique

In Section 3.2, we will need to use Patarin's H-Coefficient technique [Pat08]; more precisely, a revisited formulation by Chen and Steinberger [CS14], given in Lemma 1.

Lemma 1 (H-Coefficient Technique). *Consider a fixed deterministic adversary $\mathcal{A}$. Let $\mathcal{T} = \mathcal{T}_{good} \sqcup \mathcal{T}_{bad}$ be a partition of $\mathcal{T}$ into a set of good transcripts and bad transcripts. And suppose that:*

$$\begin{aligned} \frac{\mathbf{Pr}[D_X = \tau]}{\mathbf{Pr}[D_Y = \tau]} &\geq 1 - \varepsilon, \quad \forall \tau \in \mathcal{T}_{good} , \\ \mathbf{Pr}[D_Y \in \mathcal{T}_{bad}] &\leq \delta . \end{aligned}$$

Then, we have:

$$\Delta_{\mathcal{A}}(X, Y) \leq \varepsilon + \delta .$$

The label $\mathcal{T}_{good}$ is "good" in the sense that we expect the value $\frac{\mathbf{Pr}[D_X=\tau]}{\mathbf{Pr}[D_Y=\tau]}$ to be close to 1 for transcripts τ in $\mathcal{T}_{good}$. A proof of this lemma can be found in [CS14].

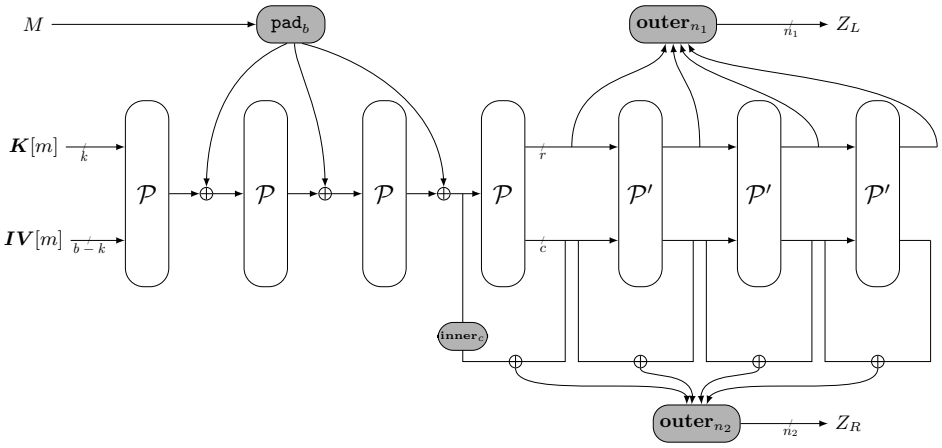


Fig. 1: The MacaKey construction. The output string is $Z_L \| Z_R$.

2.5 Multicollision Result

We will use the following multicollision result of Choi et al. [CLL19, page 187], and later slightly improved by Lefevre and Mennink [LM24].

Lemma 2. *Let $q, R \in \mathbb{N}$. Consider the experiment of throwing uniformly at random q balls in R bins. For $u \in [R]$, denote by S_u the size of the bin number u. Then, we have:*

$$\mathbb{E}\left[\max_{u \in [R]} S_u\right] \leq \frac{2q}{R} + 3\ln(R) + 4\,.$$

The proof of this lemma can be found in [LM24].

3 MacaKey Construction

3.1 Description of MacaKey

Let $k, b, r, c \in \mathbb{N}$ such that $b = r + c$ and $k \leq b$. Let $\mathcal{P}, \mathcal{P}' \in \boldsymbol{P}(b)$ be two cryptographic permutations. Moreover, let $\boldsymbol{IV}$ be a public array of μ elements from $\{0,1\}^{b-k}$, and $\boldsymbol{K}$ an array of μ elements from $\{0,1\}^k$. The construction MacaKey based on the key array $\boldsymbol{K}$, the IVs $\boldsymbol{IV}$, and the permutations $\mathcal{P}$ and $\mathcal{P}'$, takes as input an index $m \in [\mu]$, a message $M \in \{0,1\}^*$, the output length $n \in \mathbb{N}$, and it outputs a string $Z \in \{0,1\}^n$, so that:

$$\texttt{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}} : [\mu] \times \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*\,.$$

The algorithmic details of MacaKey are provided in Algorithm 1, with an illustration in Fig. 1. At a high level, MacaKey builds on the FSKS construction

by augmenting the squeezing phase through the integration of the summation-truncation hybrid technique. This is done by XORing together two consecutive inner states, and appending them to the output. This process can also be implemented in a `CENC` fashion [Iwa06], where the state before the squeezing phase is XORed with all subsequent states. Both approaches can be seen as equivalent, since the output contains the same information, i.e., the output of one can be reconstructed from the output of the other.

Using two different permutations during the absorbing and squeezing phases prevents that the adversary makes use of the collisions that Chakraborty and Dhar exploited in their attack [CD25]. Alternatively, a similar effect can be achieved by using a single permutation but reserve the lowest input bit, fixing it to 0 during absorbing and to 1 during squeezing. Another option is to employ carefully chosen non-cryptographic permutations, as done, for example, with the Merkle-Damgård construction using a permutation [HPY07], the sponge construction [Hir18], and more recently in an `Ascon-PRF`-like construction [LBM25]. Due to this strict separation during absorbing and squeezing phases, the adversary cannot overwrite the outer part of a state anymore. This improves the security, similarly to the guarantees achieved by `Ascon-PRF`.

Parametrization of r. Looking ahead (cf. Theorem 1), maximum security is reached when $r = 0$, and in that case each output block is the XOR of two consecutive states. Thus, from a purely generic security viewpoint, $r = 0$ is optimal. However, we kept the analysis general because the memory overhead is c bits, hence choosing a larger r might make sense in memory-constrained applications, or whenever the permutation size b is itself large enough to exceed the target security level.

3.2 Security of MacaKey

In this section, we provide a security bound for `MacaKey` in Theorem 1 in the model defined in Section 2.3. Moreover, the proof will be in the random permutation model, meaning that the underlying permutations are assumed to be sampled uniformly at random at the beginning of the game. Before, let us quantify the adversarial resources.

The adversary's construction queries are quantified by the minimum number of permutation evaluations needed in the real world to compute them (excluding potential bad collisions). We denote this quantity as $\mathcal{M}$. This metric is similar to the quantity M defined by Daemen et al. [DMV17], denoting the number of distinct duplexing calls. However, in our case we include in $\mathcal{M}$ the first permutation call made for each user, thus in particular $\mathcal{M} \geq \mu$.

Moreover, we will need to describe $\boldsymbol{IV}$ using two particular metrics. To facilitate this, let us represent $\boldsymbol{IV}$ with two lists, L_1 and L_2, defined as follows: L_1 contains all elements appearing in $\boldsymbol{IV}$ without any repetition. L_2, having the same length as L_1, records the *multiplicity* of each element. In other words, the s^{th} element in L_1 appears $L_2[s]$ times in $\boldsymbol{IV}$, where $s \in [|L_1|] \leq \mu$. From this,

Algorithm 1 `MacaKey` construction based on the permutations $\mathcal{P}$ and $\mathcal{P}'$.

