IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2019, No. 3, pp. 43–80. DOI:10.13154/tosc.v2019.i3.43-80

DoveMAC: A TBC-based PRF with Smaller State, Full Security, and High Rate

Tony Grochow[1], Eik List[1] and Mridul Nandi[2]

[1] Bauhaus-Universität Weimar, Weimar, Germany
<firstname>.<lastname>(at)uni-weimar.de
[2] Indian Statistical Institute, Kolkata, India
mridul.nandi(at)gmail.com

Abstract. Recent parallelizable message authentication codes (MACs) have demonstrated the benefit of tweakable block ciphers (TBCs) for authentication with high security guarantees. With ZMAC, Iwata et al. extended this line of research by showing that TBCs can simultaneously increase the number of message bits that are processed per primitive call. However, ZMAC and previous TBC-based MACs needed more memory than sequential constructions. While this aspect is less an issue on desktop processors, it can be unfavorable on resource-constrained platforms. In contrast, existing sequential MACs limit the number of message bits to the block length of the primitive n or below.
This work proposes DoveMAC, a TBC-based PRF that reduces the memory of ZMAC-based MACs to $2n+2t+2k$ bits, where n is the state size, t the tweak length, and k the key length of the underlying primitive. DoveMAC provides $(n+\min(n+t))/2$ bits of security, and processes $n+t$ bits per primitive call. Our construction is the first sequential MAC that combines beyond-birthday-bound security with a rate above n bits per call. By reserving a single tweak bit for domain separation, we derive a single-key variant DoveMAC1k.

Keywords: Authentication · authenticated encryption · message authentication code · PRF · provable security · tweakable block cipher

1 Introduction

Message Authentication Codes (MACs) secure the integrity and authenticity of communications. Many standardized MACs, such as CMAC [Dwo16], OMAC [IK03], or PMAC [BR02] are block-cipher modes of operations with birthday-bound security[1] This fact implies hardly a problem if the state size of the underlying primitive is at least 128 bits; however, resource-limited platforms often use primitives with smaller state and key size, e.g., HIGHT [HSH+06] or PRESENT [BKL+07]. Smaller state sizes result in lower security guarantees, which may be impractical when used in a mode with birthday-bound security, as emphasized in [IMG+14, MV04]. Therefore, MACs with higher security margins are desirable for lightweight platforms, in particular, for stateless deterministic MACs that avoid the transmission of additional nonces.

Existing Parallelizable MACs. A considerable amount of research has been devoted to the construction of MACs with high security bounds. A generic approach is to sum the outputs of several independent hash functions or PRFs, e.g., Yasuda's Sum-ECBC [Yas10]

[1] The birthday attack was presented on hash functions first by Yuval [Yuv79] and on MACs by Preneel and van Oorschot [PvO95] (cf. [MvOV96]).

Received: 2018-06-01, Revised: 2019-06-01, Accepted: 2019-08-01, Published: 2019-09-20

or Iwata and Minematsu's Fr [IM16], which accumulates the outputs of multiple GMAC instances. However, this strategy implies r times the computational effort, the key material, as well as the state size as the single-hash approach.
Many block-cipher-based MACs with higher security were inspired by the classical PMAC [BR02] design. Those process the message blocks in parallel, accumulate the results, and give the sum as input into a finalization that produces the tag. One approach for (slightly) higher security is the use of counter sums, where some bits of the message inputs are used for a counter. Thus, the influence of the message length on the security bound is eliminated [BGR95, Ber99]. This approach has seen a recent revival, e.g., in LightMAC [LPTY16] or the variable-size counter modes by Dutta et al. [DJN17].
A higher security gain can be achieved by a larger accumulator, as had been proposed in PMAC$^+$ [Yas11]. This approach was adopted by various proposals, e.g., by the extensions of LightMAC [Nai17, Nai18a], but also by [IMPS17, LN17, Nai15]. In particular, the latter profited from the use of tweakable block ciphers as underlying primitive.

Tweakable Block Ciphers. Tweakable block ciphers (TBCs) extend classical block ciphers (BCs) by an additional public input, called tweak [LRW02]. Given a non-empty set of keys $\mathcal{K}$, a tweak space $\mathcal{T}$, and a state space $\mathcal{B}$, a TBC $\widetilde{\pi} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$ is a family of keyed permutations, s.t. for all combinations of key and tweak $K, T \in \mathcal{K} \times \mathcal{T}$, $\widetilde{\pi}(K, T, \cdot)$ is a permutation over $\mathcal{B}$. With the recent advent of dedicated performant proposals such as Deoxys-BC, Joltik-BC [JNP14], or Skinny [BJK$^+$16c], TBCs have been established for various cryptographic applications, including MACs, encryption, and authenticated encryption schemes [IMPS17, JNP16b, Nai15, PS16]. In those contexts, the tweak can efficiently separate domains, which can not only increase security guarantees but also lead to simpler designs. For instance, PMAC_TBC1k [Nai15], Encrypted Parallel Wegman-Carter (EPWC) [PS16] and its variant in SCT-2 [JNP16b] encode domains in the tweak space for avoiding additional input masks and multiple keys.
In general, tweakable block ciphers are slightly slower than classical block ciphers with equal security guarantees since the additional tweak needs to be processed also in a secure manner. With ZMAC, Iwata et al. [IMPS17] proposed a message authentication code that brought a considerable speed-up. In contrast to previous designs, ZMAC used both the state and the tweak input to process $n + t$ bits of message material per primitive call. Thus, it could benefit from a TBC both in terms of high security and a high rate.

State Size of A Scheme. We briefly define our intention of the required state size of a scheme. Given a TBC with a t-bit tweak, a k-bit key and a block length of n bits, processing a message block with a (T)BC implies the need to hold $n+k$ (or $n+t+k$ bits), respectively. A higher-level scheme may further require to store masking keys or accumulators. We disregard further memory for performance optimizations; e.g., on the majority of platforms, it is common to store an extended state with expanded key material. Moreover, TBC-based constructions can usually be easily adapted to reserve one or a few bits of the tweak for domain separation to avoid multiple keys. In certain settings, an outside environment can precompute and prepare eventual checksums, e.g., by appending it to the input; though, this poses a security risk as we will discuss later in Section 3, and a higher-level environment may be absent in certain settings.

Parallel Block-cipher-based MACs. In general, the PMAC-like block-cipher-based constructions above are not optimized for a small state. The block-cipher-based variants [Yas11, Yas12, Zha15] require at least $2n$ bits for the current block-cipher state and an input mask, plus k bits for at least one key, plus $2n$ bits for an accumulator. LightMAC_Plus [Nai17] is similar to PMAC$^+$, but spares n bits for the mask, yielding $3n + k$ bits of memory. LightMAC_Plus_1k [Nai17] shares the state requirements of

Table 1: Comparison of existing deterministic MACs. $r \geq 1$ is a flexible parameter for the rate; $n/k/t$ = state/key/tweak size in bits; $c \leq n$ = #bits for a counter; q = #queries; m = max. #blocks per message, σ = total #message blocks. the rate excludes the calls in the finalization.

Construction	#Keys	State (bits)	Security	Rate (#bits/call)	Finalization (#calls)
Based on permutations					
Chaskey [MMH+14]	1	$2n+k$	$O(\sigma^2/2^{2n})$	$2n$	1
Based on compression functions					
NI+-MAC [DNP16]	1	$3n+k$	$O(q^2m^2/2^{2n})$	n	2
Based on classical block ciphers					
LightMAC [LPTY16]	2	$2n+k$	$O(q^2/2^n)$	$n-c$	1
LightMAC_Plus [Nai17]	3	$3n+k$	$O(q^3/2^{2n})$	$n-c$	2
LightMAC_Plus2 [Nai17]	$3+r$	$4n+k$	$O(q^{r+1}/2^{rn}+q^2/2^{2n})$	$n-c$	$2+r$
LightMAC_Plus_1k [Nai18a]	1	$3n+k-4$	$O(q^3/2^{2n})$	$n-c$	3
PMAC+ [Yas11]	3	$5n+k$	$O(q^3m^3/2^{2n}+qm/2^n)$	n	2
PMAC w/ Parity [Yas12]	4	$(r+2)n+k$	$O(q^2/2^n+mq\sigma/2^{2n})$	$(r-1)n/r$	2
1k_PMAC+ [DDN+17]	1	$5n+k$	$O(q\sigma^2/2^{2n}+\sigma/2^n)$	n	2
3kf9 [ZWSW12]	3	$2n+k$	$O(q^3m^3/2^{2n}+qm/2^n)$	n	2
Based on tweakable block ciphers					
EPWC [PS16] (no nonces)	1	$2n+t+k$	$O(q^2/2^n)$	n	1
PMAC_TBC1k [Nai15]	1	$3n+t+k$	$O(n^2q^2/2^{2n})$	n	2
PMAC_TBC3k [Nai15]	3	$3n+t+k$	$O(q^2/2^{2n})$	n	2
ZMAC [IMPS17]	1	$4n+2t+k$	$O(\sigma^2/2^{n+\min(n,t)})$	$t+n$	4
ZMACb/ZMACt/ZMACbt [Nai18b]	1	$4n+2t+k$	$O(q^2/2^n)$	$t+n$	2
ZMAC1 [Nai18b]	1	$4n+2t+k$	$O(\sigma^2/2^{n+\min(n,t)}+q/2^n)$	$t+n$	1
DoveMAC [**This work**]	2	$2n+2t+2k$	$O(q^2m^2/2^{n+\min(n,t)}+q/2^n)$	$t+n$	1
DoveMAC1k [**This work**]	1	$2n+2t+k$	$O(q^2m^2/2^{n+\min(n,t-1)}+q/2^n)$	$t+n-1$	1

LightMAC_Plus; it spares four bits due to its $n-2$-bit accumulators. The generalized variant of LightMAC_Plus, LightMAC_Plus2 [Nai18a], requires $4n+k$ bits in its finalization; PMAC w/ Parity [Yas12] needs r masks plus n bits each for the current block and the accumulator.

Parallel TBC-based MACs. The situation is similar for existing TBC-based MACs. ZMAC needs $3n+t+k$ bits for input, masks, and key, plus $n+t$ bits for the accumulators. Recently, Naito [Nai18b] investigated variants of ZMAC called ZMACb, ZMACt, and ZMACbt, which differed from each other in the fact that they used only the plaintext, tweak, or both inputs during the finalization. The same work developed ZMAC1, which could avoid the tweak-based domain separation in the hash function with limited loss of security. Though, his variants share the same memory requirements with ZMAC.[2]
EPWC or PMAC_TBC1k avoid the need for input masks. the former [PS16] requires a nonce for beyond-birthday security and $2n+t+k$ bits; its variant as used in SCT-2 [JNP16b] can append the nonce that is used in the finalization to the message to use the same amount of memory. Though, since we consider deterministic authentication, Table 1 contains only the nonce-ignoring variant of EPWC, although it can guarantee only birthday-bound security. Finally, PMAC_TBC1k and PMAC_TBC3k require

[2]The definition of XT1 [Nai18b, Algorithm 5] suggests that one could alternatively compute the input masks on-the-fly for each block. Here, we neglect this strategy, because the high rate is usually more important; however, it could reduce state by $2n$ bits and may be worth a separate investigation.

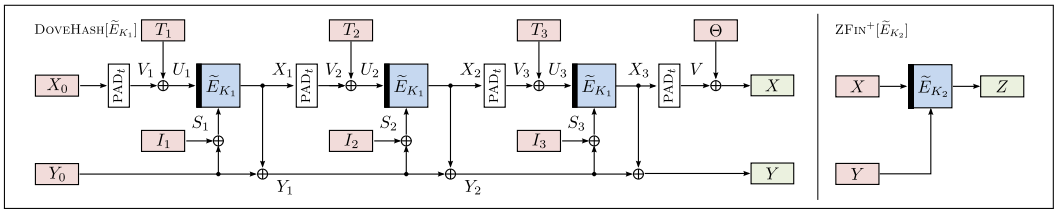

Figure 1: Hashing with DoveHash (**left**) and finalization with ZFin$^+$ (**right**) of a message $M = (M_1, M_2, M_3)$ with three $(t+n)$-bit blocks with DoveMAC$[\widetilde{E}_{K_1,K_2}]$. Each block M_i is split into a t-bit part T_i and an n-bit part I_i. $\Theta = \bigoplus_{i=1}^{m} T_i$ denotes a checksum over all tweak input parts.

$3n + k + t$ bits each due to their $2n$-bit accumulator.

Sequential MACs with High Security Guarantees. Inspired by CBC-based MACs and iterated hash functions, there exists an alternative portfolio of MACs that are inherently sequential, but employ significantly smaller amounts of memory. Clearly, the security of single-chain designs is limited by the birthday bound because of potential state collisions. A simple approach for increasing the security is to employ a primitive with larger state. This can be realized, e.g., by a compression function [DNP16, Yas08, Yas09] or with a wide-state permutation, as in Chaskey [MMH+14]. For n-bit security, Chaskey needs $2n$ bits for the primitive plus k bits for a final key. LightMAC and [DJN17] are on par with $2n + k$ bits of memory. Though, permutation-based modes such as Chaskey have to process a $2n$-bit state all the time. In contrast, tweakable primitives could set the tweak constant or transform it in more lightweight manner than the state, as performed in most recent lightweight TBCs such as e.g. Skinny [BJK+16b] or QARMA [Ava17].
One approach towards higher security is the usage of two chains for (1) primitive calls and (2) accumulating intermediate results. For instance, 3kf9 [ZWSW12] enhances CBC by such an accumulating chain; a similar example is NI$^+$-MAC [DNP16]. The former combines beyond-birthday-bound security with low memory demands of only $n + k$ bits for a CBC-like mode plus n bits for an accumulator. The latter used a $2n$-bit compression function instead of a block cipher.

Rate. Besides minimizing the memory requirements, increasing the number of message bits per primitive call is a second important factor for increasing the efficiency of MACs. Nearly all constructions considered above process n bits or less of message material per primitive call, where n is the state size. Only ZMAC and its derivatives allow to authenticate $n+t$ message bits per call, but at the cost of a much larger state. The question that arises is if and how the high rate of ZMAC could be combined while maintaining or reducing its state size.

Contribution. This work tries to give an affirmative answer to the above. We propose DoveMAC, a highly secure PRF that needs $2n + 2t + 2k$ bits of memory, based on a tweakable block cipher with n-bit state and t-bit tweak size. We show a security level of approximately $(n + \min(n, t))/2$ bits, which means full n-bit security if $t \geq n$ and $(n + t)/2$ bits otherwise. Figure 1 provides a schematic overview. It maintains a chain of t bits at the top and n bits at the bottom. Each $(t + n)$-bit message block is processed by a single call of a tweakable block cipher, such that t bits are used as tweak, and n bits as state input. The output of the primitive is XORed to the bottom lane before the next block. An accumulated checksum of all tweak inputs is added to the top lane. Finally, both lanes are used as tweak and state input, respectively, to a final call of the primitive

that uses an independent key for generating the tag.
From a high-level point of view, our construction is inspired by 3kf9, NI$^+$-MAC, and ZMAC, but possesses twice the rate than the former two, and requires less memory than the latter. We show its utility for highly secure authenticated encryption by combining it with the nonce-IV-based variant of Counter-in-Tweak [PS16].
DoveMAC requires $2n + 2t + k$ bits for hashing the message. An additional independent k-bit key is used for the finalization phase. We briefly outline also a variant called DoveMAC1k that reserves a single bit for domain separation. Then, the same key can be used in both hashing and finalization which can save key-management costs, but may require to split the inputs into unconventional block lengths for usual TBCs.

Outline. The remainder is structured as follows: after Section 2 continues with briefly stating the necessary preliminaries, Section 3 describes the details of DoveMAC. Section 4 provides an analysis of its PRF security, that is detailed in the subsequent Sections 5 and 6. Section 7 describes an instantiation of DoveMAC with Skinny-64-128 for common microcontrollers and compares it to the most efficient variant of ZMAC, ZMAC1. Appendix A describes two birthday-bound attacks on versions from earlier design phases and a forgery attack on a single-key version, which provide a further insight and rationale of our final proposal. Furthermore, Appendix C proposes an authenticated encryption scheme that combines DoveMAC for authentication, and Counter-in-Tweak [PS16] for highly secure encryption.

2 Preliminaries

General Notation. We write lowercase letters for indices and integers, uppercase letters for functions and variables, and calligraphic uppercase letters for sets. For $a, b \in \mathbb{N}_0$, we write $[a..b]$ as the set of integers $\{a, a+1, \ldots, b\}$. Given a set $\mathcal{X}$, we define $\mathcal{X}^+ = \bigcup_{i=1}^{\infty} \mathcal{X}^i$, and $\mathcal{X}^* = \bigcup_{i=0}^{\infty} \mathcal{X}^i$. We denote by $\{0,1\}^x$ the set of bit strings of length x, the concatenation of binary strings X and Y by $X \,\|\, Y$, and their XOR by $X \oplus Y$. We let $|X|$ denote the length of a variable X in bits; for a bit string X that is processed in units of blocks, we write X_i for the i-th block of X. For $X \in \{0,1\}^n$ and $i \leq n$, we denote by $\mathsf{msb}_i(X)$ the i leftmost and by $\mathsf{lsb}_i(X)$ the i rightmost bits of X. To split a string into blocks of fixed maximal length, $(X_1, \ldots, X_x) \xleftarrow{n} X$ indicates that X is split into n-bit blocks i. e., $X_1 \,\|\, \ldots \,\|\, X_x = X$, and $|X_i| = n$ for $1 \leq i \leq x-1$, and $|X_x| \leq n$. For any $X \in \{0,1\}^{n+t}$, we denote by $(X_1, X_2) \xleftarrow{n,t} \mathcal{X}$ the splitting of X into $X_1 = \mathsf{msb}_n(X)$ and $X_2 = \mathsf{lsb}_t(X)$. We denote by $\langle x \rangle_n$ the encoding of a non-negative integer x into its n-bit representation. Moreover, we write $X \twoheadleftarrow \mathcal{X}$ to indicate that an element X is chosen uniformly at random from some given set $\mathcal{X}$.
A tweaked permutation $\widetilde{\pi} : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ with tweak set $\mathcal{T}$ defines a family of permutation over $\mathcal{X}$, i.e., for every $T \in \mathcal{T}$, $\widetilde{\pi}(T, \cdot)$ is a permutation over $\mathcal{X}$. Given three sets $\mathcal{T}$, $\mathcal{X}$, and $\mathcal{Y}$, we define $\mathsf{Func}(\mathcal{X}, \mathcal{Y}) =^{\text{def}} \{F | F : \mathcal{X} \to \mathcal{Y}\}$ for the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$. Moreover, we write $\widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{X}) =^{\text{def}} \{\widetilde{\pi} | \widetilde{\pi} : \mathcal{T} \times \mathcal{X} \to \mathcal{X}\}$ for the set of all tweaked permutations over $\mathcal{X}$ with associated tweak space $\mathcal{T}$. Given sets $\mathcal{X}$ and $\mathcal{Y}$, a uniform random function $\rho : \mathcal{X} \to \mathcal{Y}$ is a mapping of inputs $X \in \mathcal{X}$ independently from other inputs and uniformly at random to outputs $Y \in \mathcal{Y}$.

