A Double-Piped Mode of Operation for MACs, PRFs and PROs: Security beyond the Birthday Barrier

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
yasuda.kan@lab.ntt.co.jp

Abstract. We revisit the double-pipe construction introduced by Lucks at Asiacrypt 2005. Lucks originally studied the construction for iterated hash functions and showed that the approach is effective in improving security against various types of collision and (second-)preimage attacks. Instead, in this paper we apply the construction to the secret-key setting, where the underlying FIL (fixed-input-length) compression function is equipped with a dedicated key input. We make some adjustments to Lucks' original design so that now the new mode works with a single key and operates as a multi-property-preserving domain extension of MACs (message authentication codes), PRFs (pseudo-random functions) and PROs (pseudo-random oracles). Though more than twice as slow as the Merkle-Damgård construction, the double-piped mode enjoys security strengthened beyond the birthday bound, most notably, high MAC security. More specifically, when iterating an FIL-MAC whose output size is n-bit, the new double-piped mode yields an AIL-(arbitrary-input-length-)MAC with security up to $O\big(2^{5n/6}\big)$ query complexity. This bound contrasts sharply with the birthday bound of $O\big(2^{n/2}\big)$, which has been the best MAC security accomplished by earlier constructions.

Keywords: domain extension, unpredictability, unforgeability, message authentication code, MAC, birthday bound.

1 Introduction

Many of the symmetric-key cryptographic schemes are usually realized by iterating a smaller, fixed-input-length (FIL) primitive. Examples include hash functions and message authentication codes (MACs), which are often built of a compression function or of a block cipher. The underlying compression function or block cipher has also a fixed output length, say n-bit. The size n corresponds to a security parameter in more ways than one, because it defines not only the final output size of the scheme but also the size of intermediate values in the iteration. For example, in the case of the Merkle-Damgård (MD) construction [18, 9] or Cipher-Block-Chaining (CBC) mode of operation [22, 15], the size of the intermediate values is exactly equal to n.

The "small" size n of intermediate values leads to various types of attacks and to limited security of the overlying scheme. For instance, the Merkle-Damgård

hash functions are known to be vulnerable to multi-collision attacks [14] and to long-message second-preimage attacks [16]. These attacks exploit the fact that internal collisions can be found with $O(2^{n/2})$ work. For iterated MACs, the internal collisions immediately yield forgery [25, 26], which confines the security of the overlying MAC scheme to $O(2^{n/2})$ query complexity. This security bound is often called the birthday "barrier."

Lucks' Double-Pipe Construction. A natural approach to precluding the above-mentioned attacks is to increase the size of intermediate values and to make it larger than the final output size n. This idea was embodied by the double-pipe hash [17] proposed by Lucks. In the double-pipe hash, the size of intermediate values was doubled to $2n$ bits by invoking two applications of the compression function per message block. The double-pipe design is more than twice as slow as the MD construction, but Lucks showed that the double-pipe hashing successfully rules out various birthday-type attacks.

In this paper we apply the double-pipe construction to the secret-key setting and analyze the security of the scheme as a MAC. We also consider stronger security notions of pseudo-random functions (PRFs) and pseudo-random oracles (PROs).[1] Of particular interest is the case of being merely a secure MAC, because there has been no known construction of MAC-Pr (preserving) domain extension with security beyond the birthday barrier. So we raise the following question:

> ***Q. Can we prove that a double-piped mode provides an AIL-(arbitrary-input-length-)MAC secure beyond the birthday barrier based on the sole assumption that the underlying compression function is a secure FIL-MAC?***

Our Results. The answer to the above question turns out to be positive. We work in the dedicated-key setting [4], where the underlying compression function is equipped with a dedicated key input like a block cipher. We then make two slight modifications to Lucks' original design and show that the new double-piped mode operates as a MAC-Pr, PRF-Pr and PRO-Pr domain extension with each type of security ensured beyond the birthday bound.

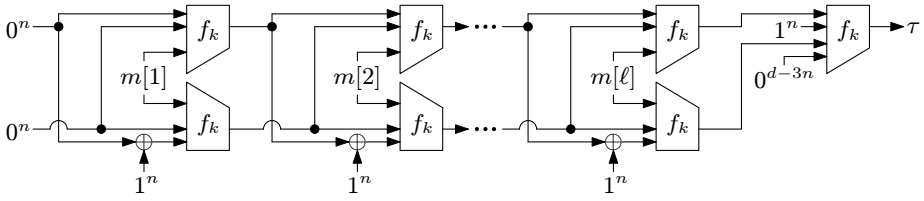

Fig. 1. Our double-piped mode of operation

Figure 1 describes our altered double-piped mode of operation iterating a compression function $f_k : \{0,1\}^d \to \{0,1\}^n$ ($d \geq 3n$) with a secret key k (A

[1] We use the term "PRO" [3] interchangeably with "indifferentiability" [19, 5].

formal definition of the scheme will be given in Sect. 4). The mode takes as its input a message $M = m[1] \parallel m[2] \parallel \cdots \parallel m[\ell]$ (being properly padded) with each block of $d - 2n$ bits and outputs the final value $\tau \in \{0,1\}^n$.

The mode basically follows Luck's original design, except that we make the following two minor changes:

- In the original, the chaining variable to the second application of the compression function was "tweaked" by swapping two hash values. Instead, we tweak the chaining variable by XOR-ing (rather than swapping) one of the two output values with a constant 1^n.
- In the original, the final iteration handled the last message block and worked just like intermediate iterations, except to omit the second application of the compression function. Instead, we arrange a special finalization step, which handles no message block and takes the last chaining variable "shifted" by 1^n.

The former has been already used by [6] in a more generalized form. The change is technical, and we adopt it only for simplicity of the proof. The latter technique was employed in the CS construction [20]. Unlike the first one, this change is essential to the security of our scheme. We remark that in principle the two changes do not affect Luck's original analysis, and the scheme (without the secret key) remains secure as a hash function.

The new double-piped mode of operation is highly secure; we obtain the following security results:

- **MAC-Pr.** This is the main result. We show that the scheme yields a MAC-Pr domain extension providing security beyond the birthday barrier. Our security proof involves new techniques of transforming collisions into a forgery. Using these techniques we are able to prove MAC security up to $O\big(2^{5n/6}\big)$ query complexity, improving on the previous best security of the birthday bound $O\big(2^{n/2}\big)$. We also provide a discussion on the gap between our bound $O\big(2^{5n/6}\big)$ and the full security $O\big(2^n\big)$.
- **PRF-Pr.** Our result for PRF is a straightforward corollary of that for PRO. This is due to the fact that we are working in the dedicated-key setting with the key being secret. In such a scenario, the notion of indistinguishability (*i.e.*, PRF) becomes strictly weaker than that of indifferentiability (*i.e.*, PRO). The new mode gives the full $O(2^n)$ security of PRF.
- **PRO-Pr.** We prove that the double-piped mode is indifferentiable beyond the birthday bound; the mode is secure up to $O(2^n)$ query complexity. This bound has been already realized by earlier constructions, but our scheme has slight performance advantages over them (see Sect. 2).

Organization. Section 2 reviews previous constructions of domain extensions for MACs, PRFs and PROs. Section 3 provides basic notation, adversary models, and security notions used in the paper. In Sect. 4 we give a formal definition of our double-piped mode of operation. Section 5 is devoted to the security proofs of our MAC-Pr result. In Sect. 6 and 7 we present the security results for PRF-Pr and PRO-Pr, respectively.