Data: input $(m, M, n) \in [\mu] \times \{0,1\}^* \times \mathbb{N}+$
Result: output $Z \in \{0,1\}^*$

$S \leftarrow \mathcal{P}(\boldsymbol{K}[m] \parallel \boldsymbol{IV}[m])$ ▷ State of the `MacaKey` construction
$(M_1, \ldots, M_\ell) \leftarrow \mathtt{pad}_b(M)$
for $1 \le i \le \ell$ **do** ▷ Absorb
  $Y_R \leftarrow \mathbf{inner}_c(S)$ ▷ XORing base string
  $S \leftarrow \mathcal{P}(S \oplus M_i)$
end for
$Z_L \leftarrow \emptyset$ ▷ Leftmost part of output string
$Z_R \leftarrow \emptyset$ ▷ Rightmost part of output string
for $1 \le i \le \lceil \frac{n}{b} \rceil$ **do** ▷ Squeeze
  $Z_L \leftarrow Z_L \parallel \mathbf{outer}_r(S)$
  $Z_R \leftarrow Z_R \parallel (\mathbf{inner}_c(S) \oplus Y_R)$
  $Y_R \leftarrow \mathbf{inner}_c(S)$
  $S \leftarrow \mathcal{P}'(S)$
end for
$Z \leftarrow Z_L \parallel Z_R$
return $Z[1:n]$

we define:

$$u_s^{\boldsymbol{IV}} := L_2[s], \qquad u_{\max}^{\boldsymbol{IV}} := \max_s u_s^{\boldsymbol{IV}}.$$

We define the PRF advantage of `MacaKey` as:

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathtt{MacaKey}}(\boldsymbol{IV}, \mathcal{M}, \mathcal{N}) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{PRF}}_{\mathtt{MacaKey}}(\mathcal{A}),$$

where the maximum is taken over all adversaries $\mathcal{A}$ having access to $\mu = |\boldsymbol{IV}|$ instances of `MacaKey` with identifiers spread according to $\boldsymbol{IV}$, making at most $\mathcal{N}$ permutation queries (to $\mathcal{P}$ and $\mathcal{P}'$ combined), and construction queries with a cost of at most $\mathcal{M}$.

Theorem 1. *Let $b, c, r, k \in \mathbb{N}$ be such that $b = r + c$ and $k \le b$. Let $\mu, \mathcal{M}, \mathcal{N} \in \mathbb{N}$. Moreover, let $\boldsymbol{IV}$ be an array of μ elements in $\{0,1\}^{b-k}$. We have*

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathtt{MacaKey}}(\boldsymbol{IV}, \mathcal{M}, \mathcal{N}) \le \sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}} - 1)}{2^{k+1}} + \frac{(\mathcal{N} + \mathcal{M}) u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{3\mathcal{M}^2}{2^b} + \frac{4\mathcal{M}\mathcal{N}}{2^b} + \frac{(9r+12)\mathcal{N}}{2^c}.$$

Interpretation of the Bound. At a high level, the bound is of the form:

$$\tilde{\mathcal{O}}\left(\sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}} - 1)}{2^{k+1}} + \frac{(\mathcal{N} + \mathcal{M}) u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{\mathcal{M}^2}{2^b} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c} \right).$$

Assuming further that $\mathcal{M} \leq \mathcal{N}$, the expression simplifies to

$$\tilde{\mathcal{O}}\left(\frac{\mathcal{N} \cdot u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c}\right).$$

We now examine how the choice of IVs, which is application-dependent, affects security. In the worst case the adversary selects them, and their best strategy is to set all IVs to be identical. In this scenario, the bound becomes:

$$\mathbf{Adv}^{\mathrm{PRF}}_{\texttt{MacaKey}}(\boldsymbol{IV}, \mathcal{M}, \mathcal{N}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N} \cdot \mu}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c}\right).$$

On the other side, if the IVs are randomly chosen, we use Lemma 2, and get:

$$\mathbb{E}\left[u_{\max}^{\boldsymbol{IV}}\right] \leq \frac{2\mu}{2^{b-k}} + 3(b-k) + 4\,.$$

This gives

$$\mathbf{Adv}^{\mathrm{PRF}}_{\texttt{MacaKey}}(\boldsymbol{IV}, \mathcal{M}, \mathcal{N}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c}\right),$$

which is of the same order as the best-case scenario when there is one unique IV per user. Therefore, IV selection influences the minimal required key size: moving from the best-case to the worst-case scenario requires an increase of approximately $\log_2(\mu)$ bits.

Assuming that all IVs are distinct, `MacaKey` is secure as long as $\mathcal{N} \ll \min\left\{2^k, 2^c, 2^b/\mathcal{M}\right\}$. In that case, assuming $\mathcal{M} \ll 2^{64}$, if we take $b = 192$, $k = 128$, and $c = 128$, we obtain a throughput of 192 bits per permutation call during the squeezing phase with 128 bits of security.

In summary, `MacaKey` can be seen as an improved version of `FSKS` that preserves security while overcoming the b-bit output limitation of existing constructions [MMH+14, BBN22, LBD23].

4 Proof of Theorem 1

In this section, we bound the advantage of an adversary in the distinguishing game from Section 2.3. In order to do that, we follow the approach of Daemen et al. [DMV17] by introducing an idealized version of `MacaKey` named `RS` in Section 4.1. Bounding the distinguishing advantage between `RS` and `RO` is more involved than in [DMV17], and requires an original proof. This added complexity comes from the unique feature of `MacaKey` that outputs the XOR of two consecutive inner parts, introducing additional constraints during the combinatorial analysis of the transcripts in the ideal world. The distance is split in Section 4.2, and the two distances are themselves analyzed in respectively Sections 4.3 and 4.4.

Algorithm 2 Description of $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ based on the injection φ and the permutations λ and Λ.

Data: input $(m, M, n) \in [\mu] \times \{0,1\}^* \times \mathbb{N}$
Result: output $Z \in \{0,1\}^*$

$S \leftarrow \varphi(m)$ ▷ State of the construction
$(M_1, \ldots, M_\ell) \leftarrow \mathtt{pad}_b(M)$
for $1 \leq i \leq \ell$ **do** ▷ Absorb
  $Y_R \leftarrow \mathbf{inner}_c(S)$ ▷ XORing base string
  $S \leftarrow \lambda(S \oplus M_i)$
end for
$Z_L \leftarrow \emptyset$ ▷ Leftmost part of output string
$Z_R \leftarrow \emptyset$ ▷ Rightmost part of output string
for $1 \leq i \leq \lceil \frac{n}{b} \rceil$ **do** ▷ Squeeze
  $Z_L \leftarrow Z \parallel \mathbf{outer}_r(S)$
  $Z_R \leftarrow Z_R \parallel (\mathbf{inner}_c(S) \oplus Y_R)$
  $Y_R \leftarrow \mathbf{inner}_c(S)$
  $S \leftarrow \Lambda(S)$
end for
$Z \leftarrow Z_L \parallel Z_R$
return $Z[1:n]$

Let $\mathcal{A}$ be any information-theoretic adversary that has access to either the real world, i.e., $\left(\mathtt{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}, \mathcal{P}'\right)$, or the ideal world, i.e., $(\mathtt{RO}, \mathcal{P}, \mathcal{P}')$. Without loss of generality, we assume that the messages queried to the construction are already padded, i.e., they are b-bit tuples that correspond to a valid padding. Any query that does not meet this criterion will be rejected by the construction oracles. We also assume w.l.o.g. that $\mathcal{A}$ requests outputs of size a multiple of b bits, as this does not increase its query complexity.

4.1 Randomized Sponge Construction

Introducing a hybrid algorithm that lies in-between `MacaKey` and `RO` will greatly simplify our analysis: the full-state randomized sponge (`RS`) construction.

`RS` behaves similarly to `MacaKey` with the exception that calls to $\mathcal{P}$ and $\mathcal{P}'$ have been replaced by three primitives: φ, λ, and Λ. φ is a uniformly-random injective mapping: $\varphi : [\mu] \to \{0,1\}^b$ that replaces the initialization calls to $\mathcal{P}$. λ and Λ are random b-bit permutations that replace respectively $\mathcal{P}$ and $\mathcal{P}'$ in the remainder of the calls.

φ, λ, and Λ are secret primitives, i.e., $\mathcal{A}$ cannot query them. Refer to Algorithm 2 for an algorithmic description of `RS` and the leftmost picture of Fig. 2 for an illustration.

