A New and Improved Reduction Proof of Cascade PRF

Mridul Nandi

Indian Statistical Institute, Kolkata
mridul@isical.ac.in

Abstract. The prefix-free security of a cascade function based on a c-bit compression function f is reduced to the q-query PRF security of f with a tightness gap ℓq where q represents the maximum number of queries to the cascade and ℓ represents the length of the longest query. A two-stage proof for this reduction was first given by Bellare et al. in FOCS-96 for an adaptive distinguisher, and later a similar two-stage reduction was proved in CRYPTO-14 by Gazi et al. for a non-adaptive distinguisher. *In this paper we prove a direct single-stage reduction with a tightness gap of σ (the total length of all queries).* This is an improvement over existing reductions whenever the lengths of queries vary widely. In the case of non-adaptive prefix-free security, we also show a reduction proof which reduces PRF advantage of the cascade to two terms – (i) a q-query PRF security of f with **a tightness gap of** q (without a factor of ℓ) and (ii) a **single query PRF security of** f with a tightness gap of σ. We further extend to a more general finer reduction to multiple terms over different limits on the queries to f. *All these reductions can be easily extended to a multiuser setup.* In particular, we reduce multiuser prefix-free PRF security of the cascade to a single user $q_{\max}$-query PRF security of f with a tightness gap $\overline{\sigma}$ (the total length of all queries to all users), where $q_{\max}$ is the maximum number of queries allowed to any user. We have shown similar improved bounds (with respect to query complexity) for non-adaptive multiuser PRF security. In addition to immediate applications to multiuser security of HMAC and NMAC, our improved analysis has the following useful applications:

1. We show that the multiuser non-adaptive PRF security of the cascade does not degrade even if f assures a weaker non-adaptive PRF security advantage.
2. The PRF security of single-keyed NMAC and Envelope MAC can be reduced to the non-adaptive multiuser prefix-free PRF security of the cascade construction and hence all improved reductions are applicable to these constructions. As a result, the constants ipad and opad used in HMAC are redundant. Moreover, the existing PRB assumption on f can be replaced by a simple regular property for the constant-free HMAC.

Keywords: PRF, HMAC, NMAC, cascade, non-adaptive security

1 Introduction

Notations. For any two integers $a < b$, we write $[a..b]$ (or simply $[b]$ when $a = 1$) to denote the set $\{a, a+1, \ldots, b\}$. A q-tuple $(x_1, \ldots, x_q)$ and $((x_1, y_1), \ldots, (x_q, y_q))$ are denoted as x^q and (x^q, y^q) respectively. In this paper, we also view a q-tuple x^q as a set $\{x_i : i \in [q]\}$ and so $|x^q|$ denotes the number of distinct elements present in the tuple. We fix two positive integers b and c and we write $\{0,1\}^b$ as B (set of blocks). We also fix a function $f : \{0,1\}^c \times \mathsf{B} \to \{0,1\}^c$ (e.g., compression function of SHA-1, SHA-256/512 [9]).

For a reasonably large integer L (e.g., 2^{100}), the set of all tuples (including the empty tuple λ) with at most L blocks is denoted as B_*. We write $\mathsf{B}_+ := \mathsf{B}_* \times \mathsf{B}$. For $m := (m[0], m[1], \ldots, m[r]) \in \mathcal{I} \times \mathsf{B}_+$ and $1 \le i \le j \le r$, we write (i) the number of blocks $\|m\| = r$, (ii) subtuple $m[i..j] := (m[i], \ldots, m[j])$, (iii) suffix $m[i..] = m[i..r]$ and (iv) prefix $m[..j] = m[0..j]$. For m^q, we write $\|m^q\| = \sum_i \|m_i\|$. For positive integers q', ℓ, σ', let

$$\mathcal{M}(q', \ell, \sigma') = \{m^q : q \le q',\ \|m_i\| \le \ell, \forall i \in [q],\ \sigma := \|m^q\| \le \sigma'\}.$$

be the set of all tuples of at most q' messages so that the longest message has at most ℓ blocks and the number of blocks in all messages is at most σ'. For $m = (x, y)$, we write $m \setminus x = y$ and $x \preceq m$ (x is called a *prefix* of m) or $x \prec m$ if $z \ne \lambda$. A q-tuple $m^q = (m_1, \ldots, m_q)$ is called *prefix-free* if m_i is not a prefix to m_j for all $i \ne j$ (and so they are necessarily distinct).

1.1 Cascade, HMAC, NMAC and Envelope MAC

The *cascade function* $f^* : \{0,1\}^c \times \mathsf{B}^* \to \{0,1\}^c$ is defined recursively as follows: For all $h \in \{0,1\}^c$, $f^*(h, \lambda) = h$ and for all $m[..i] \in \mathsf{B}^i$, $i \ge 1$,

$$f^*(h, m[..i]) = f^*\big(f(h, m[..i-1]), m[i]\big). \tag{1}$$

One can further extend the domain of f^* to the set of all arbitrary bit strings by applying an appropriate padding rule as a preprocessor of the iterated function. Let $\mathsf{pad}(\cdot)$ be the length-encoded MD-strengthening padding that maps m to $m \| \mathsf{pad}(|m|) \in \mathsf{B}^+$ injectively. The Merkle-Damgård hash (or MD hash) output $\mathsf{MD}_{\mathrm{IV}}(m)$ based on the compression function f for a message m is $f^*(\mathrm{IV}, m \parallel \mathsf{pad}(|m|))$ where $\mathrm{IV} \in \{0,1\}^c$ is some fixed initial value specified in the description of the hash function.

For keys $k, k_1, k_2 \in \{0,1\}^c$, b-bit constants $\mathsf{ipad}, \mathsf{opad}$ specified in [4] and message $m \in \{0,1\}^*$,

$$\mathsf{NMAC}_{k_1,k_2}(m) = \mathsf{MD}_{k_2}(\mathsf{MD}_{k_1}(m)),$$
$$\mathsf{HMAC}_k(m) = \mathsf{MD}\big(k^{\oplus\mathsf{opad}} \parallel \mathsf{MD}(k^{\oplus\mathsf{ipad}} \| m)\big) = \mathsf{NMAC}_{\mathsf{KDF}(k)}(m),$$

where $k^{\oplus\alpha} = (k \| 0^{b-c}) \oplus \alpha$ and $\mathsf{KDF}(k) = \big(k_1 := f(\mathrm{IV}, k^{\oplus\mathsf{ipad}}), k_2 := f(\mathrm{IV}, k^{\oplus\mathsf{opad}})\big)$. Here, we must assume that $c \le b$, which used to hold for the earlier compression functions. [1]

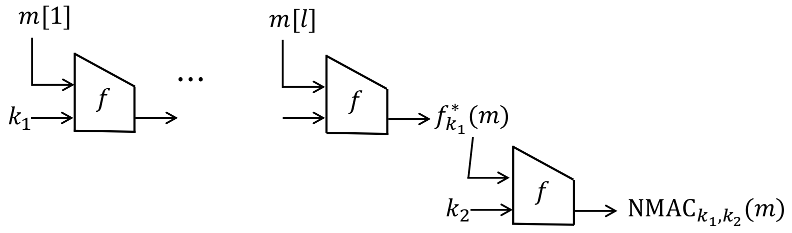

Fig. 1: $\mathsf{NMAC}_{k_1,k_2}(m)$: The top layer represents the cascade output and the bottom layer represents the finalization process applied on the output of the cascade.

Simplification. There is no loss in security if we assume message space as B_* and we ignore the padding rule. Throughout the paper, this convention is applied for all constructions. A minor change can be applied in the analysis of constructions with a more general message space.

Finally, we define another old cascade-based MAC construction, called the Envelope MAC or EvMAC. Let pad map a k-bit string to B. For example, if $k \le b$, we consider $\mathsf{pad}(K) = K \| 0^{b-k}$. We define a dual keyed function (interchanging the position of input and key) $f^{\downarrow}(K, x) := f(x, \mathsf{pad}(K))$. For any $m \in \mathsf{B}^+$, $K \in \{0,1\}^k$, we define $\mathsf{EvMAC}(K, m) = f^*(\mathrm{IV}, (K', m, K'))$ where $K' = \mathsf{pad}(K)$ and $\mathrm{IV} \in \{0,1\}^c$ is a fixed constant specified by the MD hash function based on f. Using the dual function notation, we can equivalently write

$$\mathsf{EvMAC}(K, m) = f^{\downarrow}\big(K, f^*(f^{\downarrow}(K, \mathrm{IV}), m)\big).$$

[1] Later in RFC 2014 [21] and in the special publication FIPS PUB 198-1 [12] by NIST, the MD hash was replaced by any recommended hash function H while defining HMAC. Similar to the original definition of HMAC, the new definition assumed the hash size to be less than the block size.

1.2 Existing Results on Cascade, NMAC, HMAC and EvMAC

PRF ADVANTAGE AGAINST A KEYED FUNCTION. For a function F, we write $F_K(x) = F(K, x)$ and view F as a keyed function. A *random function* $\mathsf{RF} := \mathsf{RF}_{\mathcal{X}\to\mathcal{Y}} \leftarrow_{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y})$ (the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$), where $s \leftarrow_{\$} S$ denotes the uniform sampling of s from a finite set S. We denote the PRF advantage of a distinguisher D against F as

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathsf{D}) := \Delta_{\mathsf{D}}(F\ ;\ \mathsf{RF}_{\mathcal{X}\to\{0,1\}^\tau}) := \big|\Pr(\mathsf{D}^F \to 1) - \Pr(\mathsf{D}^{\mathsf{RF}} \to 1)\big|,$$

$$\mathbf{Adv}_F^{\mathrm{prf}}(q', \ell, \sigma', T) := \max_{\mathsf{D}} \Delta_{\mathsf{D}}(F\ ;\ \mathsf{RF}_{\mathcal{X}\to\{0,1\}^\tau}),$$

where the maximum is taken over all q'-query distinguishers D running in time T such that the query tuple belongs to $\mathcal{M}(q', \ell, \sigma')$. We use the superscripts (i) nprf, (ii) pf_prf and (iii) pf_nprf when we restrict to distinguishers that are (i) non-adaptive (all queries are made before observing any responses), (ii) prefix-free (query tuples are prefix-free) and (iii) prefix-free non-adaptive, respectively. Given a function $G : \{0,1\}^k \to \{0,1\}^{k'}$ and a distinguisher D, we define the PRBG advantage of D against G as

$$\mathbf{Adv}_G^{\mathrm{prbg}}(\mathsf{D}) = |\Pr(\mathsf{D}(G(\mathsf{U})) = 1) - \Pr(\mathsf{D}(\mathsf{U}') = 1)|$$

where $\mathsf{U} \leftarrow_{\$} \{0,1\}^k$, $\mathsf{U}' \leftarrow_{\$} \{0,1\}^{k'}$. If the PRBG advantage is small for all efficient algorithms D, we call G a PRBG (pseudorandom bit generator). When G is a regular function (or an almost regular function) then the PRBG advantage for any distinguisher is zero (or negligible respectively).