Table 1. Comparing the MAC/PRF/PRO security of various domain extensions

	MAC	PRF	PRO	
Feistel network	$2^{n/6}$	2^n	$2^{n/16}$	[10, 23, 8]
Enciphered CBC	$2^{n/4}$	$2^{n/2}$	$2^{n/4}$	[11]
NI, CS, ESh, MDP	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	[1, 20, 5, 4, 12]
Proposed double-pipe	$\mathbf{2^{5n/6}}$	$\mathbf{2^n}$	$\mathbf{2^n}$	—
Benes, multilane-NMAC, one-pass	—	2^n	—	[24, 27, 28]
Prefix-free, chopMD, Maurer-Tessaro	—	2^n	2^n	[6, 7, 21]

2 Related Work

In this section we go through prior constructions of MAC-Pr, PRF-Pr and PRO-Pr domain extensions. For MACs, previous constructions provide security only up to the birthday bound. For PRFs and PROs, we mention only those constructions which have security above the birthday bound. See Table 1 for summary.[2]

MAC-Pr Constructions. The study of MAC-Pr domain extension was initiated by An and Bellare [1] who presented the NI construction. The NI construction was built on the MD iteration in the dedicated-key setting. The subsequent work, including CS [20], ESh [4] and MDP [12], is all based on the MD construction. All of these constructions have the birthday-bound security $O\big(2^{n/2}\big)$. Dodis and Puniya [10] showed that the Feistel network, when iterated sufficiently many times, yields a secure, MAC domain extension with a relatively low bound of $O\big(2^{n/6}\big)$ query complexity. A year later Dodis *et al.* [11] proposed the enciphered CBC mode, which iterates block ciphers (or "length-preserving MACs") and provides an AIL-MAC with the better security of $O\big(2^{n/4}\big)$.

PRF-Pr Constructions. The problem of PRF domain extension has been extensively studied. Patarin analyzed the Feistel network [23] and the Benes network [24], and these constructions were shown to give $O(2^n)$ security. For PRF, there exist constructions which are more efficient than the double-pipe design, such as multilane-NMAC [27] and the one-pass construction in [28].

PRO-Pr Constructions. Chang *et al.* [6] proved that a double-pipe MD construction with prefix-free padding is indifferentiable beyond the birthday bound. Chang and Nandi [7] also proved that the chopMD construction based on a $2n$-bit compression function is indifferentiable beyond the birthday bound. These constructions are similar to our double-piped mode of operation, but our construction has slight performance gains over them as our scheme requires neither a prefix-free encoding nor a $2n$-bit compression function. Maurer and Tessaro [21] treated the problem in a different setting, where the *input* size (rather than output) of the primitive was restricted to n bits.

[2] Here we focus on *deterministic* constructions without use of nonce or randomization. We also focus on *property-preserving* domain extensions. Hence other constructions—for example RMAC [13]—are excluded from our comparison.

3 Preliminaries

General Notation. We write $x \leftarrow y$ for the assignment of the value y to a variable x. Given a set X, we write $x \xleftarrow{\$} X$ for the operation of selecting an element uniformly at random from the set X and assigning its value to a variable x. The symbol $x \parallel y$ represents the concatenation of two strings x and y, $x \oplus y$ the exclusive OR of x and y, and $|x|$ the length of x in bits. For a finite string $M \in \{0,1\}^*$ and a block size b, the length of M in blocks is the quantity $\ell = \lceil \frac{|M|+1}{b} \rceil$. We adopt the notation $M \xleftarrow{b} M\|10^*$ to signify the "canonical" padding procedure, *i.e.*, $M \leftarrow M\|10^{b\ell-|M|-1}$, where ℓ is the length in blocks of the original M (before being padded). Given a padded message M, we use the notation $m[1] \parallel \cdots \parallel m[\ell] \xleftarrow{b} M$ as a shorthand for partitioning the string M into b-bit blocks and assigning each block value to $m[1], \ldots, m[\ell]$, so that each $m[i]$ is of b-bit length. Also, we write $x \parallel y \xleftarrow{a,b} z$ for dividing the string $z \in \{0,1\}^{a+b}$ into $x \in \{0,1\}^a$ and $y \in \{0,1\}^b$. Throughout the paper we fix the output size n and define $\bar{x} \stackrel{\text{def}}{=} x \oplus 1^n 0^{|x|-n}$ for a string x with $|x| \geq n$.

Adversaries and Games. An adversary is a probabilistic algorithm. An adversary A may take inputs or have access to one or more oracles. We write $y \leftarrow A^{\mathcal{O}}(x)$ to mean that the adversary A, given input x and access to oracle $\mathcal{O}$, outputs a value which is assigned to the variable y. The notation $A^{\mathcal{O}}(x) = y$ indicates the event that the output value of the adversary A, taking an input value x and interacting with oracle $\mathcal{O}$, happens to be equal to the value y.

Often it is convenient to describe oracle behavior in a game style. The notation $G(A)$ indicates running A according to the description of G. By $y \leftarrow G(A)$ we mean the operation of running A in game G, and the value returned by game G is assigned to the variable y. Likewise we define $G(A) = y$. Games frequently involve sets and flags. A set Set is used to record certain specific types of values that appear in the game. We write $\mathsf{Set} \xleftarrow{\cup} x$ for the operation $\mathsf{Set} \leftarrow \mathsf{Set} \cup \{x\}$. A flag flag is used to detect some event that happens in the game. By abuse of notation we let flag also denote the event that the flag flag gets set. We write $\overline{\mathsf{flag}}$ for the complement of the event flag. Unless otherwise stated, sets are set to $\emptyset$ and flags are set to 0 at the beginning of each game execution.

MACs. We adopt the standard single-verification model for the notion of MAC security.[3] Succinctly, for a keyed family of functions $f_k : X \to Y$ with $k \in K$ and for an adversary A define

$$\mathrm{Adv}_f^{\mathrm{mac}}(A) \stackrel{\text{def}}{=} \Pr\left[x^\star \text{ is new} \wedge f_k(x^\star) = y^\star \;\middle|\; (x^\star, y^\star) \leftarrow A^{f_k(\cdot)}, k \xleftarrow{\$} K\right]$$

as the advantage function,[4] where we call a message $x^\star$ *new* if it has not been queried to oracle $f_k(\cdot)$.

[3] It suffices to consider only single-verification adversaries, because our MACs are deterministic [2].

[4] Throughout the paper the probabilities are defined over all coins of adversaries, oracles and games.

PRFs. The notion of PRF says that a keyed family of functions $f_k : X \to Y$ with a random key $k \xleftarrow{\$} K$ should be indistinguishable from a truly random function $g : X \to Y$. Succinctly, define

$$\mathrm{Adv}^{\mathrm{prf}}_f(A) \stackrel{\mathrm{def}}{=} \Pr\big[A^{f_k(\cdot)} = 1 \;\big|\; k \xleftarrow{\$} K\big] - \Pr\big[A^{g(\cdot)} = 1 \;\big|\; g \xleftarrow{\$} \{g : X \to Y\}\big].$$

The notion of PRF implies a secure MAC [2].

PROs. Roughly speaking, the notion of indifferentiability [19, 5] corresponds to a type of indistinguishability where adversaries are given oracle access not only to the overlying scheme but also to the underlying primitive. Let $F : \{0,1\}^* \to \{0,1\}^n$ be a mode of operation which iterates a random function $f : \{0,1\}^d \to \{0,1\}^n$. Let $\mathcal{S} : \{0,1\}^d \to \{0,1\}^n$ be a (stateful) simulator having access to a random function $\mathcal{F} : \{0,1\}^* \to \{0,1\}^n$. We define

$$\mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A) \stackrel{\mathrm{def}}{=} \Pr\big[A^{F,f} = 1\big] - \Pr\big[A^{\mathcal{F},\mathcal{S}} = 1\big].$$

Here it is important to note that the simulator cannot "observe" the queries that A makes to $\mathcal{F}$.