Adversaries. An adversary $\mathbf{A}$ is an efficient Turing machine that interacts with a given set of oracles that appear as black boxes to $\mathbf{A}$. We denote by $\mathbf{A}^{\mathcal{O}}$ the output of $\mathbf{A}$ after interacting with some oracle $\mathcal{O}$. We write $\Delta_{\mathbf{A}}(\mathcal{O}^1; \mathcal{O}^2) =^{\text{def}} |\Pr[\mathbf{A}^{\mathcal{O}^1} \Rightarrow 1] - \Pr[\mathbf{A}^{\mathcal{O}^2} \Rightarrow 1]|$ for the advantage of $\mathbf{A}$ to distinguish between oracles $\mathcal{O}^1$ and $\mathcal{O}^2$. All probabilities are defined over the random coins of the oracles if any. Following [MRV15], we consider

(mostly) information-theoretic adversaries that are restricted only by their maximal number of queries and blocks. Moreover, we assume that adversaries they never ask queries to which they already know the answer. As proposed, e.g. in [CS14], information-theoretic adversaries can be safely assumed to be deterministic. All our results can be transferred to the complexity-theoretic setting by restricting the adversaries by the time needed for the evaluation of all internal primitive calls. Note that for the transferred adversaries, the setting would assume deterministic adversaries.

Adversary Characteristics. To quantify the power of adversaries, we say that an adversary $\mathbf{A}$ for a notion X against a scheme Π is a (q, m, σ)-X adversary if $\mathbf{A}$ asks at most q queries to its oracles, each of at most m blocks, and of σ blocks in total. The block size will be given in the context. We will write $\mathbf{Adv}^{\mathrm{X}}_{\Pi}(q, m, \sigma)$ for the maximum over all (q, m, σ)-X adversaries on Π. For authenticated encryption, we will represent by m the maximum number of blocks of associated data and message combined; furthermore, σ is the maximum number of blocks over all associated data and messages combined.

Block Ciphers and Tweakable Block Ciphers. Let $\mathcal{B} = \{0,1\}^n$ be a block space for a fixed integer n. A TBC $\widetilde{E}$ with associated key space $\mathcal{K}$, tweak space $\mathcal{T}$, and message space $\mathcal{B}$ is a mapping $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$ s.t. for every key $K \in \mathcal{K}$ and tweak $T \in \mathcal{T}$, it holds that $\widetilde{E}(K, T, \cdot)$ is a permutation over $\mathcal{B}$. We often write $\widetilde{E}^T_K(\cdot)$ as short form of $\widetilde{E}(K, T, \cdot)$.

Definition 1 (TPRP Advantage)**.** Let $\mathcal{K}$ be a non-empty finite set and $\mathcal{B}$, and $\mathcal{T}$ be message and tweak space, respectively. Let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$ denote a tweakable block cipher. Let $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{B})$ and $K \twoheadleftarrow \mathcal{K}$. Then, the TPRP advantage of an adversary $\mathbf{A}$ w.r.t. $\widetilde{E}$ is defined as $\mathbf{Adv}^{\mathrm{TPRP}}_{\widetilde{E}}(\mathbf{A}) =^{\mathrm{def}} \Delta_{\mathbf{A}}(\widetilde{E}_K; \widetilde{\pi})$.

PRFs and Universal Hashing. For the remaining definitions in this section, let $\mathcal{K}$, $\mathcal{X}$, and $\mathcal{Y}$ be non-empty finite sets. We restrict our considerations to $\mathcal{Y} \subseteq \{0,1\}^*$, and let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be keyed functions.

Definition 2 (PRF Advantage)**.** Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function. Let $\rho \twoheadleftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})$ and $K \twoheadleftarrow \mathcal{K}$. Then, the PRF advantage of $\mathbf{A}$ w.r.t. F is defined as $\mathbf{Adv}^{\mathrm{PRF}}_{F}(\mathbf{A}) =^{\mathrm{def}} \Delta_{\mathbf{A}}(F_K; \rho)$.

Definition 3 (Almost-Universal Hash Function)**.** Let H be defined as above. We call H ϵ-almost-universal (ϵ-AU) if for all distinct $X, X' \in \mathcal{X}$, it holds that $\Pr_{K \twoheadleftarrow \mathcal{K}}[H_K(X) = H_K(X')] \leq \epsilon$.

The collision probability is strongly related to almost-universality. Here, we define it in the context of messages that are composed of blocks of bit strings.

Definition 4 (Collision Probability between Message Pairs)**.** Let H be defined as above, with the restriction that $\mathcal{X} \subseteq \mathcal{B}^*$ for some block space $\mathcal{B} = \{0,1\}^x$. So, inputs to H are from a set of blocks. For arbitrary distinct messages $X \in \mathcal{B}^m$ and $X' \in \mathcal{B}^{m'}$ for given integers m and m', we define the collision probability by the maximum

$$\mathsf{Coll}_H(x, m, m') \stackrel{\mathrm{def}}{=} \max_{X \neq X'} \Pr_{K \twoheadleftarrow \mathcal{K}} \left[H_K(X) = H_K(X')\right].$$

We overload the notation of the collision probability from message pairs to message sets.

Definition 5 (Collision Probability among Message Sets)**.** Let H be defined as above, with the restriction that $\mathcal{X} \subseteq \mathcal{B}^*$ for some block space $\mathcal{B} = \{0,1\}^x$. So, inputs to H are from a set of blocks. Let $\mathcal{M}$ denote a set of q pairwise distinct messages $X \in \mathcal{B}^{\leq m}$ of at

most m blocks each and σ blocks in total over all messages. Then, we define the collision probability of any collision of outputs between distinct messages from $\mathcal{M}$ as

$$\mathsf{Coll}_H(x,q,m,\sigma) \stackrel{\text{def}}{=} \Pr_{K \twoheadleftarrow \mathcal{K}}\left[\exists X, X' \in \mathcal{M}, X \neq X' : H_K(X) = H_K(X')\right].$$

Given a function that outputs tuples, we will consider the collision probability between certain parts of the output. This is captured by the notion of truncated almost universality.

Definition 6 (Truncated-AU Hash Function)**.** Let $\mathcal{Y} = \{0,1\}^{n_1} \times \{0,1\}^{n_2}$ for positive integers n_1, n_2. We say that H is (n_1, n_2, ϵ)-truncated-AU (tAU) if, for all distinct $X, X' \in \mathcal{X}$, it holds that

$$\sum_{\Delta \in \{0,1\}^{n_2}} \Pr_{K \twoheadleftarrow \mathcal{K}}\left[H_K(X) \oplus H_K(X') = (0^{n_1}, \Delta)\right] \leq \epsilon.$$

The H-Coefficient Technique. The H-coefficient technique is a proof approach due to Patarin [Pat08]. It assumes that the results of the interaction of an adversary $\mathbf{A}$ with its oracles are collected in a transcript τ. The task of $\mathbf{A}$ is to distinguish the real world $\mathcal{O}_{\text{real}}$ from the ideal world $\mathcal{O}_{\text{ideal}}$ given its transcript τ. The transcript is called *attainable* if the probability to obtain it in the ideal world is greater than zero. One assumes that $\mathbf{A}$ does not ask duplicate queries or queries prohibited by the game. Θ_{real} and Θ_{ideal} denote the distribution of transcripts in the real and the ideal world, respectively. Then, the fundamental Lemma of the H-coefficient technique, the proof to which is given in [CS14, Pat08], states for information-theoretic adversaries:

Lemma 1 (Fundamental Lemma of the H-coefficient Technique [Pat08])**.** Assume, the set of attainable transcripts can be partitioned into two disjoint sets GoodT and BadT. Further assume, there exist $\epsilon_1, \epsilon_2 \geq 0$ s.t. for any transcript $\tau \in$ GoodT, it holds that

$$\frac{\Pr\left[\Theta_{\text{real}} = \tau\right]}{\Pr\left[\Theta_{\text{ideal}} = \tau\right]} \geq 1 - \epsilon_1, \quad \text{and} \quad \Pr\left[\Theta_{\text{ideal}} \in \text{BadT}\right] \leq \epsilon_2.$$

Then, for all adversaries $\mathbf{A}$, it holds that $\Delta_{\mathbf{A}}(\mathcal{O}_{\text{real}}; \mathcal{O}_{\text{ideal}}) \leq \epsilon_1 + \epsilon_2$.

3 The DoveMAC Construction

Basic Definitions. Throughout this section, we denote by two positive integers n and t the block length and tweak length, respectively. We define a non-empty key set $\mathcal{K}$, a tweak space $\mathcal{T} = \{0,1\}^t$, block space $\mathcal{B} = \{0,1\}^n$, and a tweak-block space $\mathcal{S} = \mathcal{T} \times \mathcal{B}$.
Hereafter, we use a tweakable block cipher $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$ and a tweakable permutation $\widetilde{\pi} \in \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{B})$, i.e., $\widetilde{\pi}(T, \cdot)$ is a permutation over $\mathcal{B}$ for each $T \in \mathcal{T}$. For primitives with $t \neq n$, we define a padding function $\text{PAD} : \mathbb{N} \times \mathcal{B} \to \{0,1\}^t$, where the first parameter defines the output length t. If $t = n$, the input X to PAD is returned unchanged; if $t < n$, the input X is padded by appending $n - t$ zero bits to obtain a t-bit value that is XORed to T_i. In the case that $t > n$, the least significant $t - n$ bits of X are truncated instead.

Block. First, we define a mapping Block that will be used in our hash function DoveHash to process each $(t+n)$-bit block of the input.

Definition 7 (Block)**.** For all inputs $(X_{i-1}, Y_{i-1}) \in \mathcal{B}^2$, and $M_i \in \mathcal{S}$, $\text{Block}[\widetilde{\pi}] : \mathcal{B}^2 \times \mathcal{S} \to \mathcal{B}^2$ returns (X_i, Y_i), which is computed as

$$(T_i, I_i) \xleftarrow{t,n} M_i, \qquad X_i \leftarrow \widetilde{\pi}^{T_i \oplus \text{PAD}_t(X_{i-1})}(I_i \oplus Y_{i-1}), \qquad \text{and} \qquad Y_i \leftarrow X_i \oplus Y_{i-1}.$$

Hash-then-TBC. Given a keyed hash function $H : \mathcal{K} \times \{0,1\}^* \to \mathcal{S}$, and a tweakable block cipher $\widetilde{\pi} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$. The Hash-then-TBC construction HtTBC$[\widetilde{\pi}, H]$ hashes a message input $M \in \{0,1\}^*$ with $H_K(M)$. Thereupon, it maps the hash output $(X, Y) \in \mathcal{T} \times \mathcal{B}$ to a flexible number of output blocks $Z \in \mathcal{B}^d$ by using one part as tweak, and the other as state input to $\widetilde{\pi}$. The finalization had been defined as ZFin$^+[\widetilde{\pi}]$ from [LN17] as a more efficient finalization for ZMAC. The general version XORs a counter $\langle i-1 \rangle_t$ to derive the i-th tweak input to obtain n-bit security. We briefly recall the general definitions for Hash-then-TBC PRF HtTBC$[\widetilde{\pi}, H]$ and ZFin$^+$ from [LN17].

Definition 8 (ZFin$^+$)**.** Let d be a fixed positive integer. Then, for all inputs $(X, Y) \in \mathcal{T} \times \mathcal{B}$, ZFin$^+[\widetilde{\pi}] : \mathcal{T} \times \mathcal{B} \to \mathcal{B}^d$ outputs $Z = (Z_1, \ldots, Z_d)$ where $Z_i \leftarrow \widetilde{\pi}^{X \oplus \langle i-1 \rangle_t}(Y)$, for all $1 \le i \le d$.

Definition 9 (Hash-then-TBC)**.** Let d be a fixed positive integer, and $H : \mathcal{K} \times \{0,1\}^* \to \mathcal{S}$ be a keyed hash function. Then, for all $M \in \{0,1\}^*$ and $K \in \mathcal{K}$, we define HtTBC$[\widetilde{\pi}, H] : \mathcal{K} \times \mathcal{S}^* \to \mathcal{B}^d$ as HtTBC$[\widetilde{\pi}, H_K](M) =^{\text{def}}$ ZFin$^+[\widetilde{\pi}](H_K(M))$.

DoveHash. Given the general definitions of Block, we define the basic hash function DoveHash$[\widetilde{\pi}] : \mathcal{S}^* \to \mathcal{S}$ as on the right part of Algorithm 1. It assumes a constant initial value $(X_0, Y_0) \in \mathcal{B}^2$. Internally, the hash function splits the message into m blocks M_i of $t + n$ bits each and processes each block by Block$[\widetilde{\pi}]$:

$$(X_i, Y_i) \leftarrow \text{Block}[\widetilde{\pi}](X_{i-1}, Y_{i-1}, M_i), \text{ for all } 1 \le i \le m.$$

All values T_i input are summed up to $\Theta = \bigoplus_{i=1}^{m} T_i$, which is XORed to X_m to produce the partial hash output X; the other part of the hash output Y is identical to Y_m.

DoveMAC. Now, we can define the stateless deterministic PRF DoveMAC$[\widetilde{E}_{K_1,K_2}] : \{0,1\}^* \to \mathcal{B}^d$, as in Algorithm 1. It is an instance of the HtTBC paradigm using DoveHash$[\widetilde{E}_{K_1}]$ as hash function and ZFin$^+[\widetilde{E}_{K_2}]$ for finalization, where $K_1, K_2 \in \mathcal{K}$ are independent keys. DoveMAC fixes the number of output blocks to $d = 1$ and the initial value to $(X_0, Y_0) = (0^n, 0^n)$. A given input message $\underline{M} \in \{0,1\}^*$ is padded first to $M \leftarrow \underline{M} \,\|\, 10^*$, using a one-zero padding such that the output length is the smallest multiple of $(t + n)$ bits, which is realized in Encode. The padded message M is then hashed to $(X, Y) \leftarrow$ DoveHash$[\widetilde{E}_{K_1}](M)$; in DoveHash, the message is processed in $(t + n)$-bit blocks; in each block, t bits are used as tweak, and n bits as state input, until the padded message is fully processed. In the end, a checksum of all t-bit tweak inputs is XORed to the top lane. Finally, the hash output (X, Y) is used as tweak and state input to ZFin$^+[\widetilde{E}_{K_2}]$ to produce the tag, DoveMAC$[\widetilde{E}_{K_1,K_2}](M)$ computes Z, where

$$M \leftarrow \text{Encode}_{t+n}(\underline{M}), \ (X, Y) \leftarrow \text{DoveHash}[\widetilde{E}_{K_1}](M), \text{ and } Z \leftarrow \text{ZFin}^+[\widetilde{E}_{K_2}](X, Y).$$

Internally, DoveHash requires a state of $n + t + k$ bits for calls to the TBC, plus n bits for the lower lane, which yields $2n + t + k$ bits. Since one can load the second key at the end and can supply the tweak checksum appended to the message, DoveMAC requires at least $2n + t + k$ bits of memory.

Checksum. Setting the checksum $\Theta = \bigoplus_{i=1}^{m} T_i$ as part of the message from the outside is an implementation optimization to reduce the internal state. For maximal security, this approach should be used only if the integrity of the input can be guaranteed. This could be done in a secured processor if available on the platform. Otherwise, if this part of state is critical, the security of DoveMAC would reduce, although it would still be lower bounded by the birthday bound. If the absence of the checksum was the only modification, the security of DoveMAC would reduce to the complexity of finding a collision which occurs

Algorithm 1 Authentication of a message $\underline{M}$ with construction DoveMAC, using a tweakable block cipher $\widetilde{E}$ and $d = 1$.

function DoveMAC$[\widetilde{E}_{K_1,K_2}](\underline{M})$
  $M \leftarrow \textsc{Encode}_{t+n}(\underline{M})$
  $(X, Y) \leftarrow \textsc{DoveHash}[\widetilde{E}_{K_1}](M)$
  return $\textsc{ZFin}^+[\widetilde{E}_{K_2}](X, Y)$

function $\textsc{Encode}_x(\underline{M})$
  $M \leftarrow \underline{M} \,\|\, 1$
  if $|M| \bmod x = 0$ **then**
    return M
  return $M \,\|\, 0^{x-(|M| \bmod x)}$

function $\textsc{ZFin}^+[\widetilde{E}_{K_2}](X, Y)$
  for $i = 1$ **to** d **do** $Z_i \leftarrow \widetilde{E}_{K_2}^{X \oplus \langle i-1 \rangle_t}(Y)$
  return $(Z_1, \ldots, Z_d)$

function $\textsc{pad}_t(X)$
  $n \leftarrow |X|$
  if $n = t$ **then return** X
  else if $n < t$ **then return** $X \,\|\, 0^{t-n}$
  else if $n > t$ **then return** $\mathsf{msb}_t(X)$

function $\textsc{DoveHash}[\widetilde{E}_{K_1}](M)$
  $m \leftarrow |M|/(t + n)$
  $(M_1, \ldots, M_m) \xleftarrow{t+n} M$
  $(X_0, Y_0) \leftarrow (0^n, 0^n)$
  for $i \leftarrow 1$ **to** m **do**
    $(T_i, I_i) \xleftarrow{t,n} M_i$
    $(X_i, Y_i) \leftarrow \textsc{Block}[\widetilde{E}_{K_1}](X_{i-1}, Y_{i-1}, T_i, I_i)$
  $\Theta \leftarrow \bigoplus_{i=1}^{m} T_i$
  $X \leftarrow \textsc{pad}_t(X_m) \oplus \Theta$
  $Y \leftarrow Y_m$
  return (X, Y)

function $\textsc{Block}[\widetilde{E}_{K_1}](X_{i-1}, Y_{i-1}, T_i, I_i)$
  $U_i \leftarrow T_i \oplus \textsc{pad}_t(X_{i-1})$
  $S_i \leftarrow I_i \oplus Y_{i-1}$
  $X_i \leftarrow \widetilde{E}_{K_1}^{U_i}(S_i)$
  $Y_i \leftarrow X_i \oplus Y_{i-1}$
  return (X_i, Y_i)

only in the ideal world, which yields about $n/2$ bits of security. Such a birthday-bound attack on a variant that omitted the checksum can be found in Appendix A. Though, it is relatively easy to see that a significantly better attack is not possible.

