Fine-tuning the ISO/IEC Standard LightMAC

Soumya Chattopadhyay[1], Ashwin Jha[2], and Mridul Nandi[1]

[1] Indian Statistical Institute, Kolkata, India
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
s.c.2357@gmail.com,ashwin.jha@cispa.de,mridul.nandi@gmail.com

Abstract. LightMAC, by Luykx et al., is a block cipher based message authentication code (MAC). The simplicity of design and low overhead allows it to have very compact implementations. As a result, it has been recently chosen as an ISO/IEC standard MAC for lightweight applications. LightMAC has been shown to achieve query-length independent security bound of $O(q^2/2^n)$ when instantiated with two independently keyed n-bit block ciphers, where q denotes the number of MAC queries and the query-length is upper bounded by $(n-s)2^s$ bits for a fixed counter size s. In this paper, we aim to minimize the number of block cipher keys in LightMAC. First, we show that the original LightMAC instantiated with a single block cipher key, referred as 1k-LightMAC, achieves security bound of $O(q^2/2^n)$ while the query-length is at least $(n-s)$ bits and at most $(n-s)\min\{2^{n/4}, 2^s\}$ bits. Second, we show that a minor variant of 1k-LightMAC, dubbed as LightMAC-ds, achieves security bound of $O(q^2/2^n)$ while query-length is upper bounded by $(n-s)2^{s-1}$ bits. Of independent interest, our security proof of 1k-LightMAC employs a novel sampling approach, called the *reset-sampling*, as a subroutine within the H-coefficient proof setup.

Keywords: LightMAC, MAC, PRF, single-key, lightweight, ISO/IEC standard

1 Introduction

Lightweight cryptography endeavors to safeguard communications in resource-constrained environments. The advent of Internet of Things has given a great impetus to this field of research in the last decade or so. As a result, several standardization efforts have tried to systematize the field, most notably the CAESAR competition [1], NIST lightweight cryptography standardization project [2], and the ISO/IEC standardization [3]. Specifically, the ISO/IEC 29192-6:2019 standard [3] specifies three message authentication code (or MAC) algorithms for lightweight applications. MACs are symmetric-key primitives that achieve data

Soumya Chattopadhyay and Mridul Nandi are supported by the project "Study and Analysis of IoT Security" under Government of India at R. C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata. Ashwin Jha's work was carried out in the framework of the French-German-Center for Cybersecurity, a collaboration of CISPA and LORIA.

authenticity and integrity. The ISO/IEC standard recommends LightMAC [4], Tsudik's keymode [5] and Chaskey-12 [6] as the three MAC algorithms. In this paper, we focus on LightMAC.

LightMAC, by Luykx et al. [4], is a parallelizable block cipher-based MAC. For an n-bit block cipher E instantiated with keys K_1 and K_2, and a global parameter $s < n$, a simplified[1] version of LightMAC can be defined as:

$$\mathsf{LightMAC}_{K_1,K_2}(m) := E_{K_2}(E_{K_1}(x[1]) \oplus \cdots \oplus E_{K_1}(x[\ell-1]) \oplus m[\ell] \| 10^{s-1}), \quad (1)$$

where $(m[1], \ldots, m[\ell])$ denotes the $(n-s)$-bit parsing of the input message m, and $x[i] = \langle i \rangle_s \| m[i]$ for $1 \leq i \leq \ell - 1$, where $\langle i \rangle_s$ denotes the s-bit binary representation of i. For obvious reasons s is also called the counter size. The counter-based encoding in LightMAC is inherited from some earlier MAC designs such as the XOR MACs by Bellare et al. [7] and Bernstein's protected counter sums [8]. The use of counter-based encoding limits the *rate*—ratio of the number of n-bit blocks in the message m to the number of block cipher calls required to process m. For example, LightMAC requires 4 calls to process a message of length $3n$ bits when the counter size $s = n/4$, whence the rate is $3/4$. Ideally, the rate should be as high as possible, with rate 1 or higher considered as holy grail. Dutta et al. [9] give optimal counter-based encoding strategies for some scenarios, resulting in significant speed-up. However, LightMAC still falls short on this account when compared to some other MAC schemes such as OMAC [10] and PMAC [11] etc.

However, LightMAC design is quite simple as it minimizes all auxiliary operations other than the block cipher call, which reduces the overhead to a minimum. For this reason, LightMAC is expected to have more compact implementations as compared to PMAC. Further, LightMAC is parallelizable like PMAC which enables it to exploit the parallel computing infrastructure, whenever available. As a result, LightMAC is a quite flexible algorithm, as it has qualities suitable for both memory-constrained environments as well as high performance computing.

Query-Length Independence: Yet another avenue where LightMAC gains over several other MAC schemes is its security guarantee. Many MAC algorithms, including PMAC and OMAC, have security bounds which degrade linearly with the query-length. Apparently, some sort of dependence on query-length is unavoidable in iterated MAC schemes. However, LightMAC is shown to have query-length independent security bounds.

It is well-known [12,13] that variable input length (VIL) pseudorandom functions (or PRFs) are good candidates for deterministic MACs. Indeed, almost all the security bounds on deterministic MAC schemes, in fact, quantify their PRF security. In the following discussion q and ℓ denote the number of queries and the bound on query-lengths, respectively.

Luykx et al. [4] showed that LightMAC achieves $O(q^2/2^n)$ bound on the success probability of any adversary (also referred as the PRF advantage). This

[1] assuming all messages have length $(n-s)r$ for some $1 \leq r \leq 2^s$.

bound is independent of the query-length ℓ, apart from the obvious bound of $\ell \leq (n-s)2^s$.

In comparison, arguably the most popular parallelizable MAC, PMAC, suffers from a linear degradation in security with increase in query-length. Some birthday-bound (PRF advantage is at least $q^2/2^n$) variants (or extensions) of PMAC, like PMAC with parity [14] and PMAC3 [15], do achieve query-length independence for a wide range of ℓ values. However, this costs significant increase in design complexity, such as more than two-fold increase in memory usage and relatively complex auxiliary operations like multiple masking operations or generating error correcting codes.

The situation does not improve much, when we consider birthday-bound sequential modes either. Schemes like CBC-MAC [16], XCBC [17] and OMAC exhibit similar degradation in security with increase in query-length as PMAC. EMAC [18,19] achieves query-length independence with slightly higher PRF advantage of $O(q/2^{n/2})$ while $\ell \leq 2^{n/4}$. However, EMAC only works for messages with "multiple-of-n" length. One can extend the construction to arbitrary domain by either using extra block cipher keys, as in ECBC and FCBC [17], or apply some injective padding rule on the input message before processing it through EMAC.

Beyond-the-birthday bound (BBB) secure constructions such as Sum-ECBC [20], PMAC+ [21], 3kf9 [22], PMACx [23], 1k-PMAC+ [24], and LightMAC+ [25], can also achieve query-length independent security bounds for a wide range of values of ℓ. However, these constructions require significantly more memory and additional operations (due to the BBB security requirement) as compared to LightMAC.

1.1 Motivation

ISO standards are widely used in communication protocols such as TLS, Bluetooth protocol, Zigbee etc. Being an ISO standard for lightweight cryptography, LightMAC is also widely recognized as a suitable MAC candidate for deployment in resource-constrained environments. Possibly, its simple and compact design and query-length independent security are the main reasons behind this perception. On a closer look, we see that the two independent keys greatly simplify the security argument of LightMAC. Due to the independence of keys, it can be viewed as an instance of the Hash-then-PRF paradigm [26,27], and hence the PRF security bound follows directly from LightMAC output collision probability.

However, maintaining two block cipher keys could be a burden in memory-constrained environments. Currently LightMAC with 2 keys requires 256 bits for key (128-bit block cipher key). Instead, one-key variants of LightMAC use 128 bits, which is a significant optimization in memory footprint both in hardware and software. The problem is further aggravated when implementations store precomputed round keys to reduce latency. For example, in case of AES128 [28], this precomputation would require 176 bytes of memory per key. This motivates us to look into the problem of minimizing the number of keys in LightMAC, while maintaining the query-length independence. Specifically, we ask the following question:

† : *Is there a single-key LightMAC variant which achieves similar query-length independent bounds as two-key LightMAC?*

As it turns out, the answer to this question is not straightforward. Recall the description of LightMAC from Eq. (1). Let $y_i := E_{K_1}(x_i)$ and $y^{\oplus} := y_1 \oplus \cdots \oplus y_{\ell-1} \oplus m_\ell \| 10^{s-1}$. We call x_i and y_i the i-th intermediate input and outputs, respectively and $y^{\oplus}$ and $t = E_{K_2}(y^{\oplus})$ the final input and output, respectively. There are two non-trivial bottlenecks (see section 3.2) in answering the above questions:

1. Collisions between intermediate input and final input, and
2. Collisions between intermediate output and final output.

The naive way to handle these two cases is to bound the probability of these events to $O(q^2\ell/2^n)$ as there are at most $q\ell$ intermediate inputs/outputs and q final inputs/outputs. Clearly, this naive approach leads to a degradation in the security. So,

$\star$: *we need a more sophisticated strategy to prove the security of single-key LightMAC.*

Yet another approach is to explicitly separate the final inputs from intermediate inputs by fixing some input bit to 0 in intermediate inputs and 1 in final inputs. This will help in resolving the first bottleneck. However, the second bottleneck is still present. Hence, the resulting construction is not as straightforward as two-key LightMAC. Further, domain separation also introduces slight changes in the standardized design, which is not appreciated by end-users, in general. So,

$\star\star$: *variants with very small modifications over the original LightMAC algorithm will be preferred.*

In this paper, we aim to answer † in affirmative using $\star$ and $\star\star$ as general guidelines.

1.2 Our Contributions

Our contributions are twofold:

First, in section 4, we show that *single-key* LightMAC*, denoted as* 1k-LightMAC*, is as secure as two-key* LightMAC*, while the query-lengths are lower bounded by* $(n-s)$ *bits and upper bounded by* $(n-s)\min\{2^{n/4}, 2^s\}$ *bits.* In other words, we show a security bound of $O(q^2/2^n)$ for 1k-LightMAC, while $(n-s) \leq \ell \leq (n-s)\min\{2^{n/4}, 2^s\}$.

In order to circumvent the two bottlenecks discussed in section 1.1, we use a novel sampling approach, called the *reset-sampling* – a proof style much in the same vein as the reprogramming of random oracles [29]. At the highest level, reset-sampling can be viewed as a subroutine in H-coefficient [30,31] or Expectation method [32] based proofs that can be employed in order to transform

Table 1.1: A comparative summary of several birthday-bound block cipher based MAC algorithms. Here q denotes the number of queries, ℓ denotes the bound on query-length, and s denotes the counter size.

Mode	#BC Keys	Aux. memory[1]	PRF Bound	Restriction[2]
EMAC [18,19]	2	0	$O\left(\frac{q}{2^{n/2}}\right)$ [33]	$\ell \leq n2^{n/4}$
ECBC,FCBC [17]	3	0	$O\left(\frac{q}{2^{n/2}}\right)$ [34]	$\ell \leq n2^{n/4}$
XCBC [17]	1	$2n$	$O\left(\frac{q^2\ell}{2^n}\right)$ [35]	$\ell \leq n2^{n/3}$
OMAC [10]	1	n	$O\left(\frac{q^2\ell}{2^n}\right)$ [36]	$\ell \leq n2^{n/4}$
PMAC [11]	1	n	$\Theta(\frac{q^2\ell}{2^n})$ [35,37,38]	-
PMAC3 [15]	2	$3n$	$O\left(\frac{q^2}{2^n}\right)$ [15,39]	$\ell \leq n2^{n/2}$
LightMAC [4,3]	2	s	$O\left(\frac{q^2}{2^n}\right)$ [4]	$\ell \leq (n-s)2^s$
1k-LightMAC	1	s	$O\left(\frac{q^2}{2^n}\right)$	$(n-s) \leq \ell \leq (n-s)\min\{2^{n/4}, 2^s\}$
LightMAC-ds	1	s	$O\left(\frac{q^2}{2^n}\right)$	$\ell \leq (n-s)2^{s-1}$

[1] The memory used to store masking keys or counter value.
[2] Upper bound on query-lengths for which the given security bound holds.

a possibly bad transcript into a good transcript given that certain conditions are fulfilled. In other words, it resets some bad transcript into a good transcript. For example, in our analysis we reset the intermediate outputs appropriately whenever the corresponding intermediate input collides with some final input.

Second, in section 5, we propose a close variant of 1k-LightMAC, dubbed as LightMAC-ds, and show that *LightMAC-ds is asymptotically as secure as two-key LightMAC, i.e., it achieves security bound of* $O(q^2/2^n)$ *while* $\ell \leq (n-s)2^{s-1}$. The restriction on length is due to the loss of 1-bit from counter for domain separation.

Table 1.1 gives a comparison of LightMAC, 1k-LightMAC, and LightMAC-ds with several popular birthday-bound block cipher based MAC mode of operation. We deliberately refrain from enumerating beyond-the-birthday bound modes for a fair comparison, as they require relatively more memory and/or key material (due to the BBB security requirement). From the table, it is clear that the three LightMAC candidates are overall better than other modes considering security vs block cipher key size and security vs auxiliary memory. Further, *1k-LightMAC is almost as good as LightMAC and LightMAC-ds as long as* $(n-s) \leq \ell \leq (n-s)\min\{2^{n/4}, 2^s\}$. *Note that, the lower bound on* ℓ *is necessary to avoid some trivial collision events (see section 3.2 for further details). Similarly, LightMAC-ds is as good as LightMAC as long as* $\ell \leq (n-s)2^{s-1}$.