Resources. We bound resources of adversaries in terms of its time complexity t, the number of queries q and the total length of queries σ in blocks.[5] We write, for example, $\mathrm{Adv}^{\mathrm{mac}}_f(t,q,\sigma) \stackrel{\mathrm{def}}{=} \max_A \mathrm{Adv}^{\mathrm{mac}}_f(A)$, where the max runs over all adversaries, each having a time complexity at most t, making at most q queries, the total length of queries being at most σ blocks. To measure the time complexity of adversaries, we fix a model of computation. The time complexity of an adversary includes its code size and the total time needed to perform its overlying experiment, in particular the time to access its oracles. We write Time_f for the time to perform one computation of f (for fixed-length inputs) and $\mathrm{Mem}_f(\sigma)$ the memory to store σ-many input/output pairs of f. Lastly we note that for the notion of PROs, the resources of the simulator must be also taken into consideration.

4 Definition of the Double-Piped Mode

Now we give a formal definition of our double-piped mode of operation $F_k : \{0,1\}^* \to \{0,1\}^n$ in Fig. 2. Our mode F takes a secret key $k \in \{0,1\}^\kappa$ and a message $M \in \{0,1\}^*$, outputting a tag $\tau \in \{0,1\}^n$. It iterates a single compression function $f_k : \{0,1\}^d \to \{0,1\}^n$ operating with a single key $k \in \{0,1\}^\kappa$. We require that $d \geq 3n$. We refer back to Fig. 1 for pictorial representation.

5 MAC-Pr beyond the Birthday Barrier

In this section we prove that the double-piped mode F is an AIL-MAC with security well above the birthday bound, based on the sole assumption that the compression function f is a secure FIL-MAC. First we roughly outline our proof and then proceed to its full description.

[5] The block length depends on each scheme, and in our construction it is equal to $d - 2n$ bits.

Algorithm $F_k(M)$

101 $M \xleftarrow{d-2n} M\|10^*$; $m[1] \,\|\, m[2] \,\|\, \cdots \,\|\, m[\ell] \xleftarrow{d-2n} M$
102 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
103 **for** $i = 1$ **to** ℓ
104 $\quad x_1[i] \leftarrow v_1[i] \,\|\, v_2[i] \,\|\, m[i]$; $x_2[i] \leftarrow \overline{v_1[i]} \,\|\, v_2[i] \,\|\, m[i]$
105 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$; $v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
106 **end for**
107 $\tau \leftarrow f_k\big(v_1[\ell+1] \,\|\, 1^n \,\|\, v_2[\ell+1] \,\|\, 0^{d-3n}\big)$
108 **ret** τ

Fig. 2. Definition of our double-piped mode of operation $F_k : \{0,1\}^* \to \{0,1\}^n$

5.1 Outline of the MAC-Pr Proof

We convert a forger attacking F into ones attacking f. We start by observing that the success of forging a tag value for F implies at least one of the following three events:

- **Event** forge**:** A forgery of the tag value for f occurs at the finalization step,
- **Event** ones**:** An output value of f happens to be equal to 1^n at some internal invocation to f, or
- **Event** match**:** The $2n$-bit chaining values at the input to the finalization step happen to be the same for two different queries.

The first two events immediately give us forgers attacking f without birthday degradation, but the third one does not. So we break the third event down further, showing that the match event points to at least one of the following three events:

- **Event** zeros**:** At some iteration point, the $2n$-bit chaining value happens to be equal to 0^{2n},
- **Event** twofold**:** At some iteration point, the output of the "upper" f happens to be equal to that of the "lower" f, or
- **Event** dblcoll**:** A collision of $2n$-bit chaining values occurs somewhere, yielding two collisions of such a type for f.

The zeros and twofold events instantly provide forgers attacking f without birthday degradation. To treat the dblcoll event, however, we need to partition the case depending on "how many" (single, n-bit) collisions for f are formed in the game. We introduce a threshold value θ and denote by theta the event that "θ-many" (single, n-bit) collisions for f are found. We divide the case as follows:

- **Case** theta**:** In this case we construct a forger attacking f by utilizing the accumulation of collisions. Usually the transformation of a collision into a forgery would lead to birthday degradation [1], but we succeed in maintaining a forgery probability above the birthday bound owing to the large (θ-many) number of collisions.

- **Case** $\mathsf{dblcoll} \wedge \overline{\mathsf{theta}}$**:** On the other hand, if there are not "so many" collisions, then we can create a forger attacking f with a good success probability by utilizing the double, $2n$-bit collision(s). Namely, we first detect that a collision occurs at the half n-bit value, say $v \in \{0,1\}^n$, of the chaining variable (and there are not so many such collisions). We then try to forge a tag, hoping that the remaining half n-bit value will also collide. The (predicted) tag value is chosen from previous chaining variables with its half value colliding to v (and there are not so many candidates for the tag value).

Finally, we choose the threshold value θ appropriately so that the security bound becomes optimal in our setting. This gives us the desired security of $O\big(2^{5n/6}\big)$ query complexity.

5.2 Detailed Proof of MAC-Pr

We now state our main theorem:

Theorem 1. *Let F be the double-piped mode. If the underlying compression function f is a secure FIL-MAC, then the mode F yields a secure AIL-MAC with security above the birthday bound. Specifically, we have*

$$\mathrm{Adv}^{\mathrm{mac}}_{F}(t,q,\sigma) \le 9 \cdot \sigma^{6/5} \cdot \mathrm{Adv}^{\mathrm{mac}}_{f}(t', q+2\sigma),$$

where $t' = t + (q+2\sigma)\cdot \mathrm{Time}_f + \mathrm{Mem}_f(2\sigma)$.

Proof. Let A be a forger attacking F, having a time complexity at most t, making at most q queries to $F_k(\cdot)$ oracle, the query complexity being at most σ blocks. We consider game G_1 as defined in Fig. 3. Game G_1 precisely corresponds to the game defining $\mathrm{Adv}^{\mathrm{mac}}_{F}(A)$, except that it introduces three flags forge, ones and match.

We claim that a successful forgery by A implies at least one of the events forge, ones or match. A forgery by A immediately implies the event forge as long as the value u at line 443 is new. If the value u is not new, then it must have been inserted into the set $\mathsf{Dom}(f)$ at lines 331, 345 or 431. An insertion at lines 331 or 431 implies the event ones, while that at line 345 implies match. Thus we have

$$\begin{aligned}\mathrm{Adv}^{\mathrm{mac}}_{F}(A) \stackrel{\text{def}}{=} \Pr\big[G_1(A)=1\big] &\le \Pr\big[\mathsf{forge} \vee \mathsf{ones} \vee \mathsf{match}\big] \\ &\le \Pr\big[\mathsf{forge}\big] + \Pr\big[\mathsf{ones}\big] + \Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big].\end{aligned}$$

We start with bounding $\Pr\big[\mathsf{forge}\big]$. We construct a forger B_1 attacking f as follows: B_1 runs the adversary A, simulating $F_k(\cdot)$ oracle and computing the subroutine $V_k(\cdot,\cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_1 stops at line 444 before making the query u to its $f_k(\cdot)$ oracle and submits a forgery $(u, \tau^\star)$. Note that B_1 always succeeds under the event forge, making at most $q+2\sigma$ queries to its oracle. So we get

$$\Pr\big[\mathsf{forge}\big] \le \mathrm{Adv}^{\mathrm{mac}}_{f}(B_1) \le \mathrm{Adv}^{\mathrm{mac}}_{f}(t_1, q+2\sigma),$$

Game $G_1(A)$:

201 $k \xleftarrow{\$} \{0,1\}^\kappa$
202 $(M^\star, \tau^\star) \leftarrow A^{F_k(\cdot)}$
203 **if** $M^\star \in \mathsf{Dom}(F)$ **then ret** 0 **end if**
204 **ret** $V_k(M^\star, \tau^\star)$

