XHX – A Framework for Optimally Secure Tweakable Block Ciphers from Classical Block Ciphers and Universal Hashing

Ashwin Jha[1], Eik List[2], Kazuhiko Minematsu[3],
Sweta Mishra[4], and Mridul Nandi[1]

[1] Indian Statistical Institute, Kolkata, India. {ashwin_r, mridul}@isical.ac.in
[2] Bauhaus-Universität Weimar, Weimar, Germany. eik.list@uni-weimar.de
[3] NEC Corporation, Tokyo, Japan. k-minematsu@ah.jp.nec.com
[4] IIIT, Delhi, India. swetam@iiitd.ac.in

Abstract. Tweakable block ciphers are important primitives for designing cryptographic schemes with high security. In the absence of a standardized tweakable block cipher, constructions built from classical block ciphers remain an interesting research topic in both theory and practice. Motivated by Mennink's $\widetilde{F}[2]$ publication from 2015, Wang et al. proposed 32 optimally secure constructions at ASIACRYPT'16, all of which employ two calls to a classical block cipher each. Yet, those constructions were still limited to n-bit keys and n-bit tweaks. Thus, applications with more general key or tweak lengths still lack support. This work proposes the XHX family of tweakable block ciphers from a classical block cipher and a family of universal hash functions, which generalizes the constructions by Wang et al. First, we detail the generic XHX construction with three independently keyed calls to the hash function. Second, we show that we can derive the hash keys in efficient manner from the block cipher, where we generalize the constructions by Wang et al.; finally, we propose efficient instantiations for the used hash functions.

Keywords: Provable security · ideal-cipher model · tweakable block cipher

1 Introduction

Tweakable Block Ciphers. A tweakable block cipher (TBC for short), is a cryptographic transform that adds an additional public parameter called *tweak* to the usual inputs key and plaintext of a classical block cipher. This means that a tweakable block cipher $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is a permutation on the plaintext/ciphertext space $\mathcal{M}$ for every combination of some key $K \in \mathcal{K}$ and tweak $T \in \mathcal{T}$, where $\mathcal{K}$, $\mathcal{T}$, and $\mathcal{M}$ are assumed to be non-empty sets. Tweakable block ciphers have been used first by Schroeppel and Orman in the Hasty Pudding Cipher, where the tweak still was called *Spice* [18]. Liskov, Rivest, and Wagner [11] have formalized the concept then in 2002.
In the past, one can observe a trend that the role of tweakable block ciphers has become much more prominent, last but not least due to the advent of recent

dedicated constructions, such as those proposed alongside the TWEAKEY framework [7], or e.g., SKINNY [1]. However, in the absence of a standard, tweakable block ciphers based on classical ones remain a highly interesting topic.

Blockcipher-based Constructions. Liskov et al. [11] had described two constructions, known as LRW1 and LRW2. Rogaway [17] proposed XE and XEX as refinements of LRW2 for updating tweaks efficiently and reducing the number of keys. These schemes are efficient in that they need one block cipher call plus one (computational) universal hash function. Both XE and XEX are provably secure under standard model, i.e., assuming the block cipher is a (strong) pseudorandom permutation, they are secure up to $O(2^{n/2})$ queries, when using an n-bit block cipher. Since this security bound results from the birthday paradox on collisions of inputs to the block cipher, their security is inherently limited by the birthday bound (BB-secure).