A Single-Key Variant DoveMAC1k. A single-key variant can be defined in a straightforward manner at the price of a less conventional splitting of the message blocks. Using a domain space of $\mathcal{D}$, one can assume that the tweak space is $\mathcal{T}'$ and can be split into a usable space $\mathcal{T}$ and a domain space $\mathcal{D}$: $\mathcal{T}' = \mathcal{T} \times \mathcal{D}$. Define $\mathcal{D} = \{0,1\}^\delta$. A single bit that differs for the intermediate calls and for the calls in the finalization suffices. A domain separation with $\delta \geq 1$ bit for the domain could then use $\widetilde{\pi}^{D,U_i}(S_i)$ with the domain as, e.g.,

$$D = \begin{cases} \langle 0 \rangle_\delta & \text{for } \widetilde{\pi}^{D,U_i}(V_i), \\ \langle 2 \rangle_\delta & \text{for } \widetilde{\pi}^{D,X}(Y)\,. \end{cases}$$

If $\delta \geq 2$ bits are reserved for domain purposes, further domains could be used for an encryption scheme used in combination with DoveMAC.
On the downside, this approach must split the message into potentially unconventional pieces that can potentially conflict, e.g., with byte or register alignments. The advantages and disadvantages of reserving a few tweak bits for separating domains have to be taken into account depending on the considered use case. Our security results also apply to DoveMAC1k when t is replaced by $(t-1)$ in the bounds.

4 PRF Security

If DoveHash satisfies two criteria, the PRF security of DoveMAC can be derived similarly as that of HtTBC for single-block outputs. Prior, we replace $\widetilde{E}_{K_1}$ and $\widetilde{E}_{K_2}$ in DoveMAC by two independent random permutations $\widetilde{\pi}, \widetilde{\pi}' \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{B})$, respectively. We denote the resulting construction by DoveMAC$[\widetilde{\pi}, \widetilde{\pi}']$. Using a hybrid argument, the advantage to distinguish between both settings is at most $\mathbf{Adv}^{\mathrm{TPRP}}_{\widetilde{E}_K}(\mathbf{A}')$, where $\mathbf{A}'$ is a TPRP adversary on $\widetilde{E}_K$ that asks at most $\sigma + 2q$ queries and runs in time at most $O(\sigma + 2q)$.

Theorem 1 (PRF Security of DoveMAC)**.** Let $K_1, K_2 \twoheadleftarrow \mathcal{K}$ be independent keys. Let $\widetilde{\pi}, \widetilde{\pi}' \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{B})$. Let $\mathbf{A}$ be a PRF adversary on DoveMAC$[\widetilde{\pi}, \widetilde{\pi}']$ s.t. $\mathbf{A}$ asks at most q queries that consist of at most $m < 2^{n-2}$ $(t+n)$-bit blocks after padding each, and that sum to at most $\sigma < 2^{n-2}$ $(t+n)$-bit blocks in total. Then

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{DoveMAC}[\widetilde{\pi},\widetilde{\pi}']}(\mathbf{A}) \leq \frac{4\sigma}{2^n} + \frac{q^2m^2}{2^{2n}} + \frac{2q^2 + 4qm^2 + 4q^2m^2}{2^{n+\min(n,t)}}.$$

Remark 1. While we generalized DoveMAC to arbitrary tweak lengths, our analysis for the settings $t = n$ and $t > n$ often follow the same arguments since the outputs are simply truncated before they are XORed to the next tweak input. However, a larger tweak does not yield higher security, but only increases the rate. When $t < n$, there are up to 2^{n-t} possible output values that would produce internal collision events, which reduces the bound to $2^{-(n+t)}$ in the exponent.

Proof. The proof of Theorem 1 follows from Theorem 2. The latter will use Lemmas 4 and 5 to derive concrete bounds. Theorem 2 is similar to [LN17, Corollary 1]. □

Theorem 2 (PRF Security of HtTBC)**.** Let H be short for DoveHash$[\widetilde{\pi}]$. Assume that the collision probability over all messages $\mathsf{Coll}_{H[\widetilde{\pi}]}(t+n, q, m, \sigma)$ is upper bounded by ϵ_1, and H is (t, n, ϵ_2)-tAU. Let $\mathbf{A}$ be a PRF adversary against HtTBC$[\widetilde{\pi}', H]$ that makes at most q queries consisting at most m $(t+n)$-bit blocks after padding each, that sum to at most σ $(t+n)$-bit blocks in total. Then

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathrm{HtTBC}[\widetilde{\pi}',H]}(\mathbf{A}) \leq \epsilon_1 + \frac{\binom{q}{2} \cdot \epsilon_2}{2^n}.$$

Proof. The queries by $\mathbf{A}$ are collected in a transcript τ that contains the messages from $\mathbf{A}$ as well as the outputs $\{(\underline{M}^i, X^i, Y^i, Z^i)\}_{1 \leq i \leq q})$, as well as the ideal primitives $\widetilde{\pi}$ and $\widetilde{\pi}'$. $\underline{M}^i$ denotes the i-th message, X^i and Y^i the inputs to the final call to $\widetilde{\pi}'$, and Z^i the tags. Moreover, we define the length of $\underline{M}^i$ after padding by m^i.
Both the real and the ideal worlds have an on-line and an off-line sampling phase. In the on-line phase, the real world computes the tags Z^i. The ideal world maps inputs $\underline{M}^i \in \mathcal{M}$ to uniformly random outputs $Z^i \twoheadleftarrow \mathcal{B}$.
In the off-line phase, the real world releases all internal values X^i and Y^i and $\widetilde{\pi}$. The ideal world samples $\widetilde{\pi}$ to derive those internal values X^i and Y^i in this phase and releases $\widetilde{\pi}$, X^i, and Y^i. Those parts of the transcript are revealed to the adversary after it made all its queries, but before it outputs its decision bit that represents its guess of which world it interacted with. The task of $\mathbf{A}$ is then to distinguish the real world $\mathcal{O}_{\mathrm{real}}$ from the ideal world $\mathcal{O}_{\mathrm{ideal}}$. A transcript τ is called *attainable* if the probability to obtain τ in the ideal world is non-zero.
The set of all attainable transcripts can be partitioned into two disjoint sets GoodT and BadT. We call a transcript τ bad iff $\tau \in$ BadT, and denote it as good otherwise. A transcript is called bad if at least one of the following statements holds:

- bad_1: There exist distinct $i, j \in \{1, \ldots, q\}$ s.t. $(X^i, Y^i) = (X^j, Y^j)$.
- bad_2: There exist distinct $i, j \in \{1, \ldots, q\}$ s.t. $(X^i, Z^i) = (X^j, Z^j)$. We condition the event bad_2 to exclude bad_1.

The proof of Theorem follows then from Lemmas 2 and 3 below. □

The bad events represent possible input or output collisions in the finalization: bad_1 models the event that two pairs of state and tweak inputs collide; bad_2 indicates a collision between tweak and outputs. For all bad events, the adversary could easily distinguish the worlds. However, their probability to occur is sufficiently small as is studied in Lemma 2.

Lemma 2 (Bad Transcripts)**.** Given the considerations from Theorem 2, and the bad events as defined above. Then

$$\Pr\left[\Theta_{\text{ideal}} \in \textsc{BadT}\right] \leq \epsilon_1 + \frac{\binom{q}{2} \cdot \epsilon_2}{2^n} .$$

Proof. It holds that $\Pr[\Theta_{\text{ideal}} \in \textsc{BadT}] \leq \Pr\left[\mathsf{bad}_1\right] + \Pr\left[\mathsf{bad}_2 | \neg \mathsf{bad}_1\right]$. We upper bound the probability of those bad events in the following.

Bad_1. In this case, it holds that there exists at least one tuple of distinct $i, j \in [1..q]$, s.t. $\underline{M}^i$ and $\underline{M}^j$ yielded $(X^i, Y^i) = (X^j, Y^j)$. So, the outputs would have to be equal, and with high probability, $\mathbf{A}$ could distinguish the worlds. This probability is at most

$$\Pr[\mathsf{bad}_1] \leq \mathsf{Coll}_{H[\widetilde{\pi}]}(t + n, q, m, \sigma) \leq \epsilon_1 .$$

Bad_2. In this case, it holds that $\underline{M}^i$ and $\underline{M}^j$ produced $(X^i, Z^i) = (X^j, Z^j)$. In the ideal world, the outputs Z^i and Z^j are sampled independently and uniformly at random. Given that $H[\widetilde{\pi}]$ is (t, n, ϵ_2)-tAU, the probability that $X^i = X^j$, for a fixed pair of i and j, is bounded by ϵ_2. Over q queries, it follows that

$$\Pr\left[\mathsf{bad}_2\right] \leq \binom{q}{2} \cdot \frac{\epsilon_2}{2^n}$$

Our claim in Lemma 2 follows from the sum of both terms.
Note that we can simplify the bound for bad_2 that will be treated in Lemma 5, which captures also the term for a collision between two queries of at most m $(t + n)$-bit blocks. We could generalize it over all queries to another term of the collision bound for q queries of at most σ blocks in total. However, this would treat collisions twice. Considering

$$\Pr\left[\mathsf{bad}\right] \leq \Pr\left[\mathsf{bad}_1\right] + \Pr\left[\mathsf{bad}_2 | \neg \mathsf{bad}_1\right]$$

avoids the duplicate term for the collision bound. □

The finalization of DoveMAC is identical to $\textsc{HtTBC}[\widetilde{\pi}', H]$ for single-block outputs. Hence, we can apply the following lemma whose proof is given in Lemma 3 in [LN17]; for the sake of completeness, we sketch it in Appendix B.

Lemma 3 (Interpolation Probability of Good Transcripts)**.** Given the considerations from Theorem 2, and the definition of good and bad transcripts as from its proof. Let $\tau \in \textsc{GoodT}$. Then, $\Pr\left[\Theta_{\text{real}} = \tau\right] \geq \Pr\left[\Theta_{\text{ideal}} = \tau\right]$.

The remaining analysis reduces to finding upper bounds for ϵ_1 and ϵ_2. We study these properties in the upcoming Sections 5 and 6, respectively.

5 Collision Analysis

We will show the following lemma in this section.

Lemma 4 (Collision Probability of $\textsc{DoveHash}[\widetilde{\pi}]$)**.** Let $\sigma < 2^{n-2}$. Then, it holds that

$$\mathsf{Coll}_{\textsc{DoveHash}[\widetilde{\pi}]}(t + n, q, m, \sigma) \leq \frac{4\sigma}{2^n} + \frac{4qm^2 + 4q^2m^2}{2^{n+\min(n,t)}} .$$

For our analysis of the collision probability, we need the definition of the longest common prefix between two messages.

Definition 10 (Longest Common Prefix)**.** Given an integer n, let $M, M' \in \mathcal{S}^*$, with $\mathcal{S} = \{0,1\}^{t+n}$, and denote their blocks as $(M_1, \ldots, M_m) \xleftarrow{t+n} M$ and $(M'_1, \ldots, M'_{m'}) \xleftarrow{t+n} M'$. We define the length of their longest common prefix $\mathsf{LCP}_{\mathcal{S}}$ as $\mathsf{LCP}_{\mathcal{S}}(M, M') =^{\text{def}} \max_j\{1 \leq i \leq j : M_i = M'_i\}$, and will write $\mathsf{LCP}_{t+n}(M, M')$ hereafter. Given a set $\mathcal{Q} \subseteq (\mathcal{S}^*)^*$ of elements from $\mathcal{S}^*$, we define $\mathsf{LCP}(M, \mathcal{Q}) =^{\text{def}} \max_{M' \in \mathcal{Q}} \mathsf{LCP}(M, M')$.

Proof of Lemma 4. In the following, we consider q queries M^i for $i \in [1..q]$ that are collected in a transcript τ. Most of the time, however, we study the probability of events for a single message M^i, that we will denote as M for simplicity or the probability of a collision for two distinct messages M^i and M^j. To reduce the number of indices, we will simply call them M and M' where possible.
Given two distinct messages M, M', we denote the blocks of $M = (M_1, \ldots, M_m)$, and the corresponding intermediate values X_i, Y_i, S_i, T_i, and I_i, for $1 \leq i \leq m$, as well as the blocks of $M' = (M'_1, \ldots, M'_m)$, and its intermediate values X'_i, Y'_i, etc, for $1 \leq i \leq m'$, in the intuitive way. Note that the values U_i can be derived, given T_i and X_{i-1}. We consider $m \geq m'$; the analysis of the case $m \leq m'$ is analogous.
We further define $p =^{\text{def}} \mathsf{LCP}_{t+n}(M, M')$ for the length of the longest common prefix between M and M'. If $m = m'$, it holds that $p < m$ and $M_{p+1} \neq M'_{p+1}$; otherwise, it holds that $p \leq m$ if $m > m'$. Each block M_i or M'_i represents the concatenation of a tuple $(T_i, I_i) \xleftarrow{n} M_i$ and $(T'_i, I'_i) \xleftarrow{n} M'_i$ after padding, which is XORed with the previous state to yield tweak and input of the current block: $(U_i, S_i) = (T_i, I_i) \oplus (\textsc{pad}_t(X_{i-1}), Y_{i-1})$ and $(U'_{i'}, S'_{i'}) = (T'_{i'}, I'_{i'}) \oplus (\textsc{pad}_t(X'_{i'-1}), Y'_{i'-1})$, respectively, for all $1 \leq i \leq m$ and $1 \leq i' \leq m'$. We will often denote by Δ the XOR difference between corresponding blocks of the two messages. For instance, $\Delta X_i = X_i \oplus X'_i$, $\Delta\Theta = \Theta \oplus \Theta'$, and so on.

Non-trivial Input Collisions. We say that a tweak-input tuple (U_i, S_i) beyond the common prefix (i.e. $i > p$) is **old** iff

- There exists $j \in \{0, \ldots, m\}$, $j < i$, s.t. $(U_i, S_i) = (U_j, S_j)$ and/or
- There exist $j \in \{p+1, \ldots, m'\}$, s.t. $(U_i, S_i) = (U'_j, S'_j)$.

We call such an event a **non-trivial tweak-input collision**, whereas a trivial tweak-input collision is a collision of $(U_i, S_i) = (U'_i, S'_i)$ for all $i \in \{0, \ldots, p\}$. We call a tweak-input tuple (U_i, S_i) **fresh** iff it is not old. We extend the definition of freshness to tweak-input tuples of M', $(U'_{i'}, S'_{i'})$, in the natural manner.

Structure Graphs. We briefly introduce a tool called structure graph from [BPR05, JMN17, JN16, Pie06]. We follow the description from [JMN17].

Directed Graphs. We define **directed unlabeled graphs** $G = (\mathcal{V}, \mathcal{E})$ of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ between them. Moreover, we define **directed edge-labeled graphs** $G_L = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ with $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{L}$, where $\mathcal{L}$ denotes a set of labels corresponding to edges. Here, we consider (u, v, ℓ) or $u \xrightarrow{\ell} v$ as a directed edge between $u, v \in \mathcal{V}$ and label $\ell \in \mathcal{L}$. For an edge (u, v), u is called the predecessor of v. Analogously, v is called the successor of u.

Walks. A **walk** $\mathbf{v}$ is a vertex sequence $\mathbf{v} = (v_0, v_1, \ldots, v_m)$ s.t. $v_{i-1} \to v_i$ for all $i \in [1..m]$. If all vertices of a walk are pairwise distinct, $\mathbf{v}$ is called a **path**. If all vertices $v_1, \ldots, v_m$ are pairwise distinct and $v_0 = v_m$, it is called a **cycle**. If there exist $i < j$ s.t. $v_i = v_j$, then, $\mathbf{v}_{i..j}$ is called to contain a **loop**. We denote a partial sequence of $\mathbf{v}_{i..j} \subseteq \mathbf{v}$ as $\mathbf{v}_{i..j} =^{\text{def}} (v_i, \ldots, v_j)$. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{L}_2)$ be two directed edge-labeled graphs. A function $\varphi : \mathcal{V}_1 \to \mathcal{V}_2$ is an isomorphism between G_1 and G_2, written

$G_1 \cong G_2$, iff α is bijective and $(u, v, \ell) \in \mathcal{E}_1$ iff $(\alpha(u), \alpha(v), \ell) \in \mathcal{E}_2$. G_2 is called the α-transformed graph of G_1, denoted as $G_2 = \alpha(G_1)$. Note that if α is injective, the range of α can be restricted to its domain, which yields a bijective α.

Function Graphs. A directed edge-labeled graph $G_L = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ is called a **function graph** if for all vertices $u \in \mathcal{V}$ and all labels $\ell \in \mathcal{L}$, there exists at most one successor $v \in \mathcal{V}_S(u)$ s.t. (u, v) has label ℓ. This definition is extendable to walks. If there is a walk $\mathbf{v}$ with labels (in that order) $\ell_1, \ldots, \ell_m$, then the walk must be unique.

Intermediate Inputs and States. Let $\mathcal{M} = (M^1, \ldots, M^q)$ be a tuple of q pairwise distinct messages $M^i \in \mathcal{B}^{m_i}$, for $1 \le m_i \le m$ and $1 \le i \le q$ and $\sum_{i=1}^{q} m_i \le \sigma$. Moreover, we consider that $M^i = (M^i_1, \ldots, M^i_{m_i})$ are sequences of message blocks M^i_j, for $1 \le j \le m_i$. For our purpose, we denote by in as the domain of intermediate inputs and state as the domain of intermediate states. For our purpose, we will use $\mathsf{in} = \mathcal{T} \times \mathcal{B}$ and $\mathsf{state} = \mathcal{B} \times \mathcal{B}$. We will use $\mathsf{in}(i, j) = (U^i_j, S^i_j)$ to refer to intermediate inputs and $\mathsf{state}(i, j) = (X^i_j, Y^i_j)$ for intermediate states. Sequences of inputs and states are denoted by in and state, respectively. A sequence I is attainable if there exists a function F s.t. $I = \mathsf{in}(F, \mathcal{M})$. We consider functions $F \in \mathsf{Func}(\mathcal{T} \times \mathcal{B}, \mathcal{B})$.