On query M **to** $F_k(\cdot)$:

300 $\mathsf{Dom}(F) \xleftarrow{\cup} M$; $M \xleftarrow{d-2n} M\|10^*$
301 $m[1] \parallel m[2] \parallel \cdots \parallel m[\ell] \xleftarrow{d-2n} M$
302 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
303 **for** $i = 1$ **to** ℓ
304 $\quad x_1[i] \leftarrow v_1[i] \parallel v_2[i] \parallel m[i]$
305 $\quad x_2[i] \leftarrow \overline{v_1[i]} \parallel v_2[i] \parallel m[i]$
306 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$
▶
311 $\quad v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
▶
▷
331 $\quad \mathsf{Dom}(f) \xleftarrow{\cup} x_1[i], x_2[i]$
332 $\quad$ **if** $v_1[i+1] = 1^n$ **or** $v_2[i+1] = 1^n$
333 $\quad\quad$ **then** $\mathsf{ones} \leftarrow 1$ **end if**
334 **end for**
341 $\mathsf{Dbl} \xleftarrow{\cup} v_1[\ell+1] \parallel v_2[\ell+1]$
343 $u \leftarrow v_1[\ell+1] \parallel 1^n \parallel v_2[\ell+1] \parallel 0^{d-3n}$
345 $\tau \leftarrow f_k(u)$; $\mathsf{Dom}(f) \xleftarrow{\cup} u$
347 **ret** τ

Subroutine $V_k(M, \tau)$:

400 $M \xleftarrow{d-2n} M\|10^*$
401 $m[1] \parallel m[2] \parallel \cdots \parallel m[\ell] \xleftarrow{d-2n} M$
402 $v_1[1] \leftarrow 0^n$; $v_2[1] \leftarrow 0^n$
403 **for** $i = 1$ **to** ℓ
404 $\quad x_1[i] \leftarrow v_1[i] \parallel v_2[i] \parallel m[i]$
405 $\quad x_2[i] \leftarrow \overline{v_1[i]} \parallel v_2[i] \parallel m[i]$
406 $\quad v_1[i+1] \leftarrow f_k\big(x_1[i]\big)$
▶
411 $\quad v_2[i+1] \leftarrow f_k\big(x_2[i]\big)$
▶
▷
431 $\quad \mathsf{Dom}(f) \xleftarrow{\cup} x_1[i], x_2[i]$
432 $\quad$ **if** $v_1[i+1] = 1^n$ **or** $v_2[i+1] = 1^n$
433 $\quad\quad$ **then** $\mathsf{ones} \leftarrow 1$ **end if**
434 **end for**
441 **if** $v_1[\ell+1] \parallel v_2[\ell+1] \in \mathsf{Dbl}$
442 $\quad$ **then** $\mathsf{match} \leftarrow 1$ **end if**
443 $u \leftarrow v_1[\ell+1] \parallel 1^n \parallel v_2[\ell+1] \parallel 0^{d-3n}$
444 $\tau' \leftarrow f_k(u)$
445 **if** $\tau = \tau'$ **and** $u \notin \mathsf{Dom}(f)$
446 $\quad$ **then** $\mathsf{forge} \leftarrow 1$ **end if**
447 **if** $\tau = \tau'$ **then ret** 1 **end if**
448 **ret** 0

Fig. 3. Definition of game $G_1(A)$ for MAC-Pr. The marks ▶ and ▷ indicate that more lines will be inserted in later games.

where $t_1 = t + (q + 2\sigma) \cdot \mathrm{Time}_f$.

We next treat the term $\Pr\big[\mathsf{ones}\big]$. We construct a forger B_2 attacking f as follows: B_2 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, 2\sigma\}$ and then starts running the adversary A, simulating $F_k(\cdot)$ oracle and (if necessary) computing $V_k(\cdot,\cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_2 keeps a counter, which is initialized to 0 and gets incremented as B_2 makes a call to $f_k(\cdot)$ oracle except for the finalization step at line 345 where the counter remains unchanged. Just before making the α-th query x_α to $f_k(\cdot)$ oracle, B_2 quits running A and outputs a forgery $(x_\alpha, 1^n)$.

We now argue that the forger B_2 always succeeds as long as the index α is correctly guessed; the α-th query is expected to be the *first* call to $f_k(\cdot)$ oracle such that the value 1^n gets returned. We verify that the value x_α is new if the index α is such a value. The only thing to check is whether x_α has been already queried at the finalization step (at line 345). If it had been queried, then it would mean that x_α is of the form $*1^n*$ in which the leftmost $*$ is some n-bit string, contradicting with the minimality of α. Now the choice of α is independent of

the event ones, so we get $\mathrm{Adv}^{\mathrm{mac}}_f(B_2) \geq \frac{1}{2\sigma} \cdot \Pr[\mathsf{ones}]$. Observe that the adversary B_2 makes at most $q + 2\sigma$ queries to its oracle, and hence we obtain

$$\Pr\big[\mathsf{ones}\big] \leq 2\sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_2) \leq 2\sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma).$$

We proceed to the evaluation of $\Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big]$. To do this, we consider game G_2 defined in Fig. 4. Game G_2 adds three new flags zeros, twofold and dblcoll to game G_1.

Insert following lines into game G_1 (at mark $\triangleright$):

On query M **to** $F_k(\cdot)$:

320 **if** $v_1[i+1] = v_2[i+1]$
321 **then** twofold $\leftarrow 1$ **end if**
322 $w \leftarrow v_1[i+1] \,\|\, v_2[i+1]$
323 **if** $x_1[i] \notin \mathsf{Dom}_1(f)$ **and** $w \in \mathsf{Dbl}$
324 **then** dblcoll $\leftarrow 1$ **end if**
325 $\mathsf{Dom}_1(f) \xleftarrow{\cup} x_1[i]$; $\mathsf{Dbl} \xleftarrow{\cup} w$
326 **if** $w = 0^{2n}$ **then** zeros $\leftarrow 1$ **end if**

Subroutine $V_k(M, \tau)$:

422 $w \leftarrow v_1[i+1] \,\|\, v_2[i+1]$
423 **if** $x_1[i] \notin \mathsf{Dom}_1(f)$ **and** $w \in \mathsf{Dbl}$
424 **then** dblcoll $\leftarrow 1$ **end if**
425 $\mathsf{Dom}_1(f) \xleftarrow{\cup} x_1[i]$; $\mathsf{Dbl} \xleftarrow{\cup} w$
426 **if** $w = 0^{2n}$ **then** zeros $\leftarrow 1$ **end if**

Fig. 4. Definition of game $G_2(A)$ for MAC-Pr

We show that match implies at least one of the events zeros, twofold or dblcoll. Let $(M^\star, \tau^\star)$ be the forgery output by the adversary A. The event match implies that there exists some previous query M' to $F_k(\cdot)$ oracle such that $F_k(M') = F_k(M^\star)$, making an internal collision of the value u at lines 343 and 443. If either (i) M' is a suffix of $M^\star$ with $M^\star$ producing a chaining variable of 0^{2n} or (ii) $M^\star$ is such a suffix of M', then the case immediately implies the zeros event. If neither is such a suffix of the other, then the condition $F_k(M') = F_k(M^\star)$ guarantees that there must be an internal collision of $2n$-bit chaining variables. The collision value may be of the form $v\|v \in \{0,1\}^{2n}$ leading to the event twofold, or otherwise we must have the event dblcoll. Therefore, we have

$$\begin{aligned}\Pr\big[\mathsf{match} \wedge \overline{\mathsf{ones}}\big] &\leq \Pr\big[(\mathsf{zeros} \vee \mathsf{twofold} \vee \mathsf{dblcoll}) \wedge \overline{\mathsf{ones}}\big] \\ &\leq \Pr\big[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\big] + \Pr\big[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\big] \\ &\qquad + \Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\big].\end{aligned}$$

We bound the probability $\Pr\big[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\big]$. We construct a forger B_3 attacking f as follows: B_3 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, \sigma\}$ and then starts running the adversary A, simulating game G_2 by making queries to its $f_k(\cdot)$ oracle. The adversary B_3 keeps a counter, which is initialized to 0 and gets incremented by 1 (and not by 2) as B_3 makes two calls to $f_k(\cdot)$ oracle either at lines 306-311 or at lines 406-411. Just before the α-th execution of lines 306-311 or of lines 406-411, in which two queries x^1_α, x^2_α are about to be made, B_3 quits running A and outputs a forgery $(x^1_\alpha, 0^n)$.