Constructions with Stronger Security. Beyond-birthday-bound (BBB) secure schemes that overcome this barrier have been an interesting research topic from both theory and practice. Minematsu [13] introduced a rekeying-based construction. Landecker, Shrimpton and Terashima [9] proposed a cascade of two independent LRW2 instances, called CLRW2. Both constructions are secure up to $O(2^{2n/3})$ queries, however, at the price of decreased efficiency of using two block-cipher calls per block plus per-tweak rekeying or plus two calls to a universal hash function, respectively.
For settings that require stronger security, Lampe and Seurin [8] proved that the chained cascade of more instances of LRW2 could asymptotically approach a security of up to $O(2^n)$ queries, i.e. full n-bit security. However, the downside is drastically decreased performance. An alternative direction has been initiated by Mennink [12], who also proposed TBC constructions from classical block ciphers, but proved the security in the ideal-cipher model. Mennink's constructions could achieve full n-bit security quite efficiently when both input and key are n bits. In particular, his second construction $\widetilde{F}[2]$ required only two block-cipher calls. Following Mennink's work, Wang et al. [20] proposed 32 constructions of optimally secure tweakable block ciphers from classical block ciphers. Their designs share an n-bit key, n-bit tweak and n-bit plaintext, and linearly mix tweak, key, and the result of a second offline call to the block cipher. Their constructions have the desirable property of allowing to cache the result of the first block-cipher call; moreover, given a-priori known tweaks, some of their constructions allow further to precompute the result of the key schedule.
All constructions by Wang et al. were restricted to n-bit keys and tweaks. While this limit was reasonable, it did not address tweakable block ciphers with tweaks longer than n bit. Such constructions are interesting since they allow significantly increased security than those with n bit, which is useful for e.g., authenticated encryption or variable-input-length ciphers, such as [19]. In general, extending the key length in the ideal-cipher-model is far from trivial (see, e.g., [2,6,10]), and the key size in this model does not necessarily match the required tweak length. Moreover, many ciphers, like the AES-192 or AES-256, possess key and block

lengths for which the constructions in [12,20] are inapplicable. In general, the tweak represents additional data accompanying the plaintext/ciphertext block, and no general reason exists why tweaks must be limited to the block length. Applications include but are not limited to, e.g., schemes for high-security AE or full-disk encryption. For example, Shrimpton and Terashima [19] proposed the AE scheme Protected IV, which required a TBC with variable-length tweak and BBB security. Moreover, disk-encryption schemes are typically based on TBCs, where the physical location on disk (e.g., the sector ID) is used as tweak, which can be arbitrarily long.

Before proving the security of a construction, we have to specify the employed model. The standard model is well-established in our community despite the fact that proofs base on few unproven assumptions, such as that a block cipher is a PRP, or ignore practical side-channel attacks. In the standard model, the adversary is given access only to either the *real construction* $\widetilde{E}$ or an *ideal construction* $\widetilde{\pi}$. In contrast, the ideal-cipher model differs in the sense that it assumes an ideal primitive—in our case the classical ideal block cipher E which is used in $\widetilde{E}$—which the adversary has also access to in both worlds. Although a proof in the ideal-cipher model is not an unexceptional guarantee that no attacks may exist when instantiated in practice [3], for us, it allows to capture away the details of the primitive for the sake of focus on the security of the construction, and is employed by a wide range of applications [4].

For schemes proven in the standard model, one can look at the XTX construction [14], which was proposed earlier by Minematsu and Iwata at IMACC'15. XTX extended the tweak domain of a given tweakable block cipher $\widetilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ by hashing the arbitrary-length tweak to an $(n+t)$-bit value. The first t bits serve as tweak and the latter n bits are XORed to both input and output of $\widetilde{E}$. Given an ϵ-AXU family of hash functions and an ideal tweakable cipher, XTX is secure for up to $O(2^{(n+t)/2})$ queries in the standard model. However, no alternative to XTX exists in the ideal-cipher model yet.

Recently, Naito [15] proposed the XKX framework of beyond-birthday-secure tweakable block ciphers, which shares similarities to ours. He proposed two instances, the birthday-secure $\text{XKX}^{(1)}$ and the beyond-birthday-secure $\text{XKX}^{(2)}$. More detailed, the nonce is processed by a block-cipher-based PRF which yields the block-cipher key for the current message; the counter is hashed with a universal hash function under a second, independent key to mask the input. As a contrast to other and to our proposal, Naito's construction demands both a counter plus a nonce as parameters to overcome the birthday bound; as a standalone construction, its security reduces to $n/2$ bits if an adversary could use the same "nonce" value for all queries. Hence, $\text{XKX}^{(2)}$ is tailored only to certain domains, e.g., modes of operation in nonce-based authenticated encryption schemes. Our proposal differs from XKX in four aspects: (1) we do not pose limitations on the reuse of input parameters; moreover, (2) we do not require a minimum key length of $n+k$ bits; (3) we do not use several independent keys, but employ the block cipher to derive hashing keys; (4) finally, Naito's construction is proved in the standard model, whereas we consider the ideal-cipher model.