Block-Vertex Input-Structure Graphs. A **block-vertex input-structure graph** G_L for a function F and a message tuple $\mathcal{M}$ is defined by its set of labeled edges

$$\mathcal{E} \stackrel{\text{def}}{=} \bigcup_{i=1}^{q} \{(\mathsf{in}(i, j-1), \mathsf{in}(i, j), M^i_j) : i \in [1..m_i]\}\,.$$

Thus, it is a graph-theoretic representation of the intermediate values. The vertices are inputs (U^i_j, S^i_j) to the tweakable block cipher, i.e., $\mathcal{V} = \mathcal{T} \times \mathcal{B}$; edges are transitions from one permutation state (U^i_{j-1}, S^i_{j-1}) to the next one (U^i_j, S^i_j). The labels are the message blocks $M^i_j = (T^i_j, I^i_j)$, which implies $\mathcal{L} = \mathcal{T} \times \mathcal{B}$:

$$(U^i_{j-1}, S^i_{j-1}) \xrightarrow{T^i_j, I^i_j} (U^i_j, S^i_j)\,.$$

For our purpose, the transition will be defined as

$$\begin{aligned} U^i_j &\stackrel{\text{def}}{=} \mathrm{PAD}_t\left(\widetilde{\pi}\left(U^i_{j-1}, S^i_{j-1}\right)\right) \oplus T^i_j \\ S^i_j &\stackrel{\text{def}}{=} \widetilde{\pi}\left(U^i_{j-1}, S^i_{j-1}\right) \oplus I^i_j \oplus Y^i_{j-1}\,. \end{aligned}$$

Note that we have to implicitly keep track of the values $Y^i_j = \bigoplus_{k=0}^{j-1} Y^i_k$. Given the sequence of predecessors, Y^i_{j-1} is uniquely determined; however, the notation would become unhandy. To address this issue, one could consider an isomorphic structure-graph representation instead.

Block-Vertex State-Structure Graphs. We consider a **block-vertex state-structure graph** G^S_L for a function F and a message tuple $\mathcal{M}$ is defined by its set of labeled edges

$$\mathcal{E} \stackrel{\text{def}}{=} \bigcup_{i=1}^{q} \{(\mathsf{state}(i, j-1), \mathsf{state}(i, j), M^i_j) : i \in [1..m_i]\}\,.$$

The vertices is the intermediate state (X^i_j, Y^i_j) the tweakable block cipher, i.e., $\mathcal{V} = \mathcal{T} \times \mathcal{B}$; the edges are transitions from one permutation state (X^i_{j-1}, Y^i_{j-1}) to the subsequent one

(X_j^i, Y_j^i). The labels are again the message blocks $M_j^i = (T_j^i, I_j^i)$, which implies that $\mathcal{L} = \mathcal{T} \times \mathcal{B}$:

$$\left(X_{j-1}^i, Y_{j-1}^i\right) \xrightarrow{T_j^i, I_j^i} \left(X_j^i, Y_j^i\right) ,$$

where the transition is defined as

$$\begin{aligned} X_j^i &\stackrel{\text{def}}{=} \widetilde{\pi}(\text{PAD}_t(X_{j-1}^i \oplus T_j^i, Y_{j-1}^i \oplus I_j^i)) \\ Y_j^i &\stackrel{\text{def}}{=} X_j^i \oplus Y_{j-1}^i . \end{aligned}$$

G_L^S is a union of all M^i-walks, for $M^i \in \mathcal{M}$ and $1 \leq i \leq q$. Both G_L and G_L^S are function graphs since for every vertex $u \in \mathcal{V}$, all outgoing and ingoing edges have distinct labels. Moreover, each walk is unique. Note that $\mathbf{A}$ can construct the structure graph from the information in the transcript τ. For a given message M, the structure graph $G_L(M)$ is isomorphic to $\mathbf{v}(M)$, the walk of M. For multiple messages M, M', ..., the graph $G_L(M, M', \ldots)$ represents the union of the structure graphs isomorphic to the walks of the considered messages. Over all messages of a tuple $\mathcal{M}$, $G_L(\mathcal{M})$ is isomorphic to the union of all walks $\mathbf{v}$, $\mathbf{v}'$, and so on.

Partial Walks and Structure Graphs. For our purpose, it will be sufficient to consider partial walks and partial structure graphs. In the remainder, we use $G_L(M^i, M^j)$ for the partial structure graph from the union of $G_L(M^i)$ and $G_L(M^j)$ that stops when the first loop or non-trivial state collision in M^i, M^j or between them occurs.

Core Idea. Let Coll be short-hand for the event $\mathsf{Coll}_{\text{DoveHash}[\widetilde{\pi}]}(t+n, q, m, \sigma)$. Let $\mathsf{Coll}(M^i, M^j)$ denote the event that the walks of M^i and M^j collide. So, we can bound

$$\Pr[\mathsf{Coll}] \leq \sum_{i<j} \Pr\left[\mathsf{Coll}(M^i, M^j)\right] .$$

In the following, we can concentrate on upper bounding the collision in the graphs between two messages. To reduce the number of used indices, we name them M and M'. We consider the labeled structure graph $G_L(M, M')$ of M and M' until (and including) their first non-trivial collision:

$$\Pr[\mathsf{Coll}(M, M')] \leq \sum_{G_L(M,M')} \Pr[\mathsf{Coll}(M, M') \wedge G_L(M, M')] . \tag{1}$$

Let (v_{j-1}, v_j) denote an edge from $G_L(M)$ and $(v'_{j'-1}, v'_{j'})$ an edge originally from $G_L(M')$, s.t. $v_j = v'_{j'}$ denotes the first non-trivial collision of their union graph $G_L(M, M')$. We define by $\widehat{G}_L(M, M') = (\mathcal{V}', \mathcal{E}')$ the subgraph of $G_L(M, M')$ induced by removing this first non-trivial collision, and by $\mathsf{Coll}_{j,j'}(M, M')$ the event that the subgraph $G_L(M, M')$ collided at blocks j of M and j' of M'. Note that $\widehat{G}_L(M, M')$ is determined uniquely from $G_L(M, M')$; hence, summing over all graphs $G_L(M, M')$ is equivalent to summing over all graphs $\widehat{G}_L(M, M')$. Later, we will have to determine the probability of the condition event $\mathsf{Coll}_{j,j'}(M, M')$. Thus, Equation (1) can be reformulated as

$$\begin{aligned} &\sum_{\widehat{G}_L(M,M')} \Pr\left[\mathsf{Coll}_{j,j'}(M, M') \wedge \widehat{G}_L(M, M')\right] \\ &= \sum_{\widehat{G}_L(M,M')} \Pr\left[\widehat{G}_L(M, M')\right] \cdot \Pr\left[\mathsf{Coll}_{j,j'}(M, M') \,\middle|\, \widehat{G}_L(M, M')\right] . \end{aligned} \tag{2}$$

We will distinguish between two types of graphs:

- **Bad graphs:** We call a graph $\widehat{G}_L(M, M')$ bad iff it contains a loop. Since a loop occurs in the walks of the same message, this is equivalent to the event that at least one of the structure graphs for the individual messages $\widehat{G}_L(M)$ or $\widehat{G}_L(M')$ contains a loop.
- **Good graphs:** The graph $\widehat{G}_L(M, M')$ contains no loop.

We define GoodG for the set of good partial structure graphs and BadG for the set of bad partial structure graphs. We will bound

$$\Pr\left[\mathsf{Coll}(M, M') \wedge \widehat{G}(M, M')\right] \leq \Pr\left[\left(\widehat{G}_L(M) \in \mathsf{BadG}\right) \vee \left(\widehat{G}_L(M') \in \mathsf{BadG}\right)\right] + \Pr\left[\mathsf{Coll}_{j,j'}(M, M') \wedge \widehat{G}_L(M, M') \in \mathsf{GoodG}\right].$$

To upper bound $\Pr[\mathsf{Coll}_{j,j'}(M, M')]$ for good graphs, we will later be able to use the entropy of independent values X_i, X_k from M and $X'_{i'}$, $X'_{k'}$ from M', for some indices $i, k \in [1..m]$ and $i', k' \in [1..m']$, that have not yet been inputs to the permutation π before. Thus, their corresponding outputs that lead to a collision in the subsequent permutation inputs are drawn from the set of all not yet fixed values of the permutation π. Since at most $2m$ elements have been fixed before, the probability for them to lead to a collision can then be upper bounded by

$$\Pr\left[\mathsf{Coll}(M, M') \wedge \widehat{G}_L(M, M') \wedge \widehat{G}_L(M, M') \in \mathsf{GoodG}\right] \leq \frac{2^{\max(0,n-t)}}{(2^n - 2m)^2}.$$

It follows that, for good graphs, Equation (2) can be computed as

$$\begin{aligned}&\sum_{\widehat{G}_L(M,M')} \Pr\left[\widehat{G}_L(M, M')\right] \cdot \Pr\left[\mathsf{Coll}_{j,j'}(M, M') \wedge \widehat{G}_L(M, M')\right] \\ &= \sum_{\widehat{G}_L(M,M')} \Pr\left[\widehat{G}_L(M, M')\right] \cdot \frac{2^{\max(0,n-t)}}{(2^n - 2m)^2} = \frac{2^{\max(0,n-t)}}{(2^n - 2m)^2}.\end{aligned}$$

The bad partial graphs cover all those that do not yield two independent variables. For those bad graphs, we will show instead that their number can be upper bounded by a reasonably "small" amount. Over all q queries, it follows then

$$\sum_{i<j}^{q} \Pr\left[\mathsf{Coll}(M^i, M^j)\right] \leq \sum_{i<j}^{q} \Pr\left[\mathsf{Coll}(M^i, M^j) \wedge \widehat{G}_L(M^i, M^j) \in \mathsf{GoodG}\right] + \sum_{i=1}^{q} \Pr\left[\widehat{G}_L(M^i) \in \mathsf{BadG}\right].$$

Bad Graphs. We consider four cases of bad partial structure graphs with loops:

- **Bad$_1$:** The length of the loop, r, is a single block, i.e., there exists $i \in [1..m-1]$ s.t. $(U_{i+1}, S_{i+1}) = (U_i, S_i)$.
- **Bad$_2$:** The length of the loop is a single block, i.e., $r = 1$, and loops with the begin, i.e.,$(U_2, S_2) = (U_1, S_1)$.
- **Bad$_3$:** The loop is longer than a single block, i.e., there exist distinct $i < j \in [1..m]$ s.t. $(U_j, S_j) = (U_i, S_i)$ and $j \geq i + 2$.

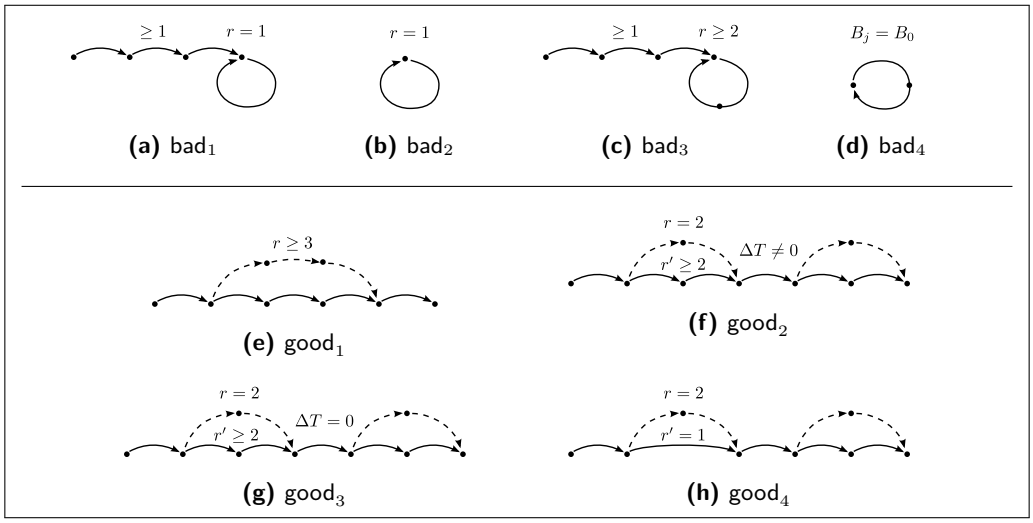

Figure 2: Structure graphs that visualize the bad graphs of message M (top), and the good structure graphs of messages M and M' (bottom).

- **Bad$_4$:** The loop is longer than a single block and collides with the initial value, i.e., there exists $i \in [3..m]$ s.t. $(U_i, S_i) = (U_1, S_1)$.

In the following, we investigate the probability of a loop in the individual cases of graphs. W.l.o.g., we consider the case that the loop is contained in the graph of M and consider $m > m'$. For each, we will distinguish between the settings where $t = n$, $t > n$, and $t < n$.

Bad$_1$. In this case, the graph of M contains a loop of length one. First, assume the Setting $t = n$. For a fixed index i, we investigate the probability that

$$\Pr\begin{bmatrix} U_{i+1} = U_i \\ S_{i+1} = S_i \end{bmatrix} = \Pr\begin{bmatrix} X_i = T_i \oplus T_{i+1} \\ X_i = X_{i+1} \oplus I_i \oplus I_{i+1} \end{bmatrix} \leq \frac{1}{2^n - (i-1)}\,.$$

Since there exist at most m blocks, the number of cases is upper bounded by

$$\frac{m}{2^n - m}.$$

The probability to fulfill the lower equation does not increase with smaller or longer tweaks. Therefore, it is equal the settings where $t > n$ and $t < n$.

Bad$_2$. In this case, the second input tuple collides with the first one. First, assume the Setting $t = n$. For a fixed index i, we investigate the probability that

$$\Pr\begin{bmatrix} U_2 = U_1 \\ S_2 = S_1 \end{bmatrix} = \Pr\begin{bmatrix} X_1 = X_0 \oplus T_1 \oplus T_2 \\ X_1 = Y_0 \oplus I_1 \oplus I_2 \end{bmatrix} \leq \frac{1}{2^n - (i-1)}\,.$$

Since there exist at most m blocks, the number of cases is upper bounded by

$$\frac{m}{2^n - m}.$$

Again, the probability to fulfill the lower equation does not increase with smaller or longer tweaks. Therefore, it is equal the settings where $t > n$ and $t < n$.

Bad$_3$. In this case, the graph of M contains a loop of length at least 2.
First, consider the Setting $t = n$. For a fixed index i, we investigate the probability that

$$\Pr\begin{bmatrix} U_i = U_j \\ S_i = S_j \end{bmatrix} = \Pr\begin{bmatrix} X_{i-1} \oplus X_{j-1} = T_i \oplus T_j \\ \bigoplus_{s=i}^{j-1} X_i = I_i \oplus I_j \end{bmatrix}. \tag{3}$$

Since we consider the first loop, all previous blocks do not form a loop. Consequently, the blocks (U_{j-1}, S_{j-1}), (U_{j-2}, S_{j-2}) are fresh and their corresponding outputs X_{j-1} and X_{j-2} are chosen randomly from a set of size at least $2^n - (j-1)$ each. Since there exist at most m blocks, the probability in this case is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - m)^2}.$$

In the Setting $t \neq n$, the top equality from Equation (3) becomes

$$\Pr\begin{bmatrix} U_i = U_j \\ S_i = S_j \end{bmatrix} = \Pr\begin{bmatrix} \mathrm{PAD}_t(X_{i-1} \oplus X_{j-1}) = T_i \oplus T_j \\ \bigoplus_{s=i}^{j-1} X_i = I_i \oplus I_j \end{bmatrix}. \tag{4}$$

Isolating the outputs X_{j-1} and X_{j-2} as in the setting $t = n$ yields that the probability that X_{j-1} fulfills it is $1/(2^n - m)$ as before. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{j-2} that can fulfill the top equality of Equation (4). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - m)^2}.$$

Bad$_4$. In this case, it holds that $(U_i, S_i) = (U_1, S_1)$, where $i > 2$.
Again, let us start with the Setting $t = n$. For a fixed index i, we investigate the probability that

$$\Pr\begin{bmatrix} U_i = U_1 \\ S_i = S_1 \end{bmatrix} = \Pr\begin{bmatrix} X_{i-1} = X_0 \oplus T_i \oplus T_1 \\ \bigoplus_{j=1}^{i-1} X_i = Y_0 \oplus I_i \oplus I_j \end{bmatrix}. \tag{5}$$

Since we consider the first loop, all previous blocks do not form a loop and are fresh. So, X_{i-1} and X_{i-2} are fresh and chosen randomly from a set of size at least $2^n - (i-1)$ each. Since there exist at most m blocks, the probability in this case is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - m)^2}.$$

In the Setting $t \neq n$, we can rewrite Equation (5) as

$$\Pr\begin{bmatrix} \mathrm{PAD}_t(X_{i-1}) = \mathrm{PAD}_t(X_0) \oplus T_i \oplus T_1 \\ \bigoplus_{j=1}^{i-1} X_i = Y_0 \oplus I_i \oplus I_j \end{bmatrix}. \tag{6}$$

We isolated the outputs X_{i-1} and X_{i-2} as in the setting $t = n$. The probability that X_{i-2} fulfills the bottom equality is $1/(2^n - m)$ as before. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{i-1} that can fulfill the top equality of Equation (6). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - m)^2}.$$

There exist at most q bad graphs. So, the probability for a bad graph is upper bounded by

$$\Pr\left[G'_L(M) \in \mathsf{BadG} \vee G'_L(M')\right] \leq \sum_{i=1}^{4} \Pr\left[\mathsf{bad}_i\right] \leq \sum_{i=1}^{q} 2 \cdot \frac{m}{2^n - \sigma} + 2 \cdot \frac{2^{\max(0,n-t)} \cdot \binom{m}{2}}{(2^n - \sigma)^2}$$
$$\leq \frac{2\sigma}{2^n - \sigma} + \frac{2^{\max(0,n-t)} \cdot 2q \cdot \binom{m}{2}}{(2^n - \sigma)^2} \leq \frac{4\sigma}{2^n} + \frac{4qm^2}{2^{n+\min(n,t)}},$$

using the fact that $\sigma < 2^{n-2}$.