The adversary B_3 always succeeds under the event $\mathsf{zeros} \wedge \overline{\mathsf{ones}}$ along with the condition that the index α is correctly guessed (*i.e.*, the α-th execution of lines 306-311 or of lines 406-411 sets the flag zeros for the first time), because the query x^1_α is guaranteed to be new if the value α is minimal (Notice that either $x^1_\alpha = x^1_{\alpha'}$ or $x^1_\alpha = x^2_{\alpha'}$ for some $\alpha' < \alpha$ implies the event zeros at index α'). In particular, the value x^1_α cannot have been queried at the finalization step (at line 345) due to the event $\overline{\mathsf{ones}}$. Now the choice of α is independent of the event $\mathsf{zeros} \wedge \overline{\mathsf{ones}}$, so we get $\mathrm{Adv}^{\mathrm{mac}}_f(B_3) \geq \frac{1}{\sigma} \cdot \Pr[\mathsf{zeros} \wedge \overline{\mathsf{ones}}]$. The adversary B_3 makes at most $q + 2\sigma$ queries to its oracle, which gives us

$$\Pr\big[\mathsf{zeros} \wedge \overline{\mathsf{ones}}\big] \leq \sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_3) \leq \sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma).$$

We then treat the term $\Pr\big[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\big]$. We construct a forger B_4 attacking f as follows: B_4 first picks an index $\alpha \xleftarrow{\$} \{1, 2, \ldots, \sigma - 1\}$ and then starts running the adversary A, simulating game G_2 by making queries to its $f_k(\cdot)$ oracle. The adversary B_4 keeps the same type of counter as the one used by B_3. Just before the α-th execution of lines 306-311, in which two queries x^1_α, x^2_α are about to be made, B_4 quits running A, makes a query $v^1_\alpha \leftarrow f_k(x^1_\alpha)$ and outputs a forgery (x^2_α, v^1_α).

The adversary B_4 always succeeds under the event $\mathsf{twofold} \wedge \overline{\mathsf{ones}}$ provided that the index α is correctly chosen (*i.e.*, the α-th execution of lines 306-311 sets the flag $\mathsf{twofold}$ for the first time), because the query x^2_α must be new if the value α is minimal (Notice that either $x^2_\alpha = x^1_{\alpha'}$ or $x^2_\alpha = x^2_{\alpha'}$ for some $\alpha' < \alpha$ implies the event $\mathsf{twofold}$ at index α'). Also, the value x^2_α cannot have been queried at the finalization step (at line 345) under the event $\overline{\mathsf{ones}}$. Now the choice of α is independent of the event $\mathsf{twofold} \wedge \overline{\mathsf{ones}}$, so we get $\mathrm{Adv}^{\mathrm{mac}}_f(B_4) \geq \frac{1}{\sigma - 1} \cdot \Pr[\mathsf{twofold} \wedge \overline{\mathsf{ones}}]$. The adversary B_4 makes at most $q + 2\sigma$ queries to its oracle, and hence

$$\Pr\big[\mathsf{twofold} \wedge \overline{\mathsf{ones}}\big] \leq (\sigma - 1) \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_4) \leq (\sigma - 1) \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma).$$

We go on to handle the term $\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\big]$. We introduce a threshold value $\theta = \theta(\sigma)$ in game G_3, as defined in Fig. 5. For the moment we just let θ be a certain function of σ. The description of θ is to be determined at the end of the proof. Game G_3 involves three sets All, Coll and Pair. The set All simply stores all input/output pairs (x, v) already computed as $f_k(x) = v$. The set Coll is a subset of All and stores only those pairs (x, v) that are colliding. The set Pair stores colliding pairs (x', x). We define a function $N(\mathsf{All}, v^\star) \stackrel{\mathrm{def}}{=} \big\{(x, v) \;\big|\; (x, v) \in \mathsf{All}, v = v^\star\big\}$. We see that

$$\begin{aligned}
\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}}\big] &= \Pr\big[(\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \mathsf{theta}) \\
&\qquad \vee (\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}})\big] \\
&\leq \Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big] \\
&\qquad + \Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big].
\end{aligned}$$

Insert following lines into game G_2 (at mark ▶):

On query M **to** $F_k(\cdot)$:
307 $C\big(x_1[i], v_1[i+1]\big)$
312 $C\big(x_2[i], v_2[i+1]\big)$

Subroutine $V_k(M, \tau)$:
407 $C\big(x_1[i], v_1[i+1]\big)$
412 $C\big(x_2[i], v_2[i+1]\big)$

Subroutine $C(x, v)$:
501 $U \leftarrow N(\mathsf{All}, v)$
502 **if** $(x, v) \notin \mathsf{All}$ **and** $U \neq \emptyset$ **then**
503 $\quad \mathsf{Coll} \xleftarrow{\cup} (x, v)$; $\mathsf{Coll} \leftarrow \mathsf{Coll} \cup U$
504 $\quad$ **for each** $(x', v') \in U$
505 $\quad\quad \mathsf{Pair} \xleftarrow{\cup} (x', x)$ **end for**
506 **end if**
507 $\mathsf{All} \xleftarrow{\cup} (x, v)$
508 **if** $|\mathsf{Pair}| \geq \theta(\sigma)$ **then** $\mathsf{theta} \leftarrow 1$ **end if**

Fig. 5. Definition of game $G_3(A)$ for MAC-Pr

We evaluate the probability $\Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big]$. We construct a forger B_5 attacking f as follows: B_5 picks two indices $\alpha, \beta \xleftarrow{\$} \{1, 2, \ldots, 2\sigma\}$ such that $\alpha < \beta$. Then B_5 starts running the adversary A, simulating $F_k(\cdot)$ oracle and (if necessary) computing $V_k(\cdot, \cdot)$ by making queries to its $f_k(\cdot)$ oracle. The adversary B_5 keeps the same type of counter as the one used by forger B_2 (counting the number of calls to $f_k(\cdot)$ oracle except at the finalization step). On making the α-th query x_α to $f_k(\cdot)$ oracle, B_5 records the returned value $v_\alpha \leftarrow f_k(x_\alpha)$. The adversary B_5 resumes running A. Then just before making the β-th query x_β to $f_k(\cdot)$, B_5 stops running A and outputs (x_β, v_α) as a forgery.

The adversary B_5 succeeds in forgery if $f_k(x_\alpha) = f_k(x_\beta)$ and the value x_β has not been queried at any previous point $\gamma < \beta$. Note that the query x_β cannot have been made at the finalization step, since we are under the event $\overline{\mathsf{ones}}$. Now we see that the number of such working pairs (α, β) is exactly $|\mathsf{Pair}|$, and this quantity $|\mathsf{Pair}|$ becomes larger than $\theta(\sigma)$ if the event theta occurs. Since there are $\binom{2\sigma}{2}$ choices of (α, β) total and this choice is independent from the event $\overline{\mathsf{ones}} \wedge \mathsf{theta}$, we obtain $\mathrm{Adv}^{\mathrm{mac}}_f(B_5) \geq \frac{\theta(\sigma)}{\binom{2\sigma}{2}} \cdot \Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big]$. This yields

$$\Pr\big[\overline{\mathsf{ones}} \wedge \mathsf{theta}\big] \leq \frac{\binom{2\sigma}{2}}{\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_5) \leq \frac{2\sigma^2}{\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_1, q + 2\sigma),$$

where we observe that B_5 makes at most $q + 2\sigma$ queries to its $f_k(\cdot)$ oracle.