MULTIUSER KEYED FUNCTIONS. Let $F : \mathcal{K}\times\mathcal{X} \to \mathcal{Y}$ be a keyed function. For any finite set $\mathcal{I}$ (called a *user-index space*), we define the *$\mathcal{I}$-folded multiuser keyed function* $F^{\otimes\mathcal{I}}(\gamma, x) := F(\mathsf{RF}(\gamma), x)$. When the user-index space is understood, we write $F^{\otimes}$. We can equivalently view this as an independent collection of keyed functions F. We denote the multiuser PRF advantage of a distinguisher D against F (with the user-index space $\mathcal{I}$) as

$$\mathbf{Adv}_F^{\mathrm{mu\_prf}}(u', q', q_{\max}, \ell, \sigma', \sigma_{\max}, T) := \max_{\mathsf{D}} \Delta_{\mathsf{D}}(F^{\otimes\mathcal{I}}\ ;\ \mathsf{RF}_{\mathcal{X}\times\mathcal{I}\to\mathcal{Y}}),$$

where the maximum is taken over all q'-query u'-user distinguishers D running in time T such that the query tuple to each user belongs to $\mathcal{M}(q_{\max}, \ell, \sigma_{\max})$ and the total length of all queries to all users is at most σ'. We also use the superscript mu_nprf when we consider non-adaptive distinguishers.

NOTATION. Throughout the paper, $T' = T + O(\sigma')$ and $\theta' = (u', q', q_{\max}, \ell, \sigma', \sigma_{\max})$. We write $A \setminus x$ or $A \cup x$ to denote $A \setminus \{x\}$ or $A \cup \{x\}$ respectively.

PRF Analysis on Cascade. The security of a fixed-length cascade construction (a special case of a cascade with a prefix-free domain) was first implicitly shown in 1984 [14] (and later published in 1986 [15]). The authors have proved asymptotically that a c-bit to $2c$-bit PRBG (pseudo-random bit generator) can be extended to a fixed-length PRF. Note that such a PRBG is equivalent to a PRF with a one-bit domain (i.e., $b = 1$). The PRBG to PRF construction is exactly the cascade construction after viewing the PRBG as a one-bit PRF. In 1996 [4], the GGM results were extended for a general value of b and with an arbitrary prefix-free domain:

$$\mathbf{Adv}_{f^*}^{\mathrm{pf\_prf}}(q', \ell, \sigma', T) \leq \ell q' \cdot \mathbf{Adv}_{f}^{\mathrm{prf}}(q', T').$$

In 2014 [13], Gazi et al. proved the above equation for non-adaptive PRF advantage.

PRF Analysis on NMAC and HMAC. Bellare [1] proved that

$$\mathbf{Adv}_{\mathsf{NMAC}^f}^{\mathrm{prf}}(q', \ell, \sigma'', T) \leq \ell q'^2 \cdot \mathbf{Adv}_{f}^{\mathrm{prf}}(2, O(\ell)) + \mathbf{Adv}_{f}^{\mathrm{prf}}(q', T').$$

Bellare assumed that a good compression function f must satisfy $\mathbf{Adv}_{f}^{\mathrm{prf}}(2, \ell) \approx \ell/2^c$ (presuming that key-guessing is the best strategy for distinguishing f from a random function) and hence the security of NMAC is dominated by the bound $\ell^2 q'^2/2^c$. Koblitz and Menezes observed that the reduction used in the preceding proof is non-constructive or existential (see [26,19] for details about different types of reductions). Moreover, the authors showed that there exists a large class of functions $f' : \{0,1\}^c \times \mathsf{B} \to \{0,1\}$ such that for all f' in the class, there exists a 1-query distinguisher $\mathcal{A}$ running in $O(1)$ time such that

$$\mathbf{Adv}_{f'}^{\mathrm{prf}}(\mathcal{A}) \geq \frac{1}{2^{c/2}}. \tag{2}$$

Later Bernstein and Lange [7] showed a distinguisher for which a similar result as 2 holds. This means that Bellare's result cannot guarantee security better than $\ell q'^2/2^{c/2}$. This violates the tightness claim of Bellare (see [18] for a detailed discussion). Later in [2], the above tightness claim was withdrawn and revised. In 2013, Koblitz-Menezes [18] also provided a constructive reduction and proved the following result (ignoring a dominated term):

$$\mathbf{Adv}_{\mathsf{NMAC}^f}^{\mathrm{prf}}(q', \ell, \sigma', T) \leq \ell q' \cdot \mathbf{Adv}_{f}^{\mathrm{prf}}(q', T'). \tag{3}$$

One year later in 2014 [13], Gazi et al. proved the following security of NMAC through a constructive reduction:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{NMAC}^f}(q',\ell,\sigma',T) \le \mathbf{Adv}^{\mathrm{nprf}}_{f^*}(q',\ell,\sigma',T') + \mathbf{Adv}^{\mathrm{prf}}_{f}(q',T') + q'^2/2^c. \tag{4}$$

$$\le \ell q' \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(q',T') + \mathbf{Adv}^{\mathrm{prf}}_{f}(q',T') + q'^2/2^c.$$

From the definition of HMAC, one can easily establish the following relation between the PRF securities of HMAC and NMAC:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{HMAC}^f}(q',\ell,\sigma',T) \le \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{NMAC}^f}(q',\ell,\sigma',T) + \mathbf{Adv}^{\mathrm{prbg}}_{\mathsf{KDF}}(T). \tag{5}$$

PRF Analysis on Envelope MAC. Yasuda proved PRF security of Envelope MAC (also called "Sandwich MAC," see [29]), along the lines of Bellare's NMAC security proof in [2]. Thus, the issues for NMAC are also present in his analysis. Koblitz and Menezes [20] proved the following constructive reduction of Envelope MAC

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{NMAC}^f}(q',\ell,\sigma',T) \le \ell q' \cdot \mathbf{Adv}^{\mathrm{prf}}_{f}(q',T') + 2q' \cdot \mathbf{Adv}^{\mathrm{prf}}_{f^{\downarrow}}(q',T') + \epsilon, \tag{6}$$

where $f^{\downarrow}$ is the dual of f. The term ϵ represents some related-key distinguishing advantage for the pair of functions $(f, f^{\downarrow})$, which cannot be derived from the PRF advantages of f and its dual.

1.3 Our Contributions

1. Multiuser PRF Security of Cascade. We provide two reductions for the multiuser PRF security of the cascade construction:

$$\mathbf{Adv}^{\mathrm{mu\_pf\_prf}}_{f^*}(\theta',T) \le \sigma' \cdot \mathbf{Adv}^{\mathrm{prf}}_{f}(q_{\max},T') \tag{7}$$

$$\mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^*}(\theta',T) \le \sigma' \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(1,T') + u \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(q_{\max},T'). \tag{8}$$

2. Non-adaptive PRF security under weak f. Due to the key guessing attack, the above bounds cannot guarantee a security better than $\sigma' T/2^c$. Suppose f is a keyed function with a higher non-adaptive PRF advantage, such as $\mathbf{Adv}^{\mathrm{nprf}}_{f}(D,T) \le DT/2^c$. We still prove a similar advantage (up to a logarithmic factor) against a non-adaptive distinguisher:

$$\mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^\star}(\theta',T) \le \frac{(\sigma' T + \sigma'^2)\cdot \log_2 q_{\max}}{2^c}$$

Applications to HMAC and NMAC. The generic reduction from NMAC to cascade (see Eq. 4) and HMAC to NMAC (see Eq. 5) can be easily extended for a multiuser setup in the following way:

$$\mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{NMAC}^f}(\theta', T) \leq \mathbf{Adv}^{\text{mu\_pf\_nprf}}_{f^*}(\theta', T') + \mathbf{Adv}^{\text{prf}}_{f}(q, T') + \frac{q'^2}{2^c} \quad (9)$$

$$\mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{HMAC}^f}(\theta', T) \leq \mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{NMAC}^f}(\theta', T) + \mathbf{Adv}^{\text{prbg}}_{\mathsf{KDF}}(T'). \quad (10)$$

Hence our results for non-adaptive PRF security of cascade can be directly applied for multiuser security of HMAC and NMAC.

3. Security of Single Keyed NMAC, constant-free HMAC and EvMAC. We prove the security of the single-keyed NMAC construction $\mathsf{1k\_NMAC}_K = \mathsf{NMAC}_{K,K}$. This helps not only to eliminate the two constants used in HMAC, but also to weaken the PRBG assumption on f. We also prove multiuser PRF security of Envelope MAC without assuming the related-key type assumption appearing in Eq. 6. In particular, we show the following three results:

$$\begin{aligned}
\mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{1k\_NMAC}^f}(\theta') \leq\ & \mathbf{Adv}^{\text{mu\_pf\_nprf}}_{f^*}(q', q', q_{\max}, \ell+1, \sigma', \sigma_{\max}, T')+ \\
& + u' \cdot \mathbf{Adv}^{\text{prf}}_{f}(2q_{\max}, T') + \frac{2q'^2}{2^c}, \\
\mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{EvMAC}}(\theta') \leq\ & \mathbf{Adv}^{\text{mu\_pf\_nprf}}_{f^*}(q', q', q_{\max}, \ell, \sigma', \sigma_{\max}, T')+ \\
& + u' \cdot \mathbf{Adv}^{\text{prf}}_{f}(q_{\max}+1, T') + \frac{q'^2}{2^c} \text{ and} \\
\mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{HMAC}'}(\theta') \leq\ & \mathbf{Adv}^{\text{mu\_prf}}_{\mathsf{1k\_NMAC}}(\theta'),
\end{aligned}$$

where HMAC' is same as HMAC when the two constants opad and ipad are replaced by the zero bit string. Moreover, the security of the modified HMAC does not require the PRBG property (it only needs the regular property as KDF does not expand the output size in the modified definition, assuming key size to be as large as the chain size c).

