# Efficient Beyond-Birthday-Bound-Secure Deterministic Authenticated Encryption with Minimal Stretch

Christian Forler[1], Eik List[2], Stefan Lucks[2], and Jakob Wenzel[2]

[1] Hochschule Schmalkalden, [2] Bauhaus-Universität Weimar
[1] cforler@hs-schmalkalden.de, [2] <firstname>.<lastname>@uni-weimar.de

**Abstract.** Block-cipher-based authenticated encryption has obtained considerable attention from the ongoing CAESAR competition. While the focus of CAESAR resides primarily on nonce-based authenticated encryption, Deterministic Authenticated Encryption (DAE) is used in domains such as key wrap, where the available message entropy motivates to omit the overhead for nonces. Since the highest possible security is desirable when protecting keys, beyond-birthday-bound (BBB) security is a valuable goal for DAE. In the past, significant efforts had to be invested into designing BBB-secure AE schemes from conventional block ciphers, with the consequences of losing efficiency and sophisticating security proofs.
This work proposes Deterministic Counter in Tweak (DCT), a BBB-secure DAE scheme inspired by the Counter-in-Tweak encryption scheme by Peyrin and Seurin. Our design combines a fast $\epsilon$-almost-XOR-universal family of hash functions, for $\epsilon$ close to $2^{-2n}$, with a single call to a $2n$-bit SPRP, and a BBB-secure encryption scheme. First, we describe our construction generically with three independent keys, one for each component. Next, we present an efficient instantiation which (1) requires only a single key, (2) provides software efficiency by encrypting at less than two cycles per byte on current x64 processors, and (3) produces only the minimal $\tau$-bit stretch for $\tau$ bit authenticity. We leave open two minor aspects for future work: our current generic construction is defined for messages of at least $2n - \tau$ bits, and the verification algorithm requires the inverse of the used $2n$-bit SPRP and the encryption scheme.

**Keywords:** deterministic authenticated encryption, symmetric cryptography, cryptographic schemes, provable security, tweakable block cipher, universal hash function.

## 1 Introduction

***Deterministic Authenticated Encryption.*** A secure authenticated encryption (AE, hereafter) scheme is nowadays widely understood as a construction which produces ciphertexts that are indistinguishable from random bitstrings and infeasible to forge. Modern AE schemes are mostly nonce-based [33], i.e.,

the user is responsible to supply an additional nonce that must be unique for every encryption. In contrast, Deterministic Authenticated Encryption (DAE) [34] is employed for settings where it is more senseful to exploit existing entropy or redundancy in the inputs to avoid the overhead of nonces, e.g., for wrapping cryptographic keys.

***Existing Designs.*** A variety of DAE and MRAE schemes has been proposed since, e.g., HADDOC [6], BCTR [9], DAEAD [10], MRO/MRS/MROS [13], GCM-SIV [14], BTM [20], HBS [21], DEOXYS [23], JOLTIK [24], HS1-SIV [25], MINICTR [28], MR-OMD [32], and SIV [34]. The naturally raising question is: Which unsolved problem requires the proposal of a novel mode?
Block-cipher-based DAE schemes are inherently efficient. While a myriad of block ciphers is available, the dominating standard state size is still 128 bits, which renders the privacy guarantees of existing DAE schemes built upon them void already after encrypting about $2^{64}$ blocks under the same key. However, since the highest attainable security is desirable for the protection of cryptographic keys, beyond-birthday-bound (BBB) security is a highly valuable goal for DAE schemes.
At least two straight-forward approaches for achieving BBB security exist in this context: first, by increasing the block size of the underlying cipher [20,21] or second, by using a wide-block permutation or compression function instead [6,13,32]. Though, previous wide-block constructions possessed significant disadvantages in terms of memory and performance, among which the latter aspect was attempted to be compensated by optimistic reduction of the underlying primitive [6,13], which implies the need for further cryptanalysis. The present work shows that neither the number of rounds nor the state size of the underlying cipher need to be modified to achieve our goal in a performant manner with the help of recent advances in the domain of tweakable block ciphers.

***Relations to Wide-Block Ciphers.*** The birthday-bound limit is also relevant for wide-block ciphers and Tweakable Enciphering Schemes (TES). TES are closely related to DAE: Hoang et al. [18] showed that the Encode-then-Encipher [4] approach can be used to transform an STPRP-secure (Strong Tweakable Pseudo-Random Permutation) TES into a provably robust AE scheme using (a hash of) the associated data as tweak. Such designs could offer even more security than necessary for DAE, i.e., *best achievable* AE security [2,18]. Thus, one could theoretically adapt any existing BBB-secure TES scheme for DAE [26,27,40]. Though, for the popular approaches Hash-Encrypt-Hash [38], Hash-Counter-Hash [41], and Protected IV [40], this strategy would also imply more operations than necessary for DAE, i.e., three passes over the plaintext. While Encrypt-Mix-Encrypt-based [17] designs employ only two passes, a BBB-secure variant of Encrypt-Mix-Encrypt with a $2n$-bit primitive would be considerably less efficient than our proposal. Therefore, a motivating observation of this work is the following: one can encode $\tau$ bits of redundancy into the message, encrypt it with the Hash-Counter-Hash approach reduced from three to two passes, and can obtain a BBB-secure DAE scheme.

A recent proposal is SIMPIRA [15,16], a family of $128b$-bit cryptographic permutations based on the AES round function. In contrast to the "modes" above, SIMPIRA is a primitive on its own. The initial version [15] based on a flawed general Feistel structure for $b = 4$, which was fixed in SIMPIRAV2 [16]. Though, the prior attacks [12,36] indicated that SIMPIRA may require more intensive studies to become fully mature. While our example instantiation employs its fixed version with $b = 2$ blocks for which no attacks are known, it can be seemlessly replaced by another secure $2n$-bit SPRP.

***Contribution.*** This work proposes Deterministic Counter in Tweak (DCT), a BBB-secure DAE scheme that combines an $\epsilon$-almost-XOR-universal (AXU) family of hash functions, for $\epsilon \approx \mathcal{O}(2^{-2n})$, with a single call to a $2n$-bit SPRP, and a $2^n$-block-secure encryption scheme. First, we propose our construction generically with three independent keys, one for each component. Next, we introduce an efficient instantiation which (1) provides software efficiency by encrypting at less than two cycles per byte on current x64 processors, (2) requires only a single key, and (3) produces only the minimal $\tau$-bit stretch. We leave open two minor aspects for future work: our current generic construction is defined for messages of $\geq 2n - \tau$ bits, and the verification algorithm requires the inverse of the SPRP and the encryption scheme. Though, since our instantiation uses a Feistel-based two-block construction as $2n$-bit SPRP and counter mode as encryption scheme, its decryption can fully reuse the components for encryption.

*Remark 1.* We stress that the HBS [21] and BTM [20] constructions by Iwata and Yasuda are similar to our work, and that Iwata and Yasuda already discussed BBB-secure adaptions. Both their BBB variants suggested the use of a $2n$-bit block cipher for encryption. Their earlier concept employed a (clearly pointed out by the authors to be inefficient) six-round Feistel network [21]; their later construction [20] used a $2n$-bit tweakable block cipher by Minematsu [26]. Both designs still required several keys, lacked software efficiency, and produced a $2n$-bit stretch. This work addresses their open questions.

***Outline.*** The rest of this paper is structured as follows: after briefly reviewing preliminaries, Section 3 describes the generic DCT framework. Section 4 recalls relevant security notions. Section 5 summarizes our security analysis, Section 6 details our instantiation, and Section 7 discusses our proposal and concludes.

## 2 Preliminaries

We use lowercase letters $x, y$ for indices and integers, uppercase letters $X, Y$ for binary strings and functions, and calligraphic uppercase letters $\mathcal{X}, \mathcal{Y}$ for sets. $\varepsilon$ denotes the empty string. We denote the concatenation of binary strings $X$ and $Y$ by $X \,\|\, Y$ and the result of their bitwise XOR by $X \oplus Y$. We indicate the length of $X$ in bits by $|X|$, and write $X_i$ for the $i$-th block, $X[i]$ for the $i$-th most significant bit of $X$, and $X[i..j]$ for the bit sequence $X[i], \ldots, X[j]$. $X \twoheadleftarrow \mathcal{X}$ denotes that $X$ is chosen uniformly at random from the set $\mathcal{X}$. We

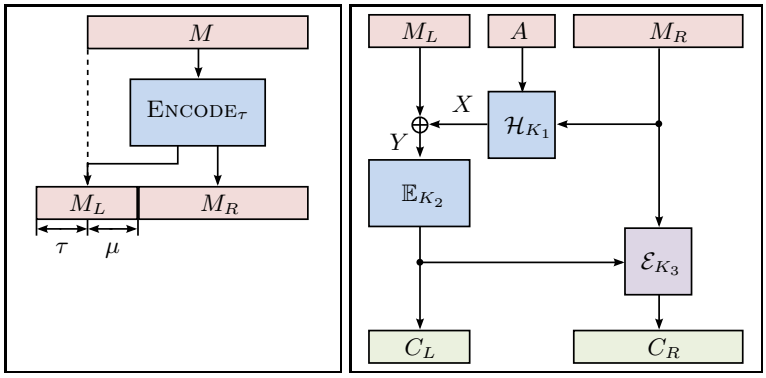


**Fig. 1:** The encrpytion process of DCT (**right**). The encoding (**left**) encodes $\tau$ bits of redundancy into the message $M$ and splits it into a $2n$-bit part $M_L$ and a variable-length part $M_R$ such that the redundancy is fully contained in $M_L$.

define three sets of particular interest: $\mathsf{Perm}(\mathcal{X})$ be the set of all permutations on $\mathcal{X}$, $\widetilde{\mathsf{Perm}}(\mathcal{T},\mathcal{X})$ the set of all tweaked permutations over $\mathcal{X}$ with associated non-empty tweak space $\mathcal{T}$, and $\mathsf{Func}(\mathcal{X},\mathcal{Y})$ the set of all functions $F : \mathcal{X} \to \mathcal{Y}$. We define by $X_1, \ldots, X_j \xleftarrow{x} X$ the injective splitting of the string $X$ into $x$-bit blocks such that $X = X_1 \,\|\, \cdots \,\|\, X_j$, with $|X_i| = x$ for $1 \le i \le j-1$, and $|X_j| \le x$. For an event $E$, we denote by $\Pr[E]$ the probability of $E$. We write $\langle x \rangle_n$ for the binary representation of an integer $x$ as an $n$-bit string, or short $\langle x \rangle$ if $n$ is clear from the context, in big-endian manner, i.e., the decimal $\langle 135 \rangle$ is encoded to $(00..00100000111)_2$.

## 3 Generic Definition of DCT

This section defines the generic DCT construction. Fix integers $n, \tau \ge 1$ with $\tau \le 2n$, and derive $\mu = 2n - \tau$. Let $\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{K}_3$ be non-empty key spaces and $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{K}_3$. Let $\mathcal{A} \subseteq \{0,1\}^*$ denote the associated-data space, $\mathcal{M} \subseteq \{0,1\}^{\ge \mu}$ the message space, and $\mathcal{C} \subseteq \{0,1\}^{\ge 2n}$ denote the ciphertext space, respectively. Let $\mathcal{H} = \left\{H | H : \mathcal{A} \times \{0,1\}^* \to \{0,1\}^{2n}\right\}$ be a family of $\epsilon$-AXU hash functions, indexed by elements from $\mathcal{K}_1$. Let $\mathbb{E} : \mathcal{K}_2 \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ denote a keyed permutation, and let $\Pi = (\mathcal{E}, \mathcal{D})$ be an IV-based encryption scheme (covered in the next section) with a non-empty key space $\mathcal{K}_3$ and an IV space $\mathcal{IV} = \{0,1\}^{2n}$.