Lastly, we assess the probability $\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big]$. For this, it is helpful to give a graphical interpretation of the sets Coll and Pair. See Fig. 6. The vertices of the *collision graph* are simply the points in Coll. Two distinct points (x, v), $(x', v') \in \mathsf{Coll}$ are connected by an edge if they are colliding, *i.e.*, if $v = v'$. Hence, the edges correspond to the points in Pair. The collision graph is always a disjoint union of complete graphs, containing no isolated vertices.

Now we construct a forger B_6 attacking f under the event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}$. First observe that the event $\overline{\mathsf{theta}}$ sets a limit on the number of vertices in the collision graph. The number is at most $2\theta(\sigma)$, for otherwise the number of edges would exceed $\theta(\sigma)$. Hence we have $|\mathsf{Coll}| \leq 2\theta(\sigma)$ throughout the game. The adversary B_6 first picks an index $\beta \xleftarrow{\$} \{2, 3, \ldots, \lceil 2\theta(\sigma) \rceil\}$ and

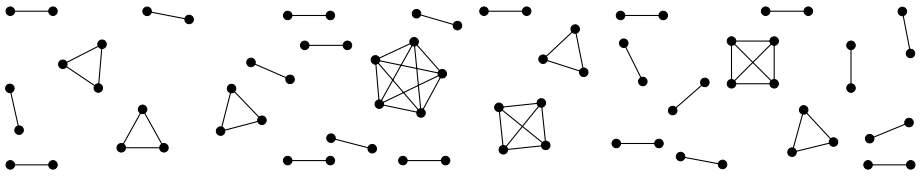

Fig. 6. A simple illustration of the collision graph. The number of vertices is equal to $|\mathsf{Coll}|$, and the number of edges is equal to $|\mathsf{Pair}|$.

then starts running A, simulating game G_3 by making queries to its $f_k(\cdot)$ oracle. B_6 keeps a counter, and on the β-th insertion of a colliding value (x_β, v_β) into the set Coll (so that at this point $|\mathsf{Coll}| = \beta$), B_6 stops running A and carries out the following operation:

1. If the β-th insertion happens to be at the "lower" pipe (*i.e.*, at lines 312 or 412), then B_6 aborts.
2. Otherwise, the β-th insertion of (x_β, v_β) occurs at the "upper" pipe (*i.e.*, at lines 307 or 407), in which case B_6 computes the following:
   (a) Choose $(x_\alpha, v_\alpha) \xleftarrow{\$} N(\mathsf{All}, v_\beta) \setminus \{(x_\beta, v_\beta)\}$ so that $x_\alpha \neq x_\beta$ and $f_k(x_\alpha) = v_\alpha = v_\beta = f_k(x_\beta)$.
   (b) Obtain the value $v'_\alpha \leftarrow f_k(\bar{x}_\alpha)$ by either querying $\bar{x}_\alpha$ to $f_k(\cdot)$ oracle again or searching the set All for the entry $(\bar{x}_\alpha, \cdot)$.
   (c) Submit $(\bar{x}_\beta, v'_\alpha)$ as a forgery.

We verify that B_6 succeeds in forgery as long as the values for β and x_α are guessed correctly (and that the case B_6 aborts is not a concern). The basic idea is that B_6 hopes to have $(v_\alpha, v'_\alpha) = (v_\beta, v'_\beta)$ with $v'_\beta \stackrel{\text{def}}{=} f_k(\bar{x}_\beta)$ and $\bar{x}_\beta$ being new. The event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll}$ guarantees the existence of such a pair (α, β) with $(v_\alpha, v'_\alpha) = (v_\beta, v'_\beta)$, $x_\alpha \neq x_\beta$ and $x_\alpha \neq \bar{x}_\beta$. So let $\alpha^\star < \beta^\star$ be the minimal indices satisfying these conditions (Note that at $(\alpha^\star, \beta^\star)$ the event $\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll}$ does not necessarily occur, since the query $x_{\alpha^\star}$ may well be in the "lower" pipe). The minimality ensures that $\bar{x}_{\beta^\star}$ is new under the event $\overline{\mathsf{ones}}$.

We evaluate the success probability of B_6. In order to do this, we need to count the number of possible values for x_α at step 2(a). We claim that this number is at most $\sqrt{2\theta(\sigma)}$. To see this, observe that the values for x_α come from the vertices of the connected component corresponding to the output collision value v_β. Such a connected component is a complete graph, and the number of vertices must be at most $\sqrt{2\theta(\sigma)} + 1$, for otherwise the number of edges would exceed $\binom{\lceil \sqrt{2\theta(\sigma)}+1 \rceil}{2} \geq \frac{\left(\sqrt{2\theta(\sigma)}+1\right)\sqrt{2\theta(\sigma)}}{2} > \theta(\sigma)$, contradicting with the event $\overline{\mathsf{theta}}$. Hence the number of possible candidates for x_α is at most $\left(\sqrt{2\theta(\sigma)} + 1\right) - 1 = \sqrt{2\theta(\sigma)}$, excluding the point x_β. Now observe that the choices of β and x_α are completely hidden from the transcript of A, and these values do not affect the probability $\Pr\left[\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\right]$, yielding

$\mathrm{Adv}^{\mathrm{mac}}_f(B_6) \geq \frac{1}{2\theta(\sigma)} \cdot \frac{1}{\sqrt{2\theta(\sigma)}} \cdot \Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big]$. Hence we get

$$\begin{aligned}\Pr\big[\,\overline{\mathsf{twofold}} \wedge \mathsf{dblcoll} \wedge \overline{\mathsf{ones}} \wedge \overline{\mathsf{theta}}\,\big] &\leq 2\theta(\sigma)\sqrt{2\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_6) \\ &\leq 3 \cdot \theta(\sigma)^{3/2} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_2, q + 2\sigma),\end{aligned}$$

where $t_2 = t + (q + 2\sigma) \cdot \mathrm{Time}_f + \mathrm{Mem}_f(2\sigma)$. Note that B_6 makes at most $q + 2\sigma$ queries to its oracle and maintains the list All, which consumes at most $\mathrm{Mem}_f(2\sigma)$ amount of time complexity.

It remains to determine the threshold function $\theta(\sigma)$. To do this, we sum up the terms obtained:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{mac}}_F(A) \leq\ & \mathrm{Adv}^{\mathrm{mac}}_f(B_1) + 2\sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_2) \\ &+ \sigma \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_3) + (\sigma - 1) \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_4) \\ &+ \frac{2\sigma^2}{\theta(\sigma)} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_5) + 3 \cdot \theta(\sigma)^{3/2} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(B_6).\end{aligned}$$

Now we simply set $\theta(\sigma) \stackrel{\mathrm{def}}{=} \sigma^{4/5}$. This choice leads to the coefficients of $\frac{2\sigma^2}{\theta(\sigma)} = 2\sigma^{6/5}$ and $3 \cdot \theta(\sigma)^{3/2} = 3\sigma^{6/5}$. Rounding off the terms yields $\mathrm{Adv}^{\mathrm{mac}}_F(A) \leq 9 \cdot \sigma^{6/5} \cdot \mathrm{Adv}^{\mathrm{mac}}_f(t_2, q + 2\sigma)$, as desired. □

5.3 On the Tightness of the Bound $O\big(2^{5n/6}\big)$

At the current stage the best attack we know is the birthday attack, which requires $O(2^n)$ query complexity. Hence the bound obtained $O\big(2^{5n/6}\big)$ is not tight. The gap originates in our construction of adversary B_6. Recall that when B_6 makes a choice of β it assumes the worst case scenario of $2\theta(\sigma)$-many vertices, the collision graph being a disjoint union of numerous 2-complete graphs. On the other hand, when B_6 makes a choice of x_α it assumes the other worst case scenario of $\sqrt{2\theta(\sigma)}$-many vertices, the collision graph being a single gigantic complete graph. Hence we are considering two extreme cases which cannot happen concurrently. It remains an open problem to fill in the gap between our proof bound $O\big(2^{5n/6}\big)$ and the best known attack bound $O(2^n)$.