Once again, all of our non-adaptive multiuser PRF securities for the cascade construction are applicable to these variants. We finally note that all reductions in our analysis are constructive and so the bounds are applicable to a uniform setting when we naturally extend the result in an asymptotic set-up.

1.4 Key Intuition in Our Proof

Reduction for Cascade Construction. Let us consider a single user distinguisher of cascade which makes prefix-free queries $m_1, \ldots, m_q$. We view a rooted directed tree T over all prefixes of the queries. We define a directed edge (x, y) where $x = \mathsf{chop}(y)$ (removing the last block) and so λ is the root and $m_1, \ldots, m_q$ are all leaves (as queries are prefix-free). Now we define a depth-first ordering $\leq$ on the set of all intermediate nodes V' (i.e., excluding leave nodes). This is a linear order and it orders first all intermediate prefixes of m_1 as $\lambda < m_1[1] < \cdots < m_1[1..\ell_i - 1]$ where ℓ_i denotes the number of blocks present in m_i and $m_i[1..j]$ represents the j blocks of m_i. After ordering the first message we define the next element as the shortest intermediate prefix of m_2 which is not yet ordered. If there is no such element we can move to the next message, otherwise, starting from the shortest node for m_2 we can order till $m_2[\ell_2 - 1]$ (the last intermediate node for m_2). We continue this until we exhaust all elements of V'. Let $v_0 < v_1 < \cdots$ denote the complete ordering on V'.

Now we define hybrid oracles O_h for all $0 \leq h \leq |V'|$. All these hybrid oracles assign some random keys to some of the prefix-free nodes so that for every message m_i there is a prefix of m_i where a random key is assigned. From that node we can execute f^* function to define the output for m_i. To do so, we apply the function f in a sequence while we travel from that key node to m_i. If λ is the node where the random key is assigned then we actually compute f^*. On the other hand, if all leave nodes assign random keys then it represents a random function. All other key assigning will represent a hybrid oracle. We define O_0 as f^* oracle. We now define O_1 as the hybrid oracle where random keys are assigned to all children of λ. Suppose we have defined O_h which assigns random keys on the set of nodes C_h and the smallest element from $C_h \setminus m^q$ is v_h. We define recursively O_{h+1} where we assign random keys to all children of v_h instead of v_h and the rest of keys are the same as O_h. It is easy to see that $\mathsf{O}_{|V'|}$ will assign random keys to all the leave nodes.

After defining a set of hybrid oracles, we can simulate with the help of an oracle $\mathcal{O}$ which is either $f(K, \cdot)$ or Γ (random function). $\mathsf{Sim}^{\mathcal{O}}(h)$ assigns random keys to all nodes used for O_h except v_h. For all messages m_i with v_h as a prefix, $\mathsf{Sim}(h)$ calls $\mathcal{O}$ as if the key of the oracle is placed on v_h. So, when $\mathcal{O} = f(K, \cdot)$, the simulator actually simulates O_h. On the other hand, when $\mathcal{O} = \Gamma$, simulator evaluates random keys on the children of v_h through the oracle Γ. In other words, it simulates O_{h+1}. This completes the hybrid argument. We obtain a tightness gap as the maximum number of prefixes of all queries.

At a first glance, one may think that the above reduction is defined for non-adaptive distinguisher only. However, the way we have defined the depth-first ordering, the simulator can be defined for adaptive algorithm too. Moreover, for non-adaptive distinguisher we actually have an improved reduction. After observing all queries, we can identify the set of h values for which the number of children of v_h is one. By a simple argument, we can show that the number of nodes with at least two children is at most q. Show we can reduce to two algorithms one makes only one query and the other makes at most q queries with a tightness gap q. It can be easily extended for multiple reduction with a fine tuned trade-off between tightness gaps and query complexity.

Reduction for Single-keyed NMAC. We have proved PRF security of single-keyed NMAC. Note that $\mathsf{1k\_NMAC}_K(m) := g(f^*(g(m[1]), m[2..\ell]))$ where $g(x) = f(K, x)$ for a random key K. Due to standard hybrid argument, it is sufficient to analyze the hybrid construction $\mathsf{\Gamma}(f^*(\mathsf{\Gamma}(m[1]), m[2..\ell]))$. Let y_i denote $f^*(\mathsf{\Gamma}(m_i[1]), m_i[2..\ell])$ for the ith query m_i. As long as there is no collision in y_i values and no collision between y_i and $m_i[1]$ values, $\mathsf{\Gamma}(y_i)$ values behave perfectly random. Let us define the bad event bad true whenever such collision happened. As the final output also hides the messages by the $\mathsf{\Gamma}$, we can reduce the PRF security to a non-adaptive adversary which triggers the above bad event by its queries. The y_i values can be viewed as a multiuser cascade output where the first block of the message is the user index and the remaining message as an input. So, our multiuser analysis of cascade can be applied provided the queries are prefix-free. So we append one extra block, say x, to every message so that the new messages are prefix-free. We can define a new bad event bad' as $f(y_i, x) = f(y_j, x)$, $i \neq j$, or $f(y_i, x) = f(m_j[1], x)$. Clearly, bad' holds whenever bad holds and we can bound the probability of bad' using the prefix-free multiuser PRF adversary of cascade construction.

2 Preliminaries

2.1 Distinguisher

Oracle Algorithm. An $(\mathcal{X}, \mathcal{Y})$-oracle $\mathcal{O}$ is an interactive probabilistic algorithm that takes inputs from the set $\mathcal{X}$ and returns elements from the set $\mathcal{Y}$. *A keyed function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is viewed as an $(\mathcal{X}, \mathcal{Y})$-oracle where the key $K \leftarrow\!\!\$\, \mathcal{K}$, and then for each query x it returns $F_K(x)$. The $\mathcal{I}$-folded multiuser F can be similarly viewed as an $(\mathcal{I} \times \mathcal{X}, \mathcal{Y})$-oracle.* Conversely, any $(\mathcal{I} \times \mathcal{X}, \mathcal{Y})$-oracle F' such that $F'(\gamma, x) := F(\mathsf{RF}(\gamma), x)$ can be viewed as an $\mathcal{I}$-folded multiuser oracle F.

Oracle. A q'-query t-time $(\mathcal{X},\mathcal{Y})$-oracle algorithm $\mathcal{A}$ is an interactive algorithm that can interact with any $(\mathcal{X},\mathcal{Y})$-oracle $\mathcal{O}$ (also called a compatible oracle), that makes at most q' queries to its oracle, runs for time t, and finally returns some output (if the output is a single bit, we also call it a *distinguisher*). We call $\mathcal{A}$ *non-adaptive* if the tuple of all queries x^q does not depend on its oracle or its responses. In other words, all queries are computed before the interaction begins. We denote the oracle as $[x^q]$. When $\mathcal{O}$ is a multiuser oracle, $\mathcal{A}$ is called a u'-user oracle algorithm if the number of distinct user indices queried among all q' queries is at most u'. Let $\tau(\mathcal{A}^{\mathcal{O}}) := (\tau_{\text{query}}(\mathcal{A}^{\mathcal{O}}) := x^q, \tau_{\text{resp}}(\mathcal{A}^{\mathcal{O}}) := y^q)$ denote the pair of tuples of queries and responses and call it the *transcript*. Here, q represents the number of queries and so $q \leq q'$. In the case of a u'-user, q'-query oracle algorithm $\mathcal{A}$ and for a query transcript $\tau_{\text{query}}(\mathcal{A}^{\mathcal{O}}) := (\gamma^q, x^q)$, we have $q \leq q'$, $|\gamma^q| \leq u'$ (when γ^q is viewed as a set).

2.2 Complexity Notation for Distinguisher

A distinguisher D is called an $(\mathcal{M}, t)$-complexity distinguisher if D runs for time t and for all compatible oracles $\mathcal{O}$, the transcript $\tau_{\text{query}}(\mathsf{D}^{\mathcal{O}}) \in \mathcal{M}$. We call $\mathcal{M}$ the *joint query space* for D. A joint query space is called prefix-closed if for all $m^q \in \mathcal{M}$ and for all $i \in [q]$, $m^i \in \mathcal{M}$.

Examples of Joint Query Space. We quickly recall that $\mathcal{M}(q', \ell', \sigma')$ represents the set of all tuples of at most q' messages so that the longest message has at most ℓ' blocks and the number of blocks in all messages is at most σ'. For any $(\gamma^q, m^q) \in \mathcal{I}^q \times \mathcal{M}(q', \ell', \sigma')$ with distinct (γ_i, m_i), let $\gamma'_1, \ldots, \gamma'_u$ denote the distinct elements of γ^q (i.e., $|\gamma^q| = u$). Let $M_i = \{m_j : \gamma_j = \gamma'_i\}$. We define $\sigma_{\max}(\gamma^q, m^q) = \max_i \sum_{j \in M_i} \|m_j\|$. Let

$$\mathcal{M}(u', q', q'_{\max}, \ell', \sigma', \sigma'_{\max}) = \{(\gamma^q, m^q) \in \mathcal{M}(u', q', \ell', \sigma') : \mathsf{mcoll}(\gamma^q) \leq q'_{\max}, \sigma_{\max}(\gamma^q, m^q) \leq \sigma'_{\max}\}.$$

When $\ell = 1$ (e.g. $\mathcal{X} = \mathsf{B}$), we simply write the above sets as $\mathcal{M}^{\text{mu}}(u', q', q'_{\max})$. For any joint query space $\mathcal{M}$, we write $\mathcal{M}_{\text{pf}} = \mathcal{M} \cap \mathcal{P}$ where $\mathcal{P}$ is the set of all prefix-free block tuples.