***Encoding.*** Let $\textsc{Encode} : \mathbb{N}_0 \times \mathcal{M} \to \{0,1\}^{2n} \times \{0,1\}^*$ be an injective function that takes an integer $\tau \in \mathbb{N}_0$ and a bit string $M$ as inputs and produces two outputs $(M_L, M_R)$ such that $|M_L| = 2n$ and $|M_R| = |M| - \mu$. Since $\textsc{Encode}_\tau(\cdot)$ is injective, there exists a corresponding unique decoding function $\textsc{Decode} : \mathbb{N}_0 \times \{0,1\}^{2n} \times \{0,1\}^{\ge \mu} \to \{0,1\}^* \cup \{\perp\}$ such that, for a fixed $\tau \in \mathbb{N}_0$ and all $X, Y \in \{0,1\}^{2n} \times \{0,1\}^*$, $\textsc{Decode}_\tau(X, Y)$ returns the unique $M \in \mathcal{M}$ such that $\textsc{Encode}_\tau(M) = (X, Y)$ if such an $M$ exists, and $\perp$ otherwise.

**Algorithm 1** Encryption and decryption of the generic DCT construction.

| | |
|---|---|
| 1: **function** $\widetilde{\mathcal{E}}_{K_1,K_2,K_3}(A, M)$ | 11: **function** $\widetilde{\mathcal{D}}_{K_1,K_2,K_3}(A, C)$ |
| 2: $(M_L, M_R) \leftarrow \textsc{Encode}_\tau(M)$ | 12: $(C_L, C_R) \leftarrow C$ |
| 3: $X \leftarrow \mathcal{H}_{K_1}(A, M_R)$ | 13: $M_R \leftarrow \mathcal{D}_{K_3}(C_L, C_R)$ |
| 4: $Y \leftarrow M_L \oplus X$ | 14: $X \leftarrow \mathcal{H}_{K_1}(A, M_R)$ |
| 5: $C_L \leftarrow \mathbb{E}_{K_2}(Y)$ | 15: $Y \leftarrow \mathbb{E}^{-1}_{K_2}(C_L)$ |
| 6: $C_R \leftarrow \mathcal{E}_{K_3}(C_L, M_R)$ | 16: $M_L \leftarrow X \oplus Y$ |
| 7: **return** $(C_L \,\|\, C_R)$ | 17: **return** $\textsc{Decode}_\tau(M_L, M_R)$ |

***Encryption.*** For encryption, $\textsc{Encode}_\tau(M)$ encodes $\tau$ bits redundancy into an input message $M$ and splits the result into a $2n$-bit part $M_L$, and a variable-length part $M_R$, such that the redundancy is *fully contained in* $M_L$.[1] The latter part, $M_R$, is hashed together with the associated data $A$ to a $2n$-bit hash value: $X \leftarrow \mathcal{H}_{K_1}(A, M_R)$. Next, $X$ is XORed to $M_L$, producing $Y \leftarrow X \oplus M_L$, and the result $Y$ is encrypted by $\mathbb{E}$ to the fixed-length part of the ciphertext: $C_L \leftarrow \mathbb{E}_{K_2}(Y)$. This composition of a hash function and a final call to a PRF is a well-known method for constructing an efficient PRF [8]. Next, the PRF output $C_L$ is used as IV for an encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ which enciphers the variable-length part of the message: $C_R \leftarrow \mathcal{E}_{K_3}(C_L, M_R)$. Finally, $(C_L \,\|\, C_R)$ is returned as the ciphertext. Figure 1 illustrates the encryption process schematically.

***Decryption.*** For decryption, the ciphertext $C$ is split into $(C_L, C_R) \leftarrow C$, such that $|C_L| = 2n$. The variable-length part $C_R$ is decrypted to $M_R \leftarrow \mathcal{D}_{K_3}(C_L, C_R)$. Next, the scheme evaluates $X \leftarrow \mathcal{H}_{K_1}(A, M_R)$ and $Y \leftarrow \mathbb{E}^{-1}_{K_2}(C_L)$, and XORs both results: $M_L \leftarrow X \oplus Y$. $\textsc{Decode}_\tau(M_L, M_R)$ can either efficiently remove the redundancy from $M_L$ and determine $M$ with $\textsc{Encode}_\tau(M) = (M_L, M_R)$ if such an $M$ exists; otherwise, it can efficiently detect the invalid redundancy. The decryption returns $M$ in the former case, and $\perp$ in the latter.

***Limitations.*** We define DCT for messages of length at least $\mu$ bits, and ciphertexts of at least $2n$ bits length. For simplicity, we assume that, whenever smaller plain- or ciphertexts are passed to the encryption or decryption algorithms, respectively, the response will be $\perp$. We are aware of this current limitation of our proposal, and work actively to overcome it.

**Definition 1 (Generic DCT).** *Given the definitions above, we define the* DAE *scheme* $\mathrm{DCT}_{\mathcal{H},\mathbb{E},\Pi} = (\widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ *with deterministic encryption algorithm* $\widetilde{\mathcal{E}} : \mathcal{K} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$*, and deterministic decryption algorithm* $\widetilde{\mathcal{D}} : \mathcal{K} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$*, as given in Algorithm 1.*

For all $K \in \mathcal{K}$, $A \in \mathcal{A}$, $M \in \mathcal{M}$, and $C \in \mathcal{C}$ holds: if $\widetilde{\mathcal{E}}^A_K(M) = C$, then $\widetilde{\mathcal{D}}^A_K(C) = M$, and if $\widetilde{\mathcal{D}}^A_K(C) = M \neq \perp$, then $\widetilde{\mathcal{E}}^A_K(M) = C$.

[1] Note that encoding redundancy into $M_R$ would require a chosen-ciphertext-secure encryption scheme $\Pi$.

## 4 Security Notions

### 4.1 Adversaries and Advantages

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

$$\underset{\mathbf{A}}{\Delta}(\mathcal{O}^L; \mathcal{O}^R) := \left| \Pr\left[\mathbf{A}^{\mathcal{O}^L} \Rightarrow 1\right] - \Pr\left[\mathbf{A}^{\mathcal{O}^R} \Rightarrow 1\right] \right|$$

for the advantage of $\mathbf{A}$ to distinguish between oracles $\mathcal{O}^L$ and $\mathcal{O}^R$. All probabilities are defined over the random coins of the oracles and those of the adversary, if any. We write $\mathbf{Adv}_F^X(q, \ell, t) := \max_{\mathbf{A}} \left\{ \mathbf{Adv}_F^X(\mathbf{A}) \right\}$ for the maximal advantage over all $X$-adversaries $\mathbf{A}$ on $F$ that run in time at most $t$ and pose at most $q$ queries of at most $\ell$ blocks in total to its oracles. Wlog., we assume that $\mathbf{A}$ never asks queries to which it already knows the answer.
If the oracles $\mathcal{O}_i$, $\mathcal{O}_j$ represent a family of algorithms indexed by inputs, the indices must match. For example, when $\widetilde{\mathcal{E}}_K^A(M)$ and $\widetilde{\mathcal{D}}_K^A(C)$ represent encryption and decryption algorithms with a fixed key $K$ and indexed by $A$, then $\widetilde{\mathcal{E}}_K \hookrightarrow \widetilde{\mathcal{D}}_K$ says that $\mathbf{A}$ queries first $\widetilde{\mathcal{E}}_K^A(M)$ and later $\widetilde{\mathcal{D}}_K^A(C)$. We often write $\widetilde{\mathcal{E}}_K^A(M)$ and $\widetilde{\mathcal{D}}_K^A(C)$ as short forms for $\widetilde{\mathcal{E}}(K, A, M)$ and $\widetilde{\mathcal{D}}(K, A, C)$.
We define $\perp$, when in place of an oracle, to always return the invalid symbol $\perp$. We define $\$^{\mathcal{O}}$ for an oracle that, given an input $X$, chooses uniformly at random a value $Y$ equal in length of the expected output, $|Y| = |\mathcal{O}(X)|$, and returns $Y$. We assume that $\$^{\mathcal{O}}$ performs lazy sampling, i.e., $\$^{\mathcal{O}}(X)$ returns the same value $Y$ when queried with the same input $X$. We often omit the key for brevity, e.g., $\$^{\widetilde{\mathcal{E}}}(X)$ will be short for $\$^{\widetilde{\mathcal{E}}_K}(X)$.

### 4.2 Security Definitions for Universal Hashing

**Definition 2 ($\epsilon$-Almost-(XOR-)Universal Hash Functions).** *Let $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^*$. Let $\mathcal{H} = \{H \mid H : \mathcal{X} \to \mathcal{Y}\}$ denote a family of hash functions. $\mathcal{H}$ is called $\epsilon$-almost-universal ($\epsilon$-AU) iff for all distinct elements $X, X' \in \mathcal{X}$, it holds that $\Pr_{H \leftarrow\!\!\$\, \mathcal{H}}[H(X) = H(X')] \leq \epsilon$.*
*$\mathcal{H}$ is called* $\epsilon$-almost-XOR-universal *($\epsilon$-AXU) iff for all distinct elements $X, X' \in \mathcal{X}$ and $Y \in \mathcal{Y}$, it holds that $\Pr_{H \leftarrow\!\!\$\, \mathcal{H}}[H(X) \oplus H(X') = Y] \leq \epsilon$.*

In [7], Boesgaard et al. showed the following theorem.