4.2 Distance Splitting

By the triangle inequality, we have:

$$\begin{aligned}&\Delta_{\mathcal{A}}(\mathtt{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}, \mathcal{P}, \mathcal{P}' \;;\; \mathtt{RO}, \mathcal{P}, \mathcal{P}') \\ &\leq \underbrace{\Delta_{\mathcal{A}}(\mathtt{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}, \mathcal{P}, \mathcal{P}' \;;\; \mathtt{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}')}_{\text{(1.A)}} + \underbrace{\Delta_{\mathcal{A}}(\mathtt{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}' \;;\; \mathtt{RO}, \mathcal{P}, \mathcal{P}')}_{\text{(1.B)}},\end{aligned} \tag{1}$$

where $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ denotes the construction from Section 4.1 based on a random injection function φ and random permutations λ and Λ. At a high level $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ behaves like $\mathtt{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}$, but online calls to $\mathcal{P}$ and $\mathcal{P}'$ have been replaced by calls to secret primitives.

Furthermore, we can simplify (1.B) by noting that both $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ and $\mathtt{RO}$ do not depend on $\mathcal{P}$. Hence:

$$\Delta_{\mathcal{A}}(\mathtt{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}' \;;\; \mathtt{RO}, \mathcal{P}, \mathcal{P}') \leq \Delta_{\mathcal{B}}(\mathtt{RS}^{\varphi,\lambda,\Lambda} \;;\; \mathtt{RO}), \tag{2}$$

where $\mathcal{B}$ is an adversary with the same resources as $\mathcal{A}$, except with no access to $\mathcal{P}$ and $\mathcal{P}'$.

It remains to bound (2) and (1.A). This follows from Lemmas 3 and 4, whose proofs will be given in Sections 4.3 and 4.4, respectively.

Lemma 3. *For any adversary $\mathcal{A}$, with resources as in Section 3.2, but **no access** to $\mathcal{P}$ and $\mathcal{P}'$, the following holds:*

$$\Delta_{\mathcal{A}}(\mathtt{RS}^{\varphi,\lambda,\Lambda} \;;\; \mathtt{RO}) \leq \frac{\mathcal{M}^2}{2^b}. \tag{3}$$

Lemma 4. *For any adversary $\mathcal{A}$, with resources as in Section 3.2, the following holds:*

$$\begin{aligned}&\Delta_{\mathcal{A}}(\mathtt{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}, \mathcal{P}, \mathcal{P}' \;;\; \mathtt{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}') \leq \\ &\sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} + \frac{4\mathcal{M}\mathcal{N}}{2^b} + \frac{(9r+12)\mathcal{N}}{2^c}.\end{aligned} \tag{4}$$

Noticing that $\mu \leq \mathcal{M}$ we simplify (3) and (4) using that:

$$\frac{\mathcal{M}^2}{2^b} + \frac{2\mathcal{M}\mu}{2^b} \leq \frac{3\mathcal{M}^2}{2^b}. \tag{5}$$

Finally, combining (2) to (5) together with (1) ends the proof. □

4.3 Distance Between RS and RO

In this section, we bound the advantage of distinguishing between the randomized sponge $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ and a random oracle $\mathtt{RO}$ (bound (3) of Lemma 3). Therefore, we consider the adversary $\mathcal{A}$ described in Lemma 3. The distinguishing experiment is described in Fig. 2.

For the derivation we rely in the H-coefficient technique of Lemma 1. That means that we must first find a suitable description of the transcripts.

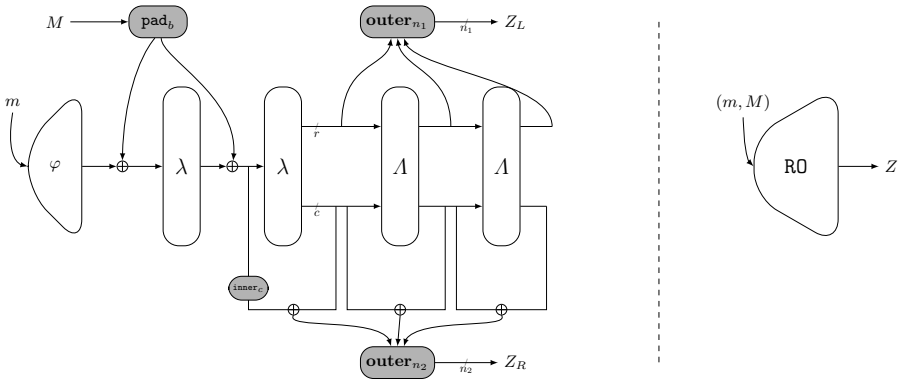


Fig. 2: Distinguishing experiment between $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ and $\mathtt{RO}$.

Counting the Queries Made by $\mathcal{A}$. We denote by $\mathcal{Q}$ the number of queries made by $\mathcal{A}$ to its oracle, *without doubly counting queries with repeated user and message* (cf. examples 1 and 2). We note that $\mathcal{Q}$ depends on the specific behavior of $\mathcal{A}$, while $\mathcal{M}$ is a fixed resource of $\mathcal{A}$. Nevertheless, it always holds that $\mathcal{Q} \leq \mathcal{M}$.

Example 1. For simplicity, in the examples of this section we use the 10^* padding function. Suppose $\mathcal{A}$ makes only the following two queries in the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world:

$$\begin{aligned}(m_1, M_1, n_1) &= (1, 0^{b-1}\|1, b), \text{ and}\\ (m_1, M_2, n_2) &= (1, 0^{b-1}\|1, 2b)\,.\end{aligned} \tag{6}$$

To compute the first query, it is necessary to compute $\varphi(1)$ and $\lambda\left(\varphi(1) \oplus 0^{b-1}\|1\right)$. To compute the second query, we need to compute the aforementioned values, together with $\Lambda \circ \lambda\left(\varphi(1) \oplus 0^{b-1}\|1\right)$. So in this case $\mathcal{M} = 3$ and $\mathcal{Q} = 1$. ♣

Example 2. Suppose $\mathcal{A}$ makes only the following two queries in the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world:

$$\begin{aligned}(m_1, M_1, n_1) &= (1, 0^{b-1}\|1, b), \text{ and}\\ (m_1, M_2, n_2) &= (1, 1^{b}, b)\,.\end{aligned} \tag{7}$$

To compute the first query, we proceed as in example 1. To compute the second query, we need to compute $\varphi(1)$ and $\lambda\left(\varphi(1) \oplus 1^{b}\right)$. So in this case we have $\mathcal{M} = 3$, but $\mathcal{Q} = 2$. ♣

Description of Transcripts. In the setting we consider in this section, $\mathcal{A}$ has access to a single oracle, either $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ or $\mathtt{RO}$. During the interaction $\mathcal{A}$ makes

queries of the form (m, M, n), where m is the index of the user making the query, n is the length of the output in bits and M is the message queried and it gets a response $Z = Z_L \| Z_R \in \{0,1\}^n$, where Z is the output string of length n, Z_L gathers the part of Z originating from truncation, while Z_R gathers the part originating from summation (cf. Fig. 2).

To simplify our analysis, at the end of the interaction between $\mathcal{A}$ and the oracles, but before $\mathcal{A}$ outputs its decision, we disclose additional information $T_1, \ldots, T_{\mathcal{Q}}$, in the following manner:

- In the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world, $T_i = \left(t_i^0, t_i^1, \ldots\right) \in (\{0,1\}^b)^*$, for $i \in [\mathcal{Q}]$, consists of the outputs of φ, λ, and Λ generated while answering the i-th query of $\mathcal{A}$. We do not include in the transcript any value t_i^j that is generated from a repeated evaluation to φ, λ, or Λ (cf., example 3).
- In the $\mathtt{RO}$ world, we will specify a procedure to generate t_i^j, in such a way that they are uniformly random and match the intrinsic properties of the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world.

Before we describe how the transcript is generated in the $\mathtt{RO}$ world we need to introduce some necessary concepts.