Definition 1. *Two oracles $\mathcal{O}$ and $\mathcal{O}'$ are called* equivalent *on a joint query space $\mathcal{M}$, denoted as $\mathcal{O} \cong_{\mathcal{M}} \mathcal{O}'$, if for all $x^q \in \mathcal{M}$, $\tau_{\text{resp}}([x^q]^{\mathcal{O}}) \cong \tau_{\text{resp}}([x^q]^{\mathcal{O}'})$ (follow the same probability distribution).*

Let $\mathscr{D}(\mathcal{M}, t)$ (and $\mathscr{D}_{\text{na}}(\mathcal{M}, t)$) denote the set of all $(\mathcal{M}, t)$-complexity distinguishers (and non-adaptive distinguishers, respectively). If $\mathcal{M}(\theta)$

is determined by some complexity parameter θ, we also write the class of distinguishers as $\mathscr{D}(\theta, t)$ or $\mathscr{D}_{\text{na}}(\theta, t)$. We also use superscript "mu" to denote the multiuser distinguisher class to avoid any confusion. For simplicity, the time parameter is sometimes ignored.

2.3 Distinguishing Security Notions

Definition 2 (distinguishing advantage). *Let F and G be $(\mathcal{X}, \mathcal{Y})$-oracles. We define the* distinguishing advantage *of a distinguisher* D *as* $\varDelta_{\mathsf{D}}(F\ ;\ G) := \left|\varDelta^*_{\mathsf{D}}(F\ ;\ G)\right|$ *where* $\varDelta^*_{\mathsf{D}}(F\ ;\ G) = \Pr(\mathsf{D}^F = 1) - \Pr(\mathsf{D}^G = 1)$. *For a joint query space $\mathcal{M}(\theta)$,*

$$\varDelta_{\theta,t}(F\ ;\ G) = \max_{\mathsf{D} \in \mathscr{D}(\theta,t)} \varDelta_{\mathsf{D}}(F\ ;\ G), \quad \varDelta^{\text{na}}_{\theta,t}(F\ ;\ G) = \max_{\mathsf{D} \in \mathscr{D}_{\text{na}}(\theta,t)} \varDelta_{\mathsf{D}}(F\ ;\ G).$$

In the above definition when G is a random function, we call the distinguishing advantages the PRF-advantages of D against F. More precisely, the (multiuser) PRF-advantage of D against F are $\mathbf{Adv}^{\text{prf}}_F(\mathsf{D}) := \varDelta_{\mathsf{D}}(F\ ;\ \mathsf{RF}_{\mathcal{X} \to \mathcal{Y}})$ and $\mathbf{Adv}^{\text{mu\_prf}}_F(\mathsf{D}) := \varDelta_{\mathsf{D}}(F^{\otimes}\ ;\ \mathsf{RF}^{\otimes}_{\mathcal{X} \to \mathcal{Y}})$. We define

$$\mathbf{Adv}^{\text{prf}}_F(\theta, t) = \max_{\mathsf{D} \in \mathscr{D}(\theta,t)} \mathbf{Adv}^{\text{prf}}_F(\mathsf{D}), \quad \mathbf{Adv}^{\text{nprf}}_F(\theta, t) = \max_{\mathsf{D} \in \mathscr{D}_{\text{na}}(\theta,t)} \mathbf{Adv}^{\text{prf}}_F(\mathsf{D}).$$

We use the superscript mu_prf and mu_nprf when we consider multiuser distinguishers against multiuser keyed functions. We sometimes write $\mathbf{Adv}^{\text{nprf}}_F(\cdot)$ when we consider only non-adaptive distinguishers.

2.4 Reduction Algorithm

A **reduction algorithm** Sim is a (two-sided) interactive algorithm that can simultaneously interact with (i) an oracle algorithm $\mathcal{A}$ (with which the reduction algorithm behaves like a compatible oracle) and (ii) a compatible oracle $\mathcal{O}$. Thus $\mathsf{Sim}^{\mathcal{O}}$ is an oracle for $\mathcal{A}$, whereas $\mathcal{A} \triangleright \mathsf{Sim}$ (the combined algorithm) behaves like an oracle algorithm. We call the reduction algorithm $(\mathcal{M}, t) \to (\mathcal{M}', t')$ if for every $(\mathcal{M}, t)$-complexity algorithm $\mathcal{A}$, $\mathcal{A} \triangleright \mathsf{Sim}$ is an $(\mathcal{M}', t')$-complexity algorithm. Note that $t' = t + t''$, where t'' denotes the time taken by the reduction algorithm. So, we also call it an $\mathcal{M} \to \mathcal{M}'$ reduction.

(Randomized) Hybrid Reduction. The definition of a reduction algorithm can be straightforwardly extended to one that has an initial input, called a **hybrid reduction**. For any input a, $\mathsf{Sim}(a)$ would behave exactly the same as a reduction algorithm. For a set I, $\mathsf{Sim}(h \leftarrow_{\$} I)$ works as follows:

– $h \leftarrow_{\$} I$.
– If $\mathsf{Sim}(h)$ is not defined, abort.
– Else, run $\mathsf{Sim}(h)$.

Definition 3 (d-step reduction). *A hybrid reduction algorithm* Sim *is called a d-step oracle reduction from a pair of oracles (F', G') to a pair of oracles (F, G) on a joint query space $\mathcal{M}$ if there is a set $I = \{i_1, \ldots, i_{d+1}\}$, called* hybrid-index set, *and $d + 1$ oracles $\mathcal{H}_{i_1}, \ldots, \mathcal{H}_{i_{d+1}}$, called a* hybrid oracles *with $\mathcal{H}_{i_1} \cong_{\mathcal{M}} F'$, $\mathcal{H}_{i_{d+1}} \cong_{\mathcal{M}} G'$ such that for all $h \in I$,*

– $\mathsf{Sim}(h)^F \cong_{\mathcal{M}} \mathcal{H}_{i_h}$ *and* $\mathsf{Sim}(h)^G \cong_{\mathcal{M}} \mathcal{H}_{i_{h+1}}$.

When G and G' are random functions, we call Sim *a d-step reduction from F' to F.*

Remark 1. Note that in the above definition, for any $1 \le h < h' \le d + 1$, $(\mathcal{H}_{i_h}, \mathcal{H}_{i_{h'}})$ is reduced to (F, G) in $(h' - h)$ steps.

Lemma 1 (hybrid reduction). *Let* Sim *be an $\mathcal{M} \to \mathcal{M}'$ reduction algorithm running for time t_{Sim}. Suppose it is a d-step oracle reduction from (F', G') to (F, G) on a query space $\mathcal{M}$ with hybrid-index set I. Then for any $(\mathcal{M}, t)$-complexity distinguisher* D*, we have an $(\mathcal{M}', t + t_{\mathsf{Sim}})$-complexity distinguisher* $\mathsf{D}' := \mathsf{D} \triangleright \mathsf{Sim}(h \leftarrow_{\$} I)$ *such that*

$$\Delta_{\mathsf{D}'}(F \ ; \ G) = \frac{1}{d} \cdot \Delta_{\mathsf{D}}(F' \ ; \ G').$$

Lemma 2. *Let $\mathcal{I}, \mathcal{X}, \mathcal{Y}$ be finite sets. Let* $\mathsf{RF} := \mathsf{RF}_{\mathcal{X} \to \mathcal{Y}}$ *and* $\mathsf{RF}' := \mathsf{RF}_{\mathcal{X} \times \mathcal{I} \to \mathcal{Y}}$*. Suppose $(\mathsf{RF}_i)_{i \in \mathcal{I}}$ is an independent collection of random functions from $\mathcal{X}$ to $\mathcal{Y}$. Then, for all $i^q \in \mathcal{I}^q, x^q \in \mathcal{X}^q$,*

$$\big(\mathsf{RF}_{i_1}(x_1), \ldots, \mathsf{RF}_{i_q}(x_q)\big) \ \cong \ \big(\mathsf{RF}'(i_1, x_1), \ldots, \mathsf{RF}'(i_q, x_q)\big)$$

(i.e., the the probability distributions of the above two random vectors are the same).

The above result is a simple property of multiuser random functions which is used in this paper. When all (i_j, x_j)'s are distinct both random vectors have uniform distribution over $\mathcal{Y}^q$. This is obvious for the R.H.S. vector. This can be also shown for the L.H.S. vector as random functions with different user indices are independent.

3 A New Reduction for Cascade

In this section we provide a new reduction proof for cascade, both for adaptive and non-adaptive distinguishers. Moreover, our reduction is a straight reduction unlike the existing two-stage reduction.

3.1 Prefix-Tree

Let $\mathcal{X} = \mathcal{I} \times \mathsf{B}_+$ and $\mathcal{X}' = \mathcal{I} \times \mathsf{B}_*$. Let us fix a parameter tuple $\theta' = (u', q', q_{\max}, \ell, \sigma', \sigma_{\max})$ and $m^q \in \mathcal{M}(\theta)$. For $1 \leq i \leq q \leq q'$, let $m_i = m_i[0..\ell_i] \in \mathcal{X}$ (i.e., $\|m_i\| = \ell_i$). Note, $\sigma := \sum_i \ell_i \leq \sigma'$ and $u = |m^q[0]| \leq u'$ where u denotes the number of distinct $m_i[0]$ in m^q. . Considering elements of $\mathcal{I}$ as blocks, we can extend the notion of prefixes to $\mathcal{X}$.

1. $V = \mathsf{Prefix}(m^q) = \{x : x \preceq m_i,\ i \in [q]\}$ and
2. $V' = \mathsf{Prefix}'(m^q) := \{x : x \prec m_i,\ i \in [q]\}$.