**Theorem 1 (Theorem 3 from [7]).** *Let $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^*$. Further, let $\mathcal{H} = \{H \mid H : \mathcal{X} \to \mathcal{Y}\}$ be a family of $\epsilon$-AXU hash functions. Then, the family $\mathcal{H}' = \{H' \mid H' : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}\}$ with $H'(X, Y) := H(X) \oplus Y$, is $\epsilon$-AU.*

### 4.3 Security Definitions for Functions and Ciphers

**Definition 3 ((Strong) PRP Advantage).** *Fix integers $n, k \geq 1$. Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and $\mathbf{A}$ ($\mathbf{A}'$) be a computationally*

*bounded adversary with access to an oracle (two oracles). Let* $K \twoheadleftarrow \{0,1\}^k$ *and* $\pi \twoheadleftarrow \mathsf{Perm}(\{0,1\}^n)$*. Then, the* PRP *and* SPRP *advantages of* $\mathbf{A}$ *and* $\mathbf{A}'$ *with respect to* $E$ *are defined as* $\mathbf{Adv}^{\mathrm{PRP}}_{E}(\mathbf{A}) := \Delta_{\mathbf{A}}(E_K; \pi)$ *and* $\mathbf{Adv}^{\mathrm{SPRP}}_{E,E^{-1}}(\mathbf{A}') := \Delta_{\mathbf{A}'}(E_K, E_K^{-1}; \pi, \pi^{-1})$*, respectively.*

**Definition 4 ((Strong) Tweakable PRP Advantage).** *Fix two integers* $n, k \geq 1$*. Let* $\mathcal{T}$ *denote a non-empty set. Let* $\widetilde{E} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *be a tweakable block cipher and* $\mathbf{A}$ *(*$\mathbf{A}'$*) a computationally bounded adversary with access to an oracle (two oracles). Let* $K \twoheadleftarrow \{0,1\}^k$ *and* $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \{0,1\}^n)$*. Then, the* TPRP *and* STPRP *advantages of* $\mathbf{A}$ *and* $\mathbf{A}'$ *with respect to* $\widetilde{E}$ *are defined as* $\mathbf{Adv}^{\mathrm{TPRP}}_{\widetilde{E}}(\mathbf{A}) := \Delta_{\mathbf{A}}(\widetilde{E}_K; \widetilde{\pi})$ *and* $\mathbf{Adv}^{\mathrm{STPRP}}_{\widetilde{E},\widetilde{E}^{-1}}(\mathbf{A}') := \Delta_{\mathbf{A}'}(\widetilde{E}_K, \widetilde{E}_K^{-1}; \widetilde{\pi}, \widetilde{\pi}^{-1})$*, respectively.*

### 4.4 Security Definitions for IV-Based Encryption Schemes

An *IV-based encryption scheme* [3] is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ of encryption and decryption algorithms $\mathcal{E} : \mathcal{K} \times \mathcal{IV} \times \mathcal{M} \to \mathcal{C}$ and $\mathcal{D} : \mathcal{K} \times \mathcal{IV} \times \mathcal{C} \to \mathcal{M}$, with associated non-empty key space $\mathcal{K}$, non-empty IV space $\mathcal{IV}$, and where $\mathcal{M}, \mathcal{C} \subseteq \{0,1\}^*$ denote message and ciphertext space, respectively. We assume *correctness* for all $K \in \mathcal{K}$, $IV \in \mathcal{IV}$, and $M \in \mathcal{M}$, i.e., if $\mathcal{E}_K^{IV}(M) = C$, then $\mathcal{D}_K^{IV}(C) = M$. Moreover, we assume *tidiness*, i.e., if $\mathcal{D}_K^{IV}(C) = M$, then $\mathcal{E}_K^{IV}(M) = C$. The security of IV-based encryption schemes is defined as the distinguishing advantage from random bits. For every query $M$, the encryption oracle samples uniformly at random $IV \twoheadleftarrow \mathcal{IV}$ and computes $C \leftarrow \mathcal{E}_K^{IV}(M)$. The real oracle outputs $(IV, C)$, whereas $\$^{\mathcal{E}}$ outputs $|(IV \,\|\, C)|$ random bits.

**Definition 5 (ivE Advantage).** *Let* $\Pi = (\mathcal{E}, \mathcal{D})$ *be an IV-based encryption scheme and* $K \twoheadleftarrow \mathcal{K}$*. Let* $\mathbf{A}$ *be a computationally bounded adversary with access to an oracle. Then, the* ivE *advantage of* $\mathbf{A}$ *with respect to* $\Pi$ *is defined as* $\mathbf{Adv}^{\mathrm{ivE}}_{\Pi}(\mathbf{A}) := \Delta_{\mathbf{A}}(\mathcal{E}_K; \$^{\mathcal{E}})$*.*

### 4.5 Security Definitions for DAE Schemes

A deterministic AE scheme [34] is a tuple $\widetilde{\Pi} = (\widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ of deterministic algorithms $\widetilde{\mathcal{E}} : \mathcal{K} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$ and $\widetilde{\mathcal{D}} : \mathcal{K} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ with associated non-empty key space $\mathcal{K}$, associated-data space $\mathcal{A}$, and message/ciphertext space $\mathcal{M}, \mathcal{C} \subseteq \{0,1\}^*$. For each $K \in \mathcal{K}$, $A \in \mathcal{A}$, $M \in \mathcal{M}$, $\widetilde{\mathcal{E}}_K^A(M)$ maps $(A, M)$ to an output $C$ such that $|C| = |M| + \tau$ for fixed stretch $\tau$. $\widetilde{\mathcal{D}}_K^A(C)$ outputs the corresponding message $M$ iff $C$ is valid, and $\perp$ otherwise. We assume correctness, i.e., for all $K, A, M \in \mathcal{K} \times \mathcal{A} \times \mathcal{M}$, it holds that $\widetilde{\mathcal{D}}_K^A(\widetilde{\mathcal{E}}_K^A(M)) = M$. Moreover, we assume tidiness, i.e., if there exists an $M$ such that $\widetilde{\mathcal{D}}_K^A(C) = M$, then it holds that $\widetilde{\mathcal{E}}_K^A(\widetilde{\mathcal{D}}_K^A(C)) = C$.

**Definition 6 (detPriv, detAuth, and DAE Advantages [34]).** *Let* $\widetilde{\Pi} = (\widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ *be a DAE scheme and* $K \twoheadleftarrow \mathcal{K}$*. Let* $\mathbf{A}_1$*,* $\mathbf{A}_2$*,* $\mathbf{A}_3$ *denote computationally bounded adversaries;* $\mathbf{A}_1$ *has access to one oracle;* $\mathbf{A}_2$ *and* $\mathbf{A}_3$ *have access to*

*two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ each. $\mathbf{A}_2$ and $\mathbf{A}_3$ never submit queries $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$. Then, the* detPriv, detAuth, *and* DAE *advantages of $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ with respect to $\widetilde{\Pi}$, are defined as*

$$\begin{aligned}\mathbf{Adv}^{\text{detPriv}}_{\widetilde{\Pi}}(\mathbf{A}_1) &:= \underset{\mathbf{A}_1}{\Delta}(\widetilde{\mathcal{E}}_K; \$^{\widetilde{\mathcal{E}}}),\\ \mathbf{Adv}^{\text{detAuth}}_{\widetilde{\Pi}}(\mathbf{A}_2) &:= \Pr\left[\mathbf{A}_2^{\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K} \text{ forges}\right], \text{ and}\\ \mathbf{Adv}^{\text{DAE}}_{\widetilde{\Pi}}(\mathbf{A}_3) &:= \underset{\mathbf{A}_3}{\Delta}(\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K; \$^{\widetilde{\mathcal{E}}}, \perp),\end{aligned}$$

*where "forges" means that $\widetilde{\mathcal{D}}_K$ returns anything other than $\perp$ for a query.*

**Theorem 2 (Proposition 8 in [35]).** *Let $\widetilde{\Pi} = (\widetilde{\mathcal{E}}, \widetilde{\mathcal{D}})$ be a DAE scheme and $K \twoheadleftarrow \mathcal{K}$. Let $\mathbf{A}$ be a computationally bounded* DAE *adversary on $\widetilde{\Pi}$ with access to two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ such that $\mathbf{A}$ never queries $\mathcal{O}_1 \hookrightarrow \mathcal{O}_2$, and $\mathbf{A}$ runs in time at most $t$ and submits at most $q$ queries of at most $\ell$ blocks in total. Then, there exist a computationally bounded* detPriv *adversary $\mathbf{A}_1$ and a computationally bounded* detAuth *adversary $\mathbf{A}_2$ both on $\widetilde{\Pi}$, such that*

$$\mathbf{Adv}^{\text{DAE}}_{\widetilde{\Pi}}(\mathbf{A}) \le \mathbf{Adv}^{\text{detPriv}}_{\widetilde{\Pi}}(\mathbf{A}_1) + \mathbf{Adv}^{\text{detAuth}}_{\widetilde{\Pi}}(\mathbf{A}_2),$$

*where $\mathbf{A}_1$ and $\mathbf{A}_2$ make at most $q$ queries of at most $\ell$ blocks and run in time $O(t)$ each.*

## 5 Security Results for the Generic DCT Construction

This section summarizes the security bounds for the generic DCT construction.

**Theorem 3 (DAE Security of Generic DCT).** *Let $\widetilde{\Pi} = \text{DCT}_{\mathcal{H},\mathbb{E},\Pi}$ be as defined in Definition 1. Let $\mathbf{A}$ be a computationally bounded* DAE *adversary on $\widetilde{\Pi}$ that asks at most $q$ queries of at most $\ell$ blocks in total, and runs in time at most $t$. Then, $\mathbf{Adv}^{\text{DAE}}_{\widetilde{\Pi}}(\mathbf{A})$ is upper bounded by*

$$\frac{3q^2\epsilon}{2} + \frac{2q^2}{2^{2n}} + \frac{3q\epsilon \cdot 2^{2n}}{2^{\tau}} + 3 \cdot \mathbf{Adv}^{\text{SPRP}}_{\mathbb{E},\mathbb{E}^{-1}}(q, O(t)) + 2 \cdot \mathbf{Adv}^{\text{IvE}}_{\Pi}(q, \ell, O(t)).$$

The proof of Theorem 3 follows from Theorem 2 and the individual bounds for the detPriv and detAuth security in Lemmata 1 and 2. The proofs are deferred to Appendices A and B.

**Lemma 1 (detPriv Security of Generic DCT).** *Let $\widetilde{\Pi} = \text{DCT}_{\mathcal{H},\mathbb{E},\Pi}$ be as defined in Definition 1. Let $\mathbf{A}$ be a computationally bounded* detPriv *adversary on $\widetilde{\Pi}$ that submits at most $q$ queries of at most $\ell$ blocks in total and runs in time at most $t$. Then*

$$\mathbf{Adv}^{\text{detPriv}}_{\widetilde{\Pi}}(\mathbf{A}) \le q^2\left(\epsilon + \frac{2}{2^{2n}}\right) + 2\left(\mathbf{Adv}^{\text{PRP}}_{\mathbb{E}}(q, O(t)) + \mathbf{Adv}^{\text{IvE}}_{\Pi}(q, \ell, O(t))\right).$$

**Lemma 2 (detAuth Security of Generic DCT).** *Let* $\widetilde{\Pi} = \mathrm{DCT}_{\mathcal{H},\mathbb{E},\Pi}$ *be as defined in Definition 1. Let* $\mathbf{A}$ *be a computationally bounded* detAuth *adversary on* $\widetilde{\Pi}$ *that submits at most* $q$ *queries of at most* $\ell$ *blocks in total, and runs in time at most* $t$*. Then*

$$\mathbf{Adv}^{\text{detAuth}}_{\widetilde{\Pi}}(\mathbf{A}) \leq \epsilon \cdot \left(\frac{q^2}{2} + \frac{3q \cdot 2^{2n}}{2^{\tau}}\right) + \mathbf{Adv}^{\text{SPRP}}_{\mathbb{E},\mathbb{E}^{-1}}(q, O(t)).$$

## 6 Instantiation

Our proposed instantiation of DCT requires (1) an efficient $2n$-bit SPRP, (2) a beyond-birthday-bound-secure IV-based encryption scheme, and (3) an $\epsilon$-AXU family of hash functions. This section describes the components in detail.

### 6.1 Components

***Efficient Tweakable Block Ciphers.*** The TWEAKEY framework by Jean et al. [22] provides a set of software-efficient tweakable block ciphers based on the AES. Therefore, they allow to exploit AES native instructions on current x64 processor architectures. Among the three available TWEAKEY ciphers Kiasu-BC, Joltik-BC, and Deoxys-BC, we concentrate on Deoxys-BC-128-128 [23] for its support of 128-bit tweaks. In the remainder, we denote it as $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$, with a state size of $n = 128$ bits, and with $\mathcal{K} = \mathcal{T} = \{0,1\}^n$.