Good Graphs. It remains to bound the number of collisions in good graphs. We denote by r the minimum distance of blocks from the block p where M diverges from M' after their longest common prefix until the first collision. This means, $r = \min_{i>1} |\{i : (U_{i+p}, S_{i+p}) \in \mathbf{v}'\}|$. Analogously, we define the distance of blocks from the block p where M' diverges from M after their longest common prefix until the first collision: $r' = \min_{i>1} |\{i : (U'_{i+p}, S'_{i+p}) \in \mathbf{v}\}|$. The values r and r' do not have to be equal. Note that, for the block directly following the longest common prefix, the inputs to π must differ by definition, i.e., $(U_{p+1}, S_{p+1}) \neq (U'_{p+1}, S'_{p+1})$. So, it must hold that at least one of r and r' has length at least 2: $(U_i, S_i) = (U'_{i'}, S'_{i'})$, $i = p + r$.
We consider the following mutually exclusive cases:

- **Good$_1$:** $r \geq 3$, i.e., there exists i and j s.t. $(U_i, S_i) = (U'_j, S'_j)$ with $i \geq p + 3$.
- **Good$_2$:** $r = 2$, $r' \geq 2$, and $\Delta T \neq 0$.
- **Good$_3$:** $r = 2$, $r' \geq 2$, and $\Delta T = 0$.
- **Good$_4$:** $r = 2$ and $r' = 1$.

Recall that we consider the first collision. Since the graphs are good, they exclude loops.

Good$_1$. In this case, the graphs of M and M' diverge and converge again, where $r \geq 3$ blocks are between division and conversion in the graph of M.
We start again in the Setting $t = n$. For a fixed index i, we investigate the probability that

$$\Pr\begin{bmatrix} U_i = U'_j \\ S_i = S'_j \end{bmatrix} = \Pr\begin{bmatrix} X_{i-1} = X'_{j-1} \oplus T_i \oplus T_j \\ \bigoplus_{k=p+1}^{i-1} X_k \oplus \bigoplus_{k=p+1}^{j-1} X'_k = I_i \oplus I_{i+1} \end{bmatrix}. \tag{7}$$

We can isolate the blocks X_{i-1} and X_{i-2}. Since we consider the first collision at X_i and no loop, the inputs (U_{i-1}, S_{i-1}) and (U_{i-2}, S_{i-2}) that produced X_{i-1} and X_{i-2} are fresh. So, X_{i-1} and X_{i-2} are sampled randomly from sets of size at least $2^n - (i-1)$. Thus, the probability that they fulfill Equation (7) is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - 2m)^2}.$$

In the Setting $t \neq n$, Equation (7) becomes

$$\Pr\begin{bmatrix} \mathrm{PAD}_t(X_{i-1}) = \mathrm{PAD}_t(X'_{j-1}) \oplus T_i \oplus T_j \\ \bigoplus_{k=p+1}^{i-1} X_k \oplus \bigoplus_{k=p+1}^{j-1} X'_k = I_i \oplus I_{i+1} \end{bmatrix}. \tag{8}$$

The probability that X_{i-2} fulfills the bottom equality is still $1/(2^n - 2m)$. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{i-1} that can fulfill the top equality of Equation (8). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - 2m)^2}.$$

Good$_2$. In this case, the graphs of M and M' diverge and converge again after two blocks in the i-th block and it holds that $T_i \neq T'_i$.
In the Setting $t = n$, we investigate the probability for fixed i that

$$\Pr\begin{bmatrix} U_i = U'_{i'} \\ S_i = S'_{i'} \end{bmatrix}.$$

We can assume that $i = i'$ holds for both messages, which means the graphs of both messages diverge at the $i-1$-th block, and join again at the i-th block. Otherwise, if $r' > 2$, we could swap M and M' and are in the case of good_1. So, we assume $r = r' = 2$.
In this case, we also consider that $T_i \oplus T'_i = \Delta T_i \neq 0$, i.e., $T_i \neq T'_i$. So, we study the probability

$$\Pr\begin{bmatrix} X_{i-1} = X'_{i-1} \oplus T_i \oplus T'_i \\ X_{i-1} = X'_{i-1} \oplus I_i \oplus I'_i \end{bmatrix}. \tag{9}$$

From $T_i \neq T'_i$, it follows that $X_{i-1} \neq X'_{i-1}$. So, it must hold that $Y_i \neq Y'_i$, i. e., the bottom lanes of both messages differ. If the graphs of M and M' would be common in all blocks $M_j = M'_j$, for $i \leq j \leq m$, the bottom lanes would differ until the end: $Y_m \oplus Y'_m = X_{i-1} \neq X'_{i-1}$. Since the bottom lane is used as state input, this would imply no collision at the end.
To obtain a collision at the end, there must exist a second diversion phase between the graphs, i.e., there must exist some fresh index $j > i$ s.t. $(U_{j-1}, S_{j-1}) \neq (U'_{j-1}, S'_{j-1})$ or even $j > m'$. This means also the probability

$$\Pr\left[X_{j-1} = \left(\bigoplus_{\substack{k=1 \\ k\neq j-1}}^{m'} \Delta X_k\right) \oplus \left(\bigoplus_{\substack{k=m'+1 \\ k\neq j-1}}^{m} X_k\right)\right]. \tag{10}$$

Since $i-1$ and $j-1$ are distinct indices and fresh, X_{i-1} and X_{j-1} are random values from sets of at least $2^n - 2m$ elements each. Thus, the probability that they fulfill Equation (9) is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - 2m)^2}.$$

In the Setting $t \neq n$, we can rewrite Equation (9) to

$$\Pr\begin{bmatrix} \mathrm{PAD}_t(X_{i-1}) = \mathrm{PAD}_t(X'_{i-1}) \oplus T_i \oplus T'_i \\ X_{i-1} = X'_{i-1} \oplus I_i \oplus I'_i \end{bmatrix}. \tag{11}$$

The probability that X_{j-1} fulfills the bottom equality is still at most $1/(2^n - 2m)$. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{i-1} that can fulfill the top equality of Equation (11). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - 2m)^2}.$$

Good$_3$. In this case, the graphs of M and M' diverge and converge again after two blocks in the i-th block, and it holds that $T_i = T'_i$.
Here, Setting $t = n$ investigates the probability

$$\Pr\begin{bmatrix} U_i = U'_{i'} \\ S_i = S'_{i'} \end{bmatrix}.$$

We can assume that $i = i'$ holds for both messages, which means the graphs of both messages diverge at the $i-1$-th block, and join again at the i-th block. Otherwise, if $r' > 2$, we could swap M and M' and are in the case of good_1. So, we assume $r = r' = 2$. Since $T_i = T'_i$, we consider the probability

$$\Pr\left[X_{i-1} = X'_{i-1}\right] . \tag{12}$$

In order to allow $X_i = X'_i$ to collide, it follows that the tweaks must have differed in the $i-1$-th block: $U_{i-1} \neq U'_{i-1}$. Since the outputs from the penultimate blocks $X_{i-2} = X'_{i-2}$ were equal, this implies that $T_{i-1} \neq T'_{i-1}$.
If the graphs of M and M' are identical until the last block, it would follow that the tweak checksums would differ: $\Theta \neq \Theta'$. So, as for good_2, there must exist a second diversion phase between the graphs. This means, there must exist some index $j > i$ s.t. either $(U_{j-1}, S_{j-1}) \neq (U'_{j-1}, S'_{j-1})$ or alternatively $j > m'$. This means also the probability

$$\Pr\left[X_{j-1} = \left(\bigoplus_{\substack{k=1\\k\neq j-1}}^{m'} \Delta X_k\right) \oplus \left(\bigoplus_{\substack{k=m'+1\\k\neq j-1}}^{m} X_k\right)\right] . \tag{13}$$

Since $i-1$ and $j-1$ are distinct indices and fresh, X_{i-1} and X_{j-1} are chosen randomly from sets of at least $2^n - 2m$ elements each. Thus, the probability that they fulfill Equation (12) is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - 2m)^2} .$$

In the Setting $t \neq n$, Equation (12) becomes

$$\Pr\left[\mathrm{PAD}_t(X_{i-1}) = \mathrm{PAD}_t(X'_{i-1})\right] . \tag{14}$$

The probability that X_{j-1} fulfills Equation (13) equality is still $1/(2^n - 2m)$. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{i-1} that can fulfill Equation (14). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - 2m)} .$$

Good_4. In this case, the graphs of M and M' diverge at the $(p+1)$-th block and converge again after two blocks for one graph, which hits the $p+1$-th block of the respective other. We will study the sub-case $(U_{p+2}, S_{p+2}) = (U'_{p+1}, S'_{p+1})$. The sub-case $(U'_{p+2}, S'_{p+2}) = (U_{p+1}, S_{p+1})$ is analogous.
Setting $t = n$ investigates the probability

$$\Pr\begin{bmatrix} U_{p+2} = U'_{p+1} \\ S_{p+2} = S'_{p+1} \end{bmatrix} = \Pr\begin{bmatrix} X_p = X_{p+1} \oplus T_{p+1} \oplus T'_{p+1} \\ X_{p+1} = I_{p+2} \oplus I'_{p+1} \end{bmatrix} . \tag{15}$$

Since we assume that the graphs are good, there exist no previous non-trivial collisions. Since $\widetilde{\pi}$ is a random permutation, the values X_p and X_{p+1} are randomly chosen from at least $2^n - 2m$ elements each. Thus, the probability that they fulfill Equation (15) is upper bounded by

$$\frac{\binom{m}{2}}{(2^n - 2m)^2} .$$

In the Setting $t \neq n$, Equation (15) becomes

$$\Pr\left[\begin{matrix}\textsc{pad}_t(X_p) = \textsc{pad}_t(X_{p+1}) \oplus T_{p+1} \oplus T'_{p+1} \\ X_{p+1} = I_{p+2} \oplus I'_{p+1}\end{matrix}\right]. \tag{16}$$

The probability that X_{p+1} fulfills the bottom equality is at most $1/(2^n - 2m)$. If $t > n$, the longer tweak cannot increase the probability. If $t < n$, there exist up to 2^{n-t} values for X_{i-1} that can fulfill the top equality of Equation (16). The probability is therefore upper bounded by

$$\frac{2^{n-t} \cdot \binom{m}{2}}{(2^n - 2m)^2}.$$

Over all cases of good graphs and $\binom{q}{2}$ query pairs, the probability of a collision is upper bounded by

$$4 \cdot \binom{q}{2} \cdot \frac{2^{\max(0,n-t)} \cdot \binom{m}{2}}{(2^n - 2\sigma)^2} \leq \frac{4 \cdot 2^{\max(0,n-t)} \cdot \binom{q}{2} \cdot \binom{m}{2}}{(2^n - 2\sigma)^2} \leq \frac{4q^2m^2}{2^{n+\min(n,t)}},$$

using again our assumption of $\sigma < 2^{n-2}$. Taking the sum over all cases, we can derive our bound in Lemma 4. □

6 tAU Analysis

In addition to the collision analysis, we need an upper bound on the probability that two distinct messages yield a collision in $X = X'$. This is captured by the following Lemma.

Lemma 5 (tAU Upper Bound of $\textsc{DoveHash}[\widetilde{\pi}]$)**.** Let $m < 2^{n-2}$. Then, $\textsc{DoveHash}[\widetilde{\pi}]$ is (t, n, ϵ)-tAU for

$$\epsilon \leq \mathsf{Coll}_{\textsc{DoveHash}[\widetilde{\pi}]}(t+n, 2, m, 2m) + \frac{2m^2}{2^n} + \frac{4}{2^{\min(n,t)}}.$$

Proof. The proof follows from a similar but simpler argumentation as in our collision analysis. We study the probability of events for a single message M^i, that we will denote as M for simplicity, or the probability of a collision for two distinct messages M^i and M^j, that we will call M and M' for simplicity.
As in the collision analysis, we will consider a directed, edge-labeled function graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ of our construction. Though, here, we differ from the previous graph by considering permutation outputs. So, the values (X_i, Y_i) will represent the vertices of the graph. The set of edges $\mathcal{E}$ are the transitions between vertices; the set of labels $\mathcal{L}$ consists of exactly those input tuples $(T_i, I_i) \in \mathcal{T} \times \mathcal{B}$ that map $(X_{i-1}, Y_{i-1}) \xrightarrow{T_i, I_i} (X_i, Y_i)$. So, we interpret the blocks of a message $M = (M_1, \ldots, M_m)$ as the labels of the graph. Again, we consider walks $\mathbf{v}$ and $\mathbf{v}'$ associated with M and M', respectively. In the following, we differentiate walks according to **non-trivial output collisions**, i.e., collisions between two permutation outputs

- $X_i = X'_j$, where $i > \mathsf{LCP}_{t+n}(M, M')$ for some $i \geq 1$ and $j \geq 0$ or
- $X_i = X_j$ for $i > j$, or
- $X'_i = X'_j$ for $i > j$.

We call the event of an output collision in the same message, $X_i = X_j$ or $X'_i = X'_j$, for some $i \neq j$, an **output loop**. Moreover, we exclude **non-trivial tweak-input collisions** from consideration here since their probability has already been studied in the collision analysis. Clearly, their probability is upper bounded by

$$\mathsf{Coll}_{\textsc{DoveHash}[\widetilde{\pi}]}\left(t+n, 2, m, 2m\right) .$$

Instead, we will focus on three mutually exclusive cases:

- **Bad Walks:**
  - bad_1: The partial walk $\mathbf{v}$ contains an output loop.
  - bad_2: The partial walks $\mathbf{v}$ and $\mathbf{v}'$ contain no output loop but a non-trivial output collision.
- **Good Walks:** The walks $\mathbf{v}$ and $\mathbf{v}'$ contain no output loops and no non-trivial output collision. We call such walks $\mathbf{v}$ good walks.

Bad_1. In the following, we investigate the probability of an output loop. We stop at the first loop and assume no further non-trivial tweak-input collision. W.l.o.g., we consider the case that the loop is contained in the walk of M and consider $m > m'$. For each, we will distinguish between the settings where $t = n$, $t > n$, and $t < n$.
Assume the Setting $t = n$. In this case, we consider the probability

$$\Pr\left[X_i = X_j\right] .$$

Since the input has been fresh, the probability that both values are equal is at most $1/(2^n - 2m)$. Over at most $\binom{m}{2}$ possible combinations of blocks, the probability is upper bounded by

$$\frac{\binom{m}{2}}{2^n - 2m} .$$

Note that we compare the non-truncated and non-padded values X_i. moreover, $X_0 \in \mathcal{B}$; so, the probability upper bound holds also in the Setting $t \neq n$.

Bad_2. Here, we consider non-trivial output collisions between two messages M and M'. Again, we study the first non-trivial output collision $X_i = X'_{i'}$. and excluded non-trivial state-input collisions and bad walks. So, the input tuple (U_i, S_i) has been fresh. Thus, the probability that two permutation outputs are equal is at most $1/(2^n - 2m)$. Over at most $\binom{m}{2}$ possible combinations of blocks, the probability is upper bounded by

$$\frac{\binom{m}{2}}{2^n - 2m} .$$

Again, we compare the non-truncated and non-padded values X_i. Thus, the upper bound also holds in the Setting $t \neq n$.
So, the probability that a walk is bad is at most

$$2 \cdot \frac{\binom{m}{2}}{2^n - 2m} \leq \frac{2m^2}{2^n} ,$$

using the assumption that $m < 2^{n-2}$.

Good Walks. It remains to study the probability of collisions $X = X'$ *and* no output loops or non-trivial output collisions occurred. For a collision at the end, it has to hold that

$$\Pr\left[\textsc{pad}_t(X_m \oplus X'_{m'}) = \Theta \oplus \Theta'\right] .$$

We study two cases depending on $\Delta\Theta = \Theta \oplus \Theta'$: (1) $\Delta\Theta \neq 0^t$ and (2) $\Delta\Theta = 0^t$.

Case (1): $\Delta\Theta \neq 0^t$. Here, it must hold that

$$\textsc{pad}_t(X_m \oplus X'_{m'}) = \left(\bigoplus_{i=1}^{m} T_i\right) \oplus \left(\bigoplus_{i=1}^{m'} T'_i\right) \neq 0^t .$$

Since we assume no further non-trivial output nor non-trivial tweak-input collisions, the tweak-input tuple (U_m, S_m) was fresh. In the Setting $t = n$, the probability is at most

$$\frac{1}{2^n - 2m} .$$

The probability does not increase in the Setting $t > n$. In the Setting $t < n$, there can exist at most 2^{n-t} outputs X_m that lead to the collision. So, the probability is upper bounded by

$$\frac{2^{n-\min(n,t)}}{2^n - 2m} .$$

Case (2): $\Delta\Theta = 0^t$. Here, it must hold that

$$\textsc{pad}_t(X_m \oplus X'_{m'}) = 0^t . \tag{17}$$

We differentiate between our three settings.
In the Setting $t = n$, the fact that $X_m \oplus X'_{m'}$ must hold implies an output collision. This would contradict our assumption that no non-trivial output collisions have occurred. Thus, in this setting, the case has probability zero here. A similar argument holds in the Setting $t > n$.
In the Setting $t < n$, Equation (17) can hold either due to a non-trivial output collision, or that the most significant t bits collided from two different outputs. Only the latter event is relevant and would mean that $X_m \neq X'_{m'}$. Again, it implies, together with our assumptions of no further non-trivial output nor non-trivial tweak-input collisions, that tweak-input tuple (U_m, S_m) was fresh.
In the Setting $t < n$, there can exist at most 2^{n-t} outputs X_m that lead to the collision on the most significant t bits. Then, the probability is upper bounded by

$$\frac{2^{n-\min(n,t)}}{2^n - 2m} .$$

So, the probability for an output collision when the transcript is good is at most

$$2 \cdot \frac{2^{n-\min(n,t)}}{2^n - 2m} \leq \frac{4}{2^{\min(n,t)}} ,$$

using the assumption that $m < 2^{n-2}$. From all cases, we obtain our bound in Lemma 5. □

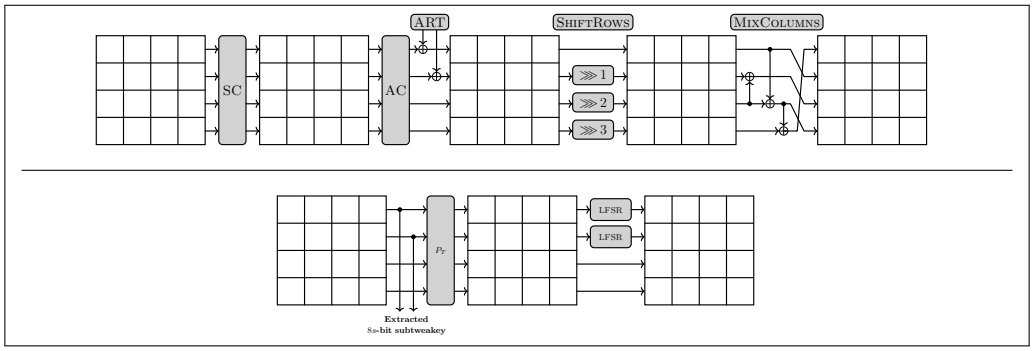

Figure 3: The round function (**top**) and the tweakey schedule of Skinny (**bottom**) [Jea16].