We associate a directed tree (called prefix tree) $T_{\mathfrak{m}}$ over the vertex set V consisting of all directed edges of the form $\mathsf{chop}(y) \to y$ for $y \in V \setminus \lambda$. It is a rooted tree with λ as the root (it is the only vertex with in-degree zero). For every $v \in V$, we define the set of outwards nodes as $\mathsf{ch}(v) = \{u : v \to u\}$. Let L denote the set of leaf nodes (having zero out-degree). The set V' is the set of all non-leaf or intermediate nodes. When m^q is prefix-free, V' is disjoint with m^q and $L = m^q$. We denote $d := |V'| - 1$. It is easy to see that for prefix-free m^q, $d \leq \sigma \leq \sigma'$.

Depth-First Ordering. Let m^q be prefix-free. We define a bijective function $\mathsf{DF} : V' \to [0..d]$ and a sequence $v_0 = \lambda, v_1, \ldots, v_d$ where $\mathsf{DF}(\lambda) = 0$ and for all $x \in V' \setminus \lambda$ we compute as follows.

```
1. Initialize $ctr = 1$
2. for $i = 1$ to $q$
3. for $j = 1$ to $\ell_i - 1$
       if $\mathsf{DF}(m_i[..j])$ is not defined then
          $\mathsf{DF}(m_i[..j]) = ctr$, $v_{ctr} = m_i[..j]$ and $ctr \leftarrow ctr + 1$.
```

Note that $v_h = \mathsf{DF}^{-1}(h)$ for all $h \leq d$. For all $h > d$, v_h is undefined. We note that the definition of v_h depends on the choices of m^q. We write $u <_{\mathrm{DF}} v$ (or $u \leq_{\mathrm{DF}} v$) if $\mathsf{DF}(u) < \mathsf{DF}(v)$ (or $\mathsf{DF}(u) \leq \mathsf{DF}(v)$ respectively). Note that for all $x \prec y$, $\mathsf{DF}(x) < \mathsf{DF}(y)$.

3.2 Adaptive Reduction of Multiuser Cascade

Root Assignment Function. Let $\mathsf{chop}(m_i) := m_i[..\ell_i - 1]$. For every $h \in [0..\sigma']$, let $\mathsf{cut}_h(m^i) = m_i$ if $\mathsf{DF}(\mathsf{chop}(m_i)) < h$, otherwise it is defined as the shortest prefix x of m_i such that $\mathsf{DF}(x) \geq h$. Note that $\mathsf{DF}(x)$, for all $x \preceq m_i$, depend only on m^i and hence $\mathsf{cut}_h(m^i)$ can be computed adaptively. We call cut_h *root assignment function*. By definition of cut_h, for all $m^q \in \mathcal{M}$, $i \in [q]$, $\mathsf{cut}_h(m^i) \prec m_i$. Let $C_h := \{\mathsf{cut}_h(m^i) : i \in [q]\}$.

1. We observe that C_h is prefix-free. If not, then there would exist $i \neq j$ such that $\mathsf{cut}_h(m_i) \prec \mathsf{cut}_h(m_j)$. Hence, $\mathsf{cut}_h(m_j)$ would be a prefix of m_j with DF a value of at least h. This contradicts the fact that $\mathsf{cut}_h(m_j)$ is the shortest such prefix.
2. Note that $\mathsf{cut}_0(m_i) = \lambda$, $\mathsf{cut}_1(m_i) = m_i[0]$. Note that for all $x \in C_h \setminus m^q$, $\mathsf{DF}(x) \geq h$. So, $\mathsf{cut}_h(m_i) = m_i$ for all $i \in [q]$, $h \geq d$. Moreover, for all $h \leq |V'|$, $v_h \in C_h$.

Lemma 3. *For all $h < d$, $C_{h+1} = (C_h \setminus v_h) \cup \mathsf{ch}(v_h)$.*

Proof. Clearly $v_h \notin C_{h+1}$ as its DF value is less than $h+1$. Let $j = \|v_h\|$. For every i with $v_h \prec m_i$ (equivalently, $\mathsf{cut}_h(m_i) = v_h$), by definition, $\mathsf{cut}_{h+1}(m_i) = m_i[0..j] \in \mathsf{ch}(v_h)$. This proves that $\mathsf{ch}(v_h) \subseteq C_{h+1}$. For all i with $\mathsf{cut}_h(m_i) \neq v_h$, we have $\mathsf{cut}_{h+1}(m_i) = \mathsf{cut}_h(m_i)$. This completes the proof. □

Let $\mathcal{K}$ be a finite set. Given a function $\mathbf{K} : \mathcal{X}_\lambda \to \mathcal{K}$, called key assignment function, and a root assignment function cut_h, we define an oracle $\mathsf{O}(\mathbf{K}, \mathsf{cut}_h)$ as follows. On ith query m_i, for $i \geq 1$:

```
1. $u_i = \mathsf{cut}_h(m^i)$
2. $K_i = \mathbf{K}(u_i)$
3. return $z_i = f^*(K_i, m_i \setminus u_i)$
```

In this paper we mostly consider a random function as the key assignment function. In other words, $\mathbf{K} = \mathsf{RF} := \mathsf{RF}_{\mathcal{X}_\lambda \to \mathcal{K}}$. We write O_h to denote $\mathsf{O}(\mathsf{RF}, \mathsf{cut}_h)$. Let $\mathcal{M} := \mathcal{M}(\theta')$.

Lemma 4. *(1) $\mathsf{O}_1 \equiv_{\mathcal{M}_{\mathrm{pf}}} f^{*\otimes}$ with $\mathcal{I}$ as user index space and B_+ as input space. (2) $\mathsf{O}_{\sigma'} \equiv_{\mathcal{M}_{\mathrm{pf}}} \mathsf{RF}$.*

Proof. Note that $\mathsf{cut}_1(m^i) = m_i[0]$ for all i. So, $\mathsf{O}_{1,f^*}(m_i) = f^*(\mathsf{RF}(m_i[0]), m_i[1..])$. This proves (1). To prove (2), we note that $\mathsf{cut}_d(m_i) = m_i$ for all i. So, $\mathsf{O}_d(m_i) = f^*(\mathsf{RF}(m_i[0..\ell_i - 1]), \lambda) = \mathsf{RF}(m_i)$. □

Reduction from $f^{*\otimes}$ to f. We now define the σ'-step $\mathcal{M}_{\mathrm{pf}} \to \mathsf{B}^{\leq q'_{\max}}$ reduction $\mathsf{Sim}^{\mathcal{O}}$ from $f^{*\otimes}$ to f. It keeps its own random function, denoted as $\mathsf{RF}_{\mathsf{Sim}}$. For every $h \in [\sigma']$, $\mathsf{Sim}(h)$ runs as follows, where $j = \|v_h\|$ whenever v_h is defined: On ith query m_i, for $i \geq 1$:

```
1. $u_i = \mathsf{cut}_h(m^i)$
2. if $v_h$ is defined and $u_i = v_h$ then return $f^*(\mathcal{O}(m_i[j]), m_i[j+1..])$
3. else return $z_i = f^*(\mathsf{RF}_{\mathsf{Sim}}(u_i), m_i \setminus u_i)$
```

Lemma 5. *For all $h < \sigma'$, (1) $\mathsf{Sim}^{f_K}(h) \cong_{\mathcal{M}_{\rm pf}} \mathsf{O}_h$ and (2) $\mathsf{Sim}^{\Gamma}(h) \cong_{\mathcal{M}_{\rm pf}} \mathsf{O}_{h+1}$.*

Proof. It is obvious that Sim makes at most $q'_{\max}$ queries to its oracle and hence it is an $\mathcal{M}_{\rm pf} \to \mathcal{M}(q'_{\max})$ reduction algorithm. We denote $u_i = \mathsf{cut}_h(m_i)$ for all i.

$\mathcal{O} = f_K$: For all query m_i with $\mathsf{cut}_h(m^i) = v_h$, we define $\mathbf{K}(v_h) = K$ and it returns

$$\begin{aligned} z_i = f^*(f(K, m_i[j]), m_i[j+1..]) &= f^*(K, m_i \setminus u_i) \\ &= f^*(\mathbf{K}(v_h), m_i \setminus u_i). \end{aligned}$$

Similarly when $u_i := \mathsf{cut}_h(m^i) \neq v_h$, we define $\mathbf{K}(u_i) = \mathsf{RF}_{\mathsf{Sim}}(u_i)$ and it returns $z_i = f^*(\mathsf{RF}_{\mathsf{Sim}}(u_i), m_i \setminus u_i) = f^*(\mathbf{K}(v_h), m_i \setminus u_i)$. Note that $\mathbf{K}$ behaves exactly like a random function (the key K is uniform and independent to the random function $\mathsf{RF}_{\mathsf{Sim}}$). Combining both cases, we have $\mathsf{Sim}^{f_K}(h) \cong_{\mathcal{M}_{\rm pf}} \mathsf{O}_h$.

$\mathcal{O} = \Gamma$: For all m_i with $u_i := \mathsf{cut}_h(m^i) = v_h$, we have $z_i = f^*(\Gamma(m_i[j]), m_i[j+1..])$. Similarly when $u_i := \mathsf{cut}_h(m^i) \neq v_h$: the final output $z_i = f^*(\mathsf{RF}_{\mathsf{Sim}}(u_i), m_i \setminus u_i)$. So we define $\mathbf{K}$ on C_{h+1} as follows: For all $(v_h, a) \in \mathsf{ch}(v_h)$, $\mathbf{K}(x) = \Gamma(a)$ and for all other $x \in C_{h+1}$, $\mathbf{K}(x) = \mathsf{RF}_{\mathsf{Sim}}(x)$. Once again, $\mathbf{K}$ behaves like a random function defined over C_{h+1} (by Lemma 2) and so $\mathsf{Sim}^{\Gamma}(h) \cong_{\mathcal{M}_{\rm pf}} \mathsf{O}(\mathbf{K}, \mathsf{cut}_{h+1}) \cong_{\mathcal{M}_{\rm pf}} \mathsf{O}_{h+1}$. □

Example 1. In this example, we consider $b = 1$, $q = 6$, $m_1 = 000, m_2 = 0010, m_3 = 10, m_4 = 1100, m_5 = 1101, m_6 = 111$. The DF function over all intermediate nodes is illustrated in Fig.2. For example, $\mathsf{DF}(1) = 4$. The figure (1) (and similarly (2)) describes how $\mathsf{Sim}(4)^{\mathcal{O}}$ (respectively $\mathsf{Sim}(5)^{\mathcal{O}}$) works. It uses oracle $\mathcal{O}$ for all queries where $v_4 = 1$ (or $v_5 = 11$ respectively) is a prefix. The figures (3) and (4) represent the oracles O_4 and O_5. It is easy to see from the figure that $\mathsf{Sim}(4)^{\mathsf{RF}(\cdot)} \equiv \mathsf{O}_5 \equiv \mathsf{Sim}(5)^{f(K,\cdot)}$.