***$2n$-bit* SPRP.** Simpira [15] is a recently proposed family of $128b$-bit cryptographic permutations based on the AES round function by Gueron and Mouha. We employ Simpira with two-block inputs ($b = 2$), which is similar to a Feistel network with 15 rounds (the output halves are swapped). The round function $F$ consists of two AES rounds (an AES round is also denoted by `aesenc` in the pseudocode); the first AES round in $F$ uses a round counter $c$ and $b = 2$ as key; the second round an all-zeroes key. The construction is used in an Even-Mansour design, so that a 256-bit ciphertext is computed by $C_L \leftarrow \text{Simpira}(Y \oplus (K \,\|\, 0^{128})) \oplus (K \,\|\, 0^{128})$, with a 128-bit secret key $K$, and where $Y \leftarrow M_L \oplus X$.
Since it is hard to prove the security of Simpira, a possible provably secure $2n$-bit SPRP would be the $\Psi_3$ construction by Coron et al. [11], which consists of three invocations of a tweakable block cipher; though, this construction is a little less performant than our current choice and requires the inverse for decryption.

***Encryption Scheme.*** The recently proposed CTRT mode by Peyrin and Seurin [29] is an IV-based encryption scheme that can provide security for close to $2^n$ blocks encrypted under the same key. Originally, the authors proposed it as a *nonce-IV*-based mode that requires an $n$-bit nonce $V$ as input to the block cipher $\widetilde{E}$ and an $n$-bit IV as tweak that is converted by a regular function[2] $\text{Conv} : \{0,1\}^n \to \mathcal{T}$ to an $(n-d)$-bit tweak for a fixed $d \leq n$.

[2] $F : \mathcal{X} \to \mathcal{Y}$ is called regular iff all outputs $Y \in \mathcal{Y}$ are produced by an equal number of preimages $X \in \mathcal{X}$.

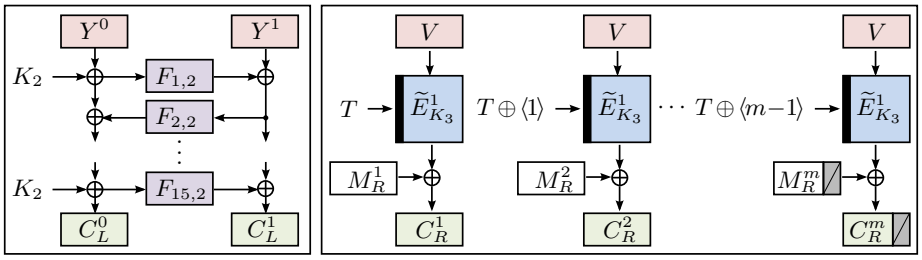


**Fig. 2:** The components of our instantiation of DCT: the two-block Simpira construction [15] for $\mathbb{E}$ (**left**) and the CTRT$[\widetilde{E}]$ mode [29] (**right**) for $\mathcal{E}$. $F$ denotes two AES rounds used in the Simpira construction.

Basically, CTRT represents a counter mode built upon a tweakable block cipher, where only the tweak is incremented for each block. We denote encryption and decryption algorithms, instantiated with a tweakable block cipher $\widetilde{E}$, by CTRT$[\widetilde{E}] =$ (CTRT.$\mathcal{E}[\widetilde{E}]$, CTRT.$\mathcal{D}[\widetilde{E}]$). The $i$-th message block $M_i$ is encrypted to a ciphertext block $C_i$ by $C_i \leftarrow \widetilde{E}_K^{D \,\|\, T+i}(V) \oplus M_i$, where $D \in \{0,1\}^d$ denotes the domain. For our instantiation of DCT, we adopt the purely IV-based variant of CTRT from [29, Appendix C], with a minor modification: to eliminate carry-bit concerns, we XOR the counter to the tweak instead of adding it modulo $2^n$. Clearly, since the IV is expected to be random, this modification does not change the probability distribution of tweaks to occur. Thus, the bounds from [29] apply to our adapted mode in a straight-forward manner. Our variant is defined in Algorithm 2; the encryption process is depicted in Figure 2.
Similar to [23,29], we encode a domain into the tweak to simplify our security analysis and to avoid multiple keys for the instances of $\widetilde{E}$ inside CTRT$[\widetilde{E}]$ and for key generation. For the calls to $\widetilde{E}$ inside CTRT$[\widetilde{E}]$, we set the most significant bit to 1 as domain and truncate the most significant bit of the IV $U$ to derive the tweak: $T \leftarrow U[2..n]$.

***Universal Hash Function.*** We considered several approaches for efficient hashing. Recent works pointed out weak-key issues [1,30,37] of Horner-based polynomials (e.g. GHASH or Poly1305) that modern AE schemes should avoid, which motivated our choice of BRW polynomials [5,31]. If it will turn out that similar attacks apply also to Bernstein-Rabin-Winograd (BRW) polynomials, one can easily switch to a different family of hash functions with similar security guarantees as our construction.
BRW polynomials require only a single $n$-bit key, half the number of multiplications compared to Horner-based polynomials, and a negligible number of $\lceil \log_2(m) \rceil$ additional squarings. Hereafter, we denote by $\mathbb{GF}(2^n)$ the Galois Field with a given irreducible polynomial $p(x)$ of degree $n$. We represent the elements in the field by $n$-bit strings. In this context, we use big-endian encoding where the most significant bit is on the left, e.g., $M = (10000111)_2$ represents the polynomial $\mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + \mathtt{1}$. For $n = 128$, we fix $p(x) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + \mathtt{1}$. Given an

**Algorithm 2** Definition of our instantiation of DCT, with $n = 128$ and $\tau \leq 2n$.

**function** $\widetilde{\mathcal{E}}_{SK}(A, M)$
  $(K_1^1, K_1^2, K_2, K_3) \leftarrow \textsc{KeyGen}(SK)$
  $(M_L, M_R) \leftarrow \textsc{Encode}_\tau(M)$
  $X \leftarrow \mathcal{H}_{K_1^1 \,\|\, K_1^2}(A, M_R)$
  $Y \leftarrow M_L \oplus X$
  $C_L \leftarrow \textsc{Simpira}_{K_2}(Y)$
  $C_R \leftarrow \text{CTRT}.\mathcal{E}[\widetilde{E}]_{K_3}(C_L, M_R)$
  **return** $(C_L \,\|\, C_R)$

**function** $\textsc{Encode}_\tau(M)$
  $\mu \leftarrow 2n - \tau$
  $M_L \leftarrow 0^\tau \,\|\, M[1..\mu]$
  $M_R \leftarrow M[(\mu+1)..|M|]$
  **return** $(M_L, M_R)$

**function** $\textsc{Simpira}_K(Y)$
  $(Y^0, Y^1) \xleftarrow{n} Y$
  $Y^1 \leftarrow Y^1 \oplus K$
  **for** $c \leftarrow 1$ **to** 15 **do**
    $l \leftarrow (c-1) \bmod 2$
    $r \leftarrow c \bmod 2$
    $Y^r \leftarrow Y^r \oplus \textsc{Simpira}.F_{c,2}(Y^l)$
  $C_L^0 \leftarrow Y^0 \oplus K$
  $C_L^1 \leftarrow Y^1$
  **return** $(C_L^0 \,\|\, C_L^1)$

**function** $\textsc{Simpira}.F_{c,b}(X)$
  $a \leftarrow c \oplus b$
  $L \leftarrow (a, \texttt{0x10} \oplus a, \texttt{0x20} \oplus a, \texttt{0x30} \oplus a)$
  **return** $\texttt{aesenc}(\texttt{aesenc}(X, L), 0)$

**function** $\text{CTRT}.\mathcal{E}[\widetilde{E}]_K(IV, M_R)$
  $(U, V) \xleftarrow{n} IV$
  $T \leftarrow U[2..n]$
  $m \leftarrow \lceil M_R/n \rceil$
  $(M_R^1, \ldots, M_R^m) \xleftarrow{n} M_R$
  **for** $i \leftarrow 1$ to $m-1$ **do**
    $C_R^i \leftarrow \widetilde{E}_K^{1, T \oplus \langle i-1 \rangle}(V) \oplus M_R^i$
  $\kappa_m \leftarrow \widetilde{E}_K^{1, T \oplus \langle m-1 \rangle}(V)$
  $C_R^m \leftarrow \kappa_m[1..|M_R^m|] \oplus M_R^m$
  **return** $(C_R^1 \,\|\, \cdots \,\|\, C_R^m)$

**function** $\text{CTRT}.\mathcal{D}[\widetilde{E}]_K(IV, C_R)$
  **return** $\text{CTRT}.\mathcal{E}[\widetilde{E}]_K(IV, C_R)$

**function** $\widetilde{E}_K^{D,T}(X)$
  **return** $\textsc{Deoxys-BC-}n\textsc{-}n_K^{D \,\|\, T}(X)$

**function** $\widetilde{\mathcal{D}}_{SK}(A, C)$
  $(K_1^1, K_1^2, K_2, K_3) \leftarrow \textsc{KeyGen}(SK)$
  $(C_L, C_R) \leftarrow C$
  $M_R \leftarrow \text{CTRT}.\mathcal{D}[\widetilde{E}]_{K_3}(C_L, C_R)$
  $X \leftarrow \mathcal{H}_{K_1^1 \,\|\, K_1^2}(A, M_R)$
  $Y \leftarrow \textsc{Simpira}_{K_2}^{-1}(C_L)$
  $M_L \leftarrow X \oplus Y$
  **return** $\textsc{Decode}_\tau(M_L, M_R)$

**function** $\textsc{Decode}_\tau(M_L, M_R)$
  $R \leftarrow M_L[1..\tau]$
  $M \leftarrow M_L[(\tau+1)..2n] \,\|\, M_R$
  **if** $R = 0^\tau$ **then return** $M$
  **return** $\perp$

**function** $\textsc{Simpira}_K^{-1}(C_L)$
  $(C_L^0, C_L^1) \xleftarrow{n} C_L$
  $Y^0 \leftarrow C_L^0 \oplus K$
  $Y^1 \leftarrow C_L^1$
  **for** $c \leftarrow 15$ **downto** 1 **do**
    $l \leftarrow (c-1) \bmod 2$
    $r \leftarrow c \bmod 2$
    $Y^r \leftarrow Y^r \oplus \textsc{Simpira}.F_{c,2}(Y^l)$
  $Y^0 \leftarrow Y^0 \oplus K$
  **return** $(Y^0 \,\|\, Y^1)$

**function** $\textsc{KeyGen}(SK)$
  **for** $i \leftarrow 1$ **to** 4 **do**
    $K_i \leftarrow \widetilde{E}_{SK}^{0, \langle i \rangle}(\langle i \rangle)$
  **return** $(K_1, K_2, K_3, K_4)$

**function** $\mathcal{H}_K(A, M_R)$
  $W \leftarrow \textsc{Encode}'(A, M_R)$
  $(K_1^1, K_1^2) \xleftarrow{n} K$
  $H_1 \leftarrow K_1^1 \cdot \text{BRW}_{K_1^1}(W)$
  $H_2 \leftarrow K_1^2 \cdot \text{BRW}_{K_1^2}(W)$
  **return** $(H_1 \,\|\, H_2)$