Practical Significance: Our results are restricted in terms of the length of messages, especially, 1k-LightMAC which effectively bounds the message length

to roughly $2^{35.5}$ bytes for 128-bit block size. However, we believe that this is a minor issue. Indeed, many real life communication protocols limit the message lengths to much less than 1 Gigabyte. For example, SRTP [40] limits the payload length to at most 1 Megabyte. So, the impact of length restriction could, in fact, be minimal in most applications. Furthermore, we emphasize that 1k-LightMAC can be used as a drop-in replacement, since the required changes are minimal. This is particularly a compelling feature for the intended application area of the ISO/IEC-29192-6:2019 standard, i.e. resource constrained environments, where additional deployment or maintenance cost is highly undesirable. In summary, our results have significant practical importance due to the ISO/IEC standardization of LightMAC and the inherent advantages of 1k-LightMAC and LightMAC-ds over LightMAC.

2 Preliminaries

Notational Setup: For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. The set of all bit strings (including the empty string) is denoted $\{0,1\}^*$. The length of any bit string $X \in \{0,1\}^*$, denoted $|X|$, is the number of bits in X. For $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of all bit strings of length n, and $\{0,1\}^{\leq n} := \bigcup_{i=0}^{n} \{0,1\}^i$. For any $A, B \in \{0,1\}^*$, we write $A \| B$ to denote the concatenation of A and B. For $n \in \mathbb{N}$ and $X \in \{0,1\}^*$, $(X_1, \ldots, X_l) \xleftarrow{n} X$ denotes the n-bit parsing of X where $|X_i| = n$ for all $1 \leq i \leq l-1$ and $0 \leq |X_l| \leq n-1$. For any $n \in \mathbb{N}$ and $M \in \{0,1\}^*$, we define $\mathsf{pad}_n(M) := M \| 10^d$ where d is the smallest integer such that $|\mathsf{pad}_n(M)|$ is a multiple of n. For $i, m \in \mathbb{N}$ such that $i < 2^m$, we define $\langle i \rangle_m$ as the m-bit binary encoding of the integer i. For $0 \leq k \leq n$, we define the falling factorial $(n)_k := n!/(n-k)! = n(n-1)\cdots(n-k+1)$. The set of all functions from $\mathcal{X}$ to $\mathcal{Y}$ is denoted $\mathcal{F}(\mathcal{X}, \mathcal{Y})$, and the set of all permutations of $\mathcal{X}$ is denoted $\mathcal{P}(\mathcal{X})$. We simply write $\mathcal{F}(a, b)$ and $\mathcal{P}(a)$, whenever $\mathcal{X} = \{0,1\}^a$ and $\mathcal{Y} = \{0,1\}^b$.

For a pair of q-tuples $\widetilde{X} = (X_1, \ldots, X_q)$ and $\widetilde{Y} = (Y_1, \ldots, Y_q)$, $(\widetilde{X}, \widetilde{Y})$ denotes the $2q$-tuple $(X_1, \ldots, X_q, Y_1, \ldots, Y_q)$. Similarly, one can extend notation for more than 2 tuples. Two q-tuples $\widetilde{X}$ and $\widetilde{Y}$ are said to be *permutation compatible*, denoted as $\widetilde{X} \leftrightsquigarrow \widetilde{Y}$, if $(X_i = X_j) \iff (Y_i = Y_j)$, for all $i \neq j$. By an abuse of notation, we also use $\widetilde{X}$ to denote the set $\{X_i : i \in [q]\}$.

For a finite set $\mathcal{X}$, $\mathsf{X} \leftarrow_{\$} \mathcal{X}$ denotes the uniform at random sampling of X from $\mathcal{X}$, and $\widetilde{\mathsf{X}} \leftarrow_{\#} \mathcal{X}$ denotes the without replacement sampling of a tuple $\widetilde{\mathsf{X}}$ from the set $\mathcal{X}$.

A Useful Lemma: The following result from linear algebra will be very useful in later analysis.

Lemma 2.1. *Let $(\mathsf{Y}_1, \ldots, \mathsf{Y}_l) \leftarrow_{\#} \mathcal{S} \subset \{0,1\}^n$ with $|\mathcal{S}| = N > l$. Let A be a $k \times l$ binary matrix with rank r. We write the column vector $(\mathsf{Y}_1, \ldots, \mathsf{Y}_l)^{tr}$ as $\widetilde{\mathsf{Y}}$. Then, for any $c \in (\{0,1\}^n)^k$, we have*

$$\Pr\left[A \cdot \widetilde{\mathsf{Y}} = c\right] \leq \frac{1}{(N-l)^r}$$

Proof. Since the rank of the matrix A is r, we can identify $1 \le i_1 < \cdots < i_r \le l$ such that $\mathsf{Y}_{i_1}, \ldots \mathsf{Y}_{i_r}$ will be uniquely determined by fixing the value for the remaining $l-r$ variables. By conditioning on the values of these $l-r$ variables, the probability that $A \cdot \widetilde{\mathsf{Y}} = c$ is bounded by at most $\frac{1}{(N-l+r)_r}$ which is less than $\frac{1}{(N-l)^r}$. □

We will often employ this lemma for $k \ge 2$ cases.

2.1 Security Definitions

Distinguishers: A (q,T)-distinguisher $\mathscr{A}$ is an oracle Turing machine, that makes at most q oracle queries, runs in time at most T, and outputs a single bit. For any oracle $\mathcal{O}$, we write $\mathscr{A}^{\mathcal{O}}$ to denote the output of $\mathscr{A}$ after its interaction with $\mathcal{O}$. By convention, $T = \infty$ denotes computationally unbounded (information-theoretic) and deterministic distinguishers. In this paper, we assume that the distinguisher is non-trivial, i.e., it never makes a duplicate query. Let $\mathbb{A}(q,T)$ be the class of all non-trivial distinguishers limited to q queries and T computations.

Pseudorandom Function: A $(\mathcal{K},\mathcal{X},\mathcal{Y})$-*keyed function* F with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We write $F_K(X)$ for $F(K,X)$.

The *pseudorandom function* or PRF advantage of any distinguisher $\mathscr{A}$ against a $(\mathcal{K},\mathcal{X},\mathcal{Y})$-keyed function F is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{F}(\mathscr{A}) = \mathbf{Adv}_{F;\Gamma}(\mathscr{A}) := \left| \Pr_{\mathsf{K} \xleftarrow{\$} \mathcal{K}} \left[\mathscr{A}^{F_{\mathsf{K}}} = 1\right] - \Pr_{\Gamma \xleftarrow{\$} \mathcal{F}(\mathcal{X},\mathcal{Y})} \left[\mathscr{A}^{\Gamma} = 1\right] \right|. \quad (2)$$

The *PRF security* of F against $\mathbb{A}(q,T)$ is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{F}(q,T) := \max_{\mathscr{A} \in \mathbb{A}(q,T)} \mathbf{Adv}^{\mathsf{prf}}_{F}(\mathscr{A}).$$

Pseudorandom Permutation: A $(\mathcal{K},\{0,1\}^n)$-*block cipher* E with key space $\mathcal{K}$ and block space $\{0,1\}^n$ is a $(\mathcal{K},\{0,1\}^n,\{0,1\}^n)$-keyed function, such that $E(K,\cdot)$ is a permutation over $\{0,1\}^n$ for any key $K \in \mathcal{K}$. We write $E_K(X)$ for $E(K,X)$.

The *pseudorandom permutation* or PRP advantage of any distinguisher $\mathscr{A}$ against a $(\mathcal{K},\{0,1\}^n)$-block cipher E is defined as

$$\mathbf{Adv}^{\mathsf{prp}}_{E}(\mathscr{A}) = \mathbf{Adv}_{E;\Pi}(\mathscr{A}) := \left| \Pr_{\mathsf{K} \xleftarrow{\$} \mathcal{K}} \left[\mathscr{A}^{E_{\mathsf{K}}} = 1\right] - \Pr_{\Pi \xleftarrow{\$} \mathcal{P}(n)} \left[\mathscr{A}^{\Pi} = 1\right] \right|. \quad (3)$$

The *PRP security* of E against $\mathbb{A}(q,T)$ is defined as

$$\mathbf{Adv}^{\mathsf{prp}}_{E}(q,T) := \max_{\mathscr{A} \in \mathbb{A}(q,T)} \mathbf{Adv}^{\mathsf{prp}}_{E}(\mathscr{A}).$$

2.2 H-coefficient Technique

The H-coefficient technique by Patarin [30,31] is a tool to upper bound the distinguishing advantage of any deterministic and computationally unbounded distinguisher $\mathscr{A}$ in distinguishing the real oracle $\mathcal{R}$ from the ideal oracle $\mathcal{I}$. The collection of all queries and responses that $\mathscr{A}$ made and received to and from the oracle, is called the transcript of $\mathscr{A}$, denoted as τ.

Let $\mathbb{R}$ and $\mathbb{I}$ denote the transcript random variable induced by $\mathscr{A}$'s interaction with $\mathcal{R}$ and $\mathcal{I}$, respectively. Let $\mathcal{T}$ be the set of all transcripts. A transcript $\tau \in \mathcal{T}$ is said to be *attainable* if $\Pr[\mathbb{I} = \tau] > 0$, i.e., it can be realized by $\mathscr{A}$'s interaction with $\mathcal{I}$. Following these notations, we state the main result of H-coefficient technique in Theorem 2.1. A proof of this theorem is available in multiple papers, including [41,42].

Theorem 2.1 (H-coefficient). *For $\epsilon_1, \epsilon_2 \geq 0$, suppose there is a set $\mathcal{T}_{\mathsf{bad}} \subseteq \mathcal{T}$, that we call the set of all bad transcripts, such that the following conditions hold:*

- $\Pr[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}] \leq \epsilon_1$*; and*
- *For any $\tau \notin \mathcal{T}_{\mathsf{bad}}$, τ is attainable and* $\dfrac{\Pr[\mathbb{R} = \tau]}{\Pr[\mathbb{I} = \tau]} \geq 1 - \epsilon_2$.

Then, for any computationally unbounded and deterministic distinguisher $\mathscr{A}$, we have

$$\mathbf{Adv}_{\mathcal{R};\mathcal{I}}(\mathscr{A}) \leq \epsilon_1 + \epsilon_2.$$

3 Revisiting LightMAC

LightMAC is a block cipher-based parallelizable PRF construction by Luykx et al. [4]. It uses a counter-based encoding of input message blocks, much in the same vein as some of the previously proposed constructions like XMACC and XMACR [7] and protected counter sums [8]. Algorithm 3.1 gives the algorithmic description of LightMAC and Figure 3.1 gives a pictorial illustration.

Throughout the rest of this paper, we refer to $x[i]$ and $y[i]$ as *intermediate input* and *output*, respectively, for all $i \in [\ell - 1]$ and $y^{\oplus}$ and t are referred as the *final input* and *output*, respectively.

Note that, the block size n and counter size s are application specific parameters that are fixed before any invocation. In order to argue the security of LightMAC, we must have $\langle i \rangle_s \neq \langle j \rangle_s$. When $i = 2^s + j$ for some $j \in [2^s - 1]$, then $\langle i \rangle_s = \langle j \rangle_s$. So, the maximum number of blocks in the padded message, denoted $\ell_{\max}$, must be less than 2^s. This will ensure that all the counters will be different.

3.1 Hash-then-PRP and the Security of LightMAC

For some $\epsilon \geq 0$, a $(\mathcal{K}, \{0,1\}^{\leq (n-s)2^s}, \{0,1\}^n)$-keyed function H is called an ϵ-*universal hash function* if for all distinct $m, m' \in \{0,1\}^{\leq (n-s)2^s}$, we have

$$\Pr_{\mathsf{K} \xleftarrow{\$} \mathcal{K}}[H_{\mathsf{K}}(m) = H_{\mathsf{K}}(m')] \leq \epsilon.$$

Algorithm 3.1 LightMAC based on an n-bit block cipher E instantiated with two keys K_1, K_2. Here s denotes the counter size.

function $\mathsf{LightMAC}_{E_{K_1},E_{K_2}}(m)$
  $y^{\oplus} \leftarrow 0^n$
  $(m[1], \ldots, m[\ell]) \xleftarrow{n-s} m$
  for $i = 1$ **to** $\ell - 1$ **do**
    $x[i] \leftarrow \langle i \rangle_s \| m[i]$ ▷ encoding $\langle i \rangle_s$ and $m[i]$ into $x[i]$
    $y[i] \leftarrow E_{K_1}(x[i])$ ▷ encrypting the encoded input
    $y^{\oplus} \leftarrow y^{\oplus} \oplus y[i]$ ▷ accumulating the intermediate output
  end for
  $y^{\oplus} \leftarrow y^{\oplus} \oplus \mathsf{pad}_n(m[\ell])$ ▷ accumulating final block of message
  $t \leftarrow E_{K_2}(y^{\oplus})$ ▷ tag generation
  return t
end function

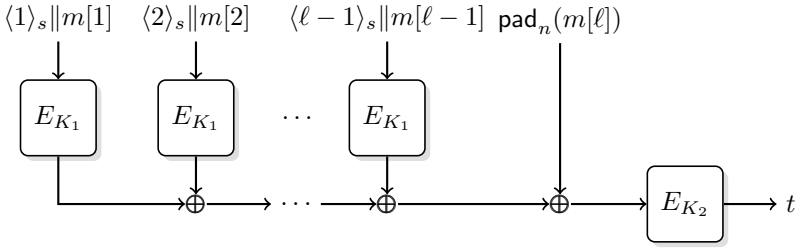

Fig. 3.1: LightMAC evaluated over an ℓ-block padded message m.