For a given i-th query, we can split T_i into two tuples: T_i^a, made of the values that are generated during the absorb phase, and T_i^b, made of the values that are generated during the squeeze phase. We denote $\theta(i) = |T_i^a|$, and $\chi(i) = |T_i| - 1$, so that:

$$\begin{aligned} T_i^a &= \left(t_i^0, t_i^1, \ldots, t_i^{\theta(i)-1}\right) , \\ T_i^b &= \left(t_i^{\theta(i)}, \ldots, t_i^{\chi(i)}\right) . \end{aligned} \tag{8}$$

The following example serves to clarify this concept.

Example 3. Suppose $\mathcal{A}$ makes the same queries as in example 2 in the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world:

$$\begin{aligned} (m_1, M_1, n_1) &= (1, 0^{b-1} \| 1, b), \text{ and} \\ (m_1, M_2, n_2) &= (1, 1^b, b) . \end{aligned}$$

For the first query, we have $T_1 = \left(\varphi(1), \lambda(\varphi(1) \oplus 0^{b-1} \| 1)\right)$ and for the second query, we must compute $\varphi(1)$ and $\lambda(\varphi(1) \oplus 1^b)$. However, since $\varphi(1)$ has already been computed when the first query was made, we simply have $T_2 = \left(\lambda(\varphi(1) \oplus 1^b)\right)$.

Therefore, we have: $T_1^a = (\varphi(1))$ and $T_1^b = \left(\lambda(\varphi(1) \oplus 0^{b-1} \| 1)\right)$, while $T_2^a = \emptyset$ and $T_2^b = \left(\lambda(\varphi(1) \oplus 1^b)\right)$. ♣

Putting all together, we describe the extended transcript[2] τ as an ordered list consisting of tuples:

$$\tau = (m_i, n_i, M_i, T_i)_{i=1}^{\mathcal{Q}} ,$$

[2] Henceforth we will simply call it the transcript.

where $m_i \in [\mu]$ is the index of the user making the query, $n_i \in \mathbb{N}$ is the length of the output in bits, M_i is the message queried and T_i is as in (8). For $i \in [\mathcal{Q}]$, $j \in [\theta(i)]$, we define $M_i^j \in \{0,1\}^b$ to be the message block absorbed before obtaining the state t_i^j. Finally, for $j > 0$ we define

$$\mathbf{inSt}(i,j) = \begin{cases} M_i^j \oplus t_i^{j-1} & \text{if } j \le \theta(i) \text{ and } t_i^j \text{ appears in } \tau \\ \emptyset & \text{otherwise}. \end{cases}$$

In other words, $\mathbf{inSt}(i,j)$ is defined only when the j^{th} permutation evaluation during query i is not redundant and occurs during the absorbing phase, in which case it represents the input to the corresponding λ-evaluation.

Common Maximal Prefixes. Before we explain how to generate transcripts in the RO world, we need one last concept. A query (m, M, ν) is said to share a *common maximal prefix* with an *earlier query* (m', M', ν') whenever $m = m'$ and there exists $M_1, M_2 \in \left(\{0,1\}^b\right)^*$, $M_B \in \{0,1\}^b$ such that $M = M_1 \| M_B$ and $M' = M_1 \| M_2$. In other words, M deprived of its last b-bit block is a prefix of M'.

Example 4. Suppose that $\mathcal{A}$ makes two queries $(1, M_1 \| M_2, b)$ and $(1, M_1 \| M_3, b)$, where $M_1, M_2, M_3 \in \{0,1\}^b$ and M_2 is distinct from M_3. Then $\theta(1) = 2, \theta(2) = 0$ and *a fortiori* $T_2^a = \emptyset$. This is because the second query only differs from the first one in the last b-bit block. This means that the second query simply makes the same absorb calls as the first one, hence all the intermediate states from this phase are repeated and therefore discarded. See Fig. 3 for reference. More generally, it is easy to see that for the i-th query, (m_i, n_i, M_i, T_i), we have that $T_i^a = \emptyset$ if and only if there is a previous query for the same user sharing a *maximal common prefix* with M_i. ♣

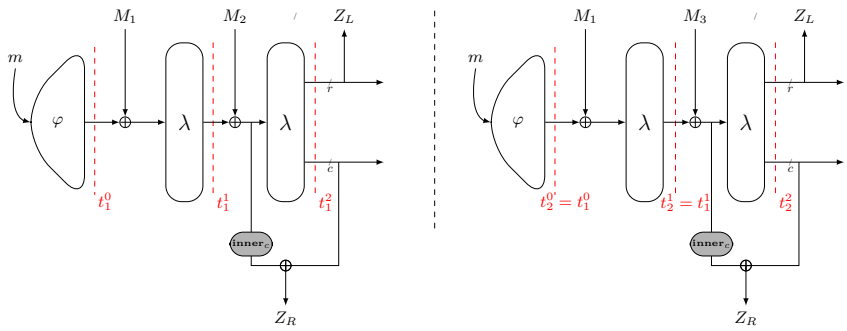


Fig. 3: Comparison of queries sharing a *common maximal prefix*.

Following example 4, we define the common prefix function, $\mathtt{CP} : [\mathcal{Q}] \to \{0,1\}^b$ as:

$$\mathtt{CP}(i) = \begin{cases} t_i^{\theta(i)-1}, & \text{if } \theta(i) \geq 1\,, \\ t_i^{\theta(i')-1}, & \text{otherwise, where } i' \text{ is the earliest query made} \\ & \text{sharing a common maximal prefix with query i.} \end{cases} \tag{9}$$

Generating Transcripts in the RO *World.* In order to maintain consistency with the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world, the transcripts in the RO world must satisfy certain constraints.

Fix any $i \in [\mathcal{Q}]$, let Z_L^i and Z_R^i be respectively the response to the query generated via truncation and via summation (cf. Fig. 1). We split Z_L^i and Z_R^i into strings of b-bits: $Z_L^{i,\theta(i)} \| \cdots \| Z_L^{i,\chi(i)}$ and $Z_R^{i,\theta(i)} \| \cdots \| Z_R^{i,\chi(i)}$, respectively.

We now consider the consistency equations, i.e., the equations that we must follow when generating the transcript in the RO world:

$$\begin{cases} Z_L^{i,j} &= \mathbf{outer}_r\left(t_i^j\right)\,, & \forall\, \theta(i) \leq j \leq \chi(i)\,, \\ Z_R^{i,\theta(i)} &= \mathbf{inner}_c\left(\mathtt{CP}(i)\right) \oplus \mathbf{inner}_c\left(t_i^{\theta(i)}\right) \oplus \mathbf{inner}_c\left(M_i^{\theta(i)}\right)\,, & \\ Z_R^{i,j} &= \mathbf{inner}_c\left(t_i^{j-1}\right) \oplus \mathbf{inner}_c\left(t_i^j\right)\,, & \forall\, \theta(i)+1 \leq j \leq \chi(i)\,. \end{cases} \tag{10}$$

Every choice of $\mathtt{CP}(i)$ fixes all the $t_{i'}^j$ such that $\mathtt{CP}(i) = \mathtt{CP}(i')$. Hence, we choose $\mathtt{CP}(i) \xleftarrow{\$} \{0,1\}^b$, for every query i that does not share a *common maximal prefix* with an earlier query. I.e., to solve (10), we choose $\mathtt{CP}(i)$ uniformly at random, for every query with different *common maximal prefix*. We now describe how we generate $T_1, \ldots, T_{\mathcal{Q}}$ in the RO world.

- Firstly, we consider the case where $j < \theta(i) - 1$. That is, the values which in the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world t_i^j would correspond to an intermediate state during the absorb phase, but not in the last permutation call of the absorb phase. Since in this case t_i^j does not appear in (10), we sample it uniformly at random i.e.:
$$t_i^j \xleftarrow{\$} \{0,1\}^b\,.$$
- Secondly, in the case where $j = \theta(i) - 1$, $\mathbf{outer}_r\left(t_i^j\right)$ is sampled uniformly at random, since it is not constrained by (10). As for the inner part, by (10) we have: $\mathbf{inner}_c\left(t_i^j\right) = \mathbf{inner}_c\left(\mathtt{CP}(i)\right)$. If $j = \theta(i) - 1$, then only $\mathbf{inner}_c\left(t_i^j\right)$ is constrained by (10). Therefore, we compute t_i^j as follows:
$$s \xleftarrow{\$} \{0,1\}^r\,, \quad t_i^j \leftarrow s \| \mathbf{inner}_c\left(\mathtt{CP}(i)\right)\,;$$
- In the remaining case: $j \geq \theta(i)$, we have that t_i^j is completely determined by (10):
  - First of all, we have $\mathbf{outer}_r\left(t_i^j\right) = Z_R^{i,j}$.
  - Lastly, we have $\mathbf{inner}_c\left(t_i^j\right) = \mathbf{inner}_c\left(t_i^{j-1}\right) \oplus Z_R^{i,j}$, which we can solve iteratively.