The following result follows from the hybrid reduction lemma.

Theorem 1. *Every (θ', T)-distinguisher D can be reduced to a $(q'_{\max}, T')$-distinguisher $\mathsf{D}' := \mathsf{D} \triangleright \mathsf{Sim}$ such that*

$$\mathbf{Adv}^{\rm pf\_prf}_{f^{*\otimes}}(\mathsf{D}) \leq \sigma' \cdot \mathbf{Adv}^{\rm prf}_{f}(\mathsf{D}'). \tag{11}$$

Hence, $\mathbf{Adv}^{\rm mu\_pf\_prf}_{f^{*\otimes}}(\theta', T) \leq \sigma' \cdot \mathbf{Adv}^{\rm prf}_{f}(q'_{\max}, T')$. Moreover, the reduction is non-adaptive whenever D is non-adaptive.

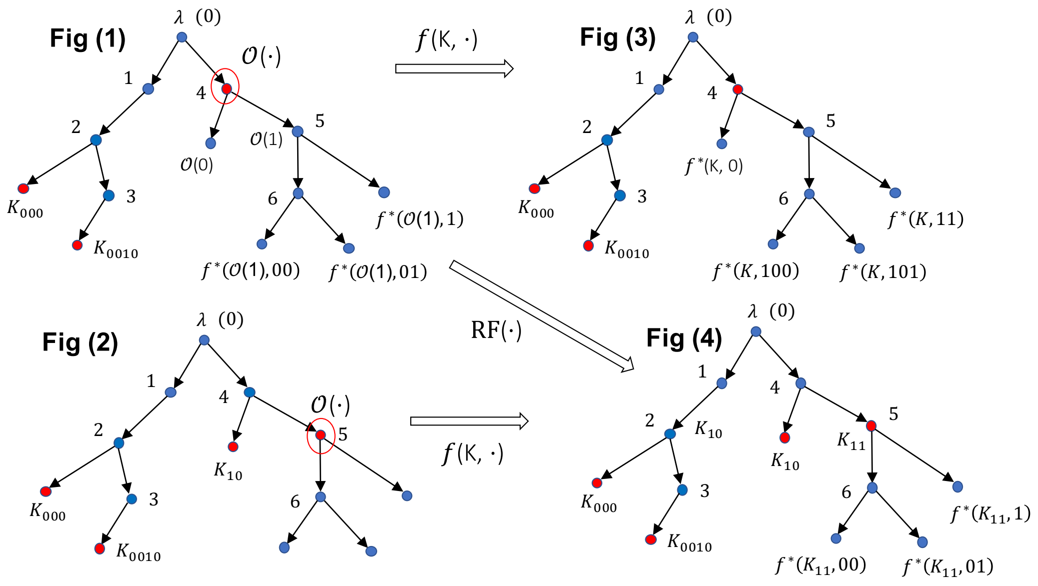

Fig. 2: Fig (1) is $\mathsf{Sim}(4)$, Fig (2) is $Sim(5)$, Fig(3) is O_4 and Fig (4) is O_5.

3.3 Improved Bound for Non-Adaptive Distinguisher

In the last subsection, we have shown a multiplicative gap of σ' for the cascade, which is definitely an improvement over the previously known bound of $\ell' q'$. Now we show that we can improve the query complexity in case of the non-adaptive bound. In Theorem 1, we reduce to a $q_{\max}$-query algorithm. However, it is easy to see that the number of queries can be much less (depending on the choice of h and m^q). In fact, we show that given m^q, except about q choices, all other h-values reduce to a single-query algorithm. To conclude this, we need the following simple result on a rooted tree.

Lemma 6. *Let $V(T)$ and $L(T)$ denote the set of nodes and the set of leaf nodes of a rooted tree T respectively and $V' = V \setminus L$. Then, $\sum_{v \in V'}(|\mathsf{ch}(v)| - 1) = |L(T)| - 1$. Hence, we have*

$$(1) \quad |\{v : |\mathsf{ch}(v)| > 1\}| \le |L(T)| - 1, \quad (2) \quad |\{v : |\mathsf{ch}(v)| > i\}| < |L(T)|/i.$$

Proof. The two equations (1) and (2) directly follow from the first part. We prove the first part. Note that for every rooted tree, the sum of number of all children is $|V(T)| - 1$ (all nodes except the root node are children). As leaf nodes do not have any children it is equivalent to summing over all vertices from V'. So

$$\sum_{v\in V'}(|\mathsf{ch}(v)|-1) = \sum_{v\in V}|\mathsf{ch}(v)| - |V'|$$
$$= |V| - 1 - |V'| = |L| - 1. \quad \square$$

We now a define reduction for a non-adaptive algorithm that uses the simulator $\mathsf{Sim}(h)$ defined as before for the adaptive distinguisher in the previous subsection. For all $h \le \sigma'$, let $\mathsf{q}(h, m^q)$ denote the number of queries $\mathsf{Sim}(h)$ makes to its oracle whenever m^q denotes all non-adaptive queries. It is easy to see that $\mathsf{q}(h, x^q) = |\mathsf{ch}(v_h)|$, whenever v_h is defined. We have seen that for all h, m^q, $\mathsf{q}(h, m^q) \le q_{\max}$. Let

$$\mathcal{N}_{[i..j]}(m^q) := \{h \in [\sigma'] : i \le \mathsf{q}(h, m^q) \le j\}$$

and $N_{[i..j]}$ denote the maximum size of the set $\mathcal{N}_{[i..j]}$, varying over all $m^q \in \mathcal{M}_{\mathrm{pf}}$. Note that $|\mathcal{N}_{[q']}| = \sigma'$ for all random coins as q' is the maximum number of queries made by Sim. Now, we split the interval by setting $q_0 = -1 \le q_1 < \cdots < q_c = q_{\max}$. So, $[0..q_{\max}]$ is partitioned into intervals $(q_0, q_1], \ldots, (q_{c-1}, q_c]$ where $(a, b]$ represents the set $[a+1..b]$. The values of q_i will be determined later. We denote $N_i := N_{(q_{i-1}, q_i]}$. Let $\perp_i$ be a set of size i so that for every $h \in \perp_i$, $\mathsf{Sim}(h)$ is not defined. We now define Sim_i as

$$\mathsf{Sim}(h \leftarrow\!\!\$\, \mathcal{N}_{(q_{i-1}, q_i]} \cup \perp_{\delta_i}),$$

where $\delta_i = N_i - \mathsf{N}_i$ and $|\mathcal{N}_{(q_{i-1}, q_i]}| = \mathsf{N}_i$. Now for any distinguisher D, let $\mathsf{D}_i = \mathsf{D} \triangleright \mathsf{Sim}_i$ and $\mathcal{C}_h = \mathsf{D} \triangleright \mathsf{Sim}(h)$. By definition, D_i makes at most q_i queries and

$$\varDelta^*_{\mathsf{D}_i}(f \ ;\ \mathsf{\Gamma}) = \frac{1}{N_i} \cdot \sum_{h \in \mathcal{N}_{(q_{i-1}, q_i]}} (\Pr(\mathcal{C}_h^f = 1) - \Pr(\mathcal{C}_h^{\mathsf{\Gamma}} = 1)).$$

So,

$$\begin{aligned}
\varDelta^*_{\mathsf{D}}(f^{*\otimes} \ ;\ \mathsf{RF}) &= \Pr(\mathsf{D}^{\mathsf{Sim}(1)\,\triangleright\, f} = 1) - \Pr(\mathsf{D}^{\mathsf{Sim}(d)\,\triangleright\, \mathsf{\Gamma}} = 1) \\
&= \sum_{i=1}^{\sigma'} \big(\Pr(\mathsf{D}^{\mathsf{Sim}(i)\,\triangleright\, f} = 1) - \Pr(\mathsf{D}^{\mathsf{Sim}(i)\,\triangleright\, \mathsf{\Gamma}} = 1)\big) \\
&= \sum_{i=1}^{c} \sum_{h \in \mathcal{N}(q_{i-1}..q_i]} \varDelta^*_{\mathcal{C}_h}(f \ ;\ \mathsf{\Gamma}) \\
&= \sum_{i=1}^{c} N_i \cdot \varDelta^*_{\mathsf{D}_i}(f \ ;\ \mathsf{\Gamma}).
\end{aligned}$$

So, we have proved following general reduction for a non-adaptive distinguisher:

Theorem 2. *Following the notations as described above, let* D *be a* q'*-query* $(\mathcal{M}_{\mathrm{pf}}(\theta'), t)$ *non-adaptive distinguisher. Then,*

$$\Delta_{\mathsf{D}}(f^{*\otimes}\ ;\ G') \leq \sum_{i=1}^{c} N_{[q_{i-1}+1..q_i]} \cdot \Delta_{\mathsf{D}_i}(f;\ \Gamma).$$

Now we consider two partitions of $[0..q_{\max}]$. For each such choice, we have the following immediate corollary: (1) $N_{[2..q]} = q' - 1$ and $N_1 \leq \sigma'$.