7 Instantiation

This section reports on an instantiation of DoveMAC with Skinny-64-128 [BJK+16c]. We provide a rationale, results of an implementation on two common Atmel microcontrollers, and a brief comparison with an implementation of ZMAC1 on the same platform and with the same primitive [IMPS17].

Choice of A Primitive. DoveMAC would benefit from a tweakable block cipher with a tweak length of at least the block size. In contrast to ZHash, DoveHash does not require an additional counter. Moreover, the term $O(\sigma^2/2^{n+\min(n,t)})$ in the bound limits the security to at most n bits. So, while larger tweaks would increase the rate, the security does not increase for $t > n$. So, we considered performant lightweight tweakable block ciphers for instantiation with $t = n$. Among the available lightweight primitives, the search focused on Skinny-64-128 [BJK+16c], Joltik-BC-64-192 [JNP14], MANTIS [BJK+16c], and QARMA [Ava17]. We opted for Skinny-64-128 due to its lightness and its availability, among others, for microcontrollers. We give a brief overview for the sake of self-containment.

Definition. Skinny [BJK+16c] is a lightweight tweakable block cipher that employs the TWEAKEY schedule [JNP14]. State words and tweakey words are represented by a 4×4-matrix of cells, where each cell is a nibble for Skinny-64. So, key and tweak words are considered together and are processed in a linear update function to produce the round keys. The cipher consists of a 36-round substitution-permutation network which consists of SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR), and MixColumns (MC). The primitive is optimized towards low code and area size. Compared to most earlier lightweight SPNs, Skinny omits an initial key whitening and comes with a non-MDS mixing layer that can be implemented by a few simple XORs. Moreover, the round tweakey is XORed only to half of the state, i.e., to the two topmost state rows in each round. The round function and an iteration of the tweakey schedule are illustrated in Figure 3. More details can be found in [BJK+16c].

Implementation. We implemented DoveMAC in C on ATmega 2560 [Atm14] and ATmega 328p [Atm18], which are common 8-bit RISC microcontrollers operating at 16 MHz. The former has 256 KiB flash memory, eight KiB RAM, and four KiB EEPROM; the latter 32 KiB flash memory, two KiB SRAM, and one KiB EEPROM available. As primitive, we employed the public Skinny-64-128 implementation for microcontrollers from [rwe18] referenced by the SKINNY designers' overview [BJK+16a]. Internally, this

Table 2: Rounded throughputs in cycles/byte and RAM storage in bytes of our implementations on Atmel microcontrollers.

	Message length (bytes)											
	ATmega 2560							ATmega 328p					RAM
Scheme	64	128	256	512	1 024	2 048	4 096	64	128	256	512	1 024	(bytes)
DoveMAC[Skinny-64-128]	760	616	544	508	490	481	476	758	614	542	506	488	176
ZMAC1[Skinny-64-128]	1 013	757	630	566	534	518	510	1 009	755	627	564	532	236

implementation uses two parallel four-bit S-boxes and precomputed the subkeys.
For comparison, we also implemented ZMAC1 [Nai18b], a successor of ZMAC with the same primitive on the same platform. ZMAC is the most intuitive and most illustrative choice of a MAC with a rate that is comparable to that of our proposal. For fairness, we used the recent more efficient successor ZMAC1 that spares the separate domains and employs the same finalization as DoveMAC. So, differences between the constructions are majorly due to the hash function, plus the hash function avoids to extract odd tweak portions and can also use 64-bit tweaks per primitive call.
Our instantiation uses two 64-bit keys for both DoveMAC and ZMAC1, one for the hash function and a second one for the finalization each. The results of our comparison are given in Table 2 for message lengths of up to four KiB for the ATmega 2560 and of up to one KiB for the ATmega 328p. We employed `avr-gcc` as compiler with the `-Os` option for minimizing the code size. Each measurement represents the mean of 1 000 tag computations of hash function and finalization. The storage values in RAM exclude the size of the message and keys. Since the microcontrollers are similar, the storage results were identical and are given therefore only once in Table 2.
The storage and performance values depend strongly on the primitive implementation. For the chosen setup, DoveMAC is about 7-12 percent faster compared to ZMAC1, which is likely to be caused by the doublings in the latter. After subtracting the message and key, our implementation of DoveMAC[Skinny-64-128] used 176 bytes of RAM. An implementation of ZMAC1 on the same platform and with the same block-cipher implementation employed 236 bytes of RAM. Our implementation results leave room for further minimizing the state considerably from the theoretical minimal requirements. Though, the differences reflect implementation-specific overheads in ZMAC1, e.g., the masks, as well as temporary variables for the counters. We plan to publish the source code freely available to the public domain.

8 Conclusion and Future Work

This work proposed a sequential TBC-based PRF that attempts to reduce the memory requirements of ZMAC, in the spirit of the 3kf9 and NI$^+$-MAC designs at a rate of $t + n$ bits per primitive call. Our construction is the first sequential block-cipher-based proposal that processes more message bits than the state size of the primitive while having $O((n + \min(n, t))/2)$ bits of security. We could simply derive a single-key variant DoveMAC1k that spares the second key of DoveMAC by reserving a bit from the tweak space to separate the primitive used for hashing from that used for the finalization.

Future Work. DoveMAC can be easily combined with a small-state encryption mode to an AE scheme. There exist several options: (1) an on-line scheme that uses a PRF only for authentication of the associated data, (2) an off-line nonce-based AE scheme, or (3) a deterministic off-line AE scheme. For all variants, the nonce-IV or the purely IV-based

versions of CTRT [PS16] would allow to use the already available TBC efficiently. We outline the second option in Appendix C. The third option would require a longer IV of at least $(t + n)$ bits to benefit from the high security guarantees of DoveMAC. This would require to use a longer output from the PRF. We show briefly in Section A why it is not straight-forward to derive longer outputs from DoveHash and ZFin$^+$ with high security. So, a future work is to derive a highly secure deterministic AE scheme. Though, the focus of the current work resided on a highly secure fixed-output-length PRF that also uses the tweak for message absorption and reduced the state compared to previous ZMAC-like variants.

Acknowledgments

We are highly thankful to the reviewers and editors of ToSC for their very helpful comments that has lead to considerable improvements of this work.

References

[Atm14] Atmel. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V – 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash, 2014. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf, last access 2018-11-23.

[Atm18] Atmel. ATmega48A/PA/88A/PA/168A/PA/328/P – megaAVR Data Sheet, 2018. http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf, last access 2018-11-23.

[Ava17] Roberto Avanzi. The QARMA Block Cipher Family. Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes. *IACR Trans. Symmetric Cryptol.*, 2017(1):4–44, 2017. https://doi.org/10.13154/tosc.v2017.i1.4-44.

[Ber99] Daniel J. Bernstein. How to Stretch Random Functions: The Security of Protected Counter Sums. *J. Cryptology*, 12(3):185–192, 1999. https://doi.org/10.1007/s001459900051.

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *LNCS*, pages 15–28. Springer, 1995. https://doi.org/10.1007/3-540-44750-4_2.

[BJK+16a] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY family of block ciphers, 2016. https://sites.google.com/site/skinnycipher/, last access 2018-11-23.

[BJK+16b] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY family of block ciphers – Implementations, 2016. https://sites.google.com/site/skinnycipher/implementation.

[BJK+16c] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO II*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016. Full version at https://eprint.iacr.org/2016/660.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007. https://doi.org/10.1007/978-3-540-74735-2_31.

[BPR05] Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved Security Analyses for CBC MACs. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 527–545. Springer, 2005. https://doi.org/10.1007/11535218_32.

[BR02] John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 384–397. Springer, 2002. http://dx.doi.org/10.1007/3-540-46035-7_25.

[CS14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014. Full version at https://eprint.iacr.org/2013/222.

[DDN+17] Nilanjan Datta, Avijit Dutta, Mridul Nandi, Goutam Paul, and Liting Zhang. Single Key Variant of PMAC_Plus. *IACR Trans. Symmetric Cryptol.*, 2017(4):268–305, 2017. https://doi.org/10.13154/tosc.v2017.i4.268-305.

[DJN17] Avijit Dutta, Ashwin Jha, and Mridul Nandi. A New Look at Counters: Don't Run Like Marathon in a Hundred Meter Race. *IEEE Trans. Computers*, 66(11):1851–1864, 2017. https://doi.org/10.1109/TC.2017.2710125.

[DNP16] Avijit Dutta, Mridul Nandi, and Goutam Paul. One-Key Compression Function Based MAC with Security Beyond Birthday Bound. In Joseph K. Liu and Ron Steinfeld, editors, *ACISP I*, volume 9722 of *LNCS*, pages 343–358. Springer, 2016. https://doi.org/10.1007/978-3-319-40253-6_21.

[Dwo16] Morris J Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. Technical report, 2016. Supersedes SP 800-38B (https://www.nist.gov/node/562931).

[HSH+06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *LNCS*, pages 46–59. Springer, 2006. https://doi.org/10.1007/11894063_4.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *LNCS*, pages 129–153. Springer, 2003. https://doi.org/10.1007/978-3-540-39887-5_11.

[IM16] Tetsu Iwata and Kazuhiko Minematsu. Stronger Security Variants of GCM-SIV. *IACR Trans. Symmetric Cryptol.*, 2016(1):134–157, 2016. https://doi.org/10.13154/tosc.v2016.i1.134-157.

[IMG+14] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, and Eita Kobayashi. CAESAR candidate SILC. Directions in Authenticated Ciphers, 2014. Latest submission to the CAESAR competition (v3) at https://competitions.cr.yp.to/round3/clocsilcv3.pdf.

[IMPS17] Tetsu Iwata, Kazuhiko Minematsu, Thomas Peyrin, and Yannick Seurin. ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO, Part III*, volume 10403 of *LNCS*, pages 34–65. Springer, 2017. Full version at https://eprint.iacr.org/2017/535. https://doi.org/10.1007/978-3-319-63697-9_2.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[JMN17] Ashwin Jha, Avradip Mandal, and Mridul Nandi. On The Exact Security of Message Authentication Using Pseudorandom Functions. *IACR Trans. Symmetric Cryptol.*, 2017(1):427–448, 2017. https://doi.org/10.13154/tosc.v2017.i1.427-448.

[JN16] Ashwin Jha and Mridul Nandi. Revisiting structure graphs: Applications to CBC-MAC and EMAC. *J. Mathematical Cryptology*, 10(3-4):157–180, 2016. https://doi.org/10.1515/jmc-2016-0030.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT II*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014. Full version at https://eprint.iacr.org/2014/831.pdf.

[JNP16a] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Deoxys v1.4. http://competitions.cr.yp.to/caesar-submissions.html, Oct 12 2016. Third-round submission to the CAESAR competition. Finalist.

[JNP16b] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Deoxys v1.41. http://competitions.cr.yp.to/caesar-submissions.html, Oct 12 2016. Third-round submission to the CAESAR competition; Deoxys-II became finalist.

[LN17] Eik List and Mridul Nandi. Revisiting Full-PRF-Secure PMAC and Using It for Beyond-Birthday Authenticated Encryption. In Helena Handschuh, editor, *CT-RSA*, LNCS, pages 258–274. Springer, 2017. Full version at https://eprint.iacr.org/2016/1174.

[LPTY16] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC Mode for Lightweight Block Ciphers. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 43–59. Springer, 2016. https://doi.org/10.1007/978-3-662-52993-5_3.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002. https://dx.doi.org/10.1007/3-540-45708-9_3.pdf.

[MMH+14] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In Antoine Joux and Amr M. Youssef, editors, *SAC*, volume 8781 of *LNCS*, pages 306–323. Springer, 2014. https://doi.org/10.1007/978-3-319-13051-4_19.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT II*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015. https://doi.org/10.1007/978-3-662-48800-3_19.

[MV04] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004. https://doi.org/10.1007/978-3-540-30556-9_27.

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Nai15] Yusuke Naito. Full PRF-Secure Message Authentication Code Based on Tweakable Block Cipher. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec*, volume 9451 of *LNCS*, pages 167–182. Springer, 2015. https://doi.org/10.1007/978-3-319-26059-4_9.

[Nai17] Yusuke Naito. Blockcipher-Based MACs: Beyond the Birthday Bound Without Message Length. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT III*, volume 10626 of *LNCS*, pages 446–470. Springer, 2017. https://doi.org/10.1007/978-3-319-70700-6_16. Full version available at https://eprint.iacr.org/2017/852.pdf.

[Nai18a] Yusuke Naito. Improved Security Bound of LightMAC_Plus and Its Single-Key Variant. In Nigel P. Smart, editor, *CT-RSA*, volume 10808 of *LNCS*, pages 300–318. Springer, 2018. https://doi.org/10.1007/978-3-319-76953-0_16.

[Nai18b] Yusuke Naito. On the Efficiency of ZMAC-Type Modes. In Jan Camenisch and Panos Papadimitratos, editors, *CANS*, volume 11124 of *LNCS*, pages 190–210. Springer, 2018. https://doi.org/10.1007/978-3-030-00434-7_10.

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering Generic Composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 257–274. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_15.

[Pat08] Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *LNCS*, pages 328–345. Springer, 2008. http://dx.doi.org/10.1007/978-3-642-04159-4_21.

[Pie06] Krzysztof Pietrzak. A Tight Bound for EMAC. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP II*, volume 4052 of *LNCS*, pages 168–179. Springer, 2006. https://doi.org/10.1007/11787006_15.

[PS16] Thomas Peyrin and Yannick Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO I*, volume 9814 of *LNCS*, pages 33–63. Springer, 2016. http://dx.doi.org/10.1007/978-3-662-53018-4_2.

[PvO95] Bart Preneel and Paul C. van Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *LNCS*, pages 1–14. Springer, 1995. https://doi.org/10.1007/3-540-44750-4_1.

[Rog04] Phillip Rogaway. Nonce-Based Symmetric Encryption. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 348–359. Springer, 2004. https://doi.org/10.1007/978-3-540-25937-4_22.

[rwe18] rweather. SKINNY-C (Implementation for Arduino), Apr 8 2018. https://github.com/rweather/skinny-c, last access 2018-11-23.

[Yas08] Kan Yasuda. A One-Pass Mode of Operation for Deterministic Message Authentication-Security beyond the Birthday Barrier. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 316–333. Springer, 2008. https://doi.org/10.1007/978-3-540-71039-4_20.

[Yas09] Kan Yasuda. A Double-Piped Mode of Operation for MACs, PRFs and PROs: Security beyond the Birthday Barrier. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 242–259. Springer, 2009. https://doi.org/10.1007/978-3-642-01001-9_14.

[Yas10] Kan Yasuda. The Sum of CBC MACs Is a Secure PRF. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *LNCS*, pages 366–381. Springer, 2010. https://doi.org/10.1007/978-3-642-11925-5_25.

[Yas11] Kan Yasuda. A New Variant of PMAC: Beyond the Birthday Bound. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 596–609. Springer, 2011. https://doi.org/10.1007/978-3-642-22792-9_34.

[Yas12] Kan Yasuda. PMAC with Parity: Minimizing the Query-Length Influence. In Orr Dunkelman, editor, *CT-RSA 2012*, pages 203–214. Springer, 2012. https://doi.org/10.1007/978-3-642-27954-6_13.

[Yuv79] Gideon Yuval. How to Swindle Rabin. *Cryptologia*, 3(3):187–191, 1979. https://doi.org/10.1080/0161-117991854025.

[Zha15] Yusi Zhang. Using an Error-Correction Code for Fast, Beyond-Birthday-Bound Authentication. In Kaisa Nyberg, editor, *CT-RSA*, pages 291–307. Springer, 2015. https://doi.org/10.1007/978-3-319-16715-2_16.

[ZWSW12] Liting Zhang, Wenling Wu, Han Sui, and Peng Wang. 3kf9: Enhancing 3GPP-MAC beyond the Birthday Bound. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 296–312. Springer, 2012. https://doi.org/10.1007/978-3-642-34961-4_19.

A Attacks on Preliminary Constructions

In this section, we consider birthday-bound attacks on two preliminary constructions of DoveHash. Thereupon, we demonstrate the requirement of using two keys by an instant distinguishing attack. Finally, we point out a birthday-bound distinguisher if the finalization was extended with ZFin$^+$ to multi-block outputs. For all attacks hereafter, let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{B} \to \mathcal{B}$ be a secure tweakable block cipher and $K_1, K_2 \twoheadleftarrow \mathcal{K} \times \mathcal{K}$ be independent secret random keys. Moreover, we use the setting that $t = n$ for the sake of simplicity.

A.1 Distinguisher When Omitting The Checksum

Here, we consider the variant without the tweak checksum Θ. The further details remain unchanged. If we omit Θ, there exists a birthday-bound distinguisher $\mathbf{A}$ against DoveMAC$[\widetilde{E}_{K_1,K_2}]$ that we sketch in the following.