Definition of Good and Bad Transcripts. Let $\mathcal{T}$ be the set of all attainable transcripts, i.e., all transcripts that occur with positive probability. We partition $\mathcal{T}$ into two sets: $\mathcal{T} = \mathcal{T}_{good} \sqcup \mathcal{T}_{bad}$ which we define below.

- We say that a transcript τ is in $\mathcal{T}_{bad}$ if it triggers the bad event **IColl** defined as follows:

$$\begin{aligned}
\mathbf{IColl}_\lambda : \exists i, i', j, j', \in \mathbb{N}, &\left(m_i, n_i, M_i, \left(t_i^j\right)_{j=0}^{\chi(i)}\right), \left(m_{i'}, n_{i'}, M_{i'}, \left(t_{i'}^{j'}\right)_{j=0}^{\chi(i')}\right) \in \tau \\
&\text{with } (i,j) \neq (i',j'),\ j \leq \theta(i),\ j' \leq \theta(i'), \\
&\text{and either } t_i^j = t_{i'}^{j'} \\
&\text{or } j, j' > 0,\ \mathbf{inSt}(i,j),\ \mathbf{inSt}(i',j') \neq \varepsilon, \text{ and } \mathbf{inSt}(i,j) = \mathbf{inSt}(i',j'), \\
\mathbf{IColl}_\Lambda : \exists i, i', j, j', \in \mathbb{N}, &\left(m_i, n_i, M_i, \left(t_i^j\right)_{j=0}^{\chi(i)}\right), \left(m_{i'}, n_{i'}, M_{i'}, \left(t_{i'}^{j'}\right)_{j=0}^{\chi(i')}\right) \in \tau \\
&\text{with } (i,j) \neq (i',j'),\ j \geq \theta(i),\ j' \geq \theta(i'), \text{ and } t_i^j = t_{i'}^{j'}, \\
\mathbf{IColl} : &\mathbf{IColl}_\lambda \vee \mathbf{IColl}_\Lambda .
\end{aligned}$$

  In words, $\mathbf{IColl}_\lambda$ (resp., $\mathbf{IColl}_\Lambda$) is set whenever two inputs/outputs to λ (resp., Λ) collide together. Note that cross-collisions between λ and Λ do not set **IColl**.
- We define $\mathcal{T}_{good} = \mathcal{T} - \mathcal{T}_{bad}$ to be the set of good transcripts.

Naturally, we say τ is *bad* or *good* if $\tau \in \mathcal{T}_{bad}$ or $\tau \in \mathcal{T}_{good}$, respectively.

Bounding the Probability of Good Transcripts. Consider any $\tau \in \mathcal{T}_{good}$:

$$\tau = \left\{ \left(m_i, n_i, M_i, \left(t_i^j\right)_{j=1}^{\chi(i)}\right) \right\}_{i=1}^{\mathcal{Q}} \in \mathcal{T}_{good} .$$

Fix any query $i \in [\mathcal{Q}]$. To begin, we consider the case $\mathcal{A}$ is in the RO world. By definition, the intermediate states t_i^j are generated as follows:

- If $j < \theta(i) - 1$, then t_i^j is uniformly generated;
- If $j = \theta(i) - 1$, then necessarily the query does not share a common maximal prefix with an earlier query, hence t_i^j is also uniformly generated;
- Else, necessarily $j \geq \theta(i)$. First note that the output string $Z^i = Z_L^i \| Z_R^i$ is generated from a RO call, whose entries are never accessed twice. Hence, $\mathbf{outer}_r\left(t_i^j\right) = Z_L^{i,j}$ and $\mathbf{inner}_c\left(t_i^j\right) = \mathbf{inner}_c\left(t_i^{j-1}\right) \oplus Z_R^{i,j}$ are uniformly random and independent.

Therefore, every t_i^j is uniformly distributed and independent of the other values. Finally, we have $\mathcal{M}$ different (i,j) tuples in τ. Hence,

$$\mathbf{Pr}\left[D_{\mathtt{RO}} = \tau\right] = \frac{1}{2^{b\mathcal{M}}} . \tag{11}$$

Consider the case where $\mathcal{A}$ is placed in the $\mathtt{RS}^{\varphi,\lambda,\Lambda}$ world. The absorbing and squeezing states are disjoint, so the outer part of these states cannot be overwritten with an adversary-chosen value, nor can collisions be forced. Hence, τ is attainable. Each state value t_i^j in τ is equal to an evaluation of φ, λ, or Λ. Furthermore, precisely μ of these values come from evaluations of φ on the user indexes, so they are sampled uniformly at *random without replacement*. The remaining $\mathcal{M}-\mu$ come from evaluations of λ and Λ, and are also sampled uniformly at *random without replacement*. If we split these into $\mathcal{M}_1$ evaluations to λ and $\mathcal{M}_2$ evaluations to Λ, we obtain

$$\mathbf{Pr}\left[D_{\mathtt{RS}^{\varphi,\lambda,\Lambda}}=\tau\right]=\frac{1}{(2^b)_\mu(2^b)_{\mathcal{M}_1}(2^b)_{\mathcal{M}_2}}\,, \tag{12}$$

where $\mathcal{M}_1+\mathcal{M}_2=\mathcal{M}-\mu$. Hence, from (11) and (12), we get: $\mathbf{Pr}\left[D_{\mathtt{RS}^{\varphi,\lambda,\Lambda}}=\tau\right]\geq \mathbf{Pr}\left[D_{\mathtt{RO}}=\tau\right]$, or equivalently:

$$\frac{\mathbf{Pr}\left[D_{\mathtt{RS}^{\varphi,\lambda,\Lambda}}=\tau\right]}{\mathbf{Pr}\left[D_{\mathtt{RO}}=\tau\right]}\geq 1\,. \tag{13}$$

Bounding the Probability of Bad Transcripts. Let $\tau\in\mathcal{T}_{bad}$, $i,i'\in[\mathcal{Q}]$, $j,j'\in\mathbb{N}$ with $(i,j)\neq(i',j')$ and either ($j\leq\theta(i)$ and $j'\leq\theta(i')$) or ($j\geq\theta(i)$ and $j'\geq\theta(i')$). The probability that $t_i^{j'}$ and t_i^j collide (or that $\mathbf{inSt}(i,j)=\mathbf{inSt}(i',j')\neq\emptyset$) is at most $\frac{1}{2^b}$. There are at most $2\binom{\mathcal{M}}{2}\leq\mathcal{M}^2$ such possible colliding pairs. Therefore,

$$\mathbf{Pr}\left[D_{\mathtt{RO}}\in\mathcal{T}_{bad}\right]\leq\frac{\mathcal{M}^2}{2^b}\,. \tag{14}$$

Combining (13) and (14) with Lemma 1 finishes the proof for Lemma 3. □

4.4 Distance Between MacaKey and RS

In this section, we prove Lemma 4, i.e., the distance (1.A), which is the statistical distance between $\mathtt{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}$ and $\mathtt{RS}^{\varphi,\lambda,\Lambda}$, using again the H-coefficient technique of Lemma 1. To do this, we will build upon the technique of Daemen et al. [DMV17], while accounting for the additional complexity introduced by XORing with the extra c inner bits, which requires a subtler analysis.