**function** $\textsc{Encode}'(A, M)$
  $\overline{A} \leftarrow \textsc{pad}_n(A)$
  $\overline{M} \leftarrow \textsc{pad}_n(M)$
  $L \leftarrow \langle |A| \rangle_{64} \,\|\, \langle |M| \rangle_{64}$
  **return** $(\overline{A} \,\|\, \overline{M} \,\|\, L)$

**function** $\textsc{pad}_n(X)$
  **if** $|X| \bmod n = 0$ **then return** $X$
  **return** $(X \,\|\, 0^{n - (|X| \bmod n)})$

$m$-word message $M = (M_1, \ldots, M_m)$ and a key $K \in \{0,1\}^n$, the hash function $\text{BRW}_K(M)$ is defined recursively by

- $\text{BRW}_K(\varepsilon) := 0^n$ if $m = 0$,
- $\text{BRW}_K(M_1) := M_1$ if $m = 1$,
- $\text{BRW}_K(M_1, M_2) := (M_1 \cdot K) \oplus M_2$ if $m = 2$,
- $\text{BRW}_K(M_1, M_2, M_3) := (M_1 \oplus K) \cdot (M_2 \oplus K^2) \oplus M_3$ if $m = 3$,
- $\text{BRW}_K(M_1, \ldots, M_m) := \text{BRW}_K(M_1, \ldots, M_{t-1}) \cdot (M_t \oplus K^t) \oplus \text{BRW}_K(M_{t+1}, \ldots, M_m)$ if $t \leq m < 2t$ for $t \in \{4, 8, 16, 32, \ldots\}$,

where all multiplications are in $\mathbb{GF}(2^n)$. Since BRW hashing XORs the final block $M_m$ when $m$ is *not* a multiple of four, we perform an additional multiplication, $K \cdot BRW_K(M)$, to prevent predictable output differences.

Our family of hash functions – $\mathcal{H}$ or BRW-256 hereafter – takes as inputs the associated data $A$ and the variable-length part of the message $M_R$. Therefore, we define an injective encoding function $\textsc{Encode}' : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ for merging both inputs to a single bit string before hashing. First, $\textsc{Encode}'$ pads $\overline{A} \leftarrow \textsc{pad}_n(A)$ and $\overline{M_R} \leftarrow \textsc{pad}_n(M_R)$ with the minimal number of trailing zeroes such that their lengths after padding are multiples of $n$. Next, their original lengths in bits are encoded as two 64-bit big-endian-encoded integers: $L \leftarrow \langle |A| \rangle_{64} \,\|\, \langle |M_R| \rangle_{64}$. Finally, $\textsc{Encode}'$ returns $(\overline{A} \,\|\, \overline{M_R} \,\|\, L)$, which is used as input to $\mathcal{H}$. The procedure is given in Algorithm 2.

***Key Schedule and Change of Key.*** Our instantiation requires a 128-bit user-supplied secret key $SK$. In total, our instantiation of DCT uses four independent 128-bit key words: a 256-bit key $K_1^2 \,\|\, K_1^2$ for $\mathcal{H}$, a 128-bit key $K_3$ for $\mathbb{E}$, and a 128-bit key $K_3$ for $\widetilde{E}$ used in the CTRT mode. We borrow the idea from [19] of deriving the keys for the individual components with $\widetilde{E}$ under the secret $SK$ in counter mode with distinct tweaks; neither those tweaks nor $SK$ are used any further in our mode. So, the derived keys are pairwise independent. We recommend a default stretch of $\tau = 128$ bits, at most $2^{64}$ bits be encrypted under the same key, and the maximum query length be limited to $2^{40}$ blocks.

### 6.2 Concrete Security Bounds

We derive the following conjecture from the existing analysis of Simpira [16].

*Conjecture 1 (Security of two-block* Simpira*).* Let $n = 128$ and $b = 2$. Let $\mathbb{E}$ denote Simpira for $2n$-bit inputs. Let $\mathbf{A}$ be a computationally bounded SPRP adversary on $\mathbb{E}$ with access to two oracles, where $\mathbf{A}$ asks at most $q$ queries and runs in time at most $t$. Then, there exists an absolute constant $c$ such that

$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathbb{E},\mathbb{E}^{-1}}(\mathbf{A}) \leq \frac{c \cdot q}{2^n}.$$

Theorem 1 and Appendix C in [29] provide the following theorem.

**Theorem 4 (ivE Advantage of CTRT [29]).** *Fix $n \geq 1$. Let $\mathcal{T}$ be a non-empty set and $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \{0,1\}^n)$. Let $\mathbf{A}$ be an* ivE *adversary with access to an oracle, where $\mathbf{A}$ runs in time at most $t$ and poses at most $q$ queries to its oracles with at most $8 \leq \ell \leq |\mathcal{T}|$ blocks in total. Then*

$$\mathbf{Adv}^{\mathrm{ivE}}_{\mathrm{CTRT}[\widetilde{\pi}]}(\mathbf{A}) \leq \frac{1}{2^n} + \frac{1}{|\mathcal{T}|} + \frac{4\ell \log_2(q)}{|\mathcal{T}|} + \frac{\ell \log_2(\ell)}{2^n}.$$

Theorem 5.4 in [5] and Theorem 1 in [39] show that $\mathrm{BRW}_K(M_1, \ldots, M_m)$ is a monic polynomial of degree $2^{\lfloor \log_2 m \rfloor + 1} - 1 \leq 2m - 1$. The additional multiplications for the length-encoding block and the final multiplication with $K \cdot \mathrm{BRW}_K(M)$ lead to a monic polynomial of degree $2(m+1)$. For our proposed instantiation for $\mathcal{H}$, we can derive the following statement:

**Theorem 5 (BRW Hashing).** *Let $n, m \geq 1$, and let $\mathcal{X} = \bigcup_{i=0}^{m} \mathbb{GF}(2^n)^i$. Then, the family of hash functions $\mathcal{G} = \{\mathrm{BRW} \,|\, \mathrm{BRW} : \mathcal{X} \to \mathbb{GF}(2^n)\}$ is $\epsilon$-AXU for $\epsilon \leq 2(m+1)/2^n$. Moreover, the family of hash functions $\mathcal{H} = \{\mathrm{BRW}_1, \mathrm{BRW}_2 \twoheadleftarrow \mathcal{G} \times \mathcal{G} \,|\, H(M) := \mathrm{BRW}_1(M) \,\|\, \mathrm{BRW}_2(M)\}$ with independent $\mathrm{BRW}_1$ and $\mathrm{BRW}_2$ is $\epsilon'$-AXU for $\epsilon' \leq 4(m+1)^2/2^{2n}$.*

Inserting the concrete bounds from Conjecture 1 and Theorems 4 and 5 into those from Lemmata 1 and 2, we can derive the following security statements for our proposed instantiation $\mathrm{DCT}_{\text{BRW-256,SIMPIRA,CTRT}[\widetilde{E}]}$. Appendix C discusses how the quadratic dependency on $m'$ can be reduced to a linear one for $\tau \leq n$.

**Theorem 6.** *Let $K \twoheadleftarrow \mathcal{K}$ and let $\widetilde{\Pi}$ denote $\mathrm{DCT}_{\text{BRW-256,SIMPIRA,CTRT}[\widetilde{E}]}$ as defined in Algorithm 2. Let $n = 128$, $\tau \leq 2n$, $|\mathcal{T}| = 2^{n-1}$, $m$ be the sum of the maximal number blocks of message and associated data for each query, and $c$ be an absolute constant. Define $m' = m + 1$. Then, for $8 \leq \ell \leq |\mathcal{T}|$, it holds that*

$$\mathbf{Adv}^{\text{DETPRIV}}_{\widetilde{\Pi}}(q, \ell, t) \leq 2\left(\frac{q^2(2m'^2+1)}{2^{2n}} + \frac{3 + cq + 8\ell\log_2(q) + \ell\log_2(\ell)}{2^n} + \delta^{\text{TPRP}}_{\widetilde{E}}\right)$$

$$\mathbf{Adv}^{\text{DETAUTH}}_{\widetilde{\Pi}}(q, \ell, t) \leq \frac{2q^2m'^2}{2^{2n}} + \frac{12qm'^2}{2^{\tau}} + \frac{cq}{2^n} + \delta^{\text{STPRP}}_{\widetilde{E},\widetilde{E}^{-1}},$$

*where $\delta^{\text{TPRP}}_{\widetilde{E}}$ and $\delta^{\text{STPRP}}_{\widetilde{E},\widetilde{E}^{-1}}$ denote $\mathbf{Adv}^{\text{TPRP}}_{\widetilde{E}}(q, O(t))$ and $\mathbf{Adv}^{\text{STPRP}}_{\widetilde{E},\widetilde{E}^{-1}}(q, O(t))$, respectively.*

### 6.3 Software Performance on x64 Processors

We implemented an optimized version of our proposed instance in C.[3] Table 1 summarizes the results of our benchmarks. Our code was compiled using `gcc` v5.2.1 with options `-O3 -maes -mavx2 -mpclmul -march=native`, and run (1) on an Intel Core i5-4200M (Haswell) at 2.50 GHz, and (2) on an Intel Core i5-5200U (Broadwell), both with TurboBoost, SpeedStep, and HyperThreading options *disabled*. For measuring, we used the mean from 100 medians of 10000 encryptions each in the single-message setting, where we omitted the cost for key setup, and used the `rdtsc` instruction. Starting from 512 bytes, the values in Table 1 have a standard deviation of less than 0.02 cycles per byte (cpb). The results show that our proposed instance approaches a performance of less than two cpb on current x64 processors for messages of eight KiB and longer. The difference stems from the fact that the inverse throughput of the `pclmulqdq` instruction is two cycles per instruction on Intel Haswell, but only a single cycle on Broadwell processors.

While we are not aware of faster previous beyond-birthday AE schemes, we note the comparison with SIMPIRAV2 which was updated on ePrint while this work was under review [16]. While the leftmost Feistel lane in SIMPIRAV1 was sequential in the single-message setting for messages of $b > 8$ blocks, SIMPIRAV2 resolved this limitation. So, an optimal implementation of SIMPIRAV2

[3] The source code is publicly available at `https://github.com/medsec/dct`.

| | Message length (bytes) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Construction | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
| Haswell | 6.17 | 4.48 | 3.28 | 2.65 | 2.28 | 2.09 | 2.00 | 1.96 |
| Broadwell | 6.15 | 4.45 | 3.16 | 2.51 | 2.14 | 1.98 | 1.86 | 1.81 |

**Table 1:** Performance results in cycles per byte for optimized implementations of $\mathrm{DCT}_{\text{BRW-256},\textsc{Simpira},\mathrm{CTRT}[\tilde{E}]}$ on Haswell and Broadwell, respectively. Details of our benchmarking setup are given in the text.

could achieve a performance of a little less than 1.5 cpb for arbitrary-length messages in the single-message setting, which is a little faster than our instantiation. Though, future implementations might further reduce our currently achieved 1.07 cpb for Deoxys-BC-128-128 (the theoretical optimum is 0.875 cpb for its 14 AES rounds). Additionally, more aggressive optimizations similar to those for GHASH [14] could further improve the performance of our hash function.

