Tight Security Analysis of EHtM MAC

Avijit Dutta, Ashwin Jha and Mridul Nandi

Applied Statistics Unit, Indian Statistical Institute, Kolkata.
avirocks.dutta13@gmail.com,ashwin.jha1991@gmail.com,mridul.nandi@gmail.com

Abstract. The security of a probabilistic Message Authentication Code (MAC) usually depends on the uniqueness of the random salt which restricts the security to birthday bound of the salt size due to the collision on random salts (e.g XMACR). To overcome the birthday bound limit, the natural approach to use (a) either a larger random salt (e.g MACRX$_3$ uses $3n$ bits of random salt where n is the input and output size of the underlying non-compressing pseudorandom function or PRF) or (b) a PRF with increased domain size (e.g RWMAC or Randomized WMAC). Enhanced Hash-then-Mask (EHtM), proposed by Minematsu in FSE 2010, is the first probabilistic MAC scheme that provides beyond birthday bound security without increasing the randomness of the salt and the domain size of the non-compressing PRF. The author proved the security of EHtM as long as the number of MAC query is smaller than $2^{2n/3}$ where n is the input size of the underlying non-compressing PRF. In this paper, we provide the exact security bound of EHtM and prove that this construction offers security up to $2^{3n/4}$ MAC queries. The exactness is shown by demonstrating a matching attack.

Keywords: Probabilistic MAC, EHtM, XMACR, Alternating Cycle.

1 Introduction

In recent technological advancement of digital transmission, it is important to use some cryptographic means to authenticate the transmitted message or packet over an insecure channel. As a solution to this, MAC (Message Authentication Code), a popular primitive in symmetric key cryptography, plays an important role to enable two legitimate parties (having access to a shared secret key) to authenticate their transmissions.

Pseudo Random Function or (PRF) [10] is an essential tool in many cryptographic solutions. One of the natural uses of PRF is to construct a secure MAC. Out of its enormous usage, it is used to generate a pseudorandom pad for symmetric encryption, and then applying a universal hash function for producing a secure Hash-then-Mask (HtM) [5] type MAC. The security of PRF based constructions can be compromised if one applies the PRF twice to the same input [4]. Thus, a natural way to avoid repetition of the input is for the sender to use a counter, or other forms of varying, non-repeating state (also called *nonce*), which is updated with each application of the function, giving rise to *stateful MAC*. XMACC [3], WMAC [4] etc. are some of the examples of stateful MAC which follows HtM paradigm. However, these constructions have a drawback in storing the nonce which might in some settings be impractical or unsafe. For example, maintaining a synchronized nonce across different applications of the function is sometimes unsafe or even impossible. Thus, a possibility of having a stateless scheme is to use random values (also called **salt**) on which to evaluate the pseudorandom function as adopted in MACRX$_3$ [2], XMACR [3] etc, giving rise to *stateless probabilistic MAC*.

Informally, a MAC is defined by a pair of algorithms, called tag generation and verification algorithm. The verification algorithm must verify any tag generated by the tag generation algorithm. Usually tag of a *probabilistic MAC* consists of an m-bit salt (also called *random coin*) and an n-bit **core-tag** (formally defined in the definition of probabilistic MAC) depending on the salt. A forgery algorithm makes queries to both algorithms (i.e. tag generation and verification algorithm). We say that it successfully forges if it can submit a non-trivial message tag pair to the verification algorithm which successfully verifies the message tag pair. By non-trivial message tag pair we mean, it should not be obtained through some earlier tag generation queries. Informally, a MAC is information theoretically (q_m, q_v, ϵ)-secure if there is no forgery algorithm making up to q_m many tag-generation queries and q_v many verification queries, the success probability of forging is at least ϵ.

A Brief History on Stateless Probabilistic MAC. HtM, due to Carter-Wegman [4], is a popular approach to construct a secure IV-based MAC. When the IV is random, then the resulting MAC is called probabilistic MAC. XMACR [3] is one of the popular examples of probabilistic MAC that follows HtM paradigm. The core-tag of probabilistic MAC, following HtM paradigm, is computed by masking hash output by the output of a pseudorandom function applied to the salt. More formally,

$$\mathsf{HtM}(X) = (R, \mathcal{H}_K(X) \oplus f_{K'}(R))$$

where R is the salt, $\mathcal{H}_K$ is a ϵ-AXU keyed hash function and $f_{K'}$ is a keyed function [16]. The hash function of XOR-MAC is a parallel construction of counter based AXU-hash which ensures $O(q_m^2/2^m)$ unforgeable security for m-bit salt as the security is compromised when the salt repeats after $2^{m/2}$ many tag-generation queries (where q_m denotes the total number of tag-generation queries) due to classical birthday paradox. Thus, the natural question to ask, how to uplift the security of probabilistic MAC. A natural choice is to use a larger sized salt, (e.g. $m = 2n$). But this trivial choice suffers from the following shortcomings.

- Using larger salt increases communication cost and the sender's effort for generating large randomness.
- It needs increased domain sized PRF.

The second issue has been resolved by MACRX$_3$ [2] which still uses salt of size $m = 3n$ bits to compute the mask based on n-bit PRF whereas RWMAC [13], a randomized version of nonce based WMAC provides $O(q_m/2^n)$ security using n-bit salt and a $2n$ bits to n bit PRF. As a solution to both of the problems, RMAC [11] and FRMAC [12] are known to provide optimal security with n-bit salt and n-bit PRF. However, their security proofs are based on ideal assumption on the underlying primitive [11, 12]. In FSE 2010, Minematsu [13] proposed a simple variant of HtM (called **Enhanced Hash-then-Mask** or EHtM) that uses only n-bit salt and n-bit PRF. Author has shown its security upto $O(2^{2n/3})$ tag generation queries [13]. This is the first probabilistic MAC construction, illustrated in Fig. 1, which shows that by appropriately using the salt can enhance the MAC security to go beyond birthday barrier without requiring larger domain PRF.

Although Minematsu in [13] has shown EHtM is secure against all adversaries that observes at most $2^{2n/3}$ MAC queries and can make 2^n verification queries, we observe that the scheme is secure even if adversary observes more than $2^{2n/3}$ many queries and can make 2^n verification queries. In other words, author in [13] considered the following event as bad

$$R_i = R_j, R_j \oplus \mathcal{H}_K(M_j) = R_k \oplus \mathcal{H}_K(M_k) \qquad (1)$$

for some $i, j, k \in [q_m]$ and bound the probability of the bad event (1) which gives $O(q_m^3/2^{2n})$ bound. We observe that even if the bad event holds, security of the scheme prevails and hence allows us to uplift the security of the construction. It turns out that if

$$R_i = R_j, R_j \oplus \mathcal{H}_K(M_j) = R_k \oplus \mathcal{H}_K(M_k), R_k = R_l \qquad (2)$$

holds for some $i, j, k, l \in [q_m]$ then the security of the scheme gets compromised. In fact we have proposed a forgery attack with attack complexity $2^{3n/4}$ queries that exploits event (2).

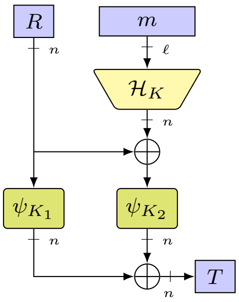

Figure 1.1: Figure of EHtM; ψ_{K_1} and ψ_{K_2} are two independent keyed functions, $\mathcal{H}_K$ is n-bit ϵ-AXU hash function. R is a n bit salt and m is message of at most ℓ blocks.

Our Contribution. In this paper, we show an **improved and exact security bound of EHtM** that provides $2^{3n/4}$ bit MAC security, in other words we show EHtM is secure for roughly about $2^{3n/4}$ MAC queries and 2^n verification queries. More precisely, we prove the following result.

Theorem 1. *Let $\psi_{K_1}, \psi_{K_2} : \{0,1\}^n \rightarrow \{0,1\}^n$ be two independent n-bit keyed instances of family of functions ψ and $\mathcal{H}_K : \{0,1\}^* \rightarrow \{0,1\}^n$ be a ϵ-AXU n-bit keyed hash function. Then we have*

$$\mathbf{Adv}^{\mathsf{RF}^\$,\perp}_{\mathsf{EHtM}}(q_m, q_v, \ell, t) \leq 2\mathbf{Adv}^{\mathrm{prf}}_{\psi}(q_m+q_v, t') + \frac{13q_m^4}{12 \cdot 2^{3n}} + \frac{q_m^2}{2^{2n+1}} + \frac{q_m^2\epsilon}{2^{n+1}} + \frac{q_m^4\epsilon}{2^{2n}} + 10q_v\epsilon + \frac{q_v}{2^n},$$

where q_m be the total number of MAC queries and q_v be the total number of verification queries, ℓ be the maximum number of message blocks and $t = t' + O((q_m + q_v)t_{\mathcal{H}})$, $t_{\mathcal{H}}$ be the maximum time for computing the hash value.

If $\epsilon \approx 2^{-n}$, then from Theorem 1, the SUF advantage of EHtM is bounded by $O(q_m^4/2^{3n}) + O(q_v/2^n)$ [1].

Second Contribution. As our second contribution in the paper, we exhibit a forgery algorithm, making roughly about $2^{3n/4}$ many MAC queries to the tag-generation oracle of EHtM and it forges with probability 1.

Note. Minematsu [13] considered two block cipher based instantiations of EHtM; MAC-R1 and MAC-R2 with same security bound as that of EHtM [13]. Although in the paper we study only EHtM construction based on PRF, we believe that our proven bound of EHtM also holds for block cipher based variants which may require a completely different analysis for its security proof.

2 Preliminaries

Symbol and Notations. We write $\perp$ and $\top$ to denote two special symbols refer to reject (or false) and accept (or true) respectively. We define $x =_? t$ to be $\top$ if $x = t$, otherwise it is defined to be $\perp$. For a set $\mathcal{X}$, $X \leftarrow_\$ \mathcal{X}$ means that X is chosen uniformly from the set $\mathcal{X}$ and *it is independent to all random variables defined so far.* We write $[q]$ to denote the set $\{1, \ldots, q\}$.

[1] Polynomial hash functions generally do not acheive 2^{-n} AXU bound; rather block cipher based UHF can acheive such a small differential probability, but in this paper we are not assuming any specific type of hash functions.