Universal hash functions are very useful in constructing PRFs via the Hash-then-PRP[2] paradigm [26,38]. In this paradigm, given independently keyed ϵ-universal hash function H_{K} and block cipher $E_{\mathsf{K}'}$, we define the Hash-then-PRP composition as $E_{K'} \circ H_K$. It is well-known that

$$\mathbf{Adv}^{\mathsf{prf}}_{E_{\mathsf{K}'} \circ H_{\mathsf{K}}}(q, T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(q, T') + \binom{q}{2}\left(\frac{1}{2^n} + \epsilon\right), \tag{4}$$

where $T' = T + qO(T_E)$ and T_E denotes the runtime of E.

We skip the proof of this result as it is available in multiple papers including [38,43]. An informal justification for Eq. (4) is based on the observation that if the input to $E_{\mathsf{K}'}$ is distinct for all q queries then the outputs behave as "almost uniform at random". The probability that some inputs to $E_{\mathsf{K}'}$ collide is bounded by $\binom{q}{2}\epsilon$.

PRF Security of LightMAC: Consider a $(\mathcal{K}, \{0,1\}^{\leq (n-s)2^s}, \{0,1\}^n)$-keyed function LightHash, defined by the following mapping:

$$\forall m \in \{0,1\}^{\leq (n-s)2^s}, \ \mathsf{LightHash}_{E_{K_1}}(m) := y^{\oplus},$$

[2] Here, we say PRP instead of PRF to highlight the use of block cipher based finalization.

where $y^{\oplus}$ is the final input corresponding to m in $\mathsf{LightMAC}_{E_{K_1},E_{K_2}}(m)$. Now, we can view LightMAC as an instantiation of Hash-then-PRP, by redefining LightMAC as

$$\mathsf{LightMAC}_{E_{K_1},E_{K_2}}(m) := E_{K_2}(\mathsf{LightHash}_{E_{K_1}}(m)).$$

Suppose, $\mathsf{LightHash}_{\Pi_1}$ is an ϵ_{LH}-universal hash for $\Pi_1 \leftarrow\!\!\$\, \mathcal{P}(n)$. Then, using Eq. (4), we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC}}(q,T) \le 2\mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma,T') + \binom{q}{2}\left(\frac{1}{2^n} + \epsilon_{\mathsf{LH}}\right), \tag{5}$$

where σ denotes the total number of blocks in all q padded queries, and $T' = T + \sigma O(T_E)$ and T_E denotes the runtime of E.

In [4,9], it has been shown that $\epsilon_{\mathsf{LH}} \le 1/(2^n - 2\ell_{\max})$, where $\ell_{\max}$ is the upper bound on the query-length in blocks. It is simply because for any $m \neq m'$ with lengths ℓ, ℓ' respectively, the event $\mathsf{LightHash}_{\Pi_1}(m) = \mathsf{LightHash}_{\Pi_1}(m')$ is identical with

$$\bigoplus_{i=1}^{\ell-1} \Pi_1(x[i]) \bigoplus_{j=1}^{\ell'-1} \Pi_1(x'[j]) = \mathsf{pad}_n(m[\ell]) \oplus \mathsf{pad}_n(m'[\ell']). \tag{6}$$

Now, since $m \neq m'$, either $(x[1],\ldots,x[\ell-1]) \neq (x'[1],\ldots,x'[\ell'-1])$, or

$$(x[1],\ldots,x[\ell-1]) = (x'[1],\ldots,x'[\ell'-1]) \wedge \mathsf{pad}_n(m[\ell]) \neq \mathsf{pad}_n(m'[\ell']).$$

The second case has zero probability. In the first case, assuming $\ell \ge \ell'$, we have at least one block say $x[i]$ which is distinct from all other blocks. Then, the probability of the event defined in Eq. (6) can be bounded above by probability that $\Pi_1(x[i])$ attains a certain value conditioned on the output of Π_1 on all other $x[j]$ and $x'[j']$ values for $j \in [\ell-1]\setminus\{i\}$ and $j' \in [\ell'-1]$. There are at most $2\ell_{\max}$ such values, i.e., Π_1 is already sampled on at most $2\ell_{\max}$ points. Therefore, the probability is bounded above by $1/(2^n - 2\ell_{\max})$.

By combining this bound with Eq. (5), we get the desired result for LightMAC in the following proposition.

Proposition 3.1. *For $\ell_{\max} < \min\{2^{n-2}, 2^s\}$, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC}}(q,T) \le 2\mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma,T') + \frac{1.5q^2}{2^n},$$

where σ denotes the total number of blocks in all q padded queries, and $T' = T + \sigma O(T_E)$ and T_E denotes the runtime of E.

3.2 Bottlenecks for Single-key LightMAC

We have just seen that the query-length independent security argument for LightMAC comes quite easily from the Hash-then-PRP paradigm. This is possible because K_1 and K_2 are independent of each other. A natural direction to explore

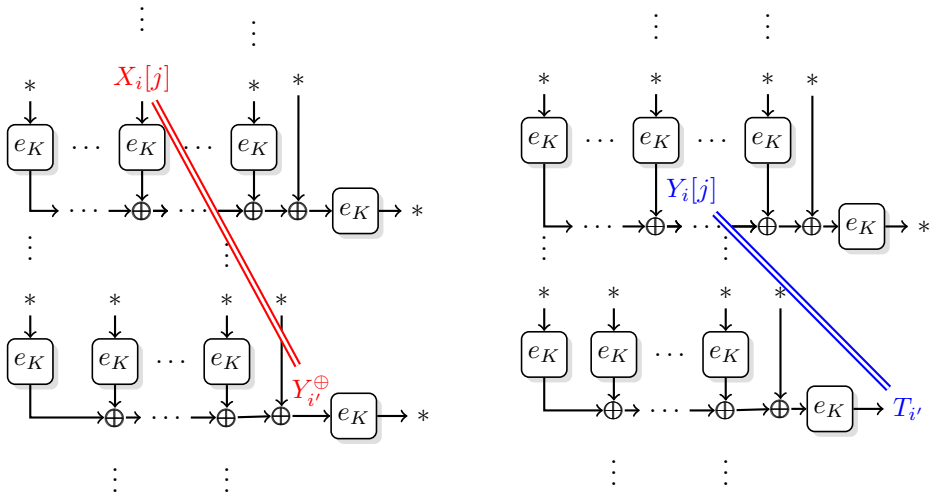

Fig. 3.2: `Icoll` (left) and `Ocoll` (right) events. In each case, labels with same color are equal, and double lines between two labels signify equality between the corresponding variables.

is the relaxation: $K_1 = K_2 = K$, i.e., LightMAC instantiated with a single key. Formally, we define the single-key LightMAC construction as follows:

$$\mathsf{1k\text{-}LightMAC}_{E_K} := \mathsf{LightMAC}_{E_K,E_K}.$$

We remark that *the additional nomenclature* 1k-LightMAC *is just for the sake of brevity. Indeed,* 1k-LightMAC *and* LightMAC *are algorithmically equivalent.* We have just instantiated $K_1 = K_2 = K$.

First thing to note is that Hash-then-PRP is no longer applicable as the hash function H_{K} and block cipher E_{K} are no longer independent. So, we have to look for a dedicated proof.

Suppose the adversary makes q queries $m_1, \ldots, m_q$ and the corresponding tuple of intermediate inputs and outputs are denoted $x_i = (x_i[1], \ldots, x_i[\ell_i - 1])$ and $y_i = (y_i[1], \ldots, y_i[\ell_i - 1])$, respectively. Similarly, the final input and output for the q queries is denoted $y_i^{\oplus}$ and t_i, respectively. Consider the events:

$$\texttt{Icoll}: \ \exists (i,a) \in [q] \times [\ell_i - 1], j \in [q], \text{ such that } x_i[a] = y_j^{\oplus};$$
$$\texttt{Ocoll}: \ \exists (i,a) \in [q] \times [\ell_i - 1], j \in [q], \text{ such that } y_i[a] = t_j;$$

`Icoll` denotes the event that a final input collides with some intermediate input and `Ocoll` denotes the analogous event for output collisions (see Figure 3.2).

In a dedicated proof we must take care of these cases as they may lead to inconsistent transcripts. For example, it is possible that $x_i[a] = y_j^{\oplus}$ (`Icoll` holds) but $y_i[a] \neq t_j$ or vice-versa. The probability of realizing such a transcript is zero in the real world. In fact, one can easily create such inconsistencies by first

making a query $m_1 = \langle 1 \rangle_s$, and then making another query $m_2 = 10^{n-s-1} \| x$, where x is any arbitrary bit string. Clearly, $x_2[1] = y_1^{\oplus}$, which implies that `Icoll` holds. This might help an adversary to mount an attack on 1k-LightMAC as it can access the internal variables using very short queries. Interestingly, if we swap the positions of counter and message block, then this trivial collision is no longer possible, and it might even be possible to show that the resulting variant is secure. Since our aim is to study the standardized algorithm, we simply assume that messages are at least $(n - s)$ bits long, thereby ensuring that at least one block cipher call is made in the hash layer. But, this only helps to avoid collisions in the corner case. We still have to consider the possibility of `Icoll` and `Ocoll` in the general case. We have to ensure that such inconsistencies do not occur with high probability. A straightforward bound on these events introduces a bound of the form $O(q^2 \ell_{\max}/2^n)$ since there are at most $q\ell_{\max}$ many (i, a) pairs and q choices for j. However, we aim to do better than this. In the next two sections, we show how we can handle these events in better way.

4 Security of 1k-LightMAC

This section is devoted to the PRF security of 1k-LightMAC. Throughout this section, we assume that messages are at least $(n - s)$-bit long. This assumption is used to avoid some trivial bad events, as discussed in section 3.2.

Theorem 4.1. *Let $q, \ell_{\min}, \ell_{\max}, \sigma, t > 0$. For $\ell_{\min} \geq 2$, $q + 4\ell_{\max} \leq 2^{n-1}$, the PRF security of 1k-LightMAC against $\mathbb{A}(q, T)$ is given by*

$$\mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}}(q, T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma + q, T') + \frac{1.5q^2}{2^n} + \frac{7.5q^3\ell_{\max}^2}{2^{2n}} + \frac{4q^4\ell_{\max}^2}{2^{3n}} + \frac{2\sigma}{2^n},$$

where q denotes the number of queries, $\ell_{\max}$ (res. $\ell_{\min}$) denotes an upper (res. lower) bound on the number of blocks in any padded query, σ denotes the total number of blocks present in all q queries, $T' = T + \sigma O(T_E)$ and T_E denotes the runtime of E.

Further assuming $\ell_{\max} \leq \min\{2^{n/4}, 2^s\}$ and $q \leq \min\{2^{\frac{3n}{4}-2}, 2^{\frac{n}{2}-1.51}\}$, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}}(q, T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma + q, T') + \frac{4q^2}{2^n} + \frac{2\sigma}{2^n}.$$

The proof of this theorem is described in the rest of this section. First of all, we switch to the information-theoretic setting, i.e., E_{K} is replaced with $\Pi \leftarrow_{\$} \mathcal{P}(n)$ via a standard hybrid argument. Formally, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}}(q, T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma + q, T') + \mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}_{\Pi}}(q, \infty). \tag{7}$$

So it is enough to bound the PRF security of 1k-LightMAC_{Π}, henceforth also referred as the real oracle. We apply the H-coefficient technique to bound this term. Fix any $\mathscr{A} \in \mathbb{A}(q, \infty)$ such that

$$\mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}_{\Pi}}(q, \infty) = \mathbf{Adv}^{\mathsf{prf}}_{\text{1k-LightMAC}_{\Pi}}(\mathscr{A}).$$

Going forward, we will bound the advantage of $\mathscr{A}$.

4.1 Description of Oracles and their Transcripts

Real Oracle: The real oracle corresponds to $\mathsf{1k\text{-}LightMAC}_{\Pi}$. It responds faithfully to all the queries made by $\mathscr{A}$. Once the query-response phase is over, it releases all the intermediate inputs and outputs to $\mathscr{A}$.