1. Initialize an empty list $\mathcal{L}$.
2. Choose $q = O(2^{n/2})$ messages M^i, for $1 \leq i \leq q$, which share the same number of m blocks of $(t+n)$ bits. Choose them such that all messages are pairwise equal in all blocks, except for one a-priori fixed t-bit block that is used as T_j in DoveHash$[\widetilde{E}_{K_1,K_2}]$, for some block $j \in \{1,\ldots,m\}$, where m is the number of blocks.
3. Ask for their corresponding outputs Z^i, for $1 \leq i \leq q$, and store the tuples (T_j^i, Z^i) in $\mathcal{L}$, indexed by Z^i.
4. If there exists any pair $Z^i = Z^{i'}$ in $\mathcal{L}$ for distinct indices i, i', lookup the corresponding blocks T_j^i and $T_j^{i'}$.
5. Choose a new message M that contains the same j-block prefix as before, i.e., $M_k = M_k^i$ for all $1 \leq k \leq j$, but differs in some of the subsequent blocks from M^i. Ask for its corresponding output Z.
6. Choose a message M' that shares the same $j-1$-block prefix as M^i and M, i.e., $M'_k = M_k$ for all $1 \leq k \leq j-1$; then uses the $M'_j = (T_j^{i'} \,\|\, I_j^{i'})$, and shares the same postfix as M: $M'_k = M_k$, for all $k > j$. Ask for Z' that corresponds to M'.
7. If $Z = Z'$, output 1 (real world); and output 0 (ideal world) otherwise.

Since all $q+2$ messages share the same $j-1$-block prefix, it holds that (X_{j-1}, Y_{j-1}) is equal for all messages. The output of the j-th block is defined as $X_j^i = \widetilde{\pi}^{T_j^i \oplus X_{j-1}}(I_j^i \oplus Y_{j-1})$ and $X^{i'} = \widetilde{\pi}^{T_j^{i'} \oplus X_{j-1}}(I_j^{i'} \oplus Y_{j-1})$, respectively. Since the tweaks differ, $T_j^i \neq T_j^{i'}$, it holds that $\Pr[X_j^i = X_j^{i'}] \leq \frac{1}{2^n}$, and over all queries

$$\sum_{\substack{i \neq i' \\ i,i' \in \{1,\ldots,q\}}} \Pr\left[X_j^i = X_j^{i'}\right] \leq \frac{\binom{q}{2}}{2^n}.$$

If $X_j^i = X_j^{i'}$, the collision event will occur also between M and M'. Thus, the distinguishing advantage of $\mathbf{A}$ is significant for $q = O(2^{n/2})$.

A.2 A Birthday-bound Forgery When The State Input Is XORed to The Bottom Row

The state size of our proposal would decrease by n bits if we would add the part of each $2n$-bit message block that will go into the state input simply to the lower lane. An illustration of the modified DoveHash function is given at the bottom of Figure 4. In this section, we sketch a birthday-bound forging adversary on this variant. Again, all further details are equal as in DoveMAC, including that the checksum is present. Let $\mathbf{A}$ be a forgery against DoveMAC$[\widetilde{E}_{K_1,K_2}]$ that we sketch in the following.

1. Initialize an empty list $\mathcal{L}$.
2. Choose $q = O(2^{n/2})$ messages M^i of $m \geq 1$ $2n$-bit blocks, s.t. all messages share the same p-block prefix, and $p = m-2$. Choose $M_{m-1}^i = M_m^i = (\langle i \rangle, \langle i \rangle)$ after padding, for $1 \leq i \leq q$, and ask for their corresponding authentication tags Z^i. Store Z^i into $\mathcal{L}$.

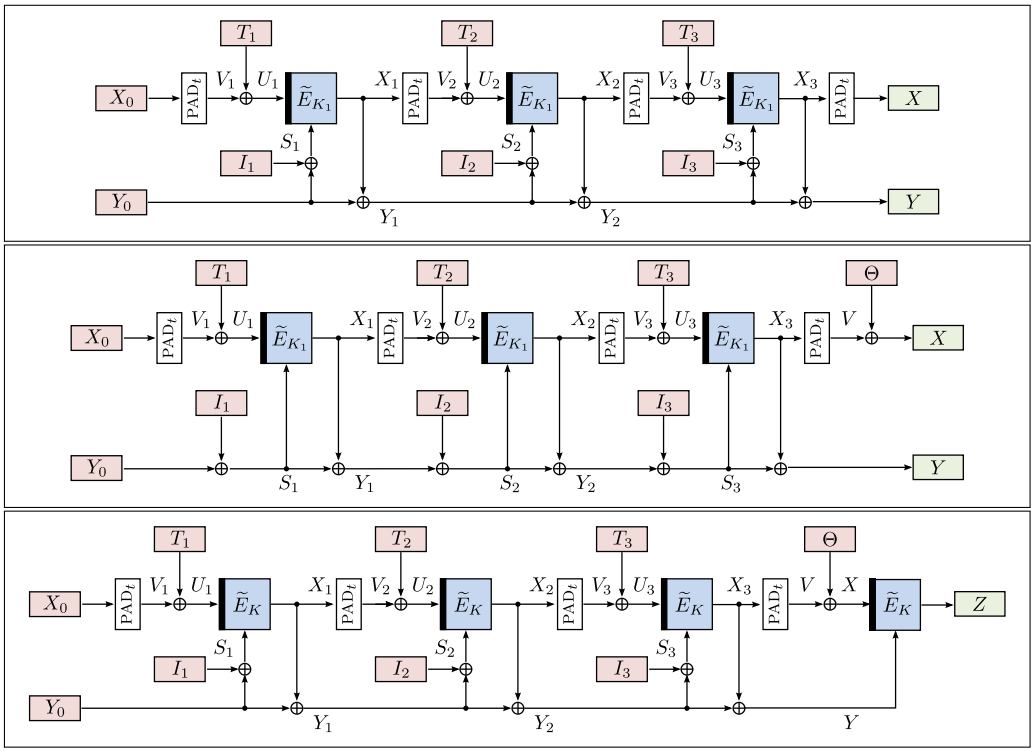

Figure 4: Earlier less secure variants of DoveHash and DoveMAC. **Top:** Variant without the final checksum. **Middle:** Variant where the lower input is XORed directly to the bottom lane. **Bottom:** Single-key variant of DoveMAC.

3. If there exists any collision $Z^i = Z^j$ in $\mathcal{L}$, for $i \neq j$, generate $M^{q+1} = M^i \,\|\, P$, where P is an arbitrary postfix. Ask for its tag Z^{q+1}.
4. Forge with $M^{q+2} = M^j \,\|\, P$ and $Z^{q+2} = Z^{q+1}$.

The forgery is valid with high probability, namely if $X^i_{m-1} \oplus X^j_{m-1} = Y^i_{m-1} \oplus Y^j_{m-1} = \langle i \rangle \oplus \langle j \rangle$. Then, the differences in X and Y cancel because $T^i_m \oplus T^j_m = I^i_m \oplus I^j_m = \langle i \rangle \oplus \langle j \rangle$.

A.3 An Insecure Single-key Variant of DoveMAC

This section illustrates an $O(1)$ forgery attack if DoveMAC would use the same key $\widetilde{E}_K$ for both the hash function and finalization. All other aspects compared to the original definition in Section 3 remain unchanged. The bottom part of Figure 4 visualizes this scheme.

The forgery attack exploits a classical length extension. For simplicity, we consider the case $t = n$ and specify the messages after the 10^* padding has been applied. We provide the messages as tuples of $M^i_j = (T^i_j, I^i_j)$, where i denotes the message index and j the block index. Then, the steps are as follows:

1. Choose $M^1 = ((0^n, 0^n), (10^{n-1}, 0^n)$. This implies that checksum is $\Theta^1 = 10^{n-1}$. Ask for the corresponding authentication tag $Z^1 = \widetilde{E}^{X^1}_K(Y^1)$.
2. Derive $M^2 = ((0^n, 0^n), (10^{n-1}, 0^n), (10^{n-1}, 0^n))$. Here, the padding is located in the third block. This implies that the checksum is $\Theta^2 = 0^n$. Ask for the corresponding authentication tag $Z^2 = \widetilde{E}^{X^2}_K(Y^2)$. From $X^2 = Z^1 \oplus \Theta^2 = Z^1$ and $Y^2 = Y_3 = Y_2 \oplus Z^1$, it follows that $Z^2 = \widetilde{E}^{Z^1}_K(Y_2 \oplus Z^1)$. So

3. Derive $M^3 = ((0^n, 0^n), (10^{n-1}, 0^n), (10^{n-1}, 0^n), (0^n, 0^n), (Z^1 \oplus Z^2, Z^2))$. Since $Z^2 \neq 0^n$ holds with probability $1 - 2^{-n}$, M^3 can be chosen s.t. our description already contains the 10^* padding. Here, the checksum is $\Theta^3 = Z^1 \oplus Z^2$. Furthermore, we observe that

$$\begin{aligned} X_4 &= Z^2 \\ Y_4 &= Y_3 \oplus X_4 = Y_2 \oplus Z^1 \oplus Z^2 \\ U_5 &= X_4 \oplus T^5 = Z^2 \oplus Z^1 \oplus Z^2 = Z^1 \\ S_5 &= Y_4 \oplus I_5 = Y_2 \oplus Z^1 \oplus Z^2 \oplus Z^2 = Y_2 \oplus Z^2 = Y_3 = Y^2. \end{aligned}$$

So, it holds that $X_5 = \widetilde{E}_K^{U_5}(S_5) = \widetilde{E}_K^{Z^1}(Y^2) = Z^2$. It follows that

$$\begin{aligned} Y^3 &= X_5 \oplus Y_4 = Z^2 \oplus Y_2 \oplus Z^1 \oplus Z^2 = Y_2 \oplus Z^1 = Y^2 \\ X^3 &= X_5 \oplus \Theta^3 = Z^2 \oplus Z^1 \oplus Z^2 = Z^1. \end{aligned}$$

So, $Z^3 = \widetilde{E}_K^{X_3}(Y^3) = \widetilde{E}_K^{X^2}(Y^2) = Z^2$.

So, the forgery message would be M^3 with tag $Z^3 = Z^2$.

A.4 A Birthday-bound Distinguisher of DoveMAC with Longer Outputs

Unfortunately, DoveMAC cannot be simply extended to multi-block outputs with the general variant of ZFin$^+$. ZFin$^+$ is given in Algorithm 1. However, we stress that it is specified only for $d = 1$. In brief, it computes d output blocks Z_i, for $i = 1, \ldots, d$ as

$$Z_i \leftarrow \widetilde{E}_{K_2}^{X \oplus \langle i-1 \rangle}(Y)\,.$$

This section illustrates a birthday-bound distinguisher on such a construction for $d = 2$. The steps are as follows, for small constant c, e.g., $c = 4$:

1. Ask $c \cdot 2^{n/2}$ messages $M^i = (T_1, I_1^i)$ after padding, where the values I_1^i are pairwise distinct and T_1 is a fixed arbitrary constant for all messages. Ask for the outputs $Z^i = (Z_1^i, Z_2^i)$ and store them in a list $\mathcal{L}$.

2. Ask $c \cdot 2^{n/2}$ further messages $M^j = (T_1 \oplus \langle 1 \rangle, I_1^j)$ after padding, where the values I_1^j are pairwise distinct. Ask for the outputs $Z^j = (Z_1^j, Z_2^j)$. With significant probability, there exists a message M^i for some $i \in [0..c2^{n/2} - 1]$ and a message M^j for some $j \in [c2^{n/2}..2c2^{n/2}]$, s.t. $X_1^i = X_1^j$. Then, it follows that $V^i = V^j$ and $Y^i = Y^j$ and $\Theta^i = \Theta^j \oplus \langle 1 \rangle$. So,

$$\begin{aligned} Z_1^i &= \widetilde{E}_{K_2}^{V^i \oplus \Theta^i}(Y^i) \\ Z_2^i &= \widetilde{E}_{K_2}^{V^i \oplus \Theta^i \oplus \langle 1 \rangle}(Y^i). \end{aligned}$$

and

$$\begin{aligned} Z_1^j &= \widetilde{E}_{K_2}^{V^j \oplus \Theta^j}(Y^j) = \widetilde{E}_{K_2}^{V^i \oplus \Theta^i \oplus \langle 1 \rangle}(Y^i) = Z_2^i \\ Z_2^j &= \widetilde{E}_{K_2}^{V^j \oplus \Theta^j \oplus \langle 1 \rangle}(Y^j) = \widetilde{E}_K^{1, V^i \oplus \Theta^i}(Y^i) = Z_1^i\,. \end{aligned}$$

Thus, $\mathbf{A}$ can expect to find some $(Z_1^i, Z_2^i) \in \mathcal{L}$ that matches some (Z_2^j, Z_1^j) with high probability. This event holds with probability $(c \cdot 2^{n/2})^2 \cdot 2^{-2n} \approx c/2^n$ for a random function but with high probability for DoveMAC if it would employ longer outputs.

B Proof of Lemma 3

For the sake of completeness, this section sketches the proof of Lemma 3 from [LN17], which bounds the ratio of good transcripts.

Lemma 6. It holds that $\Pr[\Theta_{\text{real}} = \tau] \geq \Pr[\Theta_{\text{ideal}} = \tau]$.

Proof Sketch. Let τ be a good transcript. We consider the set of values $\{Y^i\}_{1 \leq i \leq q}$ that occur as outputs of DoveHash over the queries of the transcript. This set can be rewritten as $\{\mathsf{Y}^1, \ldots, \mathsf{Y}^r\}$ with $r \leq q$ s.t. all tweaks Y^i are pairwise distinct. We further define q_i, for $1 \leq i \leq r$, for the number of queries for which DoveHash produced Y^i. Clearly, it holds that $\sum_{i=1}^{r} q_i = q$. Since the outputs are sampled independently at random in the ideal world, it holds that

$$\Pr[\Theta_{\text{ideal}} = \tau] = \prod_{i=1}^{q} \frac{1}{2^n} .$$

To bound the probability in the real world, we adopt the notion of transcript-compatible permutations. We call $\widetilde{\pi}$ compatible with τ if, for all $1 \leq i \leq q$, it holds that

$$\widetilde{\pi}(Y^i, X^i) = Z^i .$$

Let $\mathsf{Comp}(\tau)$ denote the set of all tweakable permutations $\widetilde{\pi}$ that are compatible with τ. Chen and Steinberger showed a variant of the following:

$$\Pr[\Theta_{\text{real}} = \tau] = \Pr\left[\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \{0,1\}^n) : \widetilde{\pi} \in \mathsf{Comp}(\tau)\right] .$$

Over all tweaks Y^i, for $1 \leq i \leq r$, the fraction of compatible permutation is

$$\prod_{i=1}^{r} \prod_{j=1}^{q_i} \frac{1}{2^n - (j-1)} = \prod_{i=1}^{r} \frac{1}{(2^n)_{q_i}} .$$

It holds that

$$\prod_{i=1}^{r} \frac{1}{(2^n)_{q_i}} \geq \prod_{j=1}^{q} \frac{q}{(2^n)_1} = \prod_{j=1}^{q} \frac{1}{2^n} .$$

We can derive that

$$\Pr[\Theta_{\text{real}} = \tau] \geq \Pr[\Theta_{\text{ideal}} = \tau] ,$$

which gives our claim in Lemma 3. □

C Authenticated Encryption

We can combine DoveMAC with the nonce-IV-based variant of the Counter-in-Tweak encryption scheme by Peyrin and Seurin [PS16] to a nonce-based AE scheme. We follow the notions by [NRS14, Rog04].

Nonce-IV-based Encryption. Let $\mathcal{K}$ be a non-empty set of keys. Let further $\mathcal{N}$, $\mathcal{IV}$, and $\mathcal{M}$ be a nonce space, an IV space, and a message space, respectively. A nonce-IV-based encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ is a tuple of encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{IV} \times \mathcal{M} \to \mathcal{C}$, where the encryption chooses $IV \twoheadleftarrow \mathcal{IV}$ randomly for each message, and outputs $(IV, C) \in \mathcal{IV} \times \mathcal{C}$. The nonce must not repeat over all encryption queries. $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{IV} \times \mathcal{C} \to \mathcal{M}$ is the deterministic decryption algorithm corresponding to $\mathcal{E}$ s.t. Π is correct and length-preserving assuming $\mathcal{M} = \mathcal{C} \subseteq \{0,1\}^*$.

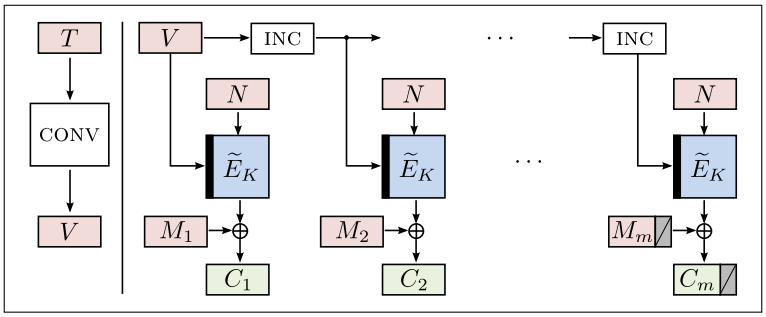

Figure 5: Encryption of a message $M = (M_1, \ldots, M_m)$ with the nonce-IV-based variant of Counter-in-Tweak [PS16] under nonce N and IV T. $\textsc{Conv} : \mathcal{IV} \to \mathcal{T}$ is a regular function and $\textsc{inc} : \mathcal{T} \to \mathcal{T}$ increments the current value of the tweak inputs s.t. $V_i = V \oplus \langle i-1 \rangle_t$, for $1 \leq i \leq m$.

Definition 11 (nivE Advantage)**.** Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a nonce-IV-based encryption scheme with signatures and assumptions as above. Let $K \twoheadleftarrow \mathcal{K}$ and let $\mathbf{A}$ be an adversary against Π with access to an oracle, s.t. $\mathbf{A}$ never repeats a nonce N. Let $\rho \twoheadleftarrow \mathsf{Func}(\mathcal{N} \times \mathcal{IV} \times \mathcal{M}, \mathcal{C})$ be a length-preserving function. For each $(N, M) \in \mathcal{N} \times \mathcal{M}$ input, the oracle samples $IV \twoheadleftarrow \mathcal{IV}$, and outputs (IV, C) where $C = \mathcal{E}_K(N, IV, C)$ in the real world and $C = \rho(N, IV, M)$ in the ideal world. The IV is not chosen by $\mathbf{A}$ in both worlds. Then $\mathbf{Adv}^{\text{nivE}}_{\Pi}(\mathbf{A}) =^{\text{def}} \Delta_{\mathbf{A}}(\mathcal{E}_K; \rho)$.