2.1 Security Definitions

Message Authentication Code. A Message Authentication Code (MAC) allows two parties to share a common secret key K to authenticate the data they send to each other. Working principle of MAC is as follows; The sender Alice applies a Tag Generation algorithm, denoted as TG, to key K and a message M to generate a tag T, and sends the message-tag pair (M, T) to receiver Bob. Bob, upon receiving (M, T), applies VeriFication algorithm, denoted as VF, to key K and the received (M, T) pair and returns either $\top$ or $\bot$ to indicate whether the received (M, T) pair is authentic or not respectively.

Definition 1. A MAC scheme is a pair $\Pi := (\mathsf{TG}, \mathsf{VF})$ of algorithms. TG is a (possibly probabilistic) algorithm takes a key $K \in \mathcal{K}$ (key space), a message $M \in \mathcal{M}$ (message space) and returns $T \in \mathcal{T}$ (tag space). We call a pair (M, T) to be **valid** for a key K if

$$\Pr[\mathsf{TG}(K, M) = T] > 0,^{2}$$

otherwise it is said to be **invalid**. VF is a deterministic algorithm that takes a key K, a message M and a tag T and returns a symbol $\cent \in \{\top, \bot\}$ such that for all valid pairs (M, T) for a key K, $\mathsf{VF}(K, M, T) = \top$ (known as the correctness condition of MAC).

Probabilistic MAC. Π is called a **probabilistic** MAC if the tag generation algorithm is probabilistic (otherwise we call it deterministic). Hence it returns a probability distribution on the tag space $\mathcal{T}$ for every pair $(K, M) \in \mathcal{K} \times \mathcal{M}$. To emphasize the probabilistic nature of the tag generation algorithm, we denote it as $\mathsf{TG}^{\$}$. In this paper, we focus on a special type of probabilistic tag generation algorithm. For each MAC query M, it first samples a random coin R uniformly from a coin space $\mathcal{R}$ and then applies an underlying deterministic algorithm $\mathsf{TG}(K, M; R)$ to message M and R which outputs the **core-tag** T. The final tag returned by $\mathsf{TG}^{\$}$ consists of the random coin R and the core-tag T. For each verification query $(\widetilde{R}, \widetilde{M}, \widetilde{T})$, the verification algorithm VF of Π simply returns $\mathsf{TG}(K, \widetilde{M}; \widetilde{R}) =_? \widetilde{T}$. We follow the convention to write the verification query in *tilde* notation.

Security Definitions of MAC. The forging adversary makes several tag generation queries and verification queries of a MAC scheme Π. We say that a verification query $(\widetilde{M}, \widetilde{T})$ is **trivial** if $\widetilde{T}$ is obtained from a previous tag generation oracle with $\widetilde{M}$ as a MAC query. Every trivial query must be valid but the converse is not true. In fact, the goal of the adversary is to find a non-trivial valid pair. Conventionally, we will assume in this paper that **all adversaries in this paper make no trivial queries**. We say that adversary wins if it submits at least one non-trivial valid verification query. We denote the probability that a forgery adversary $\mathcal{A}$ wins as $\mathbf{Adv}_{\Pi}^{\mathsf{suf}}(\mathcal{A})$. The maximum advantage of forging Π is defined as

$$\mathbf{Adv}_{\Pi}^{\mathsf{suf}}(q_m, q_v, t) := \max_{\mathcal{A}} \mathbf{Adv}_{\Pi}^{\mathsf{suf}}(\mathcal{A})$$

where maximum is taken over all $\mathcal{A}$ which makes at most q_m many tag generation oracle and q_v many verification oracle queries and runs in time at most time t. A MAC algorithm Π is called (ϵ, q_m, q_v, t)-suf MAC if $\mathbf{Adv}_{\Pi}^{\mathsf{suf}}(q_m, q_v, t) \leq \epsilon$. For an unbounded adversary, we may skip the time parameter t.

Distinguishing Game and its advantage. Let us consider $\mathcal{C}_0$ and $\mathcal{C}_1$ be two classes of functions. In a **simple distinguishing game** we consider two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ which behaves as follows: for $b \in \{0, 1\}$, $\mathcal{O}_b$ samples $\Phi_b \leftarrow_{\$} \mathcal{C}_b$ and simulates it. For an adversary $\mathcal{A}$ interacting with oracles either $\mathcal{O}_0$ or $\mathcal{O}_1$, we define the **distinguishing advantage** of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{O}_1}^{\mathcal{O}_0}(\mathcal{A}) := |\Pr[\mathcal{A}^{\mathcal{O}_0} \text{ returns } 1] - \Pr[\mathcal{A}^{\mathcal{O}_1} \text{ returns } 1]|.$$

[2] Here the probability is computed under randomness, if any, of the tag-generation algorithm. In case of deterministic algorithm, we can ignore probability and simply write it as $\mathsf{TG}(K, M) = T$.

We will use this definition of distinguishing advantage in our paper. Moreover, the above definition of advantage can be similarly extended for two or more oracles.

Random Function. The set of all functions from a set $\mathcal{D}$ to a set $\mathcal{R}$ is denoted as $\mathsf{Func}(\mathcal{D}, \mathcal{R})$. When $\mathcal{R} = \{0,1\}^n$, then we write $\mathsf{Func}_{\mathcal{D}}$ to denote the set of all functions from $\mathcal{D}$ to $\{0,1\}^n$ and we write Func when $\mathcal{D} = \mathcal{R} = \{0,1\}^n$. An n-bit **uniform random function** over a domain $\mathcal{D}$ is $\mathsf{RF}_{\mathcal{D}} \leftarrow_{\$} \mathsf{Func}_{\mathcal{D}}$. We write RF when the underlying domain set is clear from the context.

- PRF Advantage. Given an oracle adversary $\mathcal{A}$, we define prf-advantage of $\mathcal{A}$ against a keyed function F_k that output n-bit as

$$\mathbf{Adv}_F^{\text{prf}}(\mathcal{A}) := \mathbf{Adv}_{F_K}^{\mathsf{RF}}(\mathcal{A}) = |\Pr_{K \leftarrow_{\$} \mathcal{K}}[\mathcal{A}^{F_K} = 1] - \Pr_{\mathsf{RF} \leftarrow_{\$} \mathsf{Func}}[\mathcal{A}^{\mathsf{RF}} = 1]|.$$

Let $\mathbf{Adv}_F^{\text{prf}}(q,t)$ denote $\max_{\mathcal{A}} \mathbf{Adv}_F^{\text{prf}}(\mathcal{A})$ where maximum is taken over all adversaries $\mathcal{A}$ running in time t, making at most q queries.

(Almost-XOR) Universal Hash Functions. An n-bit hash function $\mathcal{H}$ is a $(\mathcal{K}, \mathcal{D})$-family of hash functions $\{\mathcal{H}_K := \mathcal{H}(K, \cdot) : \mathcal{D} \to \{0,1\}^n\}_{K \in \mathcal{K}}$ defined on its domain space $\mathcal{D}$ and indexed by the key space $\mathcal{K}$. $\mathcal{H}$ is called ϵ-Almost-XOR Universal (AXU) hash function, if for any two distinct X and X' in $\mathcal{D}$ and for any $Y \in \{0,1\}^n$,

$$\Pr_{K \leftarrow_{\$} \mathcal{K}}[\mathcal{H}_K(X) \oplus \mathcal{H}_K(X') = Y] \leq \epsilon.$$

Let $\Pi = (\mathsf{TG}^{\$}, \mathsf{VF})$ be a probabilistic MAC scheme. Let $\mathsf{RF}^{\$}$ be the the oracle that on every MAC query M, samples the coin R uniformly at random from coin space and applies random function RF on the pair (R, M) and returns $(R, \mathsf{RF}(R, M))$ as the tag. By abusing of notation, let $\perp$ be the oracle that on every verification query $(\widetilde{M}, (\widetilde{R}, \widetilde{T}))$ returns reject $\perp$. Note that, for $\mathsf{RF}^{\$}$, if the sampled random coin R is same for two different MAC queries with same message M, then the remaining part of the tag, i.e. $\mathsf{RF}(R, M)$ will be same. The same property is also true for Π. In the following, we state a result that was used in [8, 9] to bound the MAC security of Π in terms of distinguishing advantage of two pair of oracles $(\mathsf{TG}^{\$}, \mathsf{VF})$ and $(\mathsf{RF}^{\$}, \perp)$.

Lemma 1. *Let* $\Pi := (\mathsf{TG}^{\$}, \mathsf{VF})$ *a probabilistic MAC scheme. Then,*

$$\mathbf{Adv}_{\Pi}^{\text{SUF}}(q_m, q_v, t) \leq \mathbf{Adv}_{\mathsf{TG}^{\$}, \mathsf{VF}}^{\mathsf{RF}^{\$}, \perp}(q_m, q_v, t).$$

2.2 The H-Coefficient Technique

In this section, we briefly discuss the H-Coefficient Technique [15, 7, 6]. When an adversary is interacting with an oracle, the collection of all queries and response is called a *transcript*, denoted as τ. Sometimes we release more internal information about the oracle only after the adversary completes all queries. In this case a transcript also includes the additional information and clearly the maximum distinguishing advantage in this scenario can not be less than that of without additional information.

Let X_{re} (resp. X_{id}) to denote the random variable representing real world and ideal world transcript respectively. A transcript τ is said to be an *attainable transcript* if the probability of realizing τ by ideal oracle is positive (i.e. $\Pr[X_{\text{id}} = \tau] > 0$). Let $\mathcal{V}$ be the set of all attainable transcripts. For any attainable transcript τ, we similarly write $\Pr[X_{\text{re}} = \tau]$, to denote the probability of realizing an attainable transcript in real world. These probabilities are called *interpolation probabilities.* Following these notations, we state the main theorem of The H-Coefficient Technique as follows. We skip the proof of the theorem as it can be found in many papers, e.g. [15, 7, 6].