6 PRF-Pr beyond the Birthday Barrier

The PRF-Pr property of our mode immediately follows from the forthcoming PRO-Pr result. This implication is due to the following simple lemma:

Lemma 1 (PRO-Pr ⇒ PRF-Pr in the Dedicated-Key Setting). *Let $F_k : \{0,1\}^* \to \{0,1\}^n$ be a mode of operation in the dedicated-key setting, which iterates a primitive $f_k : \{0,1\}^d \to \{0,1\}^n$ with a secret key k. If the mode F is PRO-Pr in the sense of iterating a random function $f : \{0,1\}^d \to \{0,1\}^n$, then it is PRF-Pr. Specifically, for any simulator $\mathcal{S}$, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_F(t, q, \sigma) \leq \mathrm{Adv}^{\mathrm{prf}}_f(t', q') + \mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t, q, \sigma),$$

where $t' = t + q' \cdot \mathrm{Time}_f$ and q' is the number of calls to f necessary to process σ-long queries to F.

Proof. Let A be a distinguisher attacking F, having a time complexity at most t, making queries whose total complexity is at most σ blocks. We let $F^\star : \{0,1\}^* \to \{0,1\}^n$ denote the mode of operation identical to F except that the underlying primitive is replaced with a random function $f : \{0,1\}^d \to \{0,1\}^n$ (rather than a pseudo-random function f_k). Let $\mathcal{F} : \{0,1\}^* \to \{0,1\}^n$ be a random function. We can bound the advantage $\mathrm{Adv}^{\mathrm{prf}}_F(A)$ as

$$\begin{aligned}
&\Pr\left[A^{F_k(\cdot)} = 1\right] - \Pr\left[A^{\mathcal{F}(\cdot)} = 1\right] \\
&= \Pr\left[A^{F_k(\cdot)} = 1\right] - \Pr\left[A^{F^\star(\cdot)} = 1\right] + \Pr\left[A^{F^\star(\cdot)} = 1\right] - \Pr\left[A^{\mathcal{F}(\cdot)} = 1\right] \\
&\le \mathrm{Adv}^{\mathrm{prf}}_f(B) + \mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(A),
\end{aligned}$$

where we construct the distinguisher B, who attacks f, "naturally" from A who is distinguishing between F_k and $F^\star$. Note that $\mathcal{S}$ can be any simulator. We see that B makes at most q' queries to its f oracle and that A makes no queries to the underlying primitive or to the simulator. This gives us the bound. □

7 PRO-Pr beyond the Birthday Barrier

In this section we show that our double-piped mode of operation $F : \{0,1\}^* \to \{0,1\}^n$ is PRO-Pr with security above the birthday bound. It turns out that our mode provides the full PRO security of $O(2^n)$ query complexity. Our proof essentially follows the lines of [3].

In order to state our theorem, we give a description of simulators. See Fig. 7. The simulators maintain a directed, edge-labeled graph structure $\mathsf{Vert}(f)$ and $\mathsf{Edge}(f)$. The symbol $P(\mathsf{Edge}(f), w)$ denotes the set of "paths" $m_1 \| m_2 \| \cdots$ (concatenated labels) starting at the origin 0^{2n} and connecting to the vertex w. The simulators also involve arrays $f[\cdot]$ and $g[\cdot]$, which are everywhere undefined at the beginning of the game. The symbol $\mathsf{Dom}(f)$ denotes the set of already-defined domain points in the array f.

Theorem 2. *The mode of operation F is PRO-Pr beyond the birthday bound. Specifically, we have*

$$\mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t, q_f, \sigma_F) \le \frac{\sigma_F + q_f}{2^{n-1}} + \frac{5\sigma_F^2 + 18\sigma_F q_f + 17 q_f^2}{2^{2n}},$$

where the simulator $\mathcal{S}$ has a time complexity at most $t + \mathrm{Mem}_f(2\sigma_F)$ and makes at most q_f queries to $\mathcal{F}$ oracle (the idealized F).

Proof (Sketch). First we introduce a slightly modified mode of operation $H : \{0,1\}^* \to \{0,1\}^n$. See Fig. 8. The new mode H iterates two independent random functions f and g. The mode H is identical to the original F except that the finalization step is replaced with the new function g.

Initialization:
600 $\mathsf{Vert}(f) \leftarrow \{0^{2n}\}$

Simulator $\mathcal{S}(x)$:
700 **if** $x \in \mathsf{Dom}(f)$ **then ret** $f[x]$ **end if**
701 $v_1\|v_2\|m \xleftarrow{n,n,d-2n} x$
702 **if** $v_2 = 1^n$ **then** $v_2'\|m' \xleftarrow{n,d-3n} m$
703 **if** $v_1\|v_2' \in \mathsf{Vert}(f)$ **then**
704 $M \xleftarrow{\$} P\big(\mathsf{Edge}(f), v_1\|v_2'\big)$
705 $f[x] \leftarrow \mathcal{F}(M)$
706 **else** $f[x] \xleftarrow{\$} \{0,1\}^n$ **end if**
707 **else** $f[x] \xleftarrow{\$} \{0,1\}^n$; $f[\bar{x}] \xleftarrow{\$} \{0,1\}^n$
708 **if** $v_1\|v_2 \in \mathsf{Vert}(f)$ **then**
709 $\mathsf{Vert}(f) \xleftarrow{\cup} f[x] \parallel f[\bar{x}]$
710 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(v_1\|v_2, m, f[x]\|f[\bar{x}]\big)$
711 **end if**
712 **if** $\bar{v}_1\|v_2 \in \mathsf{Vert}(f)$ **then**
713 $\mathsf{Vert}(f) \xleftarrow{\cup} f[\bar{x}] \parallel f[x]$
714 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(\bar{v}_1\|v_2, m, f[\bar{x}]\|f[x]\big)$
715 **end if end if**
716 **ret** $f[x]$

Simulator $\mathcal{S}_f(x)$:
800 **if** $x \in \mathsf{Dom}(f)$ **then ret** $f[x]$ **end if**
801 $w\|m \xleftarrow{2n,d-2n} x$
802 $f[x] \xleftarrow{\$} \{0,1\}^n$; $f[\bar{x}] \xleftarrow{\$} \{0,1\}^n$
803 **if** $w \in \mathsf{Vert}(f)$ **then**
804 $\mathsf{Vert}(f) \xleftarrow{\cup} f[x] \parallel f[\bar{x}]$
805 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(w, m, f[x]\|f[\bar{x}]\big)$ **end if**
806 **if** $\bar{w} \in \mathsf{Vert}(f)$ **then**
807 $\mathsf{Vert}(f) \xleftarrow{\cup} f[\bar{x}] \parallel f[x]$
808 $\mathsf{Edge}(f) \xleftarrow{\cup} \big(\bar{w}, m, f[\bar{x}]\|f[x]\big)$ **end if**
809 **ret** $f[x]$

Simulator $\mathcal{S}_g(u)$:
900 **if** $u \in \mathsf{Dom}(g)$ **then ret** $g[u]$ **end if**
901 **if** $u \in \mathsf{Vert}(f)$ **then**
902 $M \xleftarrow{\$} P\big(\mathsf{Edge}(f), u\big)$
903 $g[u] \leftarrow \mathcal{H}(M)$
904 **else** $g[u] \xleftarrow{\$} \{0,1\}^n$ **end if**
905 **ret** $g[u]$

Fig. 7. Definitions of simulators $\mathcal{S}$, $\mathcal{S}_f$ and $\mathcal{S}_g$ for PRO-Pr

We start with noting that F is secure if H is secure. The reduction is without birthday degradation. Namely, we show that if H is PRO-Pr beyond the birthday bound, then so is F. More specifically, we obtain

$$\mathrm{Adv}^{\mathrm{pro}}_{F,\mathcal{S}}(t, q_f, \sigma_F) \le \mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}(t, q_f, q_f, \sigma_F) + \frac{\sigma_F + q_f}{2^n},$$

where in the parameters of $\mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}$ the second "q_f" is for the number of queries to g oracle and the last σ_F for the total complexity of queries made to H oracle. The proof is the same as the one for Theorem 5.2 in [3].