For simplicity, we will denote the $\mathtt{RS}$ world $\mathtt{O}_{\mathrm{RS}}$ and the $\mathtt{MacaKey}$ world by $\mathtt{O}_{\mathrm{MK}}$. Consider the distinguisher $\mathcal{A}$ as in Lemma 4. We will focus first on $\mathtt{O}_{\mathrm{RS}}$. Denote by $\mathcal{M}_1$ the online complexity induced by evaluations to λ (excluding the first initialization call), and $\mathcal{M}_2$ induced by evaluations to Λ, so that $\mathcal{M}_1+\mathcal{M}_2=\mathcal{M}-\mu$. Moreover, we split $\mathcal{N}$ as $\mathcal{N}_1+\mathcal{N}_2$, where $\mathcal{N}_1$ (resp., $\mathcal{N}_2$) records the number of queries to $\mathcal{P}$ (resp., $\mathcal{P}'$). We split the transcript into the queries made to φ, λ, Λ, $\mathcal{P}$, and $\mathcal{P}'$. Note that $\mathcal{A}$ does not have access to φ, λ, and Λ, except through queries to $\mathtt{RS}$.

– τ_φ contains the initialization part of construction queries. That is:

$$\tau_\varphi=(m,I_m)_{i=1}^{\mu}\,.$$

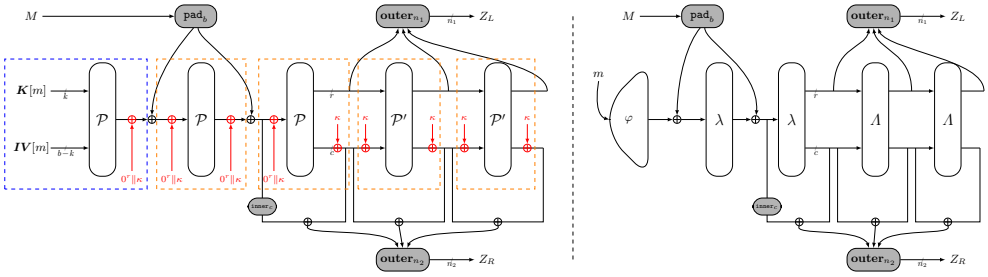


Fig. 4: Distinguishing experiment between $\texttt{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}$ and $\texttt{RS}^{\varphi,\lambda,\Lambda}$. The leftmost blue dashed box indicates $\texttt{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, while the others in dashed orange represent $\texttt{EM}^{\mathcal{P}}_{\kappa}$ and $\texttt{EM}^{\mathcal{P}'}_{\kappa}$. Besides construction queries, the adversary has additionally oracle access to $\mathcal{P}$ and $\mathcal{P}'$.

– τ_λ contains the input-output pairs to λ made during the construction queries. That is:

$$\tau_\lambda = (s_i, t_i)_{i=1}^{\mathcal{M}_1} .$$

– τ_Λ contains the input-output pairs to Λ made during the construction queries. That is:

$$\tau_\Lambda = \{(s_i, t_i)\}_{i=1}^{\mathcal{M}_2} ;$$

– $\tau_\mathcal{P}$ contains the primitive queries. That is:

$$\tau_\mathcal{P} = (x_i, y_i)_{i=1}^{\mathcal{N}_1} ;$$

– $\tau_{\mathcal{P}'}$ contains the primitive queries to $\mathcal{P}'$. That is:

$$\tau_{\mathcal{P}'} = \{(x_i, y_i)\}_{i=1}^{\mathcal{N}_2} .$$

Moreover, to simplify our analysis, after the interaction, right before $\mathcal{A}$ outputs their decision bit, we release $\boldsymbol{K}$ and κ, sampled as $\boldsymbol{K}[1], \ldots, \boldsymbol{K}[m] \xleftarrow{\$} \{0,1\}^k$ and $\kappa \xleftarrow{\$} \{0,1\}^c$. Therefore, the extended transcript is $\tau = (\tau_\varphi, \tau_\lambda, \tau_\Lambda, \tau_\mathcal{P}, \tau_{\mathcal{P}'}, \boldsymbol{K}, \kappa)$.

We now focus on $\texttt{O}_{\text{MK}}$. Given $\kappa \in \{0,1\}^c$, define $\texttt{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, $\texttt{EM}^{\mathcal{P}}_{\kappa}$, and $\texttt{EM}^{\mathcal{P}'}_{\kappa}$ as follows:

$$\begin{aligned}
\texttt{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}(m) &= \mathcal{P}\left(\boldsymbol{K}[m] \| \boldsymbol{IV}[m]\right) \oplus (0^r \| \kappa) , \\
\texttt{EM}^{\mathcal{P}}_{\kappa}(x) &= \mathcal{P}\left(x \oplus (0^r \| \kappa)\right) \oplus (0^r \| \kappa) , \\
\texttt{EM}^{\mathcal{P}'}_{\kappa}(x) &= \mathcal{P}'\left(x \oplus (0^r \| \kappa)\right) \oplus (0^r \| \kappa) .
\end{aligned}$$

For every construction query, we can replace the initialization and first $\mathcal{P}$-evaluation with $\texttt{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, and the remaining $\mathcal{P}$-evaluations (resp., $\mathcal{P}'$-evaluations)

with $\mathtt{EM}_\kappa^{\mathcal{P}}$ (resp., $\mathtt{EM}_\kappa^{\mathcal{P}'}$). This modification does not alter the construction's behavior, but allows to view MacaKey as a cascade of Even-Mansour evaluations. Intuitively, this consists of artificially adding a random *dummy* c-bit key in-between the permutation calls made in MacaKey. We remark that the addition of the dummy does not modify the outputs of MacaKey. It may seem at first glance that this does not hold for the part of the output corresponding to Z_R. However, this is the xor of two quantities, both containing exactly one κ, leading to the c-bit masks cancel each other out before producing MacaKey's output. A visual representation of this process is shown in the leftmost illustration of Fig. 4.

Let $\kappa \xleftarrow{\$} \{0,1\}^c$. We now define the transcript in the MacaKey world analogously to how it is defined in the RS world:

- τ_φ records the outputs of all calls to $\mathtt{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$;
- τ_λ records the inputs and outputs of all calls to $\mathtt{EM}^{\mathcal{P}}_{\kappa}$;
- τ_Λ records the inputs and outputs of all calls to $\mathtt{EM}^{\mathcal{P}'}_{\kappa}$;
- $\tau_{\mathcal{P}}$ and $\tau_{\mathcal{P}'}$ contain the primitive queries.

Bad Events. We will label a transcript as *bad* if there exists a collision between elements in $\tau_{\mathcal{P}}$, $\tau_{\mathcal{P}'}$ τ_φ, τ_λ, or τ_Λ that leads to a degenerate behavior in $\mathtt{O}_{\mathrm{MK}}$. More precisely, a transcript τ is *bad* if any of those events occur:

- $\mathbf{Coll}^{\varphi}$: $\exists m \neq m' \in [\mu]$ such that: $\boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$.
- $\mathbf{Connect}_F^{\mathcal{P},\varphi}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (m,I) \in \tau_\varphi$ such that: $x = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$;
- $\mathbf{Connect}_B^{\mathcal{P},\varphi}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (m,I) \in \tau_\varphi$ such that: $y = I \oplus (0^r \| \kappa)$;
- $\mathbf{Connect}_F^{\mathcal{P},\lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (u,v) \in \tau_\lambda$ such that: $x = u \oplus (0^r \| \kappa)$;
- $\mathbf{Connect}_B^{\mathcal{P},\lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (u,v) \in \tau_\lambda$ such that: $y = v \oplus (0^r \| \kappa)$;
- $\mathbf{Connect}_F^{\mathcal{P}',\Lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}'}, (u,v) \in \tau_\Lambda$ such that: $x = u \oplus (0^r \| \kappa)$;
- $\mathbf{Connect}_B^{\mathcal{P}',\Lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}'}, (u,v) \in \tau_\Lambda$ such that: $y = v \oplus (0^r \| \kappa)$;
- $\mathbf{Connect}_F^{\lambda,\varphi}$: $\exists (s,t) \in \tau_\lambda, (m,I) \in \tau_\varphi$ such that: $s \oplus (0^r \| \kappa) = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$;
- $\mathbf{Connect}_B^{\lambda,\varphi}$: $\exists (s,t) \in \tau_\lambda, (m,I) \in \tau_\varphi$ such that: $I \oplus (0^r \| \kappa) = t$;
- $\mathbf{Bad} = \mathbf{Coll}^{\varphi} \vee \mathbf{Connect}_F^{\mathcal{P},\varphi} \vee \mathbf{Connect}_B^{\mathcal{P},\varphi} \vee \mathbf{Connect}_F^{\mathcal{P},\lambda} \vee \mathbf{Connect}_B^{\mathcal{P},\lambda} \vee \mathbf{Connect}_F^{\mathcal{P}',\Lambda} \vee \mathbf{Connect}_B^{\mathcal{P}',\Lambda} \vee \mathbf{Connect}_F^{\lambda,\varphi} \vee \mathbf{Connect}_B^{\lambda,\varphi}$.