Theorem 2 (**The H-Coefficient Technique**)**.** *Let $\mathcal{V} = \mathcal{V}_{\text{good}} \sqcup \mathcal{V}_{\text{bad}}$ (disjoint union) be some partition of the set of all attainable transcripts. Suppose there exists $\epsilon_{\text{ratio}} \geq 0$ such that for any $\tau \in \mathcal{V}_{\text{good}}$,*

$$\frac{\Pr[X_{\text{re}} = \tau]}{\Pr[X_{\text{id}} = \tau]} \geq 1 - \epsilon_{\text{ratio}},$$

and there exists $\epsilon_{\text{bad}} \geq 0$ such that $\Pr[X_{\text{id}} \in \mathcal{V}_{\text{bad}}] \leq \epsilon_{\text{bad}}$. Then,

$$\mathbf{Adv}_{\mathcal{O}_{\text{re}}}^{\mathcal{O}_{\text{id}}}(\mathcal{A}) \leq \epsilon_{\text{ratio}} + \epsilon_{\text{bad}}. \tag{3}$$

When $\mathcal{O}_{\text{id}}$ is a uniform random function and $\mathcal{O}_{\text{re}}$ is some keyed construction defined over the same domain, then Eqn. (3) *says that $\mathbf{Adv}_{\mathcal{O}_{\text{re}}}^{\text{prf}}(\mathcal{A}) \leq \epsilon_{\text{ratio}} + \epsilon_{\text{bad}}$.*

2.3 Results on Alternating Cycle

In this section, we revisit to the basic definition of alternating cycle, alternating path and its related results which will be used in the MAC security proof and the MAC attack of EHtM.

Definition 2 (**Alternating Cycle** [1], [14])**.** Let us consider a tuple $\mathcal{S}$ where

$$\mathcal{S} := \big((X_1, Y_1), (X_2, Y_2), \ldots, (X_v, Y_v)\big).$$

We say that we have *an alternating-cycle in $\mathcal{S}$ of length $p \geq 2$ and p is even*, if we have p distinct indices $i_1, i_2, \ldots, i_p$ such that

$$X_{i_1} = X_{i_2}, Y_{i_2} = Y_{i_3}, X_{i_3} = X_{i_4}, \ldots, X_{i_{p-1}} = X_{i_p}, Y_{i_p} = Y_{i_1}.$$

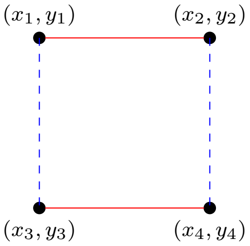

Figure 2.1: Alternating Cycle of length 4. Continuous line indicates first coordinate matches. Dotted line indicates second coordinates matches

Definition 3 (**Alternating Path** [14])**.** Given a tuple $\mathcal{S} := \big((X_1, Y_1), (X_2, Y_2), \ldots, (X_v, Y_v)\big)$, we say that we have *an alternating-path in $\mathcal{S}$ of length p, where p is odd*, if we have $p+1$ distinct indices $i_1, i_2, \ldots, i_{p+1}$ such that

$$X_{i_1} = X_{i_2}, Y_{i_2} = Y_{i_3}, X_{i_3} = X_{i_4}, \ldots, Y_{i_{p-1}} = Y_{i_p}, X_{i_p} = X_{i_{p+1}}.$$

Similarly, if p is even, then we have *an alternating-path in $\mathcal{S}$ of length p, where p is even*, if we have $p+1$ distinct indices $i_1, i_2, \ldots, i_{p+1}$ such that

$$X_{i_1} = X_{i_2}, Y_{i_2} = Y_{i_3}, X_{i_3} = X_{i_4}, \ldots, X_{i_{p-1}} = X_{i_p}, Y_{i_p} = Y_{i_{p+1}}.$$

Note: Alternating cycle of length at least 4 must contain an alternating path of length 3.

The following result gives the exact probability of the sum of two independent random functions over a pair of v inputs, where the tuple of pair of v inputs does not contain any alternating cycle, as follows. Details proof of it can be found in [1].

Lemma 2 ([1])**.** *Let f and g be two n-bit independent and uniformly distributed random functions. Let us consider a tuple $\mathcal{S} = \big((X_1, Y_1), (X_2, Y_2), \ldots, (X_v, Y_v)\big)$ such that for all $i \in [v]$, we have $(X_i, Y_i) \in (\{0,1\}^n)^2$ and $\mathcal{S}$ does not contain any alternating cycle. Then for any $T_i \in \{0,1\}^n$*

$$\Pr[f(X_i) \oplus g(Y_i) = T_i, 1 \le i \le v] = \frac{1}{2^{nv}}. \tag{4}$$

Since $\mathcal{S}$ does not contain any alternating cycle, we obtain a fresh variable [3] either $f(X_i)$ or $g(Y_i)$ from each equation $i \in [v]$. Each of these fresh variable contributes 2^{-n} for each equation, and thus 2^{-nv} for all the v equations.

A Note on the ramifications of alternating cycle. Let $\mathcal{S} = \big((X_i, Y_i), \ldots, (X_v, Y_v)\big)$ be a tuple where each $(X_i, Y_i) \in (\{0,1\}^n)^2$ and it contains an alternating cycle. Let the alternating cycle be

$$X_{i_1} = X_{i_2}, Y_{i_2} = Y_{i_3}, \ldots, Y_{i_p} = Y_{i_1}$$

of length p, where p is even and $i_1, \ldots, i_p$ are distinct indices, then

$$\bigoplus_{j \in \{i_1, \ldots, i_p\}} f(X_j) \oplus g(Y_j) = 0.$$

We would like to let the readers know in advance that we use this property to mount the forging attack on EHtM later in the paper.

3 MAC Security Proof of EHtM

In this section we show MAC advantage of EHtM is $O(\frac{q_m^4}{2^{3n}})$. In other words, we restate Theorem 1 and prove it in this section. Let us recall the construction of EHtM once again in the following.

$$\boxed{\mathsf{EHtM}(M) := \psi_{K_1}(R) \oplus \psi_{K_2}(R \oplus \mathcal{H}_K(M))}$$

where R is the random salt, ψ_{K_1} and ψ_{K_2} are two independent n-bit keyed functions and $\mathcal{H}_K$ is an ϵ-AXU n-bit keyed hash function. In the following, we bound the SUF advantage of EHtM in terms of bounding the distinguishability advantage due to Lemma 1.

Theorem 1. *Let ψ_{K_1} and ψ_{K_2} be two independent n-bit keyed instances of family of functions ψ and $\mathcal{H}_K : \{0,1\}^* \to \{0,1\}^n$ be a ϵ-AXU n-bit keyed hash function. Then we have*

$$\mathbf{Adv}_{\mathsf{EHtM}}^{(\mathsf{RF}^\$,\perp)}(q_m, q_v, \ell, t) \le 2\mathbf{Adv}_{\psi}^{\mathrm{prf}}(q_m+q_v, t')+\frac{13q_m^4}{12 \cdot 2^{3n}}+\frac{q_m^2}{2^{2n+1}}+\frac{q_m^2\epsilon}{2^{n+1}}+\frac{q_m^4\epsilon}{2^{2n}}+10q_v\epsilon+\frac{q_v}{2^n},$$

where q_m be the total number of MAC queries and q_v be the total number of verification queries, ℓ be the maximum number of message blocks and $t = t' + O((q_m + q_v)t_{\mathcal{H}})$, $t_{\mathcal{H}}$ be the maximum time for computing the hash value.

In specific, if $\epsilon \approx 2^{-n}$ then we have,

$$\mathbf{Adv}_{\mathsf{EHtM}}^{(\mathsf{RF}^\$,\perp)}(q_m, q_v, \ell, t) \le 2\mathbf{Adv}_{\psi}^{\mathrm{prf}}(q_m + q_v, t') + \frac{3q_m^4}{2^{3n}} + \frac{q_m^2}{2^{2n}} + \frac{11q_v}{2^n}.$$

Proof. Using a hybrid argument, we replace two independent keyed functions with two independent random functions f and g at the cost of $2\mathbf{Adv}_{\psi}^{\mathrm{prf}}(q_m + q_v, t')$ and call the

[3] A variable is said to be fresh if it is not equal to any of the other remaining variables

resulting construction EHtM^*. Then we apply H-Coefficient Technique on EHtM^* to prove the following

$$\mathbf{Adv}_{\mathsf{EHtM}^*}^{(\mathsf{RF}^\$,\perp)}(q_m, q_v, \ell) \leq \frac{13q_m^4}{12 \cdot 2^{3n}} + \frac{q_m^2}{2^{2n+1}} + \frac{q_m^2\epsilon}{2^{n+1}} + \frac{q_m^4\epsilon}{2^{2n}} + 10q_v\epsilon + \frac{q_v}{2^n}. \tag{5}$$

We assume $\mathcal{A}$ is computationally unbounded and hence *wlog* is deterministic and never repeats a verification query (however it can repeat MAC query). Moreover, it does not make any trivial verification query. The ideal oracle $(\mathsf{RF}^\$, \perp)$ works similar to the one as described in Lemma 1 where on each MAC query M, $\mathsf{RF}^\$$ randomly samples the coin R and then applies the random function $\mathsf{RF}(\cdot)$ on (R, M) and returns $(R, \mathsf{RF}(R, M))$ and on each verification query $(\widetilde{M}, (\widetilde{R}, \widetilde{T}))$, $\perp$ returns the reject symbol $\perp$.

As defined in Sect. 2.2, informally a transcript is a list of query and responses that is made in an interaction between the adversary and the oracle. In our case, we have different part of the transcripts as discussed below.

MAC Transcript. A MAC transcript is a finite collection of triplets where each triplet consists of the queried message, salt and the core-tag. Formally,

$$\tau_m := \{(M_1, R_1, T_1), (M_2, R_2, T_2), \ldots, (M_{q_m}, R_{q_m}, T_{q_m})\}$$

denotes the list of q_m many MAC queries of $\mathcal{A}$ and its corresponding responses. For simplicity we avoid the repetition of triplet in τ_m which can arises due to same random coin sampled for same message queried to tag-generation oracle. In this case core tag will be same for both real and ideal oracle i.e, for real oracle,

$$T_i = f(R_i) \oplus g(R_i \oplus M), T_j = f(R_j) \oplus g(R_j \oplus M) \text{ and } R_i = R_j \Rightarrow T_i = T_j.$$

Similar for ideal oracle,

$$T_i = \mathsf{RF}(R_i, M), T_j = \mathsf{RF}(R_j, M) \text{ and } R_i = R_j \Rightarrow T_i = T_j.$$

Verification Transcript. A verification transcript is a finite collection of quadruples where each quadruple consists of the verification attempted message, salt, core-tag and the response obtained from the oracle. Formally,

$$\tau_v := \{(\widetilde{M}_1, \widetilde{R}_1, \widetilde{T}_1, \mathbb{c}_1), (\widetilde{M}_2, \widetilde{R}_2, \widetilde{T}_2, \mathbb{c}_2), \ldots, (\widetilde{M}_{q_v}, \widetilde{R}_{q_v}, \widetilde{T}_{q_v}, \mathbb{c}_{q_v})\}$$

denotes the list of q_v many verification queries of $\mathcal{A}$ and its corresponding responses, where for all $a \in [q_v]$, $\mathbb{c}_a \in \{\top, \perp\}$ denotes the accept ($\mathbb{c}_a = \top$) or reject ($\mathbb{c}_a = \perp$).