Corollary 1.

$$\mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^{*\otimes}}(u', q', q_{\max}, \ell, \sigma', \sigma_{\max}, T) \leq \sigma' \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(1, T') + q' \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(q', T'). \tag{12}$$

(2) We can also fine tune the above bound by splitting the interval $[2..q']$ into smaller intervals like $[1,2], [3,4], [5..8], \ldots, [2^{p-1}+1..q']$, where $2^{p-1} < q' \leq 2^p$. So, $p < 1 + \log_2 q'$. Now by Lemma 6, we have $N_i \leq q'/2^{i-1}$. So, we can apply the result once again with this choice of intervals to obtain the following corollary.

Corollary 2. *For any* (q', ℓ', σ', t)*-non-adaptive distinguisher* D*, we have* $(2^{i-1}+1)$*-query non-adaptive distinguisher* D_i *for* $1 \leq i \leq p-1$ *such that*

$$\mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^{*\otimes}}(\mathsf{D}) \leq \sigma' \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(\mathsf{D}_1) + \sum_{i=1}^{\lceil \log_2 q' \rceil} \frac{q'}{2^{i-1}} \cdot \mathbf{Adv}^{\mathrm{nprf}}_{f}(\mathsf{D}_i). \tag{13}$$

Suppose f is a keyed function with a higher non-adaptive PRF advantage of $\mathbf{Adv}^{\mathrm{nprf}}_{f}(D, T) \leq DT/2^c$. We still prove a similar advantage (up to a logarithmic factor) against non-adaptive distinguisher:

$$\mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^{*\otimes}}(\theta', T) \leq \frac{(\sigma' T + \sigma'^2) \cdot \log_2 q_{\max}}{2^c}.$$

Applications to HMAC and NMAC. The generic reduction from NMAC to cascade (see Eq. 4) and HMAC to NMAC (see Eq. 5) can be easily extended for the multiuser setup in the following way:

$$\mathbf{Adv}^{\mathrm{mu\_prf}}_{\mathsf{NMAC}^f}(\theta', T) \leq \mathbf{Adv}^{\mathrm{mu\_pf\_nprf}}_{f^{*}}(\theta', T') + \mathbf{Adv}^{\mathrm{prf}}_{f}(q, T') + \frac{q'^2}{2^c}, \tag{14}$$

$$\mathbf{Adv}^{\mathrm{mu\_prf}}_{\mathsf{HMAC}^f}(\theta', T) \leq \mathbf{Adv}^{\mathrm{mu\_prf}}_{\mathsf{NMAC}^f}(\theta', T) + \mathbf{Adv}^{\mathrm{prbg}}_{\mathsf{KDF}}(T'). \tag{15}$$

Hence our results for non-adaptive PRF security of cascade can be directly applied to the multiuser security of HMAC and NMAC.

4 Single Keyed **NMAC** and Constant-free **HMAC**

Now we show that the single keyed $\mathsf{1k\_NMAC}_K := \mathsf{NMAC}_{K,K}$ has almost the same security as NMAC_{K_1,K_2}. For any keyed function g, we write

$$\mathsf{NMAC}^g_*(m) := g\big(f^*(g(m[1]), m[2..])\big).$$

Note that $\mathsf{NMAC}^{f_K}_*$ is the same as the single-keyed NMAC or $\mathsf{NMAC}1k$. We write $\mathsf{NMAC}' = \mathsf{NMAC}^{\Gamma}_*$. For any multiuser distinguisher $\mathsf{D} \in \mathscr{D}_{\mathrm{pf}}(\theta', T)$,

$$\begin{aligned}\Delta^*_{\mathsf{D}}(\mathsf{1k\_NMAC}^{\otimes}\ ;\ \mathsf{RF}) &= \Delta^*_{\mathsf{D}}(\mathsf{NMAC}'^{\otimes}\ ;\ \mathsf{RF}) + \Delta^*_{\mathsf{D}}(\mathsf{1k\_NMAC}^{\otimes}\ ;\ \mathsf{NMAC}'^{\otimes})\\ &= \Delta^*_{\mathsf{D}}(\mathsf{NMAC}'^{\otimes}\ ;\ \mathsf{RF}) + \Delta^*_{\mathsf{D}}(\mathsf{NMAC}^{f_K,\otimes}_*\ ;\ \mathsf{NMAC}^{\Gamma,\otimes}_*)\\ &= \Delta^*_{\mathsf{D}}(\mathsf{NMAC}'^{\otimes}\ ;\ \mathsf{RF}) + \Delta^*_{\mathsf{D}_0}(f^{\otimes}\ ;\ \mathsf{RF}),\end{aligned}$$

where $\mathsf{D}_0 = \mathsf{D} \ \triangleright\ \mathsf{NMAC}^{\mathcal{O},\otimes}_*$. Note that D_0 is a u'-user, $2q_{\max}$-query distinguisher. So we have a single user $2q_{\max}$-query distinguisher D_1 such that $\Delta^*_{\mathsf{D}_0}(f^{\otimes}\ ;\ \mathsf{RF}) \le u' \cdot \Delta^*_{\mathsf{D}_1}(f\ ;\ \mathsf{RF})$. So we focus on bounding the first term of the last expression. The main intuition for the term is the following: For any user-index γ and a message m, $\mathsf{NMAC}'^{\otimes}(\gamma, m)$ behaves like $\mathsf{RF}'(f^*(\mathsf{RF}(\gamma, m[1]), m[2..]))$ provided the lists $m_i[1]$ are different from $f^*(\mathsf{RF}(\gamma, m_i[1]), m_i[2..])$. Given that no such collision occurred between these two lists and no collision occurred in the $f^*(\mathsf{RF}(\gamma, m_i[1]), m_i[2..])$ values, $\mathsf{NMAC}'^{\otimes}$ behaves perfectly random. Now, $f^*(\mathsf{RF}(\gamma, m_i[1]), m_i[2..])$ is equivalent to the multiuser cascade with user index $(\gamma, m_i[1])$. However, our result cannot be applied as it requires a prefix-free distinguisher, which need not be the case. To get rid of the prefix-free queries, we append one block to every user so that queries become prefix-free. This can be done as the outer random function hides the internal value, which helps in reducing to a non-adaptive adversary for the internal cascade function. So, the collision event can be redefined after adjoining a block and our non-adaptive prefix-free PRF result for cascade can now be applied to bound the revised event. We now give a formal proof. We define a distinguisher $\mathsf{D}'^{\mathcal{O}}$ that works as follows:

- Run D. On ith query (γ_i, m'_i) it returns a random c-bit string z'_i. Let $m'_1, \ldots, m'_q$ be all queries. Find $x \in \mathsf{B}$ such that $m^q \in \mathcal{P}$ (prefix-free) where $m_i = (m_i[0] := \gamma_i, m'_i, x)$ for all $i \in [q]$.
- Query $\mathcal{O}^{\otimes}((\gamma, m_i[1]),\ m_i[2..])$ (with $(\gamma_i, m_i[1])$ as a user-index) and let z_i be the response, for $i \in [q]$.
- Return 1 if either of the following holds (which we call a bad event):
  1. if $z_i = z_j$ for some $i \neq j$,

   2. $z_i = f(m_j[1], x)$ for some i, j.

– Else return 0.

Lemma 7. $\Delta_{\mathsf{D}}(\mathit{NMAC}'^{\otimes}\ ;\ \mathsf{RF}) \leq \Pr(\mathsf{D}'^{f^{*\otimes}} = 1)$, *where* D' *is defined as above.*

Proof. Let $\mathsf{bad}(m^q[1], z^q) = 1$ if either $z_i = z_j$ for some $i \neq j$, or $z_i = f(m_j[1], x)$ for some i, j. Here, $m^q[1]$ represents $(m_1[1], \ldots, m_q[1])$. Thus, D' returns 1 if and only if bad is true. Let $y_i = f^{*\otimes}(\mathsf{RF}(\gamma_i, m_i[1]), m_i[2..\ell_i])$. So, $z_i = f(y_i, x)$. We now define another bad event bad' which holds if either $y_i = y_j$ for some $i \neq j$ or $y_i = m_j[1]$ for some i, j. Clearly, whenever bad' holds, bad holds and so $\Pr(\mathsf{bad}') \leq \Pr(\mathsf{D}'^{f^{*\otimes}} = 1)$. Now, if bad' does not hold, then all outputs $\mathsf{NMAC}_*^{\Gamma}(m_i)$ are random (as the input of the final Γ are fresh for all queries). So, we claim that $\Delta_{\mathsf{D}}(\mathsf{NMAC}'\ ;\ \mathsf{RF}) \leq \Pr(\mathsf{bad}')$. So the result follows. □

Clearly $\Pr(\mathsf{D}'^{\mathsf{RF}} = 1) \leq 1.5q^2/2^c$ as z_i's are uniform and independent. So, $\Pr(\mathsf{D}'^{f^{*\otimes}} = 1) \leq \Delta_{\mathsf{D}'}(f^{*\otimes}, \mathsf{RF}) + 1.5q^2/2^c$. Moreover, D' is non-adaptive and can make at most q' user-index queries (note that the first block of message is now a part of the user-index). This completes the proof of the PRF security of the single-keyed NMAC.

Theorem 3.

$$\mathbf{Adv}^{\text{mu\_prf}}_{\mathit{1k\_NMAC}^f}(\theta') \leq \mathbf{Adv}^{\text{mu\_pf\_nprf}}_{f^*}(q', q', q_{\max}, \ell+1, \sigma', \sigma_{\max}, T') + \\ + u'\mathbf{Adv}^{\text{prf}}_{f}(2q_{\max}, T') + 2q'^2/2^c.$$

The previous result can be plugged into the above expression to get the security of the constant-free variant of HMAC, denoted as HMAC', where

$$\mathsf{HMAC}'(K, m) := \mathsf{1k\_NMAC}(\mathsf{KDF}(K), m)$$

and $\mathsf{KDF}(K) = f(IV, K\|0^*)$. If $f(IV, \cdot, 0^*)$ is (almost) regular then $\mathsf{KDF}(K)$ is uniformly distributed and hence the security of the variant of HMAC' is reduced to $\mathsf{1k\_NMAC}$. So the bound for the single-keyed NMAC can be directly applied to the constant-free variant of HMAC.