Note that collisions between $\tau_{\mathcal{P}}$ and τ_Λ, between $\tau_{\mathcal{P}'}$ and τ_λ, as well as between τ_φ and $\tau_{\mathcal{P}'}$, do not lead to a degenerate behavior in $\mathtt{O}_{\mathrm{RS}}$.

Computing the Ratio for Good Transcripts. By definition of $\mathtt{O}_{\mathrm{MK}}$, every transcript which is attainable in $\mathtt{O}_{\mathrm{MK}}$ is also attainable in $\mathtt{O}_{\mathrm{RS}}$. Every transcript in $\mathtt{O}_{\mathrm{RS}}$ determines μ distinct input-outputs pairs for the injection function, $\mathcal{M}_1$

(resp., $\mathcal{M}_2$) distinct input-outputs pairs for λ (resp., Λ), and $\mathcal{N}_1$ (resp., $\mathcal{N}_2$) distinct input-outputs pairs for $\mathcal{P}$ (resp., $\mathcal{P}'$). Moreover, any good transcript which is attainable in $\mathtt{O}_{\mathrm{RS}}$ is also attainable in $\mathtt{O}_{\mathrm{MK}}$, where it defines $\mathcal{M}_1 + \mathcal{N}_1 + \mu$ (resp., $\mathcal{M}_2 + \mathcal{N}_2$) distinct permutation calls to $\mathcal{P}$ (resp., $\mathcal{P}'$). Therefore, for any good transcript τ,

$$\frac{\mathbf{Pr}\left[D_{\mathtt{O}_{\mathrm{MK}}} = \tau\right]}{\mathbf{Pr}\left[D_{\mathtt{O}_{\mathrm{RS}}} = \tau\right]} = \frac{(2^b)_{\mathcal{M}_1}(2^b)_{\mathcal{M}_2}(2^b)_{\mu}(2^b)_{\mathcal{N}_1}(2^b)_{\mathcal{N}_2}}{(2^b)_{\mathcal{M}_1+\mathcal{N}_1+\mu}(2^b)_{\mathcal{M}_2+\mathcal{N}_2}} \geq 1\,. \tag{15}$$

Bounding the Probability of Bad Transcripts in $\mathtt{O}_{RS}$. Now, it remains to evaluate the probability that a bad event occurs in $\mathtt{O}_{\mathrm{RS}}$. In this world, $\boldsymbol{K}$ and κ are sampled uniformly at random at the end of the interaction, thus the adversary cannot set this event adaptively. We evaluate each case below, we start by those that do not benefit from multicollision analysis.

– $\mathbf{Coll}^{\varphi}$: the probability that two users with the same IV have their key colliding is upper bounded by

$$\sum_{s} \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}} - 1)}{2^{k+1}}\,.$$

– $\mathbf{Connect}_F^{\mathcal{P},\varphi}$: each permutation evaluation can target at most $u_{\max}^{\boldsymbol{IV}}$ users at the same time. Therefore, this sub-event is set with a probability of at most:

$$\frac{\mathcal{N} \cdot u_{\max}^{\boldsymbol{IV}}}{2^k}\,.$$

– $\mathbf{Connect}_F^{\lambda,\varphi}$: We take a similar approach as the previous event. For each element of τ_λ, it has a probability of triggering this event of at most $u_{\max}^{\boldsymbol{IV}}/2^k$. So this happens with a probability of at most

$$\frac{\mathcal{M} \cdot u_{\max}^{\boldsymbol{IV}}}{2^k}\,.$$

– $\mathbf{Connect}_B^{\lambda,\varphi}$: Adopting a query-wise approach, we can easily see that this event happens with a probability of at most:

$$\frac{2(\mathcal{M} - \mu)\mu}{2^b} \leq \frac{2\mathcal{M}\mu}{2^b}\,.$$

In order to analyze the remaining bad events, we introduce the random variables:

$$\mathtt{Col}_F = \max_{z \in \{0,1\}^r} \#\left\{X \in \{0,1\}^b \mid \exists (X,Y) \in \tau_\lambda \cup \tau_\Lambda \text{ such that } \mathbf{outer}_r(X) = z\right\}\,,$$

$$\mathtt{Col}_B = \max_{z \in \{0,1\}^r} \#\left\{Y \in \{0,1\}^b \mid \exists (X,Y) \in \tau_\lambda \cup \tau_\Lambda \text{ such that } \mathbf{outer}_r(Y) = z\right\}\,,$$

$$\mathtt{Col}_I = \max_{z \in \{0,1\}^r} \#\left\{I \in \{0,1\}^b \mid \exists (m,I) \in \tau_\varphi \text{ such that } \mathbf{outer}_r(I) = z\right\}\,.$$

– $\mathbf{Connect}_B^{\mathcal{P},\varphi}$: Let us assume $\mathtt{Col}_I = \theta$ for some $\theta \in \mathbb{N}$. For each element of $\tau_{\mathcal{P}}$, there are by assumption at most θ elements of τ_φ that can trigger this event. For each of these, the probability of having the same inner part is at most $1/2^c$. Therefore, $\mathbf{Pr}\left[\mathbf{Connect}_B^{\mathcal{P},\varphi} \,\middle|\, \mathtt{Col}_I = \theta\right] \leq \theta\mathcal{N}/2^c$. Hence, the probability of triggering this event at is at most:

$$\begin{aligned}\mathbf{Pr}\left[\mathbf{Connect}_B^{\mathcal{P},\varphi}\right] &\leq \sum_{\theta\geq 0} \mathbf{Pr}\left[\mathbf{Connect}_B^{\mathcal{P},\varphi} \,\middle|\, \mathtt{Col}_I = \theta\right] \mathbf{Pr}\left[\mathtt{Col}_I = \theta\right] \\ &\leq \frac{\mathcal{N}}{2^c} \sum_{\theta\geq 0} \theta \mathbf{Pr}\left[\mathtt{Col}_I = \theta\right] = \frac{\mathbb{E}[\mathtt{Col}_I]\mathcal{N}}{2^c}\,.\end{aligned}$$

– $\mathbf{Connect}_F^{\mathcal{P},\lambda}$, $\mathbf{Connect}_B^{\mathcal{P},\lambda}$, $\mathbf{Connect}_F^{\mathcal{P}',\Lambda}$, and $\mathbf{Connect}_B^{\mathcal{P}',\Lambda}$ We can repeat the same reasoning, assuming $\mathtt{Col}_F = \theta_F$ and $\mathtt{Col}_B = \theta_B$ we get:

$$\begin{aligned}\mathbf{Pr}\left[\mathbf{Connect}_F^{\mathcal{P},\lambda} \vee \mathbf{Connect}_F^{\mathcal{P}',\Lambda}\right] &\leq \frac{\mathbb{E}[\mathtt{Col}_F]\mathcal{N}}{2^c} \\ \mathbf{Pr}\left[\mathbf{Connect}_B^{\mathcal{P},\lambda} \vee \mathbf{Connect}_B^{\mathcal{P}',\Lambda}\right] &\leq \frac{\mathbb{E}[\mathtt{Col}_B]\mathcal{N}}{2^c}\,.\end{aligned}$$