The pair (τ_m, τ_v) denotes the combined list of query response of $\mathcal{A}$ that constitutes the query transcript of the attack. In addition to the above values, we also release the hash key K (it is sampled independently in case of the ideal oracle) only after all queries are done. Thus, a transcript τ consists of (τ_m, τ_v, K). From a given transcript $\tau = (\tau_m, \tau_v, K)$, we compute $H_i = \mathcal{H}_K(M_i)$, for all $i \in [q_m]$ and $\widetilde{H}_a = \mathcal{H}_K(\widetilde{M}_a)$, for all $a \in [q_v]$. We also denote $S_i = H_i \oplus R_i$, for all $i \in [q_m]$ and $\widetilde{S}_a = \widetilde{R}_a \oplus \widetilde{H}_a$, for all $a \in [q_v]$. Let $\mathcal{V}$ denote the set of all attainable transcripts. For an attainable transcript $\tau = (\tau_m, \tau_v, K)$, note that $\mathbb{c}_a = \perp$ for all $a \in [q_v]$ and the ideal interpolation probability is

$$\mathsf{p}_{\mathrm{id}} = \Pr[X_{\mathrm{id}} = \tau] = \frac{1}{2^{2nq_m}} \cdot \frac{1}{|\mathcal{K}|}, \tag{6}$$

where $\mathcal{K}$ is the hash key space. This follows easily from the fact that for each MAC query, the salt R and the random function output T are uniformly and indepedently sampled. This contributes to 2^{-2nq_m} to the probability as we have removed the repetition of triplet (M_i, R_i, T_i) for all $i \in [q_m]$ from the MAC transcript. Moreover, when all the query responses are made, the hash key K is uniformly and independently distributed to all the previously sampled R and T variables. Thus, it contributes to $\frac{1}{|\mathcal{K}|}$ to the probability.

$\boxed{\text{Bad Transcript and Its Probability in Ideal World.}}$ Informally, we say an event is bad if that event either leads to pass the verification attempt in real oracle or leads to a distinguishing event. Therefore, we briefly describe here the reason about our identified bad events.

Note that for all MAC queries in real oracle, we have $f(R_i) \oplus g(S_i) = T_i$ for all $i \in [q_m]$. Hence, if there is an alternating cycle of any size (has to be even) in $\big((R_1, S_1), \ldots, (R_{q_m}, S_{q_m})\big)$, then the sum of the corresponding T_i values must be zero. This can lead to a distinguishing event. Thus, in our identified bad events, we first avoid alternating cycle of length two (also includes the verification query) which is $(R_i, S_i) = (R_j, S_j)$ where $i \neq j \in [q_m]$ (in case of verification query $R_i = \widetilde{R}_a$, $\widetilde{S}_a = S_i$ and $\widetilde{T}_a = T_i$; we need the last condition on $\widetilde{T}_a$ to make the valid verification attempt).

AC2: $\exists a \leq q_v$, there is an alternating cycle of length two in $((R_1, S_1), \ldots (R_{q_m}, S_{q_m}), (\widetilde{R}_a, \widetilde{S}_a))$. Formally,

(a) there exists $i \neq j$ such that $R_i = R_j$, $S_j = S_i$ or

(b) there exists i such that $R_i = \widetilde{R}_a$, $\widetilde{S}_a = S_i$ and $\widetilde{T}_a = T_i$.

We also avoid an alternating path of length three (which should be present for any alternating cycle of length at least 4). Similar bad event is considered in which one of the pairs in the above tuple is for the verification oracle, i.e. $(\widetilde{R}_a, \widetilde{S}_a)$ for any $a \in [q_v]$. In this case, we also need an additional condition on $\widetilde{T}_a$ value.

AP3: $\exists a \leq q_v$, there is an alternating path of length three in $((R_1, S_1), \ldots (R_{q_m}, S_{q_m}), (\widetilde{R}_a, \widetilde{S}_a))$. Formally,

(a) there exists distinct i, j, k, l such that $R_i = R_j, S_j = S_k$ and $R_k = R_l$ or

(b) there exists distinct i, j, k such that $R_i = R_j, S_j = S_k, R_k = \widetilde{R}_a$ and $\widetilde{T}_a = T_i \oplus T_j \oplus T_k$.

A transcript τ is said to be a bad transcript if either of AC2(a) or AC2(b) or AP3(a) or AP3(b) holds. In other words, a transcript τ is said to be a good transcript if none of these conditions hold.

Consequence of Bad Transcript. Suppose AC2(a) holds for some $i \neq j$ while interacting with real oracle. Then clearly, $T_i = T_j$. In other words, if we observe $R_i = R_j$ and $T_i = T_j$ for some $i \neq j$ then this may be due to the hash collision. Then, one can make additional $2 \times 2^{n/2}$ MAC queries with M_i and M_j. In this case whenever we find collision in R values, we get collision in T values. This will clearly lead a distinguishing event as this holds with negligible probability in the ideal oracle.

To exploit AC2(b), an adversary can make verification queries with $(\widetilde{R}_a = R_i, \widetilde{M}_a, \widetilde{T}_a = T_i)$ and hope that $\mathcal{H}_K(\widetilde{M}_a) = \mathcal{H}_K(M_i)$, a collision in the hash value. This would happen with probability ϵ. So the success probability is about $q_v\epsilon$. This attack would be meaningful when ϵ is large. We discuss this attack in Sect. 4.2.

Suppose AP3(a) holds for some i, j, k, l while interacting with real oracle. Now, if we assume that $S_i = S_l$ holds, which is possible to achieve due to the appropriate choice of messages and $S_i = S_l$ is a consequence of $R_i = R_j, S_j = S_k$ and $R_k = R_l$ (details of the choice of messages for which one can acheive $S_i = S_l$ and $S_i = S_l$ becomes a consequence of $R_i = R_j, S_j = S_k$ and $R_k = R_l$ can be found in Sect. 4), then it forms an alternating cycle of length 4 in $\big((R_i, S_i), (R_j, S_j), (R_k, S_k), (R_l, S_l)\big)$ and hence we have $T_i \oplus T_j \oplus T_k \oplus T_l = 0$

and that leads to a valid forgery attempt. We note here that the alternating path of length 3 is that bad event which is exploited by the adversary to mount a forgery with $2^{3n/4}$ attack complexity. Details of the attack can be found in Sect. 4.

Suppose $\mathsf{AP3}$(b) holds for some i, j, k, a while interacting with real oracle. Now, if we assume that $S_i = \widetilde{S}_a$ holds, which is possible to achieve due to the appropriate choice of messages and $S_i = \widetilde{S}_a$ is a consequence of $R_i = R_j, S_j = S_k$ and $R_k = \widetilde{R}_a$, then it creates an alternating cycle of length 4 in $\big((R_i, S_i), (R_j, S_j), (R_k, S_k), (\widetilde{R}_a, \widetilde{S}_a)\big)$ and hence we have $T_i \oplus T_j \oplus T_k \oplus f(\widetilde{R}_a) \oplus g(\widetilde{S}_a) = 0$. This implies $\widetilde{T}_a = f(\widetilde{R}_a) \oplus g(\widetilde{S}_a)$ and thus, the forging attempt with $(\widetilde{M}_a, \widetilde{R}_a, T_i \oplus T_j \oplus T_k)$ will be passed by verification oracle.

Now, we make the following claim which argues that for a good transcript τ, there is no alternating cycle in the tuple $\big((R_1, S_1), (R_2, S_2), \ldots, (R_{q_m}, S_{q_m}), (\widetilde{R}_a, \widetilde{S}_a)\big)$ for each $a \in [q_v]$.

Claim. *Let τ be a good transcript. Then there is no alternating cycle in the following tuple*

$$\mathcal{S} := \big((R_1, S_1), (R_2, S_2), \ldots, (R_{q_m}, S_{q_m}), (\widetilde{R}_a, \widetilde{S}_a)\big)$$

for each $a \in [q_v]$.

Proof. First observe that $\mathcal{S}$ cannot contain any alternating cycle of length 2, as otherwise it immediately satisfies one of the conditions of $\mathsf{AC2}$, i.e. if the alternating cycle includes $(\widetilde{R}_a, \widetilde{S}_a)$, then it satisfies $\mathsf{AC2}$(b), otherwise it satisfies $\mathsf{AC2}$(a) and hence violates the property of good transcript.

Now, we observe that $\mathcal{S}$ does not contain alternating cycle of length 4. If it contains, then it would satisfy one of the conditions of $\mathsf{AP3}$, i.e. if the alternating cycle includes $(\widetilde{R}_a, \widetilde{S}_a)$, then it satisfies $\mathsf{AP3}$(b) (Note that, alternating cycle in this case may be formed as $S_i = S_j, R_j = R_k, S_k = \widetilde{S}_a, \widetilde{R}_a = R_i$ which does not satisfy $\mathsf{AP3}$(b), but this can be fixed using reordering of indices), otherwise it satisfies $\mathsf{AP3}$(a)) (Note that, in this case also alternating cycle may be formed as $S_i = S_j, R_j = R_k, S_k = S_l, R_l = R_i$ which does not satisfy $\mathsf{AP3}$(a), but again this can be fixed using reordering of indices) and hence violates the property of good transcript.