5 Security Analysis of Enveloped MAC

We can similarly prove an improved analysis for Enveloped MAC (we get a better tightness reduction as well as eliminate the related key advantage in the existing analysis). For any keyed function g, we write

$$\mathsf{EvMAC}_*^{g}(m) := g\big(f^*(g(IV), m[1..])\big).$$

Note that $\mathsf{EvMAC}_*^{f_K^\downarrow}$ is the same as EvMAC. We write $\mathsf{EvMAC}' = \mathsf{EvMAC}_*^{\Gamma}$. By using a reduction similar to $\mathsf{1k\_NMAC}$, we have

$$\Delta^*_{\mathsf{D}}(\mathsf{EvMAC}^{\otimes}\ ;\ \mathsf{RF}) = \Delta^*_{\mathsf{D}}(\mathsf{EvMAC}'^{\otimes}\ ;\ \mathsf{RF}) + \Delta^*_{\mathsf{D}_0}(f^{\downarrow\otimes}\ ;\ \Gamma),$$

where $\mathsf{D}_0^{\mathcal{O}^{\otimes}} = \mathsf{D} \triangleright \mathsf{EvMAC}_*^{\mathcal{O}^{\otimes}}$. So we focus on bounding the multiuser PRF advantage of $\Gamma\big(f^*(\Gamma(IV), m[1..])\big)$. Now we see that if we fix $m_i[1] = IV$ in the proof of NMAC we actually reduce to EvMAC'. In other words, we need to consider the following bad event (for some x to be adjoined at the end as before to make prefix-free queries): Let $\mathsf{bad}(m^q[1], z^q) = 1$ if either $z_i = z_j$ for some $i \neq j$, or $f(z_i, x) = f(IV, x)$ for some i, j. Following a similar argument as $\mathsf{1k\_NMAC}$, we have our result.

Theorem 4.

$$\mathbf{Adv}^{\text{mu\_prf}}_{\mathit{EvMAC}}(\theta') \leq \mathbf{Adv}^{\text{mu\_pf\_nprf}}_{f^*}(q', q', q_{\max}, \ell, \sigma', \sigma_{\max}, T') + \\ + u' \cdot \mathbf{Adv}^{\text{prf}}_{f}(q_{\max}+1, T') + \frac{q'^2}{2^c}.$$

6 Final Remarks

6.1 Related Results

Verifying the integrity and authenticity of data is a prime necessity in computer systems and networks. Two parties communicating over an insecure channel use a message authentication code or MAC (or a stronger notion called a pseudorandom function or PRF) with which, the receiver validates data as being sent by the sender. MACs and PRFs are commonly constructed out of block ciphers (e.g. CBC-MAC [6,5], PMAC [8], the NIST-recommended CMAC etc. [16,11]). Popular hash functions were earlier faster than block ciphers in software. Moreover, since hash functions are not usually subject to the export restriction rules of the USA and other countries, there has been a surge of interest in constructing MACs from cryptographic hash functions. However, hash functions were not originally designed for MACs or PRFs and do not accommodate a secret key in a natural way. One of the earliest ideas for converting a hash function (mainly the Merkle-Damgård hash [23,10]) into a MAC was simply to prepend the message with the secret key. However, it was soon found that these hash functions suffer from a serious security flaw if the key is prepended in this manner – the *length extension attack* (see Example 9.64 of [22]). In CRYPTO 1996, authors of [4] proposed NMAC and HMAC and proved them secure if the underlying compression function f satisfies certain security requirements.

The Initial Results on EvMAC. Tsudik in [27] proposed envelope MAC which originally requires two independent c-bit keys. Although the author of [27] informally claimed the $2c$-bit security of the MAC, Preneel and van Oorschot [25] showed that the keys can be recovered in 2^{c+1} time if one knows approximately $2^{c/2}$ message-tag pairs. The subsequent single-keyed variants due to Kaliski and Robshaw [17] and Piermont and Simpson [24] also had security flaws. The attacks were possible only because of poor formatting of the key block processed at the end.

Multiuser analysis of AMAC. Recently, in Eurocrypt-16, AMAC (or Augmented 2-tier cascade MAC) [3] has been analyzed. The authors proved multiuser security to a leakage PRF security of f where a part of the key is leaked. The stronger security requirement is needed as AMAC does not use nested construction and applies to all messages where one message can be prefix to others. Once again authors adapted two stage reduction like cascade construction. Unfortunately our improved single stage reduction of adaptive distinguisher does not work for AMAC. The main reason of not working argument is the possibility of prefix queries. We do not see any simple way out to handle the prefix queries. However, for non-adaptive adversary a similar (in fact the improved trade-off between tightness gap and query complexity) works.

6.2 Some Remarks on Our Results

Single Stage Reduction. To quote the authors of [4]:

> "*A natural approach to such a proof is to reduce the security of f^* to that of f. However, we could not find a straightforward reduction.*"

Their proof (and all subsequent analyses of single-keyed cascade-type constructions) is divided into two parts. In the first step, the prefix-free PRF security of f^* is reduced to a multiuser PRF security of f which, in the second step, is then reduced to the single-user PRF (or simply the PRF security) of f. In this paper, all proofs are based on direct reductions.

Best Known Bound in Terms of Data and Time Complexity: Let $\epsilon(D,T) := \mathbf{Adv}_f^{\mathrm{prf}}(D,T)$. The dominating terms for the best known bound for prefix-free PRF advantage against cascade, NMAC and HMAC before this paper was $\ell q \cdot \epsilon(q, T+\sigma)$. Let D denote the user limit of an application before it re-keys. Some applications can accept a wide range of message sizes (from a single block to very large messages), and hence we must assume $\ell = q = \sigma = O(D)$ (similar guidelines are provided by NIST for lightweight cipher standardization [28]). So the best known bound in this set-up becomes $D^2 \cdot \epsilon(D, T + O(D))$. We already know that $\epsilon(D,T) \geq \max\{T/2^c, 1/2^{c/2}\}$. Both lower bounds weaken the PRF security guarantee of the cascade (and hence of HMAC and NMAC). Our result resolves these issues.

6.3 Open Problems

The following important open problems can be studies in future.

1. Similar to the reduction for non-adaptive distinguishers, can we have an improved trade-off for adaptive reductions? This problem seems to be challenging as the simulator needs to guess a bound on the number of children of a node adaptively.
2. Having an improved reduction for adaptive PRF distinguisher of AMAC is not yet solved. A different approach to handle prefix queries is needed.
3. All known bounds for cascade construction can be at the best $DT/2^c$. However, there is no such generic matching attack (matching attacks are applicable for some pathological examples). Understanding the right PRF security bound when f behaves like a random function is not yet known.

Acknowledgment. We would like to thank Alfred Menezes and Suprita Talnikar for their valuable comments.

References

1. Mihir Bellare. New proofs for NMAC and HMAC: security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer, 2006. Updated version available at `http://cseweb.ucsd.edu/~mihir/papers/hmac-new.pdf`.
2. Mihir Bellare. New proofs for nmac and hmac: security without collision resistance. *Journal of Cryptology*, 28(4):844–878, 2015.
3. Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based prfs: AMAC and its multi-user security. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 566–595. Springer, 2016.

4. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996. Extended version available at `http://cseweb.ucsd.edu/~mihir/papers/kmd5.pdf`.
5. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.
6. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
7. Daniel J Bernstein and Tanja Lange. Non-uniform cracks in the concrete: the power of free precomputation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340. Springer, 2013.
8. J. Black and P. Rogaway. A block-cipher mode of operations for parallelizable message authentication. In *Advances in Cryptology – Eurocrypt 2002*, number 2332 in Lecture Notes in Computer Science, pages 384–397, Berlin, 2002. Springer.
9. Bill Burr. Nist hash function standards status and plans. *National Institute of Standards and Technology, Gaithersburg, Maryland (csrc. nist. gov/groups/SMA/ispab/documents/minutes/2005-12/B_Burr-Dec2005-ISPAB. pdf)*, 2005.
10. Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.
11. Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. `http://csrc.nist.gov/publications/nistpubs/index.html$#$sp800-38B`.
12. PUB FIPS. 198-1. *The keyed-hash message authentication code (HMAC)*, 2008.
13. Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact prf-security of nmac and hmac. In *Annual Cryptology Conference*, pages 113–130. Springer, 2014.
14. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.
15. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.
16. T. Iwata and K. Kurosawa. Omac: One-key cbc mac. In *Fast Software Encryption, 10th International Workshop – FSE 2003*, number 2887 in Lecture Notes in Computer Science, pages 129–153, Berlin, 2003. Springer.
17. B. Kaliski and M. Robshaw. Message authentication with md. *CryptoBytes, 5*, 1(1):5–8, 1995.
18. Neal Koblitz and Alfred Menezes. Another look at HMAC. *J. Math. Cryptol.*, 7(3):225–251, 2013.
19. Neal Koblitz and Alfred Menezes. Another look at non-uniformity. *Groups Complex. Cryptol.*, 5(2):117–139, 2013.
20. Neal Koblitz and Alfred Menezes. Another look at security theorems for 1-key nested macs. In *Open Problems in Mathematics and Computational Science*, pages 69–89. Springer, 2014.
21. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: keyed-hashing for message authentication. *RFC*, 2104:1–11, 1997.

22. Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
23. Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
24. P. Piermont and W. Simpson. Ip authentication using keyed md5. *IETF RFC 1828*, August 1995.
25. Bart Preneel and Paul C Van Oorschot. On the security of iterated message authentication codes. *IEEE Transactions on Information theory*, 45(1):188–199, 1999.
26. Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progressin Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.
27. Gene Tsudik. Message authentication with one-way hash functions. *ACM SIGCOMM Computer Communication Review*, 22(5):29–38, 1992.
28. Meltem Sönmez Turan, Kerry A McKay, Çağdaş Çalık, Donghoon Chang, and Larry Bassham. Status report on the first round of the nist lightweight cryptography standardization process. *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*, 2019.
29. Kan Yasuda. "sandwich" is indeed secure: How to authenticate a message with just one hashing. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2007.