## 7 Discussion and Conclusion

This work proposed Deterministic Counter in Tweak (DCT), a beyond-birthday-bound-secure DAE scheme that combines an almost-XOR-universal family of hash functions with a single call to a double-block-length SPRP, and a beyond-birthday-bound-secure encryption scheme. DCT produces the minimal stretch, e.g., $\tau = 128$ bit for 128-bit security. Our generic construction comes with a straight-forward security proof. We proposed a software-efficient instantiation that profits greatly from the recent progress in the domain of tweakable block ciphers and encryption schemes; in particular, from the TWEAKEY framework, the tweaked counter mode as encryption scheme, and the Simpira construction as $2n$-bit SPRP – both of which allow to exploit AES-NI instructions. As a result, our instantiation can encrypt at speeds of less than two cycles per byte on current x64 processors in the single-message setting. While our generic design employs three independent keys, our instantiation requires only a single 128-bit key and provides security close to that of our generic proposal. Moreover, the use of tweaked counter mode and the Feistel-based Simpira as SPRP yields an inverse-free decryption. DCT is currently defined for messages of $\geq 2n - \tau$ bits; one solution to also allow smaller messages could be to use a padding and two additional distinct tweaks $T$ for long and small messages, respectively. For example, Gueron and Mouha proposed to use $K \cdot T$ instead of $K$ as key for Simpira, using a multiplication in $\mathbb{GF}(2^{128})$. Yet, the detailed security and efficiency implications of this approach are interesting aspects for future work.

***Acknowledgments.*** We would like to thank the reviewers of the ACISP 2016 for their fruitful comments and Nicky Mouha for helpful insights into Simpira.

## References


1. Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted Polynomials and Forgery Attacks on GCM. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT I*, volume 9056 of *Lecture Notes in Computer Science*, pages 762–786, 2015.
2. Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. Robust Authenticated Encryption and the Limits of Symmetric Cryptography. In Jens Groth, editor, *IMA Int. Conf.*, volume 9496 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2015.
3. Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.
4. Mihir Bellare and Phillip Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000.
5. Daniel J. Bernstein. Polynomial evaluation and message authentication. *http://cr.yp.to/papers*, *permanent ID: b1ef3f2d385a926123e1517392e20f8c*, 2, 2007.
6. Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche, and Ronny Van Keer. Using Keccak technology for AE: Ketje, Keyak and more, August 22 2014. SHA-3 2014 Workshop, UC Santa Barbara.
7. Martin Boesgaard, Thomas Christensen, and Erik Zenner. Badger - A Fast and Provably Secure MAC. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 176–191, 2005.
8. Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
9. Debrup Chakraborty, Cuauhtemoc Mancillas-López, and Palash Sarkar. Disk Encryption: Do We Need to Preserve Length? *IACR Cryptology ePrint Archive*, 2015:594, 2015.
10. Debrup Chakraborty and Palash Sarkar. On modes of operations of a block cipher for authentication and authenticated encryption. *Cryptography and Communications*, pages 1–57, 2015.
11. Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A Domain Extender for the Ideal Cipher. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2010.
12. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Cryptanalysis of Simpira. *IACR Cryptology ePrint Archive*, 2016:244, 2016.
13. Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 263–293. Springer, 2016.
14. Shay Gueron and Yehuda Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM Conference on Computer and Communications Security*, pages 109–119. ACM, 2015.
15. Shay Gueron and Nicky Mouha. Simpira: A Family of Efficient Permutations Using the AES Round Function. *IACR Cryptology ePrint Archive*, 2016:122 version 20160214:005409, 2016.

16. Shay Gueron and Nicky Mouha. Simpira v2: A Family of Efficient Permutations Using the AES Round Function. *IACR Cryptology ePrint Archive*, 2016:122, 2016.
17. Shai Halevi and Phillip Rogaway. A Parallelizable Enciphering Mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.
18. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust Authenticated-Encryption AEZ and the Problem That It Solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44. Springer, 2015.
19. Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.
20. Tetsu Iwata and Kan Yasuda. BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 313–330. Springer, 2009.
21. Tetsu Iwata and Kan Yasuda. HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 394–415. Springer, 2009.
22. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288, 2014.
23. Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Deoxys v1.3, 2015. Second-round submission to the CAESAR competition, `http://competitions.cr.yp.to/-caesar-submissions.html`.
24. Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Joltik v1.3, 2015. Second-round submission to the CAESAR competition, `http://competitions.cr.yp.to/-caesar-submissions.html`.
25. Ted Krovetz. HS1-SIV. `http://competitions.cr.yp.to/caesar-submissions.html`, 2014.
26. Kazuhiko Minematsu. Beyond-Birthday-Bound Security Based on Tweakable Block Cipher. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2009.
27. Kazuhiko Minematsu. Building blockcipher from small-block tweakable blockcipher. *Designs, Code and Cryptography*, 74(3):645–663, 2015.
28. Kazuhiko Minematsu. Authenticated Encryption without Tag Expansion (or, How to Accelerate AERO). In *ACISP*, 2016. to appear.
29. Thomas Peyrin and Yannick Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. Cryptology ePrint Archive, Report 2015/1049, 2015.
30. Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks against Polynomial-based MAC Schemes. In Shiho Moriai, editor, *FSE*, volume 8424 of *Lecture Notes in Computer Science - Lecture Notes in Computer Science*, pages 287–304. Springer, 2013.
31. Michael O Rabin and Shmuel Winograd. Fast evaluation of polynomials by rational preparation. *Communications on Pure and Applied Mathematics*, 25(4):433–458, 1972.
32. Reza Reyhanitabar, Serge Vaudenay, and Damian Vizár. Misuse-Resistant Variants of the OMD Authenticated Encryption Mode. In Sherman S. M. Chow, Joseph K.

Liu, Lucas Chi Kwong Hui, and Siu-Ming Yiu, editors, *ProvSec*, volume 8782 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2014.
33. Phillip Rogaway. Nonce-Based Symmetric Encryption. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2004.
34. Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.
35. Phillip Rogaway and Thomas Shrimpton. Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem. Cryptology ePrint Archive, Report 2006/221. (Full Version), 2006. `http://eprint.iacr.org/`.
36. Sondre Rønjom. Invariant subspaces in Simpira. *IACR Cryptology ePrint Archive*, 2016:248, 2016.
37. Markku-Juhani Olavi Saarinen. Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.
38. Palash Sarkar. Improving Upon the TET Mode of Operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2007.
39. Palash Sarkar. Efficient Tweakable Enciphering Schemes from (Block-Wise) Universal Hash Functions. *IEEE Transactions on Information Theory*, 55(10):4749–4760, 2009.
40. Thomas Shrimpton and R. Seth Terashima. A Modular Framework for Building Variable-Input-Length Tweakable Ciphers. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 405–423. Springer, 2013.
41. Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A Variable-Input-Length Enciphering Mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *CISC*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.

## A Proof of Lemma 1

*Proof.* Wlog., we assume that $\mathbf{A}$ asks no pointless queries, i.e. queries to which it already knows the answer or that are prohibited by the detPriv definition. In the following, we upper bound $\Delta_{\mathbf{A}}(\widetilde{\mathcal{E}}_K; \$^{\widetilde{\mathcal{E}}})$ by a game-based approach, using a sequence of Games $G_1$ through $G_4$, where the encryption oracle of the first an final game are provided in Algorithm 3. It also shows the procedure Initialize and function Finalize for all games.

We start with an initial Game $G_1$ with an oracle $\widetilde{\mathcal{E}}$ that represents the detPriv encryption oracle for the generic DCT construction. We denote by $X \leftarrow \mathcal{H}_{K_1}(A, M_R)$ the outputs of $\mathcal{H}$, and by $Y \leftarrow X \oplus M_L$ the inputs to $\mathbb{E}$. The oracle collects the values $Y$ in a set $\mathcal{Q}_Y$ and the values $C_L$ in a set $\mathcal{Q}_{C_L}$. Both sets are initialized as empty sets. Additionally, the oracle sets a flag $\mathsf{bad}_1$ if a value $Y$ repeats and a flag $\mathsf{bad}_2$ if a value $C_L$ repeats, respectively. Since $G_1$ simply models the detPriv experiment for DCT and since the outputs of $G_1$ are influenced neither by the values collected in the sets nor by the bad flags, it holds that

$$\underset{\mathbf{A}}{\Delta}(\widetilde{\mathcal{E}}_K; \$^{\widetilde{\mathcal{E}}}) \leq 2 \cdot \left|\Pr\left[G_1(\mathbf{A}) \Rightarrow 1\right] - 0.5\right|.$$

We define a second Game $G_2$, which is identical to $G_1$, but which replaces all calls to $\mathbb{E}$ by calls to a uniform random function $\$^{\mathbb{E}} \twoheadleftarrow \mathsf{Func}(\{0,1\}^{2n}, \{0,1\}^{2n})$. We omit a separate algorithmic definition due to the minor change. We can upper bound the difference between both games by the maximal advantage of a PRP adversary $\mathbf{A}_1$ on $\mathbb{E}$ that submits at most $q$ queries to its oracles and runs in time $O(t)$, plus a term $\binom{q}{2} \cdot \frac{1}{2^{2n}}$ from the PRP-PRF switching lemma:

$$\Pr\left[G_1(\mathbf{A}) \Rightarrow 1\right] \leq \Pr\left[G_2(\mathbf{A}) \Rightarrow 1\right] + \binom{q}{2}\frac{1}{2^{2n}} + \mathbf{Adv}^{\mathrm{PRP}}_{\mathbb{E}}(q, O(t)).$$

Next, we define a third Game $G_3$, which is identical to $G_2$, but which replaces all calls to $\mathcal{E}$ by calls to a uniform random function $\$^{\mathcal{E}}(\cdot,\cdot) \twoheadleftarrow \mathsf{Func}(\{0,1\}^{2n} \times \{0,1\}^*, \{0,1\}^*)$, which, for a given input tuple $(C_L, M_R)$, outputs a uniformly random chosen value with the length of the second input: $C_R \twoheadleftarrow \{0,1\}^{|M_R|}$. For brevity, we omit a separate algorithmic definition for $G_3$. Since the inputs $C_L$ are chosen uniformly at random (by $\$^{\mathbb{E}}$ in Line 411 of Algorithm 3), the difference between both games can be upper bounded by the maximal IVE advantage of an adversary $\mathbf{A}_2$ on $\Pi = (\mathcal{E}, \mathcal{D})$, where $\mathbf{A}_2$ submits at most $q$ queries of at most $\ell$ blocks in total to its oracles and runs in time $O(t)$:

$$\Pr\left[G_2(\mathbf{A}) \Rightarrow 1\right] \leq \Pr\left[G_3(\mathbf{A}) \Rightarrow 1\right] + \mathbf{Adv}^{\mathrm{IVE}}_{\Pi}(q, \ell, O(t)).$$

For the remainder, we define a fourth Game $G_4$, which is identical to $G_3$, except for the fact that the encryption oracle resamples $Y$ from the set of values that did not occur in previous queries of $\mathbf{A}$ if $Y$ repeats, i.e. $Y \twoheadleftarrow \{0,1\}^{2n} \setminus \mathcal{Q}_Y$. Similarly, $G_4$ resamples $C_L$ if it repeats, i.e. $C_L \twoheadleftarrow \{0,1\}^{2n} \setminus \mathcal{Q}_{C_L}$. The encryption function of $G_4$ is shown in Algorithm 3. One can see that the outputs of both games $G_3$ and $G_4$ differ only if a bad flag gets set. For brevity, we define a compound event $\mathsf{bad} := \mathsf{bad}_1 \vee \mathsf{bad}_2$. It holds that

$$\Pr\left[G_3(\mathbf{A}) \Rightarrow 1\right] \leq \Pr\left[G_4(\mathbf{A}) \Rightarrow 1 | \neg\mathsf{bad}\right] + \Pr[\mathsf{bad}].$$