In addition, the real oracle releases three binary variables, namely, FlagT, FlagZ, and FlagY, all of which are degenerately set to 0. The utility of these flags will become apparent from the description of ideal oracle. For now, it is sufficient to note that these flags are degenerate in the real world.
Formally, we have $\mathbb{R} := (\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}, \mathsf{FlagT}, \mathsf{FlagZ}, \mathsf{FlagY})$, where

- $\widetilde{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_q)$ denotes the q-tuple of queries made by $\mathscr{A}$, where $\mathsf{M}_i \in \{0,1\}^{\leq (n-s)2^s}$ for all $i \in [q]$. In addition, for all $i \in [q]$, let $\ell_i := \left\lfloor \frac{|\mathsf{M}_i|}{n-s} \right\rfloor + 1$.
- $\widetilde{\mathsf{T}} = (\mathsf{T}_1, \ldots, \mathsf{T}_q)$ denotes the q-tuple of final outputs received by $\mathscr{A}$, where $\mathsf{T}_i \in \{0,1\}^n$.
- $\widetilde{\mathsf{X}} = (\mathsf{X}_1, \ldots, \mathsf{X}_q)$, where X_i denotes the intermediate input tuple for the i-th query, i.e., for all $a \in [\ell_i - 1]$, $\mathsf{X}_i[a] = \langle a \rangle_s \| \mathsf{M}_i[a]$.
- $\widetilde{\mathsf{Y}} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$, where Y_i denotes the intermediate output tuple for the i-th query, i.e., for all $a \in [\ell_i - 1]$, $\mathsf{Y}_i[a] = \Pi(\mathsf{X}_i[a])$. In addition, let $\widetilde{\mathsf{Y}}^{\oplus} := (\mathsf{Y}_1^{\oplus}, \ldots, \mathsf{Y}_q^{\oplus})$, where $\mathsf{Y}_i^{\oplus} := \bigoplus_{a \in [q]} \mathsf{Y}_i[a] \oplus \mathsf{pad}_n(\mathsf{M}_i[\ell_i])$ for all $i \in [q]$.
- $\mathsf{FlagI} = 0$ for all $\mathsf{I} \in \{\mathsf{T}, \mathsf{Z}, \mathsf{Y}\}$.

Note that, $\widetilde{\mathsf{X}}$ is completely determined from $\widetilde{\mathsf{M}}$. We have included it in the transcript just for the sake of simplicity. From the definition of $\mathsf{1k\text{-}LightMAC}$, we know that $\Pi(\mathsf{Y}_i^{\oplus}) = \mathsf{T}_i$ for all $i \in [q]$. So, *in the real world we always have* $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus}) \leftrightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$, *i.e.,* $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus})$ *is permutation compatible with* $(\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$. We keep this observation in our mind when we simulate the ideal oracle.

Ideal oracle: We reuse the variable notations from the real oracle description to represent the ideal oracle transcript $\mathbb{I}$, i.e., $\mathbb{I} := (\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}, \mathsf{FlagT}, \mathsf{FlagZ}, \mathsf{FlagY})$. This should not cause any confusion, as we never consider the random variables $\mathbb{R}$ and $\mathbb{I}$ jointly, whence the probability distributions of the constituent variables will always be clear from the context. The ideal oracle transcript is described in three phases, each contingent on some predicates defined over the previous stages. Specifically, the ideal oracle first initializes $\mathsf{FlagT} = 0$, $\mathsf{FlagZ} = 0$, $\mathsf{FlagY} = 0$, and then follows the sampling mechanism given below:

Phase I (Query-Response Phase): In the query-response phase, the ideal oracle faithfully simulates $\Gamma \leftarrow\!\!\$\, \mathcal{F}(\{0,1\}^{\leq (n-s)2^s}, \{0,1\}^n)$. Formally, for $i \in [q]$, at the i-th query $\mathsf{M}_i \in \{0,1\}^{\leq (n-s)2^s}$, the ideal oracle outputs $\mathsf{T}_i \leftarrow\!\!\$\, \{0,1\}^n$. The partial transcript generated at the end of the query-response phase is given by $(\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}})$, where

- $\widetilde{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_q)$ and $\widetilde{\mathsf{T}} = (\mathsf{T}_1, \ldots, \mathsf{T}_q)$.

– $\widetilde{\mathsf{X}} = (\mathsf{X}_1, \ldots, \mathsf{X}_q)$, where $\mathsf{X}_i = (\mathsf{X}_i[1], \ldots, \mathsf{X}_i[\ell_i - 1])$ and $\mathsf{X}_i[a] := \langle a \rangle_s \| \mathsf{M}_i[a]$ for all $(i, a) \in [q] \times [\ell_i - 1]$.

Now, we define a predicate on $\widetilde{\mathsf{T}}$:

$$\mathtt{BadT}: \ \exists i \neq j \in [q], \text{ such that } \mathsf{T}_i = \mathsf{T}_j.$$

If $\mathtt{BadT}$ is true, then FlagT is set to 1, and $\widetilde{\mathsf{Y}} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$ is defined degenerately: $\mathsf{Y}_i[a] = 0^n$ for all $(i, a) \in [q] \times [\ell_i - 1]$. Otherwise, the ideal oracle proceeds to the next phase.

Phase II (Offline Initial Sampling Phase): Onward, we must have $\mathsf{T}_i \neq \mathsf{T}_j$ whenever $i \neq j$, and $\mathsf{FlagT} = 0$, since this phase is only executed when $\mathtt{BadT}$ is false. In the offline phase, the ideal oracle initially makes the following sampling:

$$(\mathsf{R}_{x_1}, \ldots, \mathsf{R}_{x_{\sigma'}}) \leftarrow_{\#} \{0,1\}^n \setminus \widetilde{\mathsf{T}},$$

where $(x_1, \ldots, x_{\sigma'})$ is an arbitrary ordering of the set

$$\mathbb{X}(\widetilde{\mathsf{X}}) := \{x : x = \mathsf{X}_i[a], (i, a) \in [q] \times [\ell_i - 1]\}.$$

Next, the ideal oracle sets

– $\mathsf{Z}_i[a] := \mathsf{R}_x$ if $x = \mathsf{X}_i[a]$, for all $(i, a) \in [q] \times [\ell_i - 1]$, and
– $\mathsf{Z}_i^{\oplus} := \bigoplus_{a=1}^{\ell_i - 1} \mathsf{Z}_i[a] \oplus \mathsf{pad}_n(\mathsf{M}_i[\ell_i])$.

At this stage we have $\mathsf{Z}_i[a] = \mathsf{Z}_j[b]$ if and only if $\mathsf{X}_i[a] = \mathsf{X}_j[b]$. In other words, $\widetilde{\mathsf{X}} \leftrightsquigarrow \widetilde{\mathsf{Z}}$. But *the same might not hold for* $\mathsf{Z}^{\oplus}$ *and* $\widetilde{\mathsf{T}}$. Now, we define four predicates on $(\widetilde{\mathsf{Z}}, \widetilde{\mathsf{X}})$:

$$\mathtt{BadZ1}: \ \exists i \neq j \in [q], \text{ such that } \mathsf{Z}_i^{\oplus} = \mathsf{Z}_j^{\oplus}.$$

$$\mathtt{BadZ2}: \ \exists (i, a) \in [q] \times [\ell_i - 1], \text{ such that } \mathsf{X}_i[a] = \mathsf{Z}_i^{\oplus}.$$

$$\mathtt{BadZ3}: \ \exists i \neq j \neq k \in [q], a \neq b \in [\ell_i - 1], \text{ such that } (\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{X}_i[b] = \mathsf{Z}_k^{\oplus}).$$

$$\mathtt{BadZ4}: \ \exists i \neq j \neq k \in [q], a \in [\ell_i - 1], b \in [\ell_j - 1], \text{ such that } (\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{X}_j[b] = \mathsf{Z}_k^{\oplus}).$$

We write $\mathtt{BadZ} := \mathtt{BadZ1} \vee \mathtt{BadZ2} \vee \mathtt{BadZ3} \vee \mathtt{BadZ4}$. Looking ahead momentarily, $\mathtt{BadZ}$ will represent bad scenarios that are difficult to fix in the third stage. For example, $\mathtt{BadZ1}$ leads to permutation incompatibility between $\mathsf{Z}^{\oplus}$ and $\widetilde{\mathsf{T}}$ which is not desirable. We will discuss utility of the other three predicates in the description of next phase.

If BadZ is true, then FlagZ is set to 1, and $\widetilde{\mathsf{Y}} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$ is again defined degenerately, as in the case of BadT. Otherwise, the ideal oracle proceeds to the next phase.

Phase III (Offline Resetting Phase): At this point, we know that BadZ is false. In this phase, we will define the complete transcript generated in the ideal world, i.e., $\mathbb{I}$, by appropriately defining $\widetilde{\mathsf{Y}}$. Remember, our goal is to maintain $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus}) \rightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$.

Definition 4.1 (full collision index). *Any query index $i \in [q]$ is called a full collision index if $\exists\ a \in [\ell_i - 1], j \in [q]$ such that $\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}$. Additionally, let*

- $\mathcal{I} := \{i \in [q] : \mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a],\ \textit{for some } a \in [\ell_i - 1], j \in [q]\}$.
- $\mathcal{J} := \{j \in [q] : \mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]\ \textit{for some } (i, a) \in [q] \times [\ell_i - 1]\}$.
- $\mathsf{FCT} := \{(i, a, j) : i, j \in [q], a \in [\ell_i - 1]\ \textit{such that}\ \mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]\}$. *Sometimes, we also use* $\widetilde{\mathsf{FCT}} := \{(i, a) \in [q] \times [\ell_i - 1] : \exists j \in [q]\ \textit{such that}\ \mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]\}$.

We refer to $i \in \mathcal{I}$ and $j \in \mathcal{J}$ as full-collision and resetting index, respectively.

Observe that we can simply set $\widetilde{\mathsf{Y}} = \widetilde{\mathsf{Z}}$, whenever $\mathcal{I} = \emptyset$, since $\neg(\mathtt{BadT} \vee \mathtt{BadZ})$ holds. However, we need a more involved method when $\mathcal{I} \neq \emptyset$. Next, we use a novel sampling approach, called *reset-sampling*, in context of the sampling for $\widetilde{\mathsf{Y}}$.

Reset-sampling: The sampling of $\widetilde{\mathsf{Y}}$ is done in two stages:

Stage 1: For all $(i, a) \in [q] \times [\ell_i - 1]$, set $\mathsf{Y}_i[a] = \mathsf{Z}_i[a]$.

Stage 2: For all $(i, a, j) \in \mathsf{FCT}$, *reset* $\mathsf{Y}_i[a] = \mathsf{T}_j$.

Finally, define $\mathsf{Y}^{\oplus} := (\mathsf{Y}_1^{\oplus}, \ldots, \mathsf{Y}_q^{\oplus})$, where $\mathsf{Y}_i^{\oplus} = \bigoplus_{a \in [q]} \mathsf{Y}_i[q] \oplus \mathsf{pad}_n(\mathsf{M}_i[\ell_i])$.

In the second stage, we have reset $\mathsf{Y}_i[a]$ from $\mathsf{Z}_i[a]$ to T_j for all $(i, a, j) \in \mathsf{FCT}$. This fixes the previous inconsistency issue, i.e., $\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}$ and $\mathsf{Y}_i[a] \neq \mathsf{T}_j$. Figure 4.1 gives a pictorial view of this step. The following must hold due to the condition $\neg$BadZ:

- For each $(i, a) \in \mathcal{I} \times [\ell_i - 1]$, there is a unique choice for j (if exists) such that $\mathsf{Y}_i[a]$ is reset to T_j. Otherwise, $\neg$BadZ1 is violated.
- Continuing the previous point, we must have $j \neq i$. Otherwise, $\neg$BadZ2 is violated. Indeed, $i = j$ incurs a trivial inconsistency: $(\mathsf{Y}_i[a] = \mathsf{T}_i) \wedge (\mathsf{X}_i[a] \neq \mathsf{Y}_i^{\oplus})$ due to the resetting mechanism.
- For each $i \in \mathcal{I}$, there exists at most one $a \in [\ell_i - 1]$, such that $\mathsf{Y}_i[a]$ is reset. Otherwise, $\neg$BadZ3 is violated.
- For all $j \in \mathcal{J}$, none of the intermediate outputs are reset. Otherwise, $\neg$BadZ4 is violated.

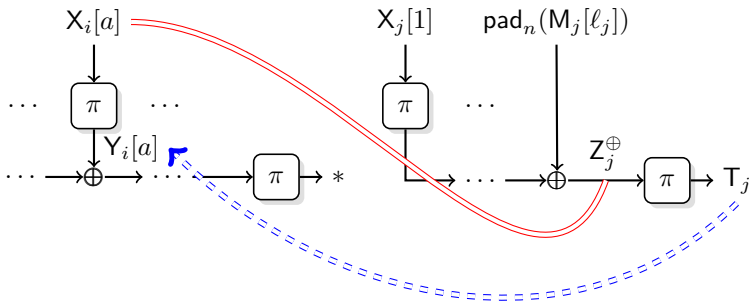

Fig. 4.1: Resetting of $\mathsf{Y}_i[a]$ due to collision $\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}$. The red double line represents a collision arising in phase II sampling. The blue dashed edge represents the corresponding resetting in phase III sampling.