Table 1: Summary of Our Results. ICM(n,k) denotes the ideal-cipher model for a block cipher with n-bit block and k-bit key; BC(n,k) and TBC(n,t,k) denote the standard-model (tweakable) block cipher of n-bit block, t-bit tweak, and k-bit key. #Enc. = #calls to the (tweakable) block cipher, and #Mult. = #multiplications over GF(2^n). $a(b)$ = b out of a calls can be precomputed with the secret key; we define $s = \lceil k/n \rceil$.

Scheme	Model	Tweak	Key	Security	Efficiency		Reference
		length in bit		in bit	#Enc.	#Mult.	
$\widetilde{F}[2]$	ICM(n,n)	n	n	n	2		[12]
$\widetilde{\text{E1}}, \ldots, \widetilde{\text{E32}}$	ICM(n,n)	n	n	n	2 (1)		[20]
XTX	TBC(n,t,k)	any ℓ	$k+2n$	$(n+t)/2$	1	$2\lceil \ell/n \rceil$	[14]
XKX$^{(2)}$	BC(n,k)	–*	$k+n$	$\min\{n, k/2\}$	1	1	[15]
XHX	ICM(n,k)	any ℓ	k	$(n+k)/2$	$s+1$ (s)	$s\lceil \ell/n \rceil$	This work
XHX	ICM(n,k)	$2n$	k	n	$s+1$ (s)	s	This work

* XKX$^{(2)}$ employs a counter as tweak.

Contribution. This work proposes the XHX family of tweakable block ciphers from a classical block cipher and a family of universal hash functions, which generalizes the constructions by Wang et al. [20]. Like them, the present work also uses the ideal-cipher model for our security analysis of XHX. As the major difference to their work, our proposal allows arbitrary tweak lengths, and works for any block cipher of n-bit block and k-bit key. The security is guaranteed up to $O(2^{(n+k)/2})$ queries, hence is full n-bit secure when $k \geq n$. Our contributions, are threefold: First, we detail the generic XHX construction with three independently keyed calls to the hash function. Second, we show that we can derive the hash keys in an efficient manner from the block cipher, generalizing the constructions by Wang et al.; finally, we propose efficient instantiations for the employed hash functions for concreteness.

The remainder is structured as follows: Section 2 briefly gives the preliminaries necessary for the rest of this work. Section 3 then defines the general construction, that we call GXHX for simplicity, which hashes the tweak to three outputs. Section 4 continues with the definition and analysis of XHX, which derives the hashing keys from the block cipher. Section 5 describes efficient instantiations for our hash functions depending on the tweak length. In particular, we propose instantiations for $2n$-bit and arbitrary-length tweaks.

2 Preliminaries

General Notation. We use lowercase letters x for indices and integers, uppercase letters X, Y for binary strings and functions, and calligraphic uppercase letters $\mathcal{X}, \mathcal{Y}$ for sets. We denote the concatenation of binary strings X and Y by $X \parallel Y$ and the result of their bitwise XOR by $X \oplus Y$. For tuples of bit strings $(X_1, \ldots, X_x)$, $(Y_1, \ldots, Y_x)$ of equal domain, we denote by $(X_1, \ldots, X_x) \oplus (Y_1, \ldots, Y_x)$ the element-wise XOR, i.e., $(X_1 \oplus Y_1, \ldots, X_x \oplus Y_x)$. We indicate the length of X in bits by $|X|$, and write X_i for the i-th block. Furthermore, we denote by $X \twoheadleftarrow \mathcal{X}$ that X is chosen uniformly at random from the set $\mathcal{X}$. We define three sets of particular interest: $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ be the set of all functions $F : \mathcal{X} \to \mathcal{Y}$, $\mathsf{Perm}(\mathcal{X})$ the set of all permutations $\pi : \mathcal{X} \to \mathcal{X}$, and $\mathsf{TPerm}(\mathcal{T}, \mathcal{X})$ for the set of tweaked permutations over $\mathcal{X}$ with associated tweak space $\mathcal{T}$. $(X_1, \ldots, X_x) \xleftarrow{n} X$ denotes that X is split into n-bit blocks i.e., $X_1 \parallel \ldots \parallel X_x = X$, and $|X_i| = n$ for $1 \leq i \leq x - 1$, and $|X_x| \leq n$. Moreover, we define $\langle X \rangle_n$ to denote the encoding of a non-negative integer X into its n-bit representation. Given a integer $x \in \mathbb{N}$, we define the function $\text{TRUNC}_x : \{0, 1\}^* \to \{0, 1\}^x$ to return the leftmost