Since $\mathcal{H}_{K_1}(\cdot,\cdot)$ is an $\epsilon$-AXU hash function, it follows from Theorem 1 that the derived hash function $\mathcal{H}'_{K_1}(A, M_L, M_R) := \mathcal{H}_{K_1}(A, M_R) \oplus M_L$ is $\epsilon$-AU. Hence, the probability that $\mathsf{bad}_1$ gets set for the $i$-th query is given by $(i-1)\cdot\epsilon$. Using the union bound, the probability over all $q$ queries of the adversary is at most $\Pr[\mathsf{bad}_1] \leq \binom{q}{2}\epsilon$.
It remains to bound $\Pr[\mathsf{bad}_2]$. Since we have replaced $\mathbb{E}$ by a uniform random function $\$^{\mathbb{E}}$, its outputs may repeat and we need to bound the probability that a collision $C_L^i = C_L^j$ occurs for some $1 \leq j < i \leq q$. Since the $2n$-bit outputs of $\$^{\mathbb{E}}$ are chosen uniformly at random, the collision probability of any two values for the $i$-th query is $(i-1)\cdot 1/2^{2n}$, and – using the union bound – over $q$ queries at most $\Pr[\mathsf{bad}_2] \leq \binom{q}{2} \cdot 1/2^{2n}$. We obtain

$$\Pr[\mathsf{bad}] \leq \Pr[\mathsf{bad}_1] + \Pr[\mathsf{bad}_2] \leq \binom{q}{2} \cdot \left(\epsilon + \frac{1}{2^{2n}}\right).$$

It remains to bound $\Pr[G_4(\mathbf{A}) \Rightarrow 1 | \neg\mathsf{bad}]$. Since $\mathbf{A}$ asks no duplicate queries, all outputs from the oracle $\widetilde{\mathcal{E}}$ are drawn independently and uniformly at random.

**Algorithm 3** Games $G_1$ (Lines 301-315) and $G_4$ (Lines 401-417) for our DET-PRIV proof. The procedure INITIALIZE and function FINALIZE are identical for all games. Games $G_1$, $G_2$, and $G_3$ lack the boxed statement, which is only part of Game $G_4$.

```
101: procedure INITIALIZE
102:   Q_Y ← Q_CL ← ∅
103:   bad_1 ← bad_2 ← false
104:   b ↞ {0,1}
105:   if b = 1 then
106:     K_1 ↞ K_1; K_2 ↞ K_2; K_3 ↞ K_3
201: function FINALIZE(b')
202:   bad ← bad_1 ∨ bad_2
203:   return b = b' [∨ bad]
```

```
301: function Ẽ(A, M)
302:   if b = 0 then
303:     return $^Ẽ(A, M)
304:   (M_L, M_R) ← ENCODE_τ(M)
305:   X ← H_{K_1}(A, M_R)
306:   Y ← X ⊕ M_L
307:   if Y ∈ Q_Y then
308:     bad_1 ← true
309:   Q_Y ← Q_Y ∪ {Y}
310:   C_L ← E_{K_2}(Y)
311:   if C_L ∈ Q_{C_L} then
312:     bad_2 ← true
313:   Q_{C_L} ← Q_{C_L} ∪ {C_L}
314:   C_R ← E_{K_3}(C_L, M_R)
315:   return (C_L ‖ C_R)
```

```
401: function Ẽ(A, M)
402:   if b = 0 then
403:     return $^Ẽ(A, M)
404:   (M_L, M_R) ← ENCODE_τ(M)
405:   X ← H_{K_1}(A, M_R)
406:   Y ← X ⊕ M_L
407:   if Y ∈ Q_Y then
408:     bad_1 ← true
409:     Y ↞ {0,1}^{2n} \ Q_Y
410:   Q_Y ← Q_Y ∪ {Y}
411:   C_L ← $^E(Y)
412:   if C_L ∈ Q_{C_L} then
413:     bad_2 ← true
414:     C_L ↞ {0,1}^{2n} \ Q_{C_L}
415:   Q_{C_L} ← Q_{C_L} ∪ {C_L}
416:   C_R ← $^E(C_L, M_R)
417:   return (C_L ‖ C_R)
```

The probability that $\mathbf{A}$ wins Game $G_4$ is therefore limited by that of correctly guessing $b$. It follows

$$\Pr\left[G_4(\mathbf{A}) \Rightarrow 1 | \neg \mathsf{bad}\right] = 0.5.$$

The bound in Lemma 1 follows then from

$$2 \cdot \left| \binom{q}{2} \left( \epsilon + \frac{2}{2^{2n}} \right) + \mathbf{Adv}^{\mathrm{PRP}}_{\mathbb{E}}(q, O(t)) + \mathbf{Adv}^{\mathrm{IvE}}_{\Pi}(q, \ell, O(t)) + 0.5 - 0.5 \right| . \quad \square$$

## B Proof of Lemma 2

*Proof.* Wlog., we assume that $\mathbf{A}$ asks no queries to which it already knows the answer or that are prohibited from the DETAUTH security definition. We say that $\mathbf{A}$'s queries are maintained in a query history $\mathcal{Q}$ wherein they are stored together with their corresponding responses of $\mathbf{A}$'s available oracles as tuples $Q^i = (A^i, M^i, C^i_L, C^i_R)$. We denote by $\mathcal{Q}^i$ the state of $\mathbf{A}$'s query history *before* $\mathbf{A}$ asked its $i$-th query, i.e. after $\mathbf{A}$ has posed $i-1$ queries. We use the symbol $*$ to

represent any string (of expected length, when required from the context) at its used position. For example, given $A, C_L, C_R$, the statement $(A, *, C_L, C_R) \in \mathcal{Q}$ represents $\exists M \in \mathcal{M}$: $(A, M, C_L, C_R) \in \mathcal{Q}$ and the statement $(A, *, C_L, C_R) \notin \mathcal{Q}$ means $\forall M \in \mathcal{M}$: $(A, M, C_L, C_R) \notin \mathcal{Q}$.
We apply again a game-based proof in the following, using a sequence of Games $G_1$ through $G_4$, where the encryption and decryption oracles of the first an final game are provided in Algorithm 4. INITIALIZE and function FINALIZE are again given for all games.
We start with Game $G_1$ that provides access to an encryption oracle $\widetilde{\mathcal{E}}$ and a decryption oracle $\widetilde{\mathcal{D}}$. $G_1$ collects the values $Y$ and $C_L$ that occur in encryption queries in two sets $\mathcal{Q}_Y$ and $\mathcal{Q}_{C_L}$, respectively. Both sets are initialized empty. Moreover, $G_1$ defines a flag $\mathsf{bad}$ which is initialized to $\mathsf{false}$ and that are set to true if a value $Y$ or $C_L$ repeats over the encryption queries of $\mathbf{A}$, respectively. A second flag, $\mathsf{forge}$, is also initialized to $\mathsf{false}$ and set to $\mathsf{true}$ in decryption queries of $\mathbf{A}$ iff $\mathbf{A}$ passes a fresh valid ciphertext to the decryption oracle $\widetilde{\mathcal{D}}$. Since $G_1$ simply models the DETAUTH setting for DCT, and since the outputs of $G_1$ are influenced neither by the $\mathsf{bad}$ flags nor by the sets, it holds that

$$\Pr\left[\mathbf{A}^{\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K} \text{ forges}\right] = \Pr\left[G_1(\mathbf{A}) \Rightarrow 1\right] = \Pr\left[\mathsf{forge}\right].$$

We define a second game $G_2$, which is identical to $G_1$, except that $G_2$ replaces all calls to $\mathbb{E}$ by calls to a uniform random $2n$-bit permutation $\pi \twoheadleftarrow \mathsf{Perm}(\{0,1\}^{2n})$ and its inverse, respectively. We omit a separate definition due to the minor change. The difference between both settings can be upper bounded by the maximal advantage of an SPRP adversary on $\mathbb{E}$, $\mathbb{E}^{-1}$ that submits at most $q$ queries to its oracles and runs in time $O(t)$:

$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathbb{E},\mathbb{E}^{-1}}(q, O(t)).$$

We consider a third game $G_3$, which functions again almost identical to $G_2$, except that $G_3$ replaces the IV-based encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ by a family of permutations $\$ : \{0,1\}^{2n} \times \{0,1\}^* \to \{0,1\}^*$ in the encryption oracle and its inverse $\$^{-1}$ in the decryption oracle, respectively, with IV space $\{0,1\}^{2n}$. This means, for all $C_L \in \{0,1\}^{2n}$, $\$(C_L, \cdot)$ is a permutation. We assume that $\$$ and $\$^{-1}$ implement lazy sampling, i.e. they collect their inputs and corresponding outputs in a set $\mathcal{Q}_\$$ as tuples $(C_L^i, M_R^i, C_R^i)$. For a given tuple $(C_L, M_R)$, $\$^{\mathcal{E}}(C_L, \cdot)$ samples an output $C_R \twoheadleftarrow \{0,1\}^{|M_R|}$ uniformly at random from the set of all values that have not previously been asked for this $C_L$, and maps $C_R$ to $(C_L, M_R)$ by storing the relation into $\mathcal{Q}_\$$. Its inverse $\$^{-1}$ works similarly such that both produce consistent outputs. In the remainder, we say that $\mathbf{A}$ can choose the outputs of $\$$ and its inverse $\$^{-1}$ as long as they are consistent to the above. This is equivalent to giving $\mathbf{A}$ the key $K_3$ for $\Pi$. Clearly, this change makes $\mathbf{A}$ strictly stronger:

$$\Pr\left[G_2(\mathbf{A}) \Rightarrow 1\right] \leq \Pr\left[G_3(\mathbf{A}) \Rightarrow 1\right].$$

Next, we consider a Game $G_4$, which is provided in Algorithm 4. $G_4$ functions again almost identical to $G_3$, except that the encryption oracle $\widetilde{\mathcal{E}}$ in $G_4$ resamples