To summarize, the ideal oracle ensures that for each full collision index at most one intermediate output is reset, and the resetting index is uniquely determined. Further, a full collision index cannot be a resetting index. Thus, $\neg\mathtt{BadZ}$ helps in avoiding trivial inconsistencies as well as keeping the resetting to a minimum. Now, we define two predicates on $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Z}}, \widetilde{\mathsf{Y}})$:

$$\mathtt{BadY1}: \ \exists i \neq j, k \in [q], \exists a \in [\ell_i - 1], b \in [\ell_k - 1], \text{ such that}$$

$$(\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{Y}_i^{\oplus} = \mathsf{X}_k[b]).$$

$$\mathtt{BadY2}: \ \exists i \neq j \neq k \in [q], \exists a \in [\ell_i - 1], \text{ such that } (\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{Y}_i^{\oplus} = \mathsf{Y}_k^{\oplus}).$$

We write $\mathtt{BadY} := \mathtt{BadY1} \vee \mathtt{BadY2}$. It is easy to see that $\mathtt{BadY}$ simply handles the new inconsistencies that may arise due to the reset sampling. For example, $\mathtt{BadY1}$ represents the scenario where resetting leads to collision between intermediate and final inputs. Similarly, $\mathtt{BadY2}$ represents the scenario where resetting leads to collision between two final inputs.

If $\mathtt{BadY}$ is true, then FlagY is set to 1, and $\widetilde{\mathsf{Y}}$ is redefined degenerately, as in the case of $\mathtt{BadT}$ and $\mathtt{BadZ}$. At this point, the ideal oracle transcript is completely defined.

Intuitively, if the ideal oracle is not sampling $\widetilde{\mathsf{Y}}$ degenerately at any stage, then we must have $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus}) \leftrightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$. We justify this intuition in the following proposition.

Proposition 4.1. *For* $\neg(\mathtt{BadT} \vee \mathtt{BadZ} \vee \mathtt{BadY})$*, we must have* $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus}) \leftrightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$.

Proof. We have

– $\widetilde{\mathsf{X}} \leftrightsquigarrow \widetilde{\mathsf{Z}}$, by definition of $\widetilde{\mathsf{Z}}$. Moreover the resetting guarantees $\widetilde{\mathsf{Z}} \leftrightsquigarrow \widetilde{\mathsf{Y}}$. Thus, $\widetilde{\mathsf{X}} \leftrightsquigarrow \widetilde{\mathsf{Y}}$.

– We have $\mathsf{Y}_i[a] = \mathsf{T}_j$ if and only if $\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}$. Now, $\neg\mathtt{BadZ4}$ implies that $j \notin \mathcal{I}$ thus, $\mathsf{Y}_j^{\oplus} = \mathsf{Z}_j^{\oplus}$. Therefore, $\mathsf{Y}_i[a] = \mathsf{T}_j \Rightarrow \mathsf{X}_i[a] = \mathsf{Y}_j^{\oplus}$. Also, $\mathsf{X}_i[a] = \mathsf{Y}_j^{\oplus}$ implies $j \notin \mathcal{I}$ (due to $\neg\mathtt{BadY1}$), thus, $\mathsf{Z}_j^{\oplus} = \mathsf{Y}_j^{\oplus}$. This gives us $\mathsf{X}_i[a] = \mathsf{Y}_j^{\oplus} \Rightarrow \mathsf{Y}_i[a] = \mathsf{T}_j$ from the second stage sampling of Y. Thus, $\mathsf{X}_i[a] = \mathsf{Y}_j^{\oplus} \Leftrightarrow \mathsf{Y}_i[a] = \mathsf{T}_j$.

– $\neg\mathtt{BadZ} \wedge \neg\mathtt{BadY}$ and definition of Y imply that $\mathsf{Y}_i^{\oplus}$'s are distinct. Also, $\neg\mathtt{BadT}$ implies that T_i's are distinct. Thus $\widetilde{\mathsf{Y}}^{\oplus} \leftrightsquigarrow \widetilde{\mathsf{T}}$.

These observations suffice to conclude that $(\widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}^{\oplus}) \leftrightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$. □

4.2 Transcript Analysis

Set of Transcripts: Given the description of transcript random variable corresponding to the ideal oracle, we can define the set of transcripts $\mathcal{T}$ as the set of all tuples $\tau = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y}, \mathrm{flagT}, \mathrm{flagZ}, \mathrm{flagY})$, where

– $\widetilde{m} = (m_1, \ldots, m_q)$, where $m_i \in \left(\{0,1\}^{\leq (n-s)2^s}\right)$ for $i \in [q]$. For $i \in [q]$, let $\ell_i = \left\lfloor \frac{|m_i|}{n-s} \right\rfloor + 1$.

– $\widetilde{t} = (t_1, \ldots, t_q)$, where $t_i \in \{0,1\}^n$ for $i \in [q]$;

– $\widetilde{x} = (x_1, \ldots, x_q)$, where $x_i = (x_i[1], \ldots, x_i[\ell_i - 1])$ for $i \in [q]$, and $x_i[a] = \langle a \rangle_s \| m_i[a]$ for all $a \in [\ell_i - 1]$;

– $\widetilde{y} = (y_1, \ldots, y_q)$, where $y_i = (y_i[1], \ldots, y_i[\ell_i - 1])$ for $i \in [q]$, and $y_i[a] \in \{0,1\}^n$ for all $a \in [\ell_i - 1]$.

– $\mathrm{flagT}, \mathrm{flagZ}, \mathrm{flagY} \in \{0,1\}$.

Furthermore, the following must always hold:

1. if $\mathrm{flagI} = 1$ for some $\mathrm{I} \in \{\mathrm{T}, \mathrm{Z}, \mathrm{Y}\}$, then $y_i[a] = 0^n$ for all $(i,a) \in [q] \times [\ell_i - 1]$.
2. if $\mathrm{flagT} = 0$, then t_i's are all distinct.
3. if $\mathrm{flagI} = 0$ for all $\mathrm{I} \in \{\mathrm{T}, \mathrm{Z}, \mathrm{Y}\}$, then $(\widetilde{x}, \widetilde{y}^{\oplus}) \leftrightsquigarrow (\widetilde{y}, \widetilde{t})$.

The first two conditions are obvious from the ideal oracle sampling mechanism. The last condition follows from Proposition 4.1 and the observation that in ideal oracle sampling for any $\mathsf{I} \in \{\mathsf{T}, \mathsf{Z}, \mathsf{Y}\}$, $\mathsf{FlagI} = 1$ if and only if $\mathtt{BadI}$ is true. Note that, condition 3 is vacuously true for real oracle transcripts.

Bad Transcript: A transcript $\tau \in \mathcal{T}$ is called *bad* if and only if the following predicate is true:

$$(\mathsf{FlagT} = 1) \vee (\mathsf{FlagZ} = 1) \vee (\mathsf{FlagY} = 1).$$

In other words, we term a transcript bad if the ideal oracle sets $\widetilde{\mathsf{Y}}$ degenerately. Let

$$\mathcal{T}_{\mathsf{bad}} := \{\tau \in \mathcal{T} : \tau \text{ is bad.}\}.$$

All other transcript $\tau' = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y}, \mathrm{flagT}, \mathrm{flagZ}, \mathrm{flagY}) \in \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$ are called *good*. From the preceding characterization of the set of transcripts, we conclude that for any good transcript τ', we must have $(\widetilde{x}, \widetilde{y}^{\oplus}) \leftrightsquigarrow (\widetilde{y}, \widetilde{t})$. Henceforth, we drop flagT, flagZ, flagY notations for any good transcript with an implicit understanding that $\mathrm{flagT} = \mathrm{flagZ} = \mathrm{flagY} = 0$.

To apply the H-coefficient theorem we have to upper bound the probability $\Pr[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}]$ and lower bound the ratio $\Pr[\mathbb{R} = \tau]/\Pr[\mathbb{I} = \tau]$ for any $\tau \in \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$.

Lemma 4.1 (bad transcript analysis). *For $4\ell_{\max} + q \le 2^{n-1}$, we have*

$$\Pr[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}] \le \frac{3q^2}{2^{n+1}} + \frac{2.5q^3\ell_{\max}^2}{2^{2n}} + \frac{4q^3\ell_{\max}}{2^{2n}} + \frac{4q^4\ell_{\max}^2}{2^{3n}} + \frac{2\sigma}{2^n}.$$

The proof of this lemma is postponed to section 4.3.

Good Transcript: Now, fix a good transcript $\tau = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y})$. Let $\sigma' := |\widetilde{x}|$. Since, τ is good, we have $(\widetilde{x}, \widetilde{y}^{\oplus}) \leftrightsquigarrow (\widetilde{y}, \widetilde{t})$. Then, we must have $|\widetilde{y}^{\oplus}| = q$. Further, let $|\widetilde{x} \cap \widetilde{y}^{\oplus}| = r$. Thus, $|\widetilde{x} \cup \widetilde{y}^{\oplus}| = q + \sigma' - r$.

Real world: In the real world, the random permutation Π is sampled on exactly $q + \sigma' - r$ distinct points. Thus, we have

$$\Pr[\mathbb{R} = \tau] = \frac{1}{(2^n)_{q+\sigma'-r}}. \tag{8}$$

Ideal world: Here, the probability computation is slightly involved due to the two stage sampling employed in the ideal oracle. First of all, we have

$$\Pr\left[\widetilde{\mathsf{T}} = \widetilde{t}\right] = \frac{1}{2^{nq}}, \tag{9}$$

since each T_i is sampled uniformly from the set $\{0,1\}^n$ independent of others. Now, observe that all the full collision and resetting indices are fully determined from the transcript τ itself. In other words, we can enumerate the set $\widetilde{\mathsf{FCT}}$. Now, since the transcript is good, we must have $|\widetilde{\mathsf{FCT}}| = |\widetilde{x} \cap \widetilde{y}^{\oplus}| = r$, and for all indices $(i,a) \notin \widetilde{\mathsf{FCT}}$, we have $\mathsf{Y}_i[a] = \mathsf{Z}_i[a]$. Thus, we have

$$\begin{aligned}\Pr\left[\mathsf{Y}_i[a] = y_a^i \wedge (i,a) \notin \widetilde{\mathsf{FCT}} \mid \widetilde{\mathsf{T}} = \widetilde{t}\right] &= \Pr\left[\mathsf{Z}_i[a] = y_a^i \wedge (i,a) \notin \widetilde{\mathsf{FCT}} \mid \widetilde{\mathsf{T}} = \widetilde{t}\right] \\ &= \frac{1}{(2^n - q)_{\sigma'-r}},\end{aligned} \tag{10}$$

where the second equality follows from the fact that truncation[3] of a without replacement sample from a set of size $(2^n - q)$ is still a without replacement sample from the same set. We have

$$\Pr[\mathbb{I} = \omega] = \Pr\left[\widetilde{\mathsf{T}} = \widetilde{t}\right] \times \Pr\left[\widetilde{\mathsf{Y}} = \widetilde{y} \mid \widetilde{\mathsf{T}} = \widetilde{t}\right]$$

[3] Removing some elements from the tuple.

$$\leq \frac{1}{2^{nq}} \times \Pr\left[\mathsf{Y}_i[a] = y_i[a] \wedge (i,a) \notin \widetilde{\mathsf{FCT}} \mid \widetilde{\mathsf{T}} = \widetilde{t}\right]$$
$$= \frac{1}{2^{nq}} \times \frac{1}{(2^n - q)_{\sigma' - r}}. \tag{11}$$

The above discussion on good transcripts can be summarized in shape of the following lemma.

Lemma 4.2. *For any $\tau \in \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$, we have*

$$\frac{\Pr[\mathbb{R} = \tau]}{\Pr[\mathbb{I} = \tau]} \geq 1.$$

Proof. The proof follows from dividing Eq. (8) by Eq. (11). □

From H-coefficient Theorem 2.1 and Lemma 4.1 and 4.2, we get

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{1k\text{-}LightMAC}_\Pi}(\mathscr{A}) \leq \frac{3q^2}{2^{n+1}} + \frac{2.5q^3\ell^2_{\max}}{2^{2n}} + \frac{4q^3\ell_{\max}}{2^{2n}} + \frac{4q^4\ell^2_{\max}}{2^{3n}} + \frac{2\sigma}{2^n}. \tag{12}$$

Theorem 4.1 follows from Eq. (7) and (12).

4.3 Proof of Lemma 4.1

We have

$$\begin{aligned}\Pr[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}] &= \Pr[(\mathsf{FlagT} = 1) \vee (\mathsf{FlagZ} = 1) \vee (\mathsf{FlagY} = 1)] \\ &= \Pr[\mathtt{BadT} \vee \mathtt{BadZ} \vee \mathtt{BadY}] \\ &\leq \Pr[\mathtt{BadT}] \times \Pr[\mathtt{BadZ} | \neg \mathtt{BadT}] \times \Pr[\mathtt{BadY} | \neg(\mathtt{BadT} \vee \mathtt{BadZ})]\end{aligned}$$

We will handle the three terms on the right hand side separately. Before delving further, we introduce few more notations.

FEW MORE NOTATIONS: For simplicity, we denote the last padded block of any message m_i by $m_i[\ell_i]$ instead of $\mathsf{pad}_n(m_i[\ell_i])$. For any (i,a) with $i \in [q], a \in [\ell_i]$, $\mathsf{Z}_i^{\oplus \setminus a}$ (res. $\mathsf{Y}_i^{\oplus \setminus a}$) denotes $\bigoplus_{b \neq a} \mathsf{Z}_i[b] \oplus m_i[\ell_i]$ (res. $\bigoplus_{b \neq a} \mathsf{Y}_i[b] \oplus m_i[\ell_i]$).