Now, if $\mathcal{S}$ contains an alternating cycle of length 6 or more, then irrespective of whether $(\widetilde{R}_a, \widetilde{S}_a)$ is included in $\mathcal{S}$ or not, the cycle must contain an alternating path of length 3 of the following form

$$R_i = R_j, S_j = S_k, R_k = R_l$$

which immediately satisfies $\mathsf{AP3}$(a). Therefore, $\mathcal{S}$ cannot contain any alternating cycle of length 6 or more. Therefore, $\mathcal{S}$ is alternating-cycle free set. □

For a good transcript, tuple of input pairs to functions f and g does not contain any alternating cycle.

Now we resume our proof. Let $\mathcal{V}_b \subseteq \mathcal{V}$ denotes the set of all bad transcripts and $\mathcal{V}_g := \mathcal{V} \backslash \mathcal{V}_b$ denotes the set of all good transcripts. Now, we bound the probability of realizing bad transcripts in ideal world in the following lemma, proof of which is postponed to Sect. 3.1.

Lemma 3. *Let X_{id} and $\mathcal{V}_b$ be defined as above. Then we have*

$$\Pr[X_{\text{id}} \in \mathcal{V}_b] \leq \epsilon_{\text{bad}} = \frac{13q_m^4}{12 \cdot 2^{3n}} + \frac{q_m^2}{2^{2n+1}} + \frac{q_m^2\epsilon}{2^{n+1}} + \frac{q_m^4\epsilon}{2^{2n}} + 10q_v\epsilon.$$

We show in the following lemma that for a good transcript τ, realizing τ is almost as likely as real and the ideal world.

Lemma 4. *Let $\tau = (\tau_m, \tau_v, K)$ be a good transcript. Then,*

$$\frac{\mathsf{p}_{\text{re}}}{\mathsf{p}_{\text{id}}} := \frac{\Pr[X_{\text{re}} = \tau]}{\Pr[X_{\text{id}} = \tau]} \geq (1 - \epsilon_{\text{ratio}}) = (1 - \frac{q_v}{2^n}).$$

Proof. Let $\tau = (\tau_m, \tau_v, k)$ be a fixed good transcript. We calculate the real interpolation probability for a good transcript as follows. Note that, the hash key K is sampled independent to all queries and responses and for each $i \in [q_m]$, random salt R_i is distributed uniformly and independently. Therefore,

$$\begin{aligned}\mathsf{p}_{\text{re}} \quad = \quad & \frac{1}{|\mathcal{K}|} \cdot \frac{1}{2^{nq_m}} \cdot \Pr[f(R_i) \oplus g(R_i \oplus \mathcal{H}_K(M_i)) = T_i, \ \forall i \in [q_m] \\ & \qquad f(\widetilde{R_a}) \oplus g(\widetilde{R_a} \oplus \mathcal{H}_K(\widetilde{M_a})) \neq \widetilde{T_a}, \ \forall a \in [q_v]]\end{aligned}$$

For the short hand of notation we have

$$\mathsf{p} := \Pr[f(R_i) \oplus g(R_i \oplus \mathcal{H}_K(M_i)) = T_i, \ \forall i \in [q_m], f(\widetilde{R_a}) \oplus g(\widetilde{R_a} \oplus \mathcal{H}_K(\widetilde{M_a})) \neq \widetilde{T_a}, \ \forall a \in [q_v]],$$

where the probability is calculated over the randomness of f, g. Now, due to the simple algebra on probabaility, we have

$$\mathsf{p} \geq \Pr[\underbrace{f(R_i) \oplus g(S_i) = T_i, \forall i \in [q_m]}_{D}] - \sum_{a=1}^{q_v} \Pr[\underbrace{f(R_i) \oplus g(S_i) = T_i, \forall i \in [q_m], f(\widetilde{R}_a) \oplus g(\widetilde{S}_a) = \widetilde{T}_a}_{D_a}]$$

Therefore, the task is now to evaluate $\Pr[D]$ and $\Pr[D_a]$ which are evaluated as follows. As τ is a good transcript, we note from Claim 1 that, for all $a \in [q_v]$, $\big((R_1, S_1) \ldots, (R_{q_m}, S_{q_m}), (\widetilde{R}_a, \widetilde{S}_a)\big)$ has no alternating cycle. Thus, by Lemma 2,

$$\Pr[D] = 2^{-nq_m} \text{ and } \Pr[D_a] = 2^{-n(q_m+1)}.$$

Hence, $\mathsf{p} \geq \frac{1}{2^{nq_m}}(1 - \frac{q_v}{2^n})$. Thus,

$$\Pr[X_{\text{re}} = \tau] \geq \frac{1}{2^{2nq_m} \times |\mathcal{K}|} \times (1 - \epsilon_{\text{ratio}}) \qquad (7)$$

where $\epsilon_{\text{ratio}} = \frac{q_v}{2^n}$. Combining Eqn. (6), Eqn. (7), Lemma 3 and Theorem 3, we obtain the result as shown in Eqn. (5). □

3.1 Bounding Probability of Bad Events

To complete the security proof, we only require to bound the probability of the bad events as identified in Sect. 3. In other words, in this section we prove Lemma 3. To bound the probability of the bad transcript in the ideal world, we have

$$\begin{aligned}\Pr[X_{\text{id}} \in \mathcal{V}_b] \quad &:= \quad \Pr[\mathsf{AC2a} \vee \mathsf{AC2b} \vee \mathsf{AP3a} \vee \mathsf{AP3b}] \\ &\leq \quad \Pr[\mathsf{AC2a}] + \Pr[\mathsf{AC2b}] + \Pr[\mathsf{AP3a}] + \Pr[\mathsf{AP3b}] \qquad (8) \\ &\overset{(1)}{\leq} \quad \frac{q_m^2\epsilon}{2^{n+1}} + (q_v\epsilon + \frac{q_m^2}{2^{2n+1}}) + \frac{q_m^4\epsilon}{2^{2n}} + (9q_v\epsilon + \frac{13q_m^4}{12 \cdot 2^{3n}}) \\ &= \quad \frac{13q_m^4}{12 \cdot 2^{3n}} + \frac{q_m^2}{2^{2n+1}} + \frac{q_m^2\epsilon}{2^{n+1}} + \frac{q_m^4\epsilon}{2^{2n}} + 10q_v\epsilon\end{aligned}$$

where (1) follows from Lemma 5, 6, 7, and 10. In the rest of this section, we bound the probability for each of these events.

Recall that, in ideal oracle, R and T are uniformly and independently sampled for each query and after all the queries are made, a hash key K is sampled uniformly and independent to all previously sampled R and T variables.

Lemma 5. $\Pr[\mathsf{AC2a}] \leq \frac{q_m^2\epsilon}{2^{n+1}}$.

Proof.

$$\begin{aligned}\Pr[\mathsf{AC2a}] &= \sum_{i<j} \Pr[R_i = R_j \wedge \mathcal{H}_K(M_i) \oplus R_i = \mathcal{H}_K(M_j) \oplus R_j] \\ &= \sum_{i<j} \Pr[R_i = R_j \wedge \mathcal{H}_K(M_i) = \mathcal{H}_K(M_j)] \\ &= \frac{\binom{q_m}{2}\epsilon}{2^n} \leq \frac{q_m^2\epsilon}{2^{n+1}}\end{aligned}$$

This follows from the fact that the hash key K is chosen independently from all salts as well as all queries made by the adversary (as we are working in the ideal oracle and it is sampled only after all queries are made). Moreover, $M_i \neq M_j$ since $R_i = R_j$ and we have already assumed that MAC transcript does not contain repetition of triplet. □

Lemma 6. $\Pr[\mathsf{AC2b}] \leq q_v\epsilon + \frac{q_m^2}{2^{2n+1}}$.

Proof. To bound the probability of AC2b, we first consider an event E: for some $i \neq j \in [q_m]$ such that $(R_i, T_i) = (R_j, T_j)$. In the ideal oracle, it is clear that

$$\Pr[E] = \frac{q_m(q_m-1)}{2^{2n+1}} \leq \frac{q_m^2}{2^{2n+1}}. \tag{9}$$

Now, we fix a tuple $((R_1, T_1), (R_2, T_2), \ldots, (R_{q_m}, T_{q_m}))$ such that the tuple satisfies $\neg E$, i.e. $(R_i, T_i) \neq (R_j, T_j)$ for all $i \neq j \in [q_m]$. Therefore, we have

$$\begin{aligned}\Pr[\mathsf{AC2b} \mid \neg E] &= \sum_{i=1}^{q_m}\sum_{a=1}^{q_v} \Pr[R_i = \widetilde{R}_a, \mathcal{H}_K(M_i) \oplus \mathcal{H}_K(\widetilde{M}_a) = R_i \oplus \widetilde{R}_a, T_i = \widetilde{T}_a \mid \neg E] \\ &= \sum_{i=1}^{q_m}\sum_{a=1}^{q_v} \Pr[R_i = \widetilde{R}_a, \mathcal{H}_K(M_i) = \mathcal{H}_K(\widetilde{M}_a), T_i = \widetilde{T}_a \mid \neg E] \\ &\overset{(1)}{=} \sum_{i=1}^{q_m}\sum_{a=1}^{q_v} \Pr[\mathcal{H}_K(M_i) = \mathcal{H}_K(\widetilde{M}_a)] \cdot \Pr[R_i = \widetilde{R}_a, T_i = \widetilde{T}_a \mid \neg E] \\ &\overset{(2)}{\leq} q_v\epsilon\end{aligned} \tag{10}$$

(1) follows due to the independence of the hash key from all salts as well as all queries and responses made by the adversary as in the ideal oracle the hash key is sampled only after all queries are made. (2) follows as the maximum number of i for which the event $R_i = \widetilde{R}_a, T_i = \widetilde{T}_a$ is satisfied is at most 1 as we have conditioned on $\neg E$. Moreover, the maximum value of the probability of that event is 1. Finally, summing over all $a \in [q_v]$, we obtain the bound $q_v\epsilon$. Also note that as we have considered only non trivial adversaries, it follows that $M_i \neq \widetilde{M}_a$ since $R_i = \widetilde{R}_a$ and $T_i = \widetilde{T}_a$.

Therefore, from Eqn. (9) and Eqn. (10) we have,

$$\Pr[\mathsf{AC2b}] \leq \Pr[\mathsf{AC2b} \mid \neg E] + \Pr[E] \leq q_v\epsilon + \frac{q_m^2}{2^{2n+1}}. \qquad \square$$

Lemma 7. $\Pr[\mathsf{AP3a}] \leq \frac{q_m^4\epsilon}{2^{2n}}$.