**Algorithm 4** Games $G_1$ (Lines 301-413) and $G_4$ (Lines 501-613) for our dETAUTH proof. Games $G_1$, $G_2$, and $G_3$ lack the boxed statements, which are only part of Game $G_4$.

```
101: procedure INITIALIZE
102:   Q_Y ← Q_CL ← ∅
103:   bad ← forge ← false
104:   K_1 ↞ K_1; K_2 ↞ K_2; K_3 ↞ K_3

201: function FINALIZE
202:   return forge ∨ bad

301: function Ẽ(A, M)
302:   (M_L, M_R) ← ENCODE_τ(M)
303:   X ← H_{K_1}(A, M_R)
304:   Y ← X ⊕ M_L
305:   if Y ∈ Q_Y then
306:     bad ← true
307:   Q_Y ← Q_Y ∪ {Y}
308:   C_L ← E_{K_2}(Y)
309:   Q_{C_L} ← Q_{C_L} ∪ {C_L}
310:   C_R ← E_{K_3}(C_L, M_R)
311:   return (C_L ∥ C_R)

401: function D̃(A, C)
402:   (C_L, C_R) ← C
403:   M_R ← D̃_{K_3}(C_L, C_R)
404:   X ← H_{K_1}(A, M_R)
405:   Y ← E^{-1}_{K_2}(C_L)
406:   if C_L ∉ Q_{C_L} then
407:     Q_{C_L} ← Q_{C_L} ∪ {C_L}
408:     Q_Y ← Q_Y ∪ {Y}
409:   M_L ← X ⊕ Y
410:   M ← DECODE_τ(M_L, M_R)
411:   if M ≠ ⊥ then
412:     forge ← true
413:   return M

501: function Ẽ(A, M)
502:   (M_L, M_R) ← ENCODE_τ(M)
503:   X ← H_{K_1}(A, M_R)
504:   Y ← X ⊕ M_L
505:   if Y ∈ Q_Y then
506:     bad ← true
507:     Y ↞ {0,1}^{2n} \ Q_Y
508:   Q_Y ← Q_Y ∪ {Y}
509:   C_L ← π(Y)
510:   Q_{C_L} ← Q_{C_L} ∪ {C_L}
511:   C_R ← $(C_L, M_R)
512:   return (C_L ∥ C_R)

601: function D̃(A, C)
602:   (C_L, C_R) ← C
603:   M_R ← $^{-1}(C_L, C_R)
604:   X ← H_{K_1}(A, M_R)
605:   Y ← π^{-1}(C_L)
606:   if C_L ∉ Q_{C_L} then
607:     Q_{C_L} ← Q_{C_L} ∪ {C_L}
608:     Q_Y ← Q_Y ∪ {Y}
609:   M_L ← X ⊕ Y
610:   M ← DECODE_τ(M_L, M_R)
611:   if M ≠ ⊥ then
612:     forge ← true
613:   return M
```

$Y$ from the domain of all $2n$-bit values except those that previously occurred at the place of $Y$, i.e. $Y \twoheadleftarrow \{0,1\}^{2n} \setminus \mathcal{Q}_Y$ as in our dETPRIV proof. Since $\pi$ is a permutation, having always distinct values $Y$ and $Y'$ implies that the values $C_L \leftarrow \pi(Y)$ are also always distinct over all encryption queries of $\mathbf{A}$. So, we do not need to consider the $\mathsf{bad}_2$ event as in our dETPRIV proof. Moreover, it holds:

- If $\mathbf{A}$ chooses a value $C_L$ in a decryption query that collides with $C'_L$ from a previous query, then trivially, their corresponding values $Y \leftarrow \pi^{-1}(C_L)$ and $Y' \leftarrow \pi^{-1}(C'_L)$ also collide.
- If $\mathbf{A}$ chooses a value $C_L$ in a decryption query that has not occurred before, it is mapped to a fresh value $Y \leftarrow \pi^{-1}(C_L)$ that also has not occurred before.

Therefore, we do not have to consider *bad* events in the decryption oracle, but only have to store values $C_L$ and $Y$ in the sets $\mathcal{Q}_{C_L}$ and $\mathcal{Q}_Y$ if they have not occurred previously.
One can see from Algorithm 4 that the outputs of $G_3$ and $G_4$ differ only if a $\mathsf{bad}$ flag is set. So, we can upper bound

$$\Pr\left[G_3(\mathbf{A}) \Rightarrow 1\right] \leq \Pr\left[G_4(\mathbf{A}) \Rightarrow 1 | \neg\mathsf{bad}\right] + \Pr[\mathsf{bad}].$$

We can apply a similar argument as in our detPriv proof to upper bound

$$\Pr[\mathsf{bad}] \leq \binom{q}{2} \cdot \epsilon.$$

It remains to upper bound $\Pr[\mathsf{forge}]$. In the following, we consider a query $Q^i = (A^i, C_L^i, C_R^i)$ which we denote as winning query. This means, we assume $Q^i$ is a valid query by $\mathbf{A}$ to the decryption oracle that has not been asked by $\mathbf{A}$ before and that makes the game set the flag $\mathsf{forge}$ to $\mathsf{true}$. We conduct a case analysis in the following, where the individual cases differ in the fact whether or not associated data $A^i$ and ciphertext components $C_L^i$ and $C_R^i$ (or subsets thereof) of $\mathbf{A}$'s winning query occurred *together* in any previous query of $\mathbf{A}$ or not. We identify three disjoint cases C1, C2, and C3, which cover all possibilities. Analogous to the cases, we define three events $E_1$, $E_2$, and $E_3$, where $E_i = \mathsf{true}$ iff $\mathbf{A}$ wins in the correspondingly indexed case C$i$ and $\mathsf{false}$ otherwise, for $1 \leq i \leq 3$.

***Case C1:*** $(A^i, *, *, *) \notin \mathcal{Q}^i$**.** In this case, we assume that the associated data of $\mathbf{A}$'s winning query is fresh, i.e. has not occured in any previous query of $\mathbf{A}$. For $\mathbf{A}$ to win, it must hold that $\textsc{Decode}_\tau(M_L^i, M_R^i) \neq \perp$. This holds for a fraction of $1/2^\tau$ of the $2^{2n}$ values $M_L^i$, which are computed by

$$M_L^i = \mathcal{H}'_{K_1}(A^i, M_R^i, Y^i) := \mathcal{H}_{K_1}(A^i, M_R^i) \oplus Y^i.$$

$\mathbf{A}$ can compute the values $M_R^i$, and choose $A^i$; though, it does not see the values $X^i$ nor $Y^i$. Since the values $A^i$ are always fresh and $\mathcal{H}_{K_1}(\cdot,\cdot)$ is an $\epsilon$-AXU hash function, it follows from Theorem 1 that $\mathcal{H}'_{K_1}(\cdot,\cdot,\cdot)$ is an $\epsilon$-AU hash function. Thus, it holds that the probability for each value $M_L^i$ to occur is at most $\epsilon$. So, the probability that $\mathbf{A}$ wins with the $i$-th query is upper bounded by $\epsilon \cdot 2^{2n}/2^\tau$. Over $q$ queries, the success probability for $\mathbf{A}$ in this case is at most $\Pr[E_1] \leq q \cdot \epsilon \cdot 2^{2n}/2^\tau$. In the remainder, we can safely assume that the associated data $A^i$ of $\mathbf{A}$'s winning query is old, i.e. has already occurred before in $\mathbf{A}$'s queries.

***Case C2:*** $(A^i, *, C_L^i, *) \notin \mathcal{Q}^i$**.** In this case, the tuple $(A^i, C_L^i)$ is fresh, i.e. did not occur before together in any previous query of $\mathbf{A}$. Since $\pi^{-1}$ is a uniform random permutation, it maps $C_L^i$ to $Y^i \leftarrow \pi^{-1}(C_L^i)$ such that the tuple $(A^i, Y^i)$ has also not occurred before. For a successful forgery, it must hold that $M_L^i$ is valid, which holds for a fraction of $1/2^\tau$ of the $2^{2n}$ values $M_L^i = \mathcal{H}'_{K_1}(A^i, M_R^i, Y^i)$. Since $\mathcal{H}'_{K_1}(\cdot,\cdot,\cdot)$ is $\epsilon$-AU, the probability that $\mathbf{A}$ wins with the $i$-th query is again upper bounded by $\epsilon \cdot 2^{2n}/2^\tau$. Over $q$ queries, the success probability for $\mathbf{A}$ in this case is at most $\Pr[E_2] \leq q \cdot \epsilon \cdot 2^{2n}/2^\tau$.

***Case C3:*** $(A^i, *, C_L^i, *) \in \mathcal{Q}^i$**.** In this case, $A^i$ and $C_L^i$ already occurred together in some previous query of $\mathbf{A}$. It is easy to see that $(A^i, *, C_L^i, C_R^i) \notin \mathcal{Q}^i$ must hold; otherwise, the winning query of $\mathbf{A}$ would have occurred before. Since $\$^{-1}(C_L^i, \cdot)$ is a permutation, it follows that the tuple $(A^i, C_L^i, M_R^i)$ has not occurred before in any of $\mathbf{A}$'s queries. This implies itself that the tuple $(A^i, Y^i, M_R^i)$ has not occurred before since $\pi^{-1}$ is a permutation. Again, the fact that $M_L^i$ is computed using the $\epsilon$-AU hash function $\mathcal{H}'_{K_1}(\cdot, \cdot, \cdot)$, the probability that $\mathbf{A}$ wins with the $i$-th query can be upper bounded by $\epsilon \cdot 2^{2n}/2^\tau$. Over $q$ queries, the success probability for $\mathbf{A}$ in this case is at most $\Pr[E_3] \leq q \cdot \epsilon \cdot 2^{2n}/2^\tau$. It follows that

$$\Pr[E_1 \vee E_2 \vee E_3] \leq \sum_{i=1}^{3} \Pr[E_i] \leq 3q \cdot \epsilon \cdot \frac{2^{2n}}{2^\tau}.$$

Summing up, our claim in Lemma 2 follows from

$$\binom{q}{2} \cdot \epsilon + 3q \cdot \epsilon \cdot \frac{2^{2n}}{2^\tau} + \mathbf{Adv}^{\mathrm{SPRP}}_{\mathbb{E},\mathbb{E}^{-1}}(q, O(t)). \quad \square$$

## C Security Bound from the Universal Hash Function

Depending on the choice of $\tau$, the term $q^2(4m'^2)/2^{2n}$ may be limited by the birthday bound. This stems from the fact that BRW hashing is an $\epsilon_{\mathrm{BRW}}$-AXU hash function with $\epsilon_{\mathrm{BRW}} \leq 2m'/2^n$, which depends linearly on the number of blocks of a query. So, the concatenation of two independent $n$-bit hashes depends quadratically on the maximal number of blocks of a query: $\epsilon^2_{\mathrm{BRW}} \leq 4m'^2/2^{2n}$. For stretch lengths $\tau$ of $\tau \leq n$ bits, we can reduce the quadratic dependency on $m'$ to a linear one. Recall from Algorithm 2, that we computed $M_L$ by $M_L \leftarrow X \oplus Y$, where $X \leftarrow \mathcal{H}_{K_1 \,\|\, K_2}(A, M_R)$ and $Y \leftarrow \mathbb{E}^{-1}_{K_3}(C_L)$. Let us write $M_L = (M_L^1 \,\|\, M_L^2)$ such that $|M_L^1| = |M_L^2| = n$, and analogously, let us denote the halves of $X$ and $Y$ by $X = (X^1 \,\|\, X^2)$ and $Y = (Y^1 \,\|\, Y^2)$. As given in Algorithm 2, for $\tau \leq n$, the redundancy is included completely in one part of $M_L$. So, the probability that $M_L^1$ is valid can be computed as follows. Since $\mathcal{H}_{K_1}(\cdot, \cdot)$ is $\epsilon_{\mathrm{BRW}}$-AXU, the family of hash functions $\mathcal{H}_{K_1}(\cdot, \cdot) \oplus Y^1$ is $\epsilon_{\mathrm{BRW}}$-AU. There are $2^n$ possible values $X^1$ each of which is valid with probability $1/2^\tau$. The probability for each to occur is at most $\epsilon_{\mathrm{BRW}}$. So, the probability that $M_L^1$ contains the expected $\tau$ bits of redundancy can be upper bounded by

$$\frac{2^n}{2^\tau} \cdot \epsilon_{\mathrm{BRW}} \leq \frac{2m'}{2^\tau}.$$