While we bound the probability of bad events, we need to deal with system of equations in Z variables. Note that Z can be viewed as $\Pi(\mathsf{X})$ for the corresponding X variable. We will often employ Lemma 2.1 implicitly (without referring at each application) to bound the probability that these system of equations hold.

1. Bounding $\Pr[\mathtt{BadT}]$: Since, we have at most $\binom{q}{2}$ choice for i, j, and for each such pair, $\mathsf{T}_i = \mathsf{T}_j$ holds with exactly 2^{-n} probability. Thus, we have
$$\Pr[\mathtt{BadT}] \leq \frac{q^2}{2^{n+1}}. \tag{13}$$
2. Bounding $\Pr[\mathtt{BadZ} | \neg\mathtt{BadT}]$: Here, we have four cases.

(a) $\texttt{BadZ1} : \exists i \neq j \in [q]$, such that $\mathsf{Z}_i^{\oplus} = \mathsf{Z}_j^{\oplus}$. This is similar to $\texttt{BadT}$ above. We have

$$\Pr\left[\texttt{BadZ1}|\neg\texttt{BadT}\right] \leq \frac{q^2}{2 \cdot (2^n - q - 2\ell_{\max})}.$$

(b) $\texttt{BadZ2} : \exists (i, a) \in [q] \times [\ell_i - 1]$, such that $\mathsf{X}_i[a] = \mathsf{Z}_i^{\oplus}$. It is easy to see that

$$\Pr\left[\texttt{BadZ2}|\neg\texttt{BadT}\right] \leq \sum_{i=1}^{q} \frac{\ell_i - 1}{2^n - q - \ell_{\max}} \leq \frac{\sigma}{2^n - q - \ell_{\max}}.$$

(c) $\texttt{BadZ3} : \exists i \neq j \neq k \in [q], a, b \in [\ell_i - 1]$, such that $(\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{X}_i[b] = \mathsf{Z}_k^{\oplus})$. Here, $j \neq k$ implies that the system of equations has rank 2. Thus, using Lemma 2.1, we have

$$\Pr\left[\texttt{BadZ3}|\neg\texttt{BadT}\right] \leq \frac{q^3\ell_{\max}^2}{12(2^n - q - 2\ell_{\max})^2}.$$

(d) $\texttt{BadZ4} : \exists i \neq j \neq k \in [q], a \in [\ell_i - 1], b \in [\ell_j - 1]$, such that $(\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{X}_j[b] = \mathsf{Z}_k^{\oplus})$. Using similar argumentation as above, we have,

$$\Pr\left[\texttt{BadZ4}|\neg\texttt{BadT}\right] \leq \frac{q^3\ell_{\max}^2}{12(2^n - q - 2\ell_{\max})^2}.$$

Combining all the four cases and assuming $q + 2\ell_{\max} \leq 2^{n-1}$, we have

$$\Pr\left[\texttt{BadZ}|\neg\texttt{BadT}\right] \leq \frac{q^2}{2^n} + \frac{0.34q^3\ell_{\max}^2}{2^{2n}} + \frac{2\sigma}{2^n} \tag{14}$$

3. Bounding $\Pr\left[\texttt{BadY}|\neg(\texttt{BadT} \vee \texttt{BadZ})\right]$: Here, we have two cases:

(a) $\texttt{BadY1} : \exists i, j, k \in [q], \exists a \in [\ell_i - 1], b \in [\ell_k - 1]$ such that $(\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{Y}_i^{\oplus} = \mathsf{X}_k[b])$. By virtue of resetting mechanism and $\neg\texttt{BadZ}$, we arrive at an equivalent system of Z-equations

$$\mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]$$
$$\mathsf{Z}_i^{\oplus\setminus a} = \mathsf{X}_k[b] \oplus \mathsf{T}_j$$

We claim that the system always has rank 2. This can be argued as follows: Suppose the system has rank less than 2. Then, we must have $\mathsf{Z}_j^{\oplus} \oplus \mathsf{X}_i[a] \oplus \mathsf{Z}_i^{\oplus\setminus a} \oplus \mathsf{X}_k[b] \oplus \mathsf{T}_j = 0^n$. However, $\widetilde{\mathsf{Z}}$ are sampled from $\{0,1\}^n \setminus \widetilde{\mathsf{T}}$. Hence, T_j does not cancel out trivially. So, we must always have rank 2. Now if the rank is 2, then we can always rewrite the system of equations such that we have an equation in T_j and another equation involving some Z variables. Then, the first equation holds with at most $1/2^n$ probability (using the randomness of T_j) and conditioned on this

the second equation holds with probability at most $1/(2^n - q - 2\ell_{\max})$. Thus, we have

$$\Pr\left[\mathtt{BadY1}|\neg(\mathtt{BadT} \vee \mathtt{BadZ})\right] \leq \frac{q^3\ell_{\max}^2}{2^n(2^n - q - 2\ell_{\max})}.$$

(b) $\mathtt{BadY2} : \exists i, j, k \in [q], \exists a \in [\ell_i - 1]$, such that $(\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus}) \wedge (\mathsf{Y}_i^{\oplus} = \mathsf{Y}_k^{\oplus})$. Here we get $\mathsf{X}_i[a] = \mathsf{Z}_j^{\oplus} \wedge \mathsf{Z}_i^{\oplus\setminus a} = \mathsf{Y}_k^{\oplus} \oplus \mathsf{T}_j$ which changes according to the following subcases:

Case A: when $k \notin \mathcal{I}$: Then the above system becomes

$$\mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]$$
$$\mathsf{Z}_i^{\oplus\setminus a} = \mathsf{Z}_k^{\oplus} \oplus \mathsf{T}_j$$

Using similar argumentation as before we can conclude that the system has rank 2. Therefore, we have

$$\Pr\left[\mathtt{BadY2} \wedge \text{Case A}|\neg(\mathtt{BadZ} \vee \mathtt{BadT})\right] \leq \frac{q^3\ell_{\max}}{(2^n - q - 3\ell_{\max})^2}.$$

Case B: when $k \in \mathcal{I}$: In this case we have the following system of equations:

$$\mathsf{Z}_j^{\oplus} = \mathsf{X}_i[a]$$
$$\mathsf{Z}_l^{\oplus} = \mathsf{X}_k[b]$$
$$\mathsf{Z}_i^{\oplus\setminus a} \oplus \mathsf{Z}_k^{\oplus\setminus b} = \mathsf{T}_j \oplus \mathsf{T}_l$$

We must have $j \neq l$. Otherwise we will have $\mathsf{Z}_i^{\oplus} = \mathsf{Z}_k^{\oplus}$ which again violates $\neg\mathtt{BadZ}$. Thus, $j \neq l$. Now, $j \neq l$ and $\neg\mathtt{BadZ}$ implies that $\mathsf{Z}_j^{\oplus} \neq \mathsf{Z}_l^{\oplus}$. Then, following a similar line of argument as before, we conclude that the system has rank 3. Therefore, we have

$$\Pr\left[\mathtt{BadY2} \wedge \text{Case B}|\neg(\mathtt{BadZ} \vee \mathtt{BadT})\right] \leq \frac{q^4\ell_{\max}^2}{2^n(2^n - q - 4\ell_{\max})^2}.$$

Combining all the cases with the assumption that $q + 4\ell_{\max} \leq 2^{n-1}$, we have

$$\Pr\left[\mathtt{BadY}|\neg(\mathtt{BadT} \vee \mathtt{BadZ})\right] \leq \frac{2q^3\ell_{\max}^2}{2^{2n}} + \frac{4q^3\ell_{\max}}{2^{2n}} + \frac{4q^4\ell_{\max}^2}{2^{3n}}. \tag{15}$$

The result follows from summing up Eq. (13)-(15). □

5 LightMAC-ds: A Minute Variant of Single-key LightMAC

In the previous section we showed that single-key LightMAC achieves query-length independent security bounds while $\ell_{\min} \geq 2$ and $\ell_{\max} \leq 2^{n/4}$. Now, we propose a simple variant of LightMAC that achieves query-length independent security unconditionally.

5.1 Description of LightMAC-ds

For any $x \in \{0,1\}^n$ and $k < n$, let $\mathsf{chop}_k(x)$ denote the most significant $n-k$ bits of x. The complete algorithmic description of LightMAC-ds is given in Algorithm 5.1. It is clear from the description that LightMAC-ds uses the familiar technique

Algorithm 5.1 LightMAC-ds based on an n-bit block cipher E instantiated with a single key K. Here the counter size is $s-1$. Highlighted lines point to the algorithmic differences with the LightMAC algorithm.

1: **function** LightMAC-ds$_{E_K}(m)$
2: $\quad y^{\oplus} \leftarrow 0^n$
3: $\quad (m[1], \ldots, m[\ell]) \xleftarrow{n-s} m$
4: $\quad$ **for** $i = 1$ **to** $\ell - 1$ **do**
5: $\quad\quad x[i] \leftarrow 0 \| \langle i \rangle_{s-1} \| m[i]$ ▷ encoding $\langle i \rangle_{s-1}$ and $m[i]$ into $x[i]$
6: $\quad\quad y[i] \leftarrow E_K(x[i])$ ▷ encrypting the encoded input
7: $\quad\quad y^{\oplus} \leftarrow y^{\oplus} \oplus y[i]$ ▷ accumulating the intermediate output
8: $\quad$ **end for**
9: $\quad y^{\oplus} \leftarrow y^{\oplus} \oplus \mathsf{pad}_n(m[\ell])$
10: $\quad t \leftarrow E_K(1 \| \mathsf{chop}_1(y^{\oplus}))$
11: $\quad$ **return** t
12: **end function**

of domain separation to generate two "almost independent" instances of E. Specifically, we fix the most significant 1-bit of the block cipher input to

- 0 in the processing of encoded message blocks (see line no. 5 in Algorithm 5.1).
- 1 in the tag generation call (see line no. 10 in Algorithm 5.1).

Since 1-bit is reserved for domain separation, the effective counter size is reduced to $s-1$ for some global parameter $s < n$. Thus, the maximum message length can be at most $(n-s)2^{s-1}$, which is a slight drop from $(n-s)2^s$ in case of LightMAC, for large value of n and s.

5.2 Security of LightMAC-ds

Surprisingly (or not), the security argument for LightMAC-ds is quite similar to the one for single-key LightMAC. In fact, it is slightly easy to argue the security here, as we have already ensured $\neg$`Icoll` (see section 3.2) by the virtue of domain separation. However, we still have to handle `Ocoll` (see section 3.2) which would require a slight care while sampling the intermediate outputs in the ideal world. Note that, such complications do not arise in case of LightMAC for the obvious reason of independence between the primitives used to generate the intermediate and final outputs. The PRF security of LightMAC-ds is presented in Theorem 5.1.

Theorem 5.1. *Let $q, \ell_{\max}, T > 0$. For $q + 2\ell_{\max} \leq 2^{n-1}$, the PRF security of $\mathscr{A}$ against $\mathbb{A}(q,T)$ is given by*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}}(q,T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma+q,T') + \frac{2.5q^2}{2^n},$$

where ℓ denotes an upper bound on the number of blocks in any padded query, $T' = T + O(T_E)$ and T_E denotes the runtime of E.

As expected, the proof is quite similar and a bit easier than the proof of theorem 4.1. As the first step, we apply the hybrid argument to get

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}}(q,T) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(\sigma+q,T') + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}_{\Pi}}(q,\infty). \tag{16}$$

We are interested in a bound on the PRF security of $\mathsf{LightMAC\text{-}ds}_{\Pi}$, henceforth also referred as the real oracle. Fix any $\mathscr{A} \in \mathbb{A}(q,\infty)$ such that

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}_{\Pi}}(q,\infty) = \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}_{\Pi}}(\mathscr{A}).$$

Going forward, we will bound the advantage of $\mathscr{A}$ using H-coefficient technique.

5.3 Description of Oracles and their Transcripts

Real Oracle: The real oracle is defined analogously as in the proof of Theorem 5.1. We describe it just for the sake of completeness. The real oracle faithfully responds to all the queries made by $\mathscr{A}$. Once the query-response phase is over, it releases all the intermediate inputs and outputs to $\mathscr{A}$. Additionally, the real oracle releases two binary flags, FlagT and FlagZ, that are degenerately set to 0. Formally, we have

$$\mathbb{R} := (\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}, \mathsf{FlagT}, \mathsf{FlagZ}),$$

where

- $\widetilde{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_q)$ denotes the q-tuple of queries made by $\mathscr{A}$, where $\mathsf{M}_i \in \{0,1\}^{\leq (n-s)2^{s-1}}$ for all $i \in [q]$. In addition, for all $i \in [q]$, let $\ell_i := \left\lfloor \frac{|\mathsf{M}_i|}{n-s} \right\rfloor + 1$.
- $\widetilde{\mathsf{T}} = (\mathsf{T}_1, \ldots, \mathsf{T}_q)$ denotes the q-tuple of final outputs received by $\mathscr{A}$, where $\mathsf{T}_i \in \{0,1\}^n$.
- $\widetilde{\mathsf{X}} = (\mathsf{X}_1, \ldots, \mathsf{X}_q)$, where X_i denotes the intermediate input tuple for the i-th query, i.e., for all $a \in [\ell_i - 1]$, $\mathsf{X}_i[a] = 0 \| \langle a \rangle_{s-1} \| \mathsf{M}_i[a]$.
- $\widetilde{\mathsf{Y}} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$, where Y_i denotes the intermediate output tuple for the i-th query, i.e., for all $a \in [\ell_i - 1]$, $\mathsf{Y}_i[a] = \Pi(\mathsf{X}_i[a])$. In addition, let $\widetilde{\mathsf{Y}}^{\oplus} := (\mathsf{Y}_1^{\oplus}, \ldots, \mathsf{Y}_q^{\oplus})$, where $\mathsf{Y}_i^{\oplus} := \bigoplus_{a \in [\ell_i - 1]} \mathsf{Y}_i[a] \oplus \mathsf{pad}_n(\mathsf{M}_i[\ell_i])$ for all $i \in [q]$.
- $\mathsf{FlagT} = \mathsf{FlagZ} = 0$.