Therefore, by grouping everything together, we obtain:

$$\begin{aligned}\mathbf{Pr}\left[\mathbf{Bad}\right] \leq \sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} \\ + \frac{(\mathbb{E}[\mathtt{Col}_I] + \mathbb{E}[\mathtt{Col}_F] + \mathbb{E}[\mathtt{Col}_B])\,\mathcal{N}}{2^c}\,.\end{aligned} \tag{16}$$

It only remains to bound the expectations, in order to do so, we use Lemma 2. Given that the inputs to λ (in particular their outer part) cannot be chosen by the adversary, we can bound $\mathtt{Col}_I$, $\mathtt{Col}_F$, and $\mathtt{Col}_B$ as follows:

$$\begin{aligned}\mathbb{E}[\mathtt{Col}_I] &\leq \frac{2\mu}{2^r} + 3r + 4\,, \\ \mathbb{E}[\mathtt{Col}_F] &\leq \frac{2(\mathcal{M}-\mu)}{2^r} + 3r + 4\,, \\ \mathbb{E}[\mathtt{Col}_B] &\leq \frac{2(\mathcal{M}-\mu)}{2^r} + 3r + 4\,.\end{aligned} \tag{17}$$

and by substituting (17) in (16), yields:

$$\begin{aligned}\mathbf{Pr}\left[\mathbf{Bad}\right] \leq \sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} \\ + \frac{4(\mathcal{M}-\mu)\mathcal{N}}{2^b} + \frac{2\mu\mathcal{N}}{2^b} + \frac{(9r+12)\mathcal{N}}{2^c}\,.\end{aligned} \tag{18}$$

We can simplify (18) a bit by noticing that $\mathcal{M} \geq \mu$:

$$\frac{4(\mathcal{M}-\mu)\mathcal{N}}{2^b} + \frac{2\mu\mathcal{N}}{2^b} \leq \frac{4\mathcal{M}\mathcal{N}}{2^b}\,. \tag{19}$$

Conclusion. From (18) and (19) we get the result:

$$\Delta_{\mathcal{A}}(\mathtt{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}\ ;\ \mathtt{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}) \leq \\ \sum_{s} \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} + \frac{4\mathcal{M}\mathcal{N}}{2^b} + \frac{(9r+12)\mathcal{N}}{2^c},$$

which concludes the proof. □

5 Conclusion

In this work, we introduced `MacaKey`, the *first* sponge-based PRF that supports both full-state absorption and full-state squeezing, without sacrificing security and improving the throughput of all previously known permutation based constructions. We provided a tight security bound for `MacaKey`, matching that of `FSKS`. An interesting future work would be to extend the duplex construction [BDPV11a] using the idea underlying `MacaKey`. This would yield broader applicability, e.g., as an authenticated encryption scheme. However, extending the duplex construction using the idea underlying `MacaKey` is far from trivial. In particular, designing an authenticated encryption scheme would require a careful treatment of nonces and domain separation to ensure that the resulting scheme is both secure and optimal. Achieving tight, fine-grained security bounds in this setting would necessitate novel analysis.

References

ADMV15. Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 364–384. Springer, 2015.

BBN22. Arghya Bhattacharjee, Ritam Bhaumik, and Mridul Nandi. A Sponge-Based PRF with Good Multi-user Security. *IACR Cryptol. ePrint Arch.*, page 1146, 2022.

BDPV07. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge Functions, 2007.

BDPV08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

BDPV11a. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada,*

August 11-12, 2011, Revised Selected Papers, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

BDPV11b. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the security of the keyed sponge construction. In *Symmetric Key Encryption Workshop*, volume 2011, 2011.

BDPV13. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2013.

BPV12. Bertoni, Peeters, and Van Assche. Permutation-based encryption, authentication and authenticated encryption. In *Directions in Authenticated Ciphers*, 2012.

BR93. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

CD25. Bishwajit Chakraborty and Chandranan Dhar. Breaking and Fixing MacaKey. Cryptology ePrint Archive, Paper 2025/2038, 2025.

CDH+12. Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. A keyed sponge construction with pseudorandomness in the standard model. In *The Third SHA-3 Candidate Conference (March 2012)*, volume 3, page 7, 2012.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

CLL19. Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of Truncated Random Permutations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 175–195. Springer, 2019.

CLM19. Yu Long Chen, Eran Lambooij, and Bart Mennink. How to Build Pseudorandom Functions from Public Random Permutations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 266–293. Springer, 2019.

CS14. Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.

DEMS21. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

DEMS24. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon MAC, PRF, and Short-Input PRF - Lightweight, Fast, and Efficient Pseudorandom Functions. In Elisabeth Oswald, editor, *Topics in Cryptology - CT-RSA 2024 - Cryptographers' Track at the RSA Conference 2024, San Francisco, CA, USA, May 6-9, 2024, Proceedings*, volume 14643 of *Lecture Notes in Computer Science*, pages 381–403. Springer, 2024.

DMV17. Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 606–637. Springer, 2017.

Dwo01. Morris Dworkin. Recommendation for Block Cipher Modes of Operation. *NIST special publication*, 800:38B, 2001.

EM97. Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *J. Cryptol.*, 10(3):151–162, 1997.

GM20. Aldo Gunsing and Bart Mennink. The Summation-Truncation Hybrid: Reusing Discarded Bits for Free. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 187–217. Springer, 2020.

GPP11. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON Family of Lightweight Hash Functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

Hir18. Shoichi Hirose. Sequential Hashing with Minimum Padding. *Cryptogr.*, 2(2):11, 2018.

HPY07. Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007.

Iwa06. Tetsu Iwata. New blockcipher modes of operation with beyond the birthday bound security. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2006.

LBD23. Charlotte Lefevre, Yanis Belkheyar, and Joan Daemen. Kirby: A Robust Permutation-Based PRF Construction. *IACR Cryptol. ePrint Arch.*, page 1520, 2023.

LBM25. Charlotte Lefevre, Mario Marhuenda Beltrán, and Bart Mennink. To pad or not to pad? padding-free arithmetization-oriented sponges. *IACR Trans. Symmetric Cryptol.*, 2025(1):97–137, 2025.

LM24. Charlotte Lefevre and Bart Mennink. Generic Security of the Ascon Mode: On the Power of Key Blinding. In Maria Eichlseder and Sébastien Gambs, editors, *Selected Areas in Cryptography, 31st International Workshop, SAC*

2024, Montréal, Quebec, Canada, August 26-27, Revised Selected Papers, Lecture Notes in Computer Science. Springer, 2024. to appear.

Men18. Bart Mennink. Key Prediction Security of Keyed Sponges. *IACR Trans. Symmetric Cryptol.*, 2018(4):128–149, 2018.

Men23. Bart Mennink. Understanding the Duplex and Its Security. *IACR Transactions on Symmetric Cryptology*, 2023(2):1–46, Jun. 2023.

MMH+14. Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2014.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

MRV15. Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 465–489. Springer, 2015.

NIS12. NIST. SHA-3 Competition. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2007-2012.

NIS19. NIST. Lightweight Cryptography, February 2019. `https://csrc.nist.gov/Projects/Lightweight-Cryptography`.

NO14. Yusuke Naito and Kazuo Ohta. Improved Indifferentiable Security Analysis of PHOTON. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2014.

NY16. Yusuke Naito and Kan Yasuda. New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2016.

Pat08. Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of*

Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

SMC+25. Meltem Sönmez Turan, Kerry A. McKay, Donghoon Chang, Jinkeon Kang, and John Kelsey. Ascon-Based Lightweight Cryptography Standards for Constrained Devices. NIST SP 800-232, August 2025. https://doi.org/10.6028/NIST.SP.800-232.