Proof. We have,

$$\begin{aligned}
\Pr[\mathsf{AP3a}] &= \sum_{i,j,k,l} \Pr[R_i = R_j, R_k = R_l, \mathcal{H}_K(M_j) \oplus \mathcal{H}_K(M_k) = R_j \oplus R_k] \\
&= \sum_{i,j,k,l} \sum_{r,r'} \Pr[R_i = R_j = r, R_k = R_l = r', \mathcal{H}_K(M_j) \oplus \mathcal{H}_K(M_k) = R_j \oplus R_k] \\
&\overset{(1)}{=} \sum_{i,j,k,l} \sum_{r,r'} \Pr[\mathcal{H}_K(M_j) \oplus \mathcal{H}_K(M_k) = r \oplus r'] \cdot \Pr[R_i = R_j = r \wedge R_k = R_l = r'] \\
&\overset{(2)}{\le} \sum_{i,j,k,l} \frac{\epsilon}{2^{2n}} \le \frac{q_m^4 \epsilon}{2^{2n}}
\end{aligned}$$

(1) follows due to independence of the sampled hash key K over the sampled salts R_i's. (2) follows due to the ϵ-AXU property of the hash function and $\sum_{r,r'} \Pr[R_i = R_j = r \wedge R_k = R_l = r']$ is exactly 2^{-2n}. Now, varying over all possible choices of i, j, k, l, we obtain the bound. □

Bounding More Events. Now, we define some more events to bound the remaining bad event $\mathsf{AP3b}$. We first define multi collision of an element. Given a tuple $(X_1, X_2, \ldots, X_v)$ and an arbitrary element X, $\mathsf{mc}(X)$ is the number of times X appears in the tuple. Let MC denotes the event that there exists $i \in [q_m]$ such that $\mathsf{mc}(R_i) \ge 4$. i.e

$$\Pr[\mathsf{MC}] = \Pr[\exists i \in [q_m] : \mathsf{mc}(R_i) \ge 4].$$

The following result establishes a bound on the probability of MC.

Lemma 8. $\Pr[\mathsf{MC}] \le \frac{q_m^4}{12 \cdot 2^{3n}}$, *when* $q_m^4 \le 2^{3n}/2$.

Proof. To bound $\Pr[\mathsf{MC}]$ we have the following

$$\begin{aligned}
\Pr[\mathsf{MC}] &= \sum_{i=1}^{q_m} \Pr[\mathsf{mc}(R_i) \ge 4] = \sum_{i=1}^{q_m} \sum_{d=4}^{q_m} \Pr[\mathsf{mc}(R_i) = d] \\
&\le \sum_{i=1}^{q_m} \sum_{d=4}^{\infty} \Pr[\mathsf{mc}(R_i) = d] = \sum_{d=4}^{\infty} \sum_{i=1}^{q_m} \Pr[\mathsf{mc}(R_i) = d] \\
&\overset{(1)}{\le} \sum_{d=4}^{\infty} \frac{q_m^d}{d! \cdot 2^{n(d-1)}} \overset{(2)}{\le} \frac{2q_m^4}{24 \cdot 2^{3n}} = \frac{q_m^4}{12 \cdot 2^{3n}} \qquad (\because q_m^4 \le 2^{3n}/2)
\end{aligned}$$

Note that, (1) is evaluated to at most $\frac{q_m^d}{d! \cdot 2^{n(d-1)}}$ as we choose a set of d many indices that are involved in the joint event $R_{i_1} = R_{i_2}, R_{i_2} = R_{i_3}, \ldots, R_{i_{d-1}} = R_{i_d}$, which gives at most $q_m^d/d!$ many choices of indices and for each such choice, the joint event is comprised of $d-1$ many independent events (or equalities) where each event contributes 2^{-n} probability. These events are independent as each R_i is independently distributed. Finally, (2) follows from simple algebraic calculation. □

Now, we define another event MC_1 which informally says that, for fixed $\{i, j\} \subseteq [q_m]$ for which $R_i = R_j$ has occured, the number of $\{k, l\} \subseteq [q_m]$ such that $T_k \oplus T_l$ collides with $T_i \oplus T_j$ and $R_k = R_l$ has occured, is at least 3. More formally, MC_1 denotes the event that there exists at least four pair of indices: $\{\{i_1, i_2\}, \{i_3, i_4\}, \{i_5, i_6\}, \{i_7, i_8\}\}$ such that we have following equalities

$$(1) = \begin{cases} T_{i_1} \oplus T_{i_2} = T_{i_3} \oplus T_{i_4} \\ T_{i_3} \oplus T_{i_4} = T_{i_5} \oplus T_{i_6} \\ T_{i_5} \oplus T_{i_6} = T_{i_7} \oplus T_{i_8} \end{cases} \qquad (2) = \begin{cases} R_{i_1} = R_{i_2} \\ R_{i_3} = R_{i_4} \\ R_{i_5} = R_{i_6} \\ R_{i_7} = R_{i_8} \end{cases}$$

In the following lemma we bound $\Pr[\mathsf{MC}_1 \wedge \neg\mathsf{MC}]$.

Lemma 9. $\Pr[\mathsf{MC}_1 \wedge \neg\mathsf{MC}] \leq \frac{q_m^4}{2^{3n}}$.

Proof. Observe that

$$\Pr[\mathsf{MC}_1 \wedge \neg\mathsf{MC}] \leq \sum_{i,j} \Pr[\mathsf{mc}(\{T_i \oplus T_j : R_i = R_j\}) \geq 4 \mid \neg\mathsf{MC}].$$

To compute the probability, we need to select four pair of indices: $\{\{i_1, i_2\}, \{i_3, i_4\}, \{i_5, i_6\}, \{i_7, i_8\}\}$ such that we have the following linear equations :

$$(1) = \begin{cases} T_{i_1} \oplus T_{i_2} = T_{i_3} \oplus T_{i_4} \\ T_{i_3} \oplus T_{i_4} = T_{i_5} \oplus T_{i_6} \\ T_{i_5} \oplus T_{i_6} = T_{i_7} \oplus T_{i_8} \end{cases} \qquad (2) = \begin{cases} R_{i_1} = R_{i_2} \\ R_{i_3} = R_{i_4} \\ R_{i_5} = R_{i_6} \\ R_{i_7} = R_{i_8} \end{cases}$$

Now, we claim that the rank of this system of 7 equations involving T's and R's is at least 6. To argue this, first observe that the rank of the system of equations (2) cannot go below 3 as that would give $\mathsf{mc}(R_i) \geq 4$. Since, we have conditioned on $\neg\mathsf{MC}$, rank of the system of equations (2) is either 3 or 4. Now, we have following cases:

- If the system of equations (2) is of full rank, then one can easily see that the rank of equations (1) is at least 2. Otherwise, there exists at least one choice of indices for which rank of (2) becomes strictly less than 4.

- When the rank of the system of equations (2) is 3 (e.g if $i_2 = i_3, i_4 = i_5$ and $i_6 = i_1$ then $R_{i_5} = R_{i_6}$ is trivially dependent on $R_{i_1} = R_{i_2}, R_{i_3} = R_{i_4}$), then the rank of the system of equations (1) is 3 (One can see from the above example after applying the equality of indices in equations (1), we obtain $T_{i_1} = T_{i_4}, T_{i_1} = T_{i_2}, T_{i_1} \oplus T_{i_4} = T_{i_7} \oplus T_{i_8}$).

Therefore, in both cases it ensures the rank of this system of equations (involving R and T) is at least 6. Therefore, (1) is bounded by $\frac{q_m^8}{2^{6n}} \leq \frac{q_m^4}{2^{3n}}$ when $q \leq 2^{3n/4}$. □

Now, we resume to the proof of bounding the final bad event $\mathsf{AP3b}$

Lemma 10. $\Pr[\mathsf{AP3b}] \leq 9q_v\epsilon + \frac{13q_m^4}{12 \cdot 2^{3n}}$.

Proof. We have the following

$$\begin{aligned} \Pr[\mathsf{AP3b}] &\leq \Pr[\mathsf{AP3b} \mid \neg\mathsf{MC} \wedge \neg\mathsf{MC}_1] + \Pr[\mathsf{MC}] + \Pr[\mathsf{MC}_1 \wedge \neg\mathsf{MC}] \\ &\leq \underbrace{\Pr[\mathsf{AP3b} \mid W]}_{(1)} + \frac{13q_m^4}{12 \cdot 2^{3n}} \qquad \text{(From Lemma 8 and 9)} \end{aligned}$$

where $W = \neg\mathsf{MC} \wedge \neg\mathsf{MC}_1$. Now, we bound (1) as follows

$$\sum_{a=1}^{q_v} \sum_{i,j,k} \Pr[\widetilde{T}_a \oplus T_i \oplus T_j \oplus T_k = 0, R_j = R_k, R_i = \widetilde{R}_a, \mathcal{H}_K(M_i) \oplus \mathcal{H}_K(M_j) = R_i \oplus R_j \mid W]$$

To bound the above probability, we first fix index a and for that fixed a, we write the event as the product of the following conditional probabilities

$$
\begin{aligned}
& \sum_{\substack{i,j,k\\ r,r'}} \Pr[T_i \oplus T_j \oplus T_k = \widetilde{T}_a, R_j = R_k = r, R_i = \widetilde{R}_a = r', \mathcal{H}_K(M_i) \oplus \mathcal{H}_K(M_j) = R_i \oplus R_j \mid W] \\
= & \sum_{\substack{i,j,k\\ r,r'}} \Pr[H_K(M_i) \oplus H_K(M_j) = r \oplus r'] \cdot \Pr[T_i \oplus T_j \oplus T_k = \widetilde{T}_a, R_j = R_k = r, R_i = \widetilde{R}_a = r' \mid W] \\
\overset{(1)}{\leq} & \ \epsilon \sum_{\substack{i,j,k\\ r,r'}} \Pr[T_i \oplus T_j \oplus T_k = \widetilde{T}_a, R_j = R_k = r, R_i = \widetilde{R}_a = r' \mid W] \\
\overset{(2)}{\leq} & \ 9\epsilon
\end{aligned}
$$

Note that hash key K is sampled independently to all the previously sampled R and T variables and (1) is evaluated to at most ϵ due to the ϵ-AXU property of the hash function. Moreover, after fixing j, the number of (k,l) for which the event $T_j \oplus T_k \oplus T_l = \widetilde{T}_i, R_k = R_l = r, R_j = \widetilde{R}_i = r' \mid W$ is satisfied, is at most 3 and the number of such j is again at most 3 as we have conditioned on $W = \neg\mathsf{MC} \wedge \neg\mathsf{MC}_1$. As the maximum value of the probability of that event is 1, therefore, (2) is evaluated to at most 9ϵ.
Thus, for fixed $a \in [q_v]$, the probability of the event is at most 9ϵ. Therefore, by varying over all such a, we obtain the bound to be $9q_v\epsilon$. □

4 A Matching Attack

In this section, we discuss a forgery attack by exploiting the specific bad event AP3a, discussed in Sect. 3. We show that the query complexity of the attack matches with the the proven bound of the construction upto a constant factor and hence proves the tightness of the security bound.