Let $\mathsf{chop}_1(\widetilde{\mathsf{Y}}^{\oplus}) = (1 \| \mathsf{chop}_1(\mathsf{Y}_i[1]), \ldots, 1 \| \mathsf{chop}_1(\mathsf{Y}_i[\ell_i - 1]))$. It is straightforward to see that in the real world we always have $(\widetilde{\mathsf{X}}, \mathsf{chop}_1(\widetilde{\mathsf{Y}}^{\oplus})) \leftrightsquigarrow (\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$, i.e., $(\widetilde{\mathsf{X}}, \mathsf{chop}_1(\widetilde{\mathsf{Y}}^{\oplus}))$ is permutation compatible with $(\widetilde{\mathsf{Y}}, \widetilde{\mathsf{T}})$.

Ideal oracle: We reuse the notations from real oracle description to represent the variables in the ideal oracle transcript $\mathbb{I}$, i.e.

$$\mathbb{I} := (\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}, \mathsf{FlagT}, \mathsf{FlagZ}).$$

The ideal oracle transcript is described in two phases, with the second one contingent on some predicate defined over the first stage. Specifically, the ideal oracle initializes $\mathsf{FlagT} = \mathsf{FlagZ} = 0$, and then follows the sampling mechanism given below:

Phase I (Query-Response Phase): In the query-response phase, the ideal oracle faithfully simulates $\Gamma \leftarrow\!\!\$\, \mathcal{F}(\{0,1\}^{\leq (n-s)2^{s-1}}, \{0,1\}^n)$. Formally, for $i \in [q]$, at the i-th query $\mathsf{M}_i \in \{0,1\}^{\leq (n-s)2^{s-1}}$, the ideal oracle outputs $\mathsf{T}_i \leftarrow\!\!\$\, \{0,1\}^n$. The partial transcript generated at the end of the query-response phase is given by $(\widetilde{\mathsf{M}}, \widetilde{\mathsf{T}}, \widetilde{\mathsf{X}})$, where

- $\widetilde{\mathsf{M}} = (\mathsf{M}_1, \ldots, \mathsf{M}_q)$ and $\widetilde{\mathsf{T}} = (\mathsf{T}_1, \ldots, \mathsf{T}_q)$.
- $\widetilde{\mathsf{X}} = (\mathsf{X}_1, \ldots, \mathsf{X}_q)$, where $\mathsf{X}_i = (\mathsf{X}_i[1], \ldots, \mathsf{X}_i[\ell_i - 1])$ and $\mathsf{X}_i[a] := 0 \| \langle a \rangle_{s-1} \| \mathsf{M}_i[a]$ for all $(i, a) \in [q] \times [\ell_i - 1]$.

Now, we define a predicate on $\widetilde{\mathsf{T}}$:

$$\mathtt{BadT}: \ \exists i \neq j \in [q], \text{ such that } \mathsf{T}_i = \mathsf{T}_j.$$

If $\mathtt{BadT}$ is true, then $\mathsf{FlagT} = 1$, and $\widetilde{\mathsf{Y}} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$ is defined degenerately: $\mathsf{Y}_i[a] = 0^n$ for all $(i, a) \in [q] \times [\ell_i - 1]$. Otherwise, the ideal oracle proceeds to the next phase.

Phase II (Offline Sampling Phase): In the offline phase, the ideal oracle initially makes the following sampling:

$$(\mathsf{R}_{x_1}, \ldots, \mathsf{R}_{x_t}) \leftarrow\!\!\#\, \{0,1\}^n \setminus \widetilde{\mathsf{T}},$$

where $(x_1, \ldots, x_t)$ is an arbitrary ordering of the set

$$\mathbb{X}(\widetilde{\mathsf{X}}) := \{x : x = \mathsf{X}_i[a], (i, a) \in [q] \times [\ell_i - 1]\}.$$

Next, the ideal oracle sets

- $\mathsf{Z}_i[a] := \mathsf{R}_x$ if $x = \mathsf{X}_i[a]$, for all $(i, a) \in [q] \times [\ell_i - 1]$, and
- $\mathsf{Z}_i^{\oplus} := \bigoplus_{a=1}^{\ell_i - 1} \mathsf{Z}_i[a] \oplus \mathsf{pad}_n(\mathsf{M}_i[\ell_i])$.

At this stage we have $\mathsf{Z}_i[a] = \mathsf{Z}_j[b]$ if and only if $\mathsf{X}_i[a] = \mathsf{X}_j[b]$. In other words, $\widetilde{\mathsf{X}} \rightsquigarrow \widetilde{\mathsf{Z}}$. But *the same might not hold for* $\mathsf{chop}_1(\widetilde{\mathsf{Z}}^{\oplus})$ *and* $\widetilde{\mathsf{T}}$. Now, we define a predicate on $(\widetilde{\mathsf{Z}}, \widetilde{\mathsf{X}})$:

$$\mathtt{BadZ}: \ \exists i \neq j \in [q], \text{ such that } \mathsf{chop}_1(\mathsf{Z}_i^{\oplus}) = \mathsf{chop}_1(\mathsf{Z}_j^{\oplus}).$$

Note that, $\neg\mathtt{BadZ}$ ensures $\mathsf{chop}_1(\widetilde{\mathsf{Z}}^{\oplus}) \leftrightsquigarrow \widetilde{\mathsf{T}}$, that when coupled with the $\widetilde{\mathsf{X}} \leftrightsquigarrow \widetilde{\mathsf{Z}}$ due to the sampling mechanism ensures $(\widetilde{\mathsf{X}}, \mathsf{chop}_1(\widetilde{\mathsf{Z}}^{\oplus})) \leftrightsquigarrow (\widetilde{\mathsf{Z}}, \widetilde{\mathsf{T}})$. Intuitively, this makes the ideal world almost similar to the real world.

If $\mathtt{BadZ}$ is true, then $\mathsf{FlagZ} = 1$, and $\widetilde{\mathsf{Y}} := (\mathsf{Y}_1, \ldots, \mathsf{Y}_q)$ is again defined degenerately, as in the case of $\mathtt{BadT}$. Otherwise, $\widetilde{\mathsf{Y}} := \widetilde{\mathsf{Z}}$. At this point, the transcript random variable for the ideal world is completely determined.

5.4 Transcript Analysis

Set of Transcripts: Given the description of the transcript random variable corresponding to the ideal oracle, we can define the set of transcripts $\mathcal{T}$ as the set of all tuples $\tau = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y}, \mathrm{flagT}, \mathrm{flagZ})$, where

- $\widetilde{m} = (m_1, \ldots, m_q)$, where $m_i \in \left(\{0,1\}^{\leq (n-s)2^{s-1}}\right)$ for $i \in [q]$. Let $\ell_i = \left\lfloor \frac{|m_i|}{n-s} \right\rfloor + 1$ for $i \in [q]$.
- $\widetilde{t} = (t_1, \ldots, t_q)$, where $t_i \in \{0,1\}^n$ for $i \in [q]$;
- $\widetilde{x} = (x_1, \ldots, x_q)$, where $x_i = (x_i[1], \ldots, x_i[\ell_i - 1])$ for $i \in [q]$, and $x_i[a] = 0 \| \langle a \rangle_{s-1} \| m_i[a]$ for all $a \in [\ell_i - 1]$;
- $\widetilde{y} = (y_1, \ldots, y_q)$, where $y_i = (y_i[1], \ldots, y_i[\ell_i - 1])$ for $i \in [q]$, and $y_i[a] \in \{0,1\}^n$ for all $a \in [\ell_i - 1]$.
- $\mathrm{flagT}, \mathrm{flagZ} \in \{0,1\}$.

Furthermore, the following must always hold:

1. if $\mathrm{flagI} = 1$ for some $\mathrm{I} \in \{\mathrm{T}, \mathrm{Z}\}$, then $y_i[a] = 0^n$ for all $(i,a) \in [q] \times [\ell_i - 1]$.
2. if $\mathrm{flagT} = 0$, then t_i's are all distinct.
3. if $\mathrm{flagI} = 0$ for all $\mathrm{I} \in \{\mathrm{T}, \mathrm{Z}\}$, then $(\widetilde{x}, \mathsf{chop}_1(\widetilde{\mathsf{Y}}^{\oplus})) \leftrightsquigarrow (\widetilde{y}, \widetilde{t})$.

Bad Transcript: A transcript $\tau \in \mathcal{T}$ is called *bad* if and only if the following predicate is true:

$$(\mathsf{FlagT} = 1) \vee (\mathsf{FlagZ} = 1).$$

In other words, we term a transcript bad if the ideal oracle sets $\widetilde{\mathsf{Y}}$ degenerately. Let

$$\mathcal{T}_{\mathsf{bad}} := \{\tau \in \mathcal{T} : \tau \text{ is bad.}\}.$$

All other transcript $\tau' = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y}, \mathrm{flagT}, \mathrm{flagZ}) \in \mathcal{T} \setminus \mathcal{T}_{\mathsf{bad}}$ are called *good*. It is pretty straightforward to deduce that for any good transcript we must have $(\widetilde{x}, \mathsf{chop}_1(\widetilde{y}^{\oplus})) \leftrightsquigarrow (\widetilde{y}, \widetilde{t})$.

Lemma 5.1 (bad transcript analysis). *For $q + 2\ell_{\max} \leq 2^{n-1}$, we have*

$$\Pr[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}] \leq \frac{2.5q^2}{2^n}.$$

Proof. We have

$$\begin{aligned}\Pr\left[\mathbb{I} \in \mathcal{T}_{\mathsf{bad}}\right] &= \Pr\left[(\mathsf{FlagT} = 1) \vee (\mathsf{FlagZ} = 1)\right] \\ &= \Pr\left[\mathtt{BadT} \vee \mathtt{BadZ}\right] \\ &\leq \Pr\left[\mathtt{BadT}\right] \times \Pr\left[\mathtt{BadZ}|\mathtt{BadT}\right].\end{aligned}$$

We will handle the two terms on the right hand side separately:

1. Bounding $\Pr\left[\mathtt{BadT}\right]$: Since, we have at most $\binom{q}{2}$ choice for i, j, and for each such pair, $\mathsf{T}_i = \mathsf{T}_j$ holds with exactly 2^{-n} probability. Thus, we have

$$\Pr\left[\mathtt{BadT}\right] \leq \frac{q^2}{2^{n+1}}. \tag{17}$$

2. Bounding $\Pr\left[\mathtt{BadZ}|\neg\mathtt{BadT}\right]$: Fix two indices $i \neq j$. Now, we can have two cases:

   (a) $\ell_i = \ell_j$: Since $\mathsf{M}_i \neq \mathsf{M}_j$, we must have at least one index a, such that $\mathsf{M}_i[a] \neq \mathsf{M}_j[a]$, which implies that $\mathsf{X}_i[a] \neq \mathsf{X}_j[a]$. Further, note that $\mathsf{X}_i[a] \neq \mathsf{X}_k[b]$ for all $(k, b) \in \{i, j\} \times [\ell_k - 1]$. Then, by conditioning on the value of $\mathsf{Z}_k[b]$ for all $(k, b) \in \{i, j\} \times [\ell_k - 1] \setminus \{(i, a)\}$, we bound the probability that $\mathsf{chop}_1(\mathsf{Z}_i^{\oplus}) = \mathsf{chop}_1(\mathsf{Z}_i^{\oplus})$ to at most $2/(2^n - q - 2\ell_{\max})$, where the factor of 2 in the numerator is due to 1-bit chopping. There are at most $\binom{q}{2}$ choices for i, j, so in this case the probability is at most $q^2/(2^n - q - 2\ell_{\max})$.

   (b) $\ell_i \neq \ell_j$: Without loss of generality we assume that $\ell_i > \ell_j$. Then, applying exactly the same argumentation as used in the preceding case with $(i, a) = (i, \ell_i - 1)$, we can bound the probability in this case to at most $q^2/(2^n - q - 2\ell_{\max})$.