Counter-in-Tweak. The nonce-IV-based version of Counter-in-Tweak takes as inputs tuples of $(N, V, \underline{M}) \in \mathcal{N} \times \mathcal{IV} \times \mathcal{M}$, and produces a ciphertext C of equal length as the message: $|C| = |\underline{M}|$ for all keys $K \in \mathcal{K}$ and inputs $(N, V, \underline{M})$. Note that $V \twoheadleftarrow \mathcal{IV}$ is chosen by the oracle in the nivE model.
CTRT is a variant of the well-known Counter mode, where the nonce is used as input to all calls of $\widetilde{E}_K$ and the tweak V_i is derived from V. CTRT defines $\mathcal{IV} = \mathcal{T}$, and uses an increment function $\textsc{inc} : \mathcal{T} \to \mathcal{T}$ that increments the value of a given V. We define that $\textsc{inc}^i(V) = V \oplus \langle i-1 \rangle_t$ for all $i \geq 1$. Note that the decryption is similar to the encryption. It differs only in the fact that the decryption oracle does not choose IVs: $\text{CTRT}.\mathcal{D}[\widetilde{E}_K](N, V, C) = \text{CTRT}.\mathcal{E}[\widetilde{E}_K](N, V, C)$ for all $K, N, V, C \in \mathcal{K} \times \mathcal{N} \times \mathcal{IV} \times \mathcal{C}$.
The following security statement is part of Theorem 1 in the full version of [PS16] for nonce-respecting adversaries.

Theorem 3 (nivE Security of CTRT [PS16])**.** Let $\mathbf{A}$ be a nonce-respecting nivE adversary against $\text{CTRT}[\widetilde{E}]$ that asks at most q queries of σ blocks. Let $K \twoheadleftarrow \mathcal{K}$. Then

$$\mathbf{Adv}^{\text{nivE}}_{\text{CTRT}[\widetilde{E}]}(\mathbf{A}) \leq \mathbf{Adv}^{\text{TPRP}}_{\widetilde{E}}(\mathbf{A}') + \frac{\sigma^2}{2^{n+1}|\mathcal{T}|},$$

where $\mathbf{A}'$ asks at most σ queries.

Nonce-based Authenticated Encryption. Let $\mathcal{A}$ and $\mathcal{U}$ define non-empty sets of associated data and authentication tags, respectively. A nonce-based authenticated encryption scheme (with associated data) $\Pi = (\mathcal{E}, \mathcal{D})$ is a tuple of deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{U}$ and deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{U} \to \mathcal{M} \times \{\perp\}$ with associated key space $\mathcal{K}$. The associated data is authenticated but not encrypted. The decryption function $\mathcal{D}$ takes a tuple (N, A, C, T) and outputs either M or $\perp$ if the input is invalid. Again, we assume $\mathcal{C} = \mathcal{M} \subseteq \{0,1\}^*$, as well as correctness and length preservation. Note that [NRS14, Rog04] combine ciphertext and

tag in a single entity C. Though, it is more natural to consider two entities of ciphertext C and authentication tag T. Hereafter, we briefly recap the privacy and authenticity notions for nonce-based authenticated encryption.

Definition 12 (PRIV Advantage)**.** Let $\rho \twoheadleftarrow \mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{C} \times \mathcal{U})$ be length-preserving and $K \twoheadleftarrow \mathcal{K}$. Let $\mathbf{A}$ be an adversary on $\mathcal{E}$ that does not repeat inputs $N \in \mathcal{N}$. Then, the PRIV advantage of $\mathbf{A}$ w.r.t. Π is defined as $\mathbf{Adv}^{\text{PRIV}}_{\Pi}(\mathbf{A}) =^{\text{def}} \Delta_{\mathbf{A}}(\mathcal{E}_K; \rho)$.

Definition 13 (AUTH Advantage)**.** Let $\mathbf{A}$ be an adversary on Π that is given access to an encryption and decryption oracle where $K \twoheadleftarrow \mathcal{K}$ is a random secret, and $\mathbf{A}$ does not repeat inputs $N \in \mathcal{N}$ to the encryption oracle, and does not ask responses from the encryption oracle to the decryption oracle. Then, the AUTH advantage of $\mathbf{A}$ is defined as the probability that $\mathbf{A}$ successfully forges a valid ciphertext that is accepted by the decryption: $\mathbf{Adv}^{\text{AUTH}}_{\Pi}(\mathbf{A}) = \Pr[\mathbf{A} \text{ forges}]$.

Let $\mathcal{N} = \{0,1\}^{\nu}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{C} \subseteq \{0,1\}^*$, and $\mathcal{U} = \{0,1\}^{\tau}$. We define $\rho \twoheadleftarrow \mathsf{Func}(\mathcal{N} \times \mathcal{IV} \times \mathcal{M}, \mathcal{C} \times \mathcal{U})$ to query $(C,T) = \mathcal{E}_K(N, IV, M)$ for the secret key $K \twoheadleftarrow \mathcal{K}$ of the experiment, and to output (C', T') with $C' \twoheadleftarrow \{0,1\}^{|C|}$ and $T' \twoheadleftarrow \{0,1\}^{|T|}$, for each input (N, IV, M). So, ρ is equivalent to a random function that outputs a tuple of random bit strings of the expected length.

Definition 14 (nAE Advantage)**.** Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a nonce-based authenticated encryption scheme. Let $\rho \twoheadleftarrow \mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{C} \times \mathcal{U})$ be length-preserving. Let $\mathbf{A}$ be a nonce-respecting adversary on Π that is given access to an encryption and decryption oracle, where $K \twoheadleftarrow \mathcal{K}$ is a random secret, and $\mathbf{A}$ does not ask queries that it received as responses from the encryption oracle to the decryption oracle. Then, the nAE advantage of $\mathbf{A}$ is defined as $\mathbf{Adv}^{\text{nAE}}_{\Pi}(\mathbf{A}) =^{\text{def}} \Delta_{\mathbf{A}}(\mathcal{E}_K, \mathcal{D}_K; \rho, \perp)$, where $\perp : \mathcal{N} \times \mathcal{A} \times \mathcal{M} \times \mathcal{U} \to \{\perp\}$ returns the invalid symbol for every input.

We call $\mathbf{A}$ a (q_e, q_d, σ)-nAE-adversary if $\mathbf{A}$ asks at most q_e encryption queries, and at most q_d decryption queries that consist of at most σ blocks in total.

Fact 1. Given a nonce-respecting (q_e, q_d, σ)-nAE-adversary $\mathbf{A}$ on an nAE scheme Π, there exists a nonce-repeating (q_e, q_d, σ)-AE adversary $\mathbf{A}'$ on Π s.t. $\mathbf{Adv}^{\text{nAE}}_{\Pi}(\mathbf{A}) \leq \mathbf{Adv}^{\text{AE}}_{\Pi}(\mathbf{A}')$.

Fact 1 follows form the trivial observation that any nonce-respecting adversary $\mathbf{A}$ is also a nonce-repeating $\mathbf{A}'$, and the latter can simply copy the exact behavior from $\mathbf{A}$ to inherit $\mathbf{A}$'s advantage. It is well-known that, if $\mathbf{A}$ is an (q, q_d, σ)-nAE adversary on Π, then it holds that

$$\mathbf{Adv}^{\text{nAE}}_{\Pi}(\mathbf{A}) \leq \mathbf{Adv}^{\text{PRIV}}_{\mathcal{E}_K}(\mathbf{A}') + \mathbf{Adv}^{\text{AUTH}}_{\mathcal{E}_K, \mathcal{D}_K}(\mathbf{A}'').$$

for any (q, σ)-PRIV adversary $\mathbf{A}'$ and (q, q_d, σ)-AUTH adversary $\mathbf{A}''$.

NSIV. NSIV$[F, \Pi]$ is a SIV-like nonce-based off-line AEAD scheme by [PS16] based on a nonce-based function $F : \mathcal{K}_1 \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{U}$, and a nonce-IV-based encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ with key space $\mathcal{K}_2$ and $\mathcal{E} : \mathcal{K}_2 \times \mathcal{N} \times \mathcal{IV} \times \{0,1\}^* \to \{0,1\}^*$. The scheme defines a regular function $\text{CONV} : \mathcal{U} \to \mathcal{IV}$ that converts the output from F into an IV for Π.[3] First, nonce, associated data, and message are processed by F to produce a tag: $T \leftarrow F_{K_1}(N, A, M)$. Then, the scheme computes $V \leftarrow \text{CONV}(T)$, and encrypts the message M to $C \leftarrow \mathcal{E}_{K_2}(N, V, M)$. The output is given by (C, T). The following theorem is slightly adapted from Theorem 4a) in [PS16].

Theorem 4 (nAE Advantage of NSIV)**.** Let F and Π have signatures as above, and let CONV be a regular function. Let $K_1, K_2 \twoheadleftarrow \mathcal{K}_1 \times \mathcal{K}_2$. Let $\mathbf{A}$ be a nonce-respecting

[3] A function $F : \mathcal{X} \to \mathcal{Y}$ is called regular if all $Y \in \mathcal{Y}$ have the same number of preimages in $X \in \mathcal{X}$.

Algorithm 2 Authenticated encryption with DoveSIV$[\widetilde{E}_{K_1,K_2,K_3}]$.

1: **function** DoveSIV.$\mathcal{E}[\widetilde{E}_{K_1,K_2,K_3}](N, A, \underline{M})$
2: $M \leftarrow \textsc{Encode}_{t+n}(N, A, \underline{M})$
3: $T \leftarrow \textsc{DoveMAC}[\widetilde{E}_{K_1,K_2}](M)$
4: $V \leftarrow \textsc{Conv}(T)$
5: $C \leftarrow \textsc{CTRT}[\widetilde{E}_{K_3}](N, V, \underline{M})$
6: **return** (C, T)

11: **function** DoveSIV.$\mathcal{D}[\widetilde{E}_{K_1,K_2,K_3}](N, A, C, T)$
12: $V \leftarrow \textsc{Conv}(T)$
13: $\underline{M} \leftarrow \textsc{CTRT}[\widetilde{E}_{K_3}](N, V, C)$
14: $M \leftarrow \textsc{Encode}_{t+n}(N, A, \underline{M})$
15: $T' \leftarrow \textsc{DoveMAC}[\widetilde{E}_{K_1,K_2}](M)$
16: **if** $T \stackrel{?}{=} T'$ **then**
17: **return** $\underline{M}$
18: **return** $\perp$

21: **function** CTRT$[\widetilde{E}_K](U, V, \underline{M})$
22: $(\underline{M}_1, \ldots, \underline{M}_m) \xleftarrow{n} \underline{M}$
23: **for** $i = 1$ **to** $m-1$ **do**
24: $V_i \leftarrow V \oplus \langle i-1 \rangle_t$
25: $C_i \leftarrow \widetilde{E}_K^{V_i}(U) \oplus \underline{M}_i$
26: $V_m \leftarrow V \oplus \langle m-1 \rangle_t$
27: $C_m \leftarrow \mathsf{msb}_{|\underline{M}_m|}(\widetilde{E}_K^{V_m}(U)) \oplus \underline{M}_m$
28: $C \leftarrow (C_1 \parallel \ldots \parallel C_m)$
29: **return** C

31: **function** $\textsc{Encode}_{t+n}(N, A, \underline{M})$
32: $L \leftarrow \langle |A| \rangle_{t-1} \parallel \langle |\underline{M}| \rangle_n$
33: $S \leftarrow N \parallel A \parallel \underline{M} \parallel L$
34: **return** S

41: **function** $\textsc{Conv}(T)$
42: **return** $\mathsf{msb}_t(T)$

nAE adversary against NSIV$[F, \Pi]$ with access to two oracles such that $\mathbf{A}$ never queries outputs from its first (encryption) oracle to its second (verification) oracle. $\mathbf{A}$ asks at most q queries to its available oracles consisting of at most m blocks each and σ blocks in total. Then

$$\mathbf{Adv}^{\mathrm{nAE}}_{\mathrm{NSIV}[F,\Pi]}(\mathbf{A}) \leq \mathbf{Adv}^{\mathrm{NIVE}}_{\Pi}(\mathbf{A}') + \mathbf{Adv}^{\mathrm{PRF}}_{F}(\mathbf{A}'') + \frac{q}{2^n}\,,$$

where $\mathbf{A}'$ and $\mathbf{A}''$ ask at most q queries of at most σ blocks in total and run in time at most $O(\sigma)$.

Remark 2. Theorem 4 in [PS16] contains separate inequalities for nonce-repeating and nonce-respecting adversaries. The bound above is equivalent to the AE security statement for NSIV$[F, \Pi]$; though, it also holds for nonce-repeating adversaries. Their separation of statements was necessary since the statement of Theorem 4b) provided a significantly better bound for nonce-respecting settings for the choice of Counter-in-Tweak and the Encrypted Parallel Wegman-Carter MAC. However, since the security of DoveMAC does not depend on nonces, we can work with the nonce-repeating bound hereafter.

DoveSIV. Let $K_1, K_2, K_3 \in \mathcal{K}$ be independent keys, $\mathcal{N} = \{0,1\}^n$, and $\mathcal{T} = \{0,1\}^t$. We define DoveSIV as an instantiation of NSIV$[F, \Pi]$, where the PRF F is instantiated with DoveMAC$[\widetilde{E}_{K_1}, \widetilde{E}_{K_2}]$ and the encryption scheme Π with CTRT$[\widetilde{E}_{K_3}]$. We write DoveMAC$[\widetilde{E}_{K_1,K_2}]$ as a short form of DoveMAC$[\widetilde{E}_{K_1}, \widetilde{E}_{K_2}]$, hereafter. More detailed, we define an injective function $\textsc{Encode}_x : \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to (\{0,1\}^x)^*$ that maps nonce, associated data, and message into the block-wise format for DoveMAC: $M = \textsc{Encode}_{t+n}(N, A, \underline{M})$. The encryption scheme is CTRT$[\widetilde{E}_{K_3}]$ with XOR-based increment of the tweaks. The conversion function $\textsc{Conv} : \mathcal{B} \to \mathcal{U}$ maps the n-bit IV to a t-bit tweak for $\widetilde{E}$ by chopping off the least significant $n - t$ bits.
Now, we can combine the results from Theorems 3 and 4 with the security of CTRT and NSIV with Theorem 1 that quantifies the PRF security of DoveMAC.

Theorem 5 (nAE Advantage of DoveSIV)**.** Let F and Π have signatures as above, and Conv be given as in Algorithm 2. Let $K_1, K_2, K_3 \twoheadleftarrow \mathcal{K}^3$ be independent. Let $\mathbf{A}$ be a nonce-respecting nAE adversary against DoveSIV$[F, \Pi]$ with access to two oracles such that $\mathbf{A}$ never queries outputs from its first (encryption) oracle to its second (verification) oracle. Moreover, $\mathbf{A}$ asks at most q queries whose concatenated lengths of associated data and messages consists of at most m $(t+n)$-bit blocks each and at most $\sigma < 2^{n-4}$ $(t+n)$-bit

blocks in total. Then

$$\mathbf{Adv}^{\mathrm{NAE}}_{\textsc{DoveSIV}[F,\Pi]}(\mathbf{A}) \leq \frac{\sigma^2}{2^{t+n+1}} + \frac{4\sigma + 25q}{2^n} + \frac{2q^2 + 5q(q+1)(m+6)^2}{2^{n+\min(n,t)}} + 3 \cdot \mathbf{Adv}^{\mathrm{TPRP}}_{\widetilde{E}}(\mathbf{A}')\,,$$

where $\mathbf{A}'$ asks at most $2\sigma + 6q$ queries in total, and $\mathbf{A}$ as well as $\mathbf{A}'$ run in time at most $O(2\sigma + 6q)$.

Proof. The rightmost term follows from a standard argument: we replace $\widetilde{E}_{K_1}$, $\widetilde{E}_{K_2}$, and $\widetilde{E}_{K_3}$ by three independent random tweaked permutations $\widetilde{\pi}_1, \widetilde{\pi}_2, \widetilde{\pi}_3 \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{B})$. Clearly, the maximal advantage for an adversary $\mathbf{A}'$ to distinguish between both settings is given by $3 \cdot \mathbf{Adv}^{\mathrm{TPRP}}_{\widetilde{E}}(\mathbf{A}')$, where $\mathbf{A}'$ asks at most $2\sigma + 6q$ queries in total. This stems from the fact that the encoding function Encode may create six additional $(t+n)$-bit blocks: one that includes the nonce in the PRF, one block to append the length of the associated data, one block by appending the message length, and one block for the padding. We further include the two blocks for padding in DoveMAC and its finalization for simplicity. In the following, we define $\sigma' = \sigma + 6q$ and $m' = m + 6$.
The leftmost term in the bound represents the nivE security bound of CTRT:

$$\frac{\sigma^2}{2^t \cdot 2^{n+1}}\,.$$

From the application of Theorem 1, the PRF security of $\textsc{DoveMAC}[\widetilde{\pi}_1, \widetilde{\pi}_2]$ for so-encoded inputs can be upper bounded by

$$\mathbf{Adv}^{\mathrm{PRF}}_{\textsc{DoveMAC}[\widetilde{\pi}_1,\widetilde{\pi}_2]}(\mathbf{A}) \leq \frac{4\sigma'}{2^n} + \frac{q^2 m'^2}{2^{2n}} + \frac{2q^2 + 4qm'^2 + 4q^2m'^2}{2^{n+\min(n,t)}}\,.$$

From the assumption that $\sigma < 2^{n-4}$ and the fact that $q \leq \sigma$, we can upper bound $1/(2^n - \sigma') \leq 2/2^n$, and the above by

$$\frac{4(\sigma + 6q)}{2^n} + \frac{2q^2 + 5q(q+1)(m+6)^2}{2^{n+\min(n,t)}}\,.$$

Finally, an additional addend of $q/2^n$ stems from Theorem 4. The sum of the individual terms yields our claim in Theorem 5. □

Remark 3. Note that the requirement of multiple keys K_1, K_2, and K_3 can be easily lifted by using pairwise distinct domains, as is done, e.g., in the AE scheme Deoxys [JNP16a]. Then, one can define $d_1, d_2, d_3 \in \mathcal{D}$, and define a domain-tweak space $\mathcal{T}_D = \mathcal{D} \times \mathcal{T}$. In this case, a secure tweakable block cipher $\widetilde{E} : \mathcal{K} \times \mathcal{T}_D \times \mathcal{B} \to \mathcal{B}$ can be used for all calls in DoveSIV under a single key $K \in \mathcal{K}$. For example, the domain can be fixed to d_1 for all calls to $\widetilde{E}^{d_1}_K$ in DoveHash, fixed to d_2 in ZFin$^+$, and fixed to d_3 in CTRT.