We construct the following adversary $\mathcal{A}$ that makes $2^{3n/4}$ many tag-generation queries to the tag generation oracle of EHtM with message M_1 and another $2^{3n/4}$ many queries with a different message M_2. Thus, it makes total $2 \times 2^{3n/4}$ many tag-generation queries.

Word on notation. The response obtained from the tag generation oracle when queried with message M_2, will be denoted in prime ($'$) notation and will be followed throughout the discussion in this section.

Rationale of the attack. Let us discuss the rationale of the forgery on EHtM. $\mathcal{A}$ chooses two distinct messages M_1 and M_2 and makes $2^{3n/4}$ many MAC queries to the tag-generation oracle with each of these messages. We note here that, for any two queries with same message, if the sampled random coin happens to be same, then the core-tag will be same and $\mathcal{A}$ will exclude that response from its consideration. Now, after making a total $2 \times 2^{3n/4}$ many queries, it tries to find four indices i, j, k, l such that $i \neq k$ and $j \neq l$ and all of these following events are satisfied

1. $R_i = R'_j$,
2. $R_k = R'_l$
3. $T_i \oplus T'_j \oplus T_k \oplus T'_l = 0$

where R_i, R_k are the sampled random coin when queried with message M_1 and R'_j, R'_l for message M_2. Now, observe that for these fixed four indices i, j, k, l, if all the above events hold simultaneoulsy, then we have

$$g(S_i) \oplus g(S'_j) \oplus g(S_k) \oplus g(S'_l) = 0. \tag{11}$$

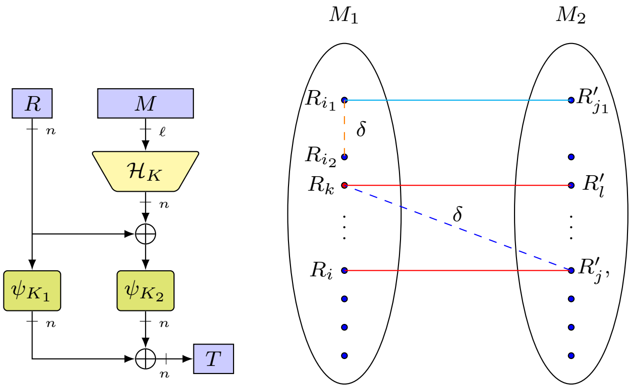

Figure 4.1: Forgery on EHtM; Guessing the hash difference in the lower part which is depicted through solid red line and dashed blue line. Forging attempt in the upper part which is depicted through solid cyan line and dashed orange line. If $R_i = R'_j, R_k = R'_l$ and $T_i \oplus T'_j \oplus T_k \oplus T'_l = 0$ then $\mathcal{A}$ computes $R_k \oplus R'_j = \delta$, an estimated hash difference. Then $\mathcal{A}$ finds $R_{i_1} = R'_{j_1}$ and $R_{i_1} \oplus R_{i_2} = \delta$ and forges with $(M_2, R_{i_2}, T_{i_1} \oplus T'_{j_1} \oplus T_{i_2})$. A little calculation shows that if δ is the correct guess of hash difference then $\psi_{K_1}(R'_{i_2}) \oplus \psi_{K_2}(R_{i_2} \oplus \mathcal{H}_K(M_2)) = T_{i_1} \oplus T'_{j_1} \oplus T_{i_2}$.

Eqn. (11) holds either **(a)** due to the randomness of g function (i.e the output of random function sums to zero) or **(b)** $\mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2) = R_k \oplus R'_j$. If case (b) holds, then $\mathcal{A}$ can retrieve the hash difference and use this difference to make the forgery attempt and if case (a) holds then we have to repeat the experiment as described in the following attack algorithm.

Attack Algorithm.

Online Phase of Attack on EHtM

```
fix $M_1$ and $i = 1$ to $2^{3n/4}$
    $M_1 \rightarrow \mathsf{EHtM.TG}; (R_i, T_i) \leftarrow \mathsf{EHtM.TG}; L_1 \leftarrow L_1 \cup \{(R_i, T_i)\}$.
fix $M_2 \neq M_1$ and $i = 1$ to $2^{3n/4}$
    $M_2 \rightarrow \mathsf{EHtM.TG}; (R'_i, T'_i) \leftarrow \mathsf{EHtM.TG}; L_2 \leftarrow L_2 \cup \{(R'_i, T'_i)\}$.
```

Offline Phase of Attack on EHtM

```
$\forall (R_i, T_i) \in L_1,$ if $(R'_j, T'_j) \in L_2$ such that $R_i = R'_j = R$ then $L \leftarrow L \cup \{(R, T_i, T'_j)\}$.
$\forall (R, T_i, T'_j), (R^*, T_k, T'_l) \in L,$ if $T_i \oplus T'_j = T_k \oplus T'_l,$ then $\delta = R \oplus R^*, L_\Delta \leftarrow L_\Delta \cup \{\delta\}$.
```

Verification Attempt on EHtM

```
$\forall \delta \in L_\Delta,$ if $(R, T_{i_1}, T'_{j_1}) \in L$ and $(R_{i_2}, T_{i_2}) \in L_1$ such that $R \oplus R_{i_2} = \delta$
  then $(R_{i_2}, M_2, (T_{i_1} \oplus T'_{j_1} \oplus T_{i_2})) \rightarrow \mathsf{EHtM.VF}$;
  if $\mathsf{EHtM.VF} \rightarrow \top$, return 1
   if no such $(R_{i_2}, T_{i_2})$ is found repeat all three phases.
```

Figure 4.2: SUF adversary $\mathcal{A}$ for EHtM.

Analysis of the attack. To start with the analysis, we assume that AXU advantage of the underlying hash function $\mathcal{H}$ is 2^{-n} (i.e. $\epsilon = 2^{-n}$). Let us fix four indices $i \neq k \in [q_m]$

and $j \neq l \in [q_m]$. For each such fixed four indices we define the indicator random variable I_{ijkl} as follows:

$$I_{ijkl} = \begin{cases} 1 & \text{if } R_i = R'_j, R_k = R'_l \text{ and } \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2) = R'_j \oplus R_k \\ 0 & \text{otherwise} \end{cases}.$$

Let X denotes the number of possible ways we can choose the set of four indices i, j, k, l such that

$$(1) = \begin{cases} R_i = R'_j \\ R_k = R'_l \\ \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2) = R'_j \oplus R_k \end{cases}$$

holds. As a result, we have

$$X = \sum_{ijkl} I_{ijkl}.$$

Now, we are interested to find $\mathbf{E}[X]$. Due to linearity of expectation, we have

$$\begin{aligned} \mathbf{E}[X] &= \sum_{ijkl} \Pr[I_{ijkl} = 1] \\ &= \sum_{ijkl} \Pr[R_i = R'_j, R_k = R'_l, \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2) = R'_j \oplus R_k] \\ &= \frac{\binom{q_m}{4}}{2^{3n}} \end{aligned}$$

Therefore, expected number of i, j, k, l for which (1) holds is $\frac{\binom{q_m}{4}}{2^{3n}}$.
Now, we fix indices $i_1 \neq i_2 \in [q_m], j_1 \in [q_m]$. For each such fixed indices, we define indicator random variable $\widetilde{I}_{i_1 j_1 i_2}$ as follows:

$$\widetilde{I}_{i_1 j_1 i_2} = \begin{cases} 1 & \text{if } R_{i_1} = R'_{j_1}, R_{i_1} \oplus R_{i_2} = \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2), R_{i_2} \notin L_2 \\ 0 & \text{otherwise} \end{cases}.$$

Let $\widetilde{X}$ denotes the number of possible ways we can choose the set of three indices i_1, j_1, i_2 such that

$$(2) = \begin{cases} R_{i_1} = R'_{j_1} \\ R_{i_1} \oplus R_{i_2} = \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2) \\ R_{i_2} \notin L_2 \end{cases}$$

holds. As a result, we have

$$\widetilde{X} = \sum_{i_1 j_1 i_2} \widetilde{I}_{i_1 j_1 i_2}.$$

As before, we are interested to find $\mathbf{E}[\widetilde{X}]$ and due to linearity of expectation, we have

$$\begin{aligned} \mathbf{E}[\widetilde{X}] &= \sum_{i_1 j_1 i_2} \Pr[R_{i_1} = R'_{j_1}, R_{i_1} \oplus R_{i_2} = \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2), R_{i_2} \notin L_2] \\ &\geq \sum_{i_1 j_1 i_2} (\Pr[R_{i_1} = R'_{j_1}, R_{i_1} \oplus R_{i_2} = \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2)] \\ &\qquad - \sum_{k=1}^{2^{3n/4}} \Pr[R_{i_1} = R'_{j_1}, R_{i_1} \oplus R_{i_2} = \mathcal{H}_K(M_1) \oplus \mathcal{H}_K(M_2), R_{i_2} = R'_k]) \\ &\geq \sum_{i_1 j_1 i_2} (\frac{1}{2^{2n}} - \frac{1}{2^{9n/4}}) \\ &\geq \frac{q_m^3}{2^{2n}} (1 - \frac{1}{2^{n/4}}) \end{aligned} \tag{12}$$

As we mentioned before, the observable events (i.e. $R_i = R'_j, R_k = R'_l, T_i \oplus T'_j \oplus T_k \oplus T'_l = 0$) implies Eqn. (11) which can be satisfied due to the randomness of g function. Using the same analysis before, it is easy to see the expected number of i, j, k, l, which we denote as $\mathbf{E}[X']$, such that the following events are satisfied simultaneously

$$(3) = \begin{cases} R_i = R'_j \\ R_k = R'_l \\ g(S_i) \oplus g(S'_j) \oplus g(S_k) \oplus g(S'_l) = 0 \end{cases}$$

is $\frac{\binom{q_m}{4}}{2^{3n}}$. Therefore, the expected number of verification attempt made by adversary is the sum of $\mathbf{E}[X]$ and $\mathbf{E}[X']$ which is $\frac{2\binom{q_m}{4}}{2^{3n}}$.