Now it remains to analyze the security of H. The analysis amounts to bounding the probability that certain "bad" events occur as in [3]. The key to keeping

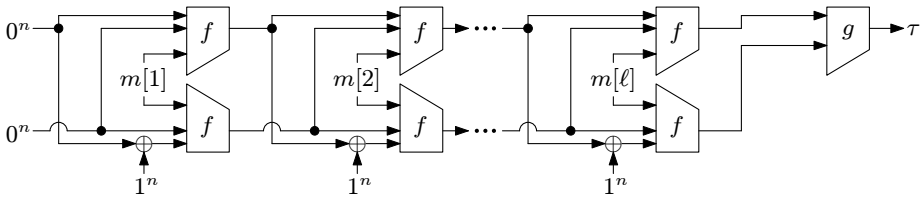

Fig. 8. Description of $H : \{0,1\}^* \to \{0,1\}^n$ iterating two random functions f and g

the security bound in $O(2^n)$ is to treat the output values of f always as pairs $\big(f(x), f(\bar{x})\big)$ and to keep track of those "bad" events associated with the $2n$-bit values. By doing so, we are able to obtain

$$\mathrm{Adv}^{\mathrm{pro}}_{H,\mathcal{S}_f,\mathcal{S}_g}(t, q_f, q_g, \sigma_H) \le \frac{\sigma_H + q_f}{2^n} + \frac{5\sigma_H^2 + 14\sigma_H q_f + 12q_f^2 + 4\sigma_H q_g + q_g + 4q_f q_g}{2^{2n}},$$

which yields the claimed bound. □

Acknowledgments

The author is most grateful to the Eurocrypt 2009 anonymous reviewers for their valuable comments. One of the reviewers carried out a thorough review of the MAC-Pr proof and pointed out a couple of typos. Some of the reviewers pointed out an inappropriate treatment of the related work. These comments were especially helpful in revising the paper.

References

1. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In Wiener, M.J., ed.: CRYPTO 1999. Volume 1666 of LNCS., Heidelberg, Springer (1999) 252–269
2. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive: Report 2004/304 (2004)
3. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In Lai, X., Chen, K., eds.: ASIACRYPT 2006. Volume 4284 of LNCS., Heidelberg, Springer (2006) 299–314
4. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A., eds.: ICALP 2007. Volume 4596 of LNCS., Heidelberg, Springer (2007) 399–410
5. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In Shoup, V., ed.: CRYPTO 2005. Volume 3621 of LNCS., Heidelberg, Springer (2005) 430–448
6. Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable security analysis of popular hash functions with prefix-free padding. In Lai, X., Chen, K., eds.: ASIACRYPT 2006. Volume 4284 of LNCS., Heidelberg, Springer (2006) 283–298
7. Chang, D., Nandi, M.: Improved indifferentiability security analysis of chopMD hash function. In Nyberg, K., ed.: FSE 2008. Volume 5086 of LNCS., Heidelberg, Springer (2008) 429–443
8. Coron, J.S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In Wagner, D., ed.: CRYPTO 2008. Volume 5157 of LNCS., Heidelberg, Springer (2008) 1–20
9. Damgård, I.: A design principle for hash functions. In Brassard, G., ed.: CRYPTO 1989. Volume 435 of LNCS., Heidelberg, Springer (1990) 416–427

10. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In Naor, M., ed.: EUROCRYPT 2007. Volume 4515 of LNCS., Heidelberg, Springer (2007) 534–554
11. Dodis, Y., Pietrzak, K., Puniya, P.: A new mode of operation for block ciphers and length-preserving MACs. In Smart, N.P., ed.: EUROCRYPT 2008. Volume 4965 of LNCS., Heidelberg, Springer (2008) 198–219
12. Hirose, S., Park, J.H., Yun, A.: A simple variant of the Merkle-Damgård scheme with a permutation. In Kurosawa, K., ed.: ASIACRYPT 2007. Volume 4833 of LNCS., Heidelberg, Springer (2007) 113–129
13. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In Daemen, J., Rijmen, V., eds.: FSE 2002. Volume 2365 of LNCS., Heidelberg, Springer (2002) 237–251
14. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In Franklin, M.K., ed.: CRYPTO 2004. Volume 3152 of LNCS., Heidelberg, Springer (2004) 306–316
15. JTC1: Data cryptographic techniques—Data integrity mechanism using a cryptographic check function employing a block cipher algorithm (1989) ISO/IEC 9797.
16. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of LNCS., Heidelberg, Springer (2005) 474–490
17. Lucks, S.: A failure-friendly design principle for hash functions. In Roy, B.K., ed.: ASIACRYPT 2005. Volume 3788 of LNCS., Heidelberg, Springer (2005) 474–494
18. Merkle, R.C.: One way hash functions and DES. In Brassard, G., ed.: CRYPTO 1989. Volume 435 of LNCS., Heidelberg, Springer (1990) 428–446
19. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Naor, M., ed.: TCC 2004. Volume 2951 of LNCS., Springer (2004) 21–39
20. Maurer, U.M., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M., eds.: ICALP 2005. Volume 3580 of LNCS., Heidelberg, Springer (2005) 472–484
21. Maurer, U.M., Tessaro, S.: Domain extension of public random functions: Beyond the birthday barrier. In Menezes, A., ed.: CRYPTO 2007. Volume 4622 of LNCS., Heidelberg, Springer (2007) 187–204
22. NIST: Computer data authentication (1985) FIPS 113.
23. Patarin, J.: Security of random Feistel schemes with 5 or more rounds. In Franklin, M.K., ed.: CRYPTO 2004. Volume 3152 of LNCS., Heidelberg, Springer (2004) 106–122
24. Patarin, J.: A proof of security in $O(2^n)$ for the Benes scheme. In Vaudenay, S., ed.: AFRICACRYPT 2008. Volume 5023 of LNCS., Heidelberg, Springer (2008) 209–220
25. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast MACs from hash functions. In Coppersmith, D., ed.: CRYPTO 1995. Volume 963 of LNCS., Heidelberg, Springer (1995) 1–14
26. Preneel, B., van Oorschot, P.C.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory **45**(1) (1999) 188–199
27. Yasuda, K.: Multilane HMAC—Security beyond the birthday limit. In Srinathan, K., Rangan, C.P., Yung, M., eds.: INDOCRYPT 2007. Volume 4859 of LNCS., Heidelberg, Springer (2007) 18–32
28. Yasuda, K.: A one-pass mode of operation for deterministic message authentication—Security beyond the birthday barrier. In Nyberg, K., ed.: FSE 2008. Volume 5086 of LNCS., Heidelberg, Springer (2008) 316–333