   Since the two cases are mutually exclusive, we have

$$\Pr\left[\mathtt{BadZ}|\neg\mathtt{BadT}\right] \leq \frac{q^2}{(2^n - q - 2\ell_{\max})}. \tag{18}$$

The result follows by summing up Eq. (17) and (18) and using $q + 2\ell_{\max} \leq 2^{n-1}$. □

Good Transcript: Fix a good transcript $\tau = (\widetilde{m}, \widetilde{t}, \widetilde{x}, \widetilde{y}, 0, 0)$. Let $\sigma' := |\widetilde{x}|$. Since, τ is good, we have $(\widetilde{x}, \mathsf{chop}_1(\widetilde{y}^{\oplus})) \rightsquigarrow (\widetilde{y}, \widetilde{t})$. Then, we must have $|\mathsf{chop}_1(\widetilde{y}^{\oplus})| = q$. Further, $\widetilde{x} \cap \mathsf{chop}_1(\widetilde{y}^{\oplus}) = \emptyset$ due to domain separation. Thus, $|\widetilde{x} \cup \mathsf{chop}_1(\widetilde{y}^{\oplus})| = q + \sigma'$.

Real world: In the real world, the random permutation Π is sampled on exactly $q + \sigma'$ distinct points. Thus, we have

$$\Pr\left[\mathbb{R} = \tau\right] = \frac{1}{(2^n)_{q+\sigma'}}. \tag{19}$$

Ideal world: In the ideal world, first $\widetilde{\mathsf{T}}$ is sampled in with replacement fashion from a set of size 2^n. Then, exactly σ' values are sampled corresponding to $\widetilde{\mathsf{Y}}$ in without replacement fashion from a set of size $2^n - q$. Thus, we have

$$\Pr\left[\mathbb{I} = \tau\right] = \frac{1}{2^{nq}} \times \frac{1}{(2^n - q)_{\sigma'}}. \tag{20}$$

On dividing Eq. (19) by (20), we get

$$\frac{\Pr\left[\mathbb{R} = \tau\right]}{\Pr\left[\mathbb{I} = \tau\right]} \geq 1.$$

From H-coefficient Theorem 2.1 and Lemma 5.1, we get

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LightMAC\text{-}ds}_\Pi}(\mathscr{A}) \leq \frac{2.5q^2}{2^n}. \tag{21}$$

Theorem 5.1 follows from Eq. (16) and (21).

6 Conclusion

In this paper we studied the single-key instance of LightMAC, an ISO/IEC standard for lightweight message authentication codes. Our main contribution is a query-length independent security bound for 1k-LightMAC. Specifically, we showed that 1k-LightMAC achieves PRF security bound of $O(q^2/2^n)$ while $(n - s) \leq \ell \leq (n - s)\min\{2^{n/4}, 2^s\}$. Further, we proposed a slight variant of LightMAC, called LightMAC-ds that achieves security bound of $O(q^2/2^n)$ while $\ell \leq (n - s)2^{s-1}$.

6.1 Future Directions in Reset-sampling

To prove the security of 1k-LightMAC, we used a novel sampling approach, called reset-sampling, that works as a subroutine within the H-coefficient proof setup. Although this approach is at a very nascent stage, we believe that reset-sampling could potentially be useful in deriving better security bounds for other single-key constructions. Indeed, OMAC [10] – another popular and standardized MAC algorithm – has a similar bottleneck as 1k-LightMAC, and might benefit from this sampling approach. In the following, we briefly discuss a possible reset-sampling approach for query-length independent security bounds for OMAC.

A simplified variant of OMAC for any message $m \in (\{0,1\}^n)^\ell$ can be defined as follows: $y[0] := 0^n$; for $1 \leq i \leq \ell - 1$, $x[i] = m[i] \oplus y[i-1]$ and $y[i] = E_K(x[i])$; $x[\ell] = m[\ell] \oplus y[\ell - 1] \oplus 2E_K(0^n)$; and $\mathsf{OMAC}_{E_K}(m) := y[\ell] = E_K(x[\ell])$.

For all $i \in [\ell - 1]$, $x[i]$ and $y[i]$ are referred as intermediate input and output, respectively, and $x[\ell]$ and $y[\ell]$ are referred as the final input and output respectively.

Suppose the adversary makes q queries. Given our analysis of 1k-LightMAC, it is easy to observe that the most contentious issue is the event when some intermediate input (res. output) collides with some final input (res. output). Intuitively, this leads to a leakage of internal values to the adversary. However, notice that this does not necessarily mean that the adversary can actually detect and exploit this to mount an attack. This is precisely the point where reset-sampling can help. As an example, consider the following sampling approach in the ideal world:

- The ideal oracle faithfully answers the q queries in the online phase.
- Once the query-response phase is over:
  - The ideal oracle samples the intermediate inputs/outputs by following the OMAC definition, except for one small change: the intermediate outputs are sampled outside the set of all final outputs. This helps in avoiding collisions between some intermediate output and some final output.
  - Now, we may have a situation where some intermediate input $x_i[a]$ collides with some final input $x_j[\ell_j]$, which is an inconsistency.
  - However, if $x_i[a+1]$ is fresh, i.e., it does not collide with any other intermediate/final input, then we can possibly reset $y_i[a]$ to $y_j[\ell_j]$ and redefine $x_i[a+1] := x'_i[a+1] = m_i[a+1] \oplus y_j[\ell_j]$.
  - This might result in a collision of the form $x'_i[a+1] = x_k[b]$, but as we have seen in case of 1k-LightMAC, the probability of such collisions are easily bounded to $O(q^3\ell^2/2^{2n})$ by considering the compound event $x_i[a] = x_j[\ell_j] \cap x'_i[a+1] = x_k[b]$. There will be some more inconsistencies arising due to the resetting. But we ignore them for the sake of brevity.
  - Finally, the ideal oracle releases the intermediate inputs and outputs.

A more formal and rigorous analysis of OMAC using reset-sampling will most probably require handling of several other bad events, and could be an interesting future research topic. Although the above description is very succinct and rough, it is expressive enough to demonstrate the idea of resetting. The technique is particularly useful for deriving improved bounds for single-key constructions, as demonstrated for 1k-LightMAC and outlined for OMAC. Interestingly, the dominating term in the bound of 1k-LightMAC is the collision probability. Indeed, the bad events introduced due to reset sampling only contribute beyond-the-brithday bound terms. In fact, this seems to be a general characteristic of reset sampling based proof, as the additional bad events are generally joint events involving two or more sources of randomness. Consequently, we believe that reset sampling may, in future, find wide applications in the analysis of single-key variant of BBB secure constructions, such as LightMAC+ [25], PMAC+ [21] etc.

References

1. CAESAR: Competition for authenticated encryption: Security, applicability and robustness. Online webpage (2014)
2. NIST: Lightweight cryptography standardization project. Online webpage (2018)

3. 27, I.J.S.: Information technology — lightweight cryptography — part 6: Message authentication codes (MACs). ISO/IEC 29192-6, International Organization for Standardization (2019)
4. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC mode for lightweight block ciphers. In: Fast Software Encryption - FSE 2016, Revised Selected Papers. (2016) 43–59
5. Tsudik, G.: Message authentication with one-way hash functions. In: IEEE - INFOCOM 1992, Proceedings. (1992) 2055–2059
6. Mouha, N.: Chaskey: a MAC algorithm for microcontrollers - status update and proposal of Chaskey-12. IACR Cryptol. ePrint Arch. **2015** (2015) 1182
7. Bellare, M., Guérin, R., Rogaway, P.: XOR macs: New methods for message authentication using finite pseudorandom functions. In: Advances in Cryptology - CRYPTO 1995, Proceedings. (1995) 15–28
8. Bernstein, D.J.: How to stretch random functions: The security of protected counter sums. J. Cryptol. **12**(3) (1999) 185–192
9. Dutta, A., Jha, A., Nandi, M.: A new look at counters: Don't run like marathon in a hundred meter race. IEEE Trans. Computers **66**(11) (2017) 1851–1864
10. Iwata, T., Kurosawa, K.: OMAC: one-key CBC MAC. In: Fast Software Encryption - FSE 2003, Revised Papers. (2003) 129–153
11. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Advances in Cryptology - EUROCRYPT 2002, Proceedings. (2002) 384–397
12. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: Symposium on Foundations of Computer Science - FOCS 1984, Proceedings. (1984) 464–479
13. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. IACR Cryptol. ePrint Arch. **2004** (2004) 309
14. Yasuda, K.: PMAC with parity: Minimizing the query-length influence. In: Topics in Cryptology - CT-RSA 2012, Proceedings. (2012) 203–214
15. Naito, Y.: The exact security of PMAC with two powering-up masks. IACR Trans. Symmetric Cryptol. **2019**(2) (2019) 125–145
16. Ehrsam, W.F., Meyer, C.H.W., Smith, J.L., Tuchman, W.L.: Message verification and transmission error detection by block chaining. Patent 4074066, USPTO (1976)
17. Black, J., Rogaway, P.: CBC macs for arbitrary-length messages: The three-key constructions. In: Advances in Cryptology - CRYPTO 2000, Proceedings. (2000) 197–215
18. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Advances in Cryptology - CRYPTO 1994, Proceedings. (1994) 341–358
19. Berendschot, A., den Boer, B., Boly, J., Bosselaers, A., Brandt, J., Chaum, D., Damgård, I., Dichtl, M., Fumy, W., van der Ham, M., Jansen, C., Landrock, P., Preneel, B., Roelofsen, G., de Rooij, P., Vandewalle, J.: Final Report of RACE Integrity Primitives. Lecture Notes in Computer Science, Springer-Verlag, 1995 **1007** (1995)
20. Yasuda, K.: The sum of CBC macs is a secure PRF. In: Topics in Cryptology - CT-RSA 2010, Proceedings. (2010) 366–381
21. Yasuda, K.: A new variant of PMAC: beyond the birthday bound. In Rogaway, P., ed.: Advances in Cryptology - CRYPTO 2011, Proceedings. (2011) 596–609
22. Zhang, L., Wu, W., Sui, H., Wang, P.: 3kf9: Enhancing 3gpp-mac beyond the birthday bound. In: Advances in Cryptology - ASIACRYPT 2012, Proceedings. (2012) 296–312

23. Zhang, Y.: Using an error-correction code for fast, beyond-birthday-bound authentication. In: Topics in Cryptology - CT-RSA 2015, Proceedings. (2015) 291–307
24. Datta, N., Dutta, A., Nandi, M., Paul, G., Zhang, L.: Single key variant of pmac_plus. IACR Trans. Symmetric Cryptol. **2017**(4) (2017) 268–305
25. Naito, Y.: Blockcipher-based macs: Beyond the birthday bound without message length. In: Advances in Cryptology - ASIACRYPT 2017, Proceedings, Part III. (2017) 446–470
26. Wegman, M.N., Carter, L.: New classes and applications of hash functions. In: Symposium on Foundations of Computer Science - FOCS 1979, Proceedings. (1979) 175–182
27. Bellare, M., Pietrzak, K., Rogaway, P.: Improved security analyses for CBC macs. In: Advances in Cryptology - CRYPTO 2005, Proceedings. (2005) 527–545
28. NIST: Announcing the ADVANCED ENCRYPTION STANDARD (AES). FIPS 197, National Institute of Standards and Technology, U. S. Department of Commerce (2001)
29. Fischlin, M., Lehmann, A., Ristenpart, T., Shrimpton, T., Stam, M., Tessaro, S.: Random oracles with(out) programmability. In: Advances in Cryptology - ASIACRYPT 2010, Proceedings. (2010) 303–320
30. Patarin, J.: Etude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES. PhD thesis, Université de Paris (1991)
31. Patarin, J.: The "coefficients H" technique. In: Selected Areas in Cryptography - SAC '08. Revised Selected Papers. (2008) 328–345
32. Hoang, V.T., Tessaro, S.: Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In: Advances in Cryptology - CRYPTO 2016, Proceedings, Part I. (2016) 3–32
33. Jha, A., Nandi, M.: Revisiting structure graphs: Applications to CBC-MAC and EMAC. J. Math. Cryptol. **10**(3-4) (2016) 157–180
34. Jha, A., Nandi, M.: Revisiting structure graphs: Applications to CBC-MAC and EMAC. IACR Cryptol. ePrint Arch. **2016** (2016) 161
35. Minematsu, K., Matsushima, T.: New bounds for pmac, tmac, and XCBC. In Biryukov, A., ed.: Fast Software Encryption - FSE 2007, Revised Selected Papers. (2007) 434–451
36. Nandi, M.: Improved security analysis for OMAC as a pseudorandom function. J. Math. Cryptol. **3**(2) (2009) 133–148
37. Nandi, M., Mandal, A.: Improved security analysis of PMAC. J. Math. Cryptol. **2**(2) (2008) 149–162
38. Gazi, P., Pietrzak, K., Rybár, M.: The exact security of PMAC. IACR Trans. Symmetric Cryptol. **2016**(2) (2016) 145–161
39. Chakraborty, B., Chattopadhyay, S., Jha, A., Nandi, M.: On length independent security bounds for the PMAC family. IACR Cryptol. ePrint Arch. **2020** (2020) 656
40. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The secure real-time transport protocol (SRTP). RFC 3711, IETF (2004)
41. Chen, S., Steinberger, J.P.: Tight security bounds for key-alternating ciphers. In: Advances in Cryptology - EUROCRYPT 2014, Proceedings. (2014) 327–350
42. Mennink, B., Neves, S.: Encrypted davies-meyer and its dual: Towards optimal security using mirror theory. In: Advances in Cryptology - CRYPTO 2017, Proceedings, Part III. (2017) 556–583
43. Jha, A., Nandi, M.: A survey on applications of h-technique: Revisiting security analysis of prp and prf. IACR Cryptol. ePrint Arch. **2018** (2018) 1130