Choice of parameters. If we choose number of MAC queries $q_m = 2^{3n/4}$, then on average there exists 1 choice of i, j, k, l such that (1) holds. For such choice of i, j, k, l we retrieve the correct hash difference. For that correct hash difference, from Eqn. (12), there exists on average at least one choice of i_1, j_1, i_2 such that (2) holds and for that choice of i_1, j_1, i_2, the verification attempt is successful. Since, the expected number of verification queries is $\frac{2\binom{q_m}{4}}{2^{3n}}$, we require on average 2 verification attempts.

Remark 1. In the discussion of Consequence of bad transcripts in Sect. 3, we stated that $S_i = S_l$ is possible to achieve due to the appropriate choices of messages and $S_i = S_l$ is a consequence of $R_i = R_j, S_j = S_k$ and $R_k = R_l$. Note that, in our attack, with two distinct messages M_1, M_2, $S_i = S'_l$ trivially follows from $R_i = R'_j, R_k = R'_l$ and $S_k = S'_j$.

4.1 Attack Implementation

In this section, we present neccessary data structures for execution of the attack and estimate required memory and time complexity. As discussed, the attack is divided into the following two parts : (a) Online phase to detect alternating cycle of length 4, and (b) Offline phase to detect alternating path of length 3. We require the following data structures for execution of the attack algorithm.

Data Structures.

1. A Master Hash Table L_{hash} with the following attributes:

$$(\mathsf{indx}_R, \mathsf{indx}_{R'}, R, R', T_R, T_{R'}).$$

2. A Collided Salt Hash Table L_{salt} with the following attributes:

$$(\mathsf{indx}_R, \mathsf{indx}_{R'}, R, R').$$

3. A Tag Collision Hash Table L_{tag} with the following attributes:

$$(\mathsf{Tag\text{-}XOR}, \mathsf{Indx}_1, \mathsf{Indx}_2).$$

With abuse of notation, we denote the blank entries for all of the above tables by $\perp$.

Online Phase. In the online phase of the attack, adversary $\mathcal{A}$ does the following:

1. For message M_1, let $\mathcal{A}$ makes i-th tag generation query. If the sampled R_i for that corresponding query already appears in the 3rd column of L_{hash}, then $\mathcal{A}$ does nothing. Otherwise, it makes the following entry in L_{hash}.

$$(i, \perp, R_i, \perp, T_i, \perp).$$

$\mathcal{A}$ continues this step until the number of entries in L_{hash} reaches to $2^{3n/4}$.

2. For message M_2, let $\mathcal{A}$ makes j-th tag generation query. If the sampled R'_j for that corresponding query already appears in the 4th column of L_{hash}, then $\mathcal{A}$ does nothing. Otherwise, if it appears in 3rd column of L_{hash}, let the entry be

   $$(i, \perp, R_i, \perp, T_i, \perp)$$

   then $\mathcal{A}$ modifies that entry as

   $$(i, j, R_i, R'_j, T_i, T'_j).$$

   Otherwise, it makes the following entry in L_{hash}.

   $$(\perp, j, \perp, R'_j, \perp, T'_j).$$

   $\mathcal{A}$ continues this step until the number of entries in L_{hash} reaches to $2^{3n/4}$.

3. Make a linear search in L_{hash} and construct two more tables in the following way

   (a) Start searching from top of L_{hash} and search for entries in the table with indx_R and $\mathsf{indx}_{R'}$ do not contain $\perp$. Let such entry be

   $$(i, j, R_i, R'_j, T_i, T'_j).$$

   .

   For each such entry, $\mathcal{A}$ does the following

   (a) Make a new entry in L_{salt} with following items

   $$(i, j, R_i, R'_j).$$

   (b) Then $\mathcal{A}$ computes $\Delta_T := T_i \oplus T'_j$.

   (c) $\mathcal{A}$ checks if Δ_T already appears in the 1st column of L_{tag}. Let the entry be

   $$(\Delta_T, k, l).$$

   Then, $\mathcal{A}$ retrieves the value (k, l) ans thus obtains four indices i, j, k, l. Otherwise, it makes a new entry to L_{tag} with items

   $$(\Delta_T, i, j).$$

Offline Phase. In the offline phase of the attack, adversary $\mathcal{A}$ does the following:

1. Given i, j, k, l, go to the row of table L_{hash} with first two entries i, j and retrieve the value of the 3rd column of that row i.e R_i. Similarly, go to the row of table L_{hash} with first two entries k, l and retrieve the value of the 3rd column of that row i.e R_k. Then compute $\delta = R_i \oplus R_k$.

2. For each entry (u, v, R_u, R'_v) in L_{salt}, search for $R_u \oplus \delta$ ($R'_v \oplus \delta$ resp.) in 3rd column (4th column resp.) of L_{hash}. If $\mathcal{A}$ finds an entry

   $$(w, \perp, R_w, \perp, T_w, \perp)$$

   in L_{hash} where $R_w = R_u \oplus \delta$ then forge with

   $$(M_2, R_w, T_u \oplus T_v \oplus T_w).$$

   or if $\mathcal{A}$ finds an entry

   $$(\perp, w', \perp, R_{w'}, \perp, T_{w'})$$

   in L_{hash}, where $R_{w'} = R_v \oplus \delta$ then forge with

   $$(M_1, R_{w'}, T_u \oplus T_v \oplus T_{w'}).$$

Memory and Time Complexity. The major part of memory consumption lies in online phase of the attack to build three tables. Expected size of table L_{hash} is $2^{3n/4}$ since we have two independently without replacement sampled random vectors of size $2^{3n/4}$ each and hence expected size of L_{salt} and L_{tag} is $2^{n/2}$. Moreover, the expected number of quadruple (i, j, k, l) for which $T_i \oplus T'_j \oplus T_k \oplus T'_l = 0$ is 1 and hence the expected number of δ value is 1 which is the correct estimation of hash difference with high probability. So, overall the memory required for the attack is roughly $2^{3n/4}$. Time complexity of online phase of the attack is dominated by building L_{hash} which is roughly $2^{3n/4}$. The time complexity of offline phase is bounded by the size of L_{salt} which is roughly $2^{n/2}$ and hence the overall time complexity of the attack is $2^{3n/4}$.

4.2 Attack Algorithm of EHtM Exploiting Bad Event AC2b

We construct an adversary $\mathcal{A}$ that forges on EHtM exploiting the bad event AC2b. This attack is simple to describe. $\mathcal{A}$ only makes a single MAC query with message M and obtains response (R, T). Now, $\mathcal{A}$ will make q_v many verification queries $(M_1, R, T), \ldots, (M_{q_v}, R, T)$ where $M_1, \ldots, M_{q_v}$ are all distinct messages, hoping to find a hash collision i.e. within q_v many verification attempts $\mathcal{A}$ hopes that $\exists i \in [q_v]$ such that $\mathcal{H}_K(M) = \mathcal{H}_K(M_i)$.

Note that, success probability of the adversary in mouting this attack is roughly about $q_v\epsilon$ where ϵ is the AXU advantage of $\mathcal{H}_K$. This attack is meaningful when ϵ is large.

Acknowledgements: We would like to thank all the anonymous reviewers for their valueable comments and suggestions to improve the quality of the paper.

References

[1] William Aiello and Ramarathnam Venkatesan. Foiling birthday attacks in length-doubling transformations - benes: A non-reversible alternative to feistel. In *Advances in Cryptology - EUROCRYPT '96*, pages 307–320, 1996.

[2] Mihir Bellare, Oded Goldreich, and Hugo Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *Advances in Cryptology - CRYPTO '99*, pages 270–287, 1999.

[3] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR macs: New methods for message authentication using finite pseudorandom functions. In *Advances in Cryptology - CRYPTO '95*, pages 15–28, 1995.

[4] John Black and Martin Cochran. MAC reforgeability. In *Fast Software Encryption, FSE 2009*, pages 345–362, 2009.

[5] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[6] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round even-mansour cipher. In *Advances in Cryptology - CRYPTO 2014,*, pages 39–56, 2014.

[7] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology - EUROCRYPT 2014,*, pages 327–350, 2014.

[8] Benoît Cogliati and Yannick Seurin. EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In *CRYPTO 2016, Proceedings, Part I*, pages 121–149, 2016.

[9] Avijit Dutta, Ashwin Jha, and Mriul Nandi. Exact security analysis of hash-then-mask type probabilistic mac constructions. Cryptology ePrint Archive, Report 2016/983, 2016. http://eprint.iacr.org/2016/983.

[10] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Advances in Cryptology, Proceedings of CRYPTO '84*, pages 276–288, 1984.

[11] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In *Fast Software Encryption, FSE 2002*, pages 237–251, 2002.

[12] Éliane Jaulmes and Reynald Lercier. Frmac, a fast randomized message authentication code. *IACR Cryptology ePrint Archive*, 2004:166, 2004.

[13] Kazuhiko Minematsu. How to thwart birthday attacks against macs via small randomness. In *Fast Software Encryption, FSE 2010*, pages 230–249, 2010.

[14] Jacques Patarin. A proof of security in $\mathrm{o}(2^{\mathrm{n}})$ for the benes scheme. In *Progress in Cryptology - AFRICACRYPT 2008*, pages 209–220, 2008.

[15] Jacques Patarin. The "Coefficients H" Technique. In *Selected Areas in Cryptography, SAC*, pages 328–345, 2008.

[16] Phillip Rogaway. Bucket Hashing and Its Application to Fast Message Authentication. *J. Cryptology*, 12(2):91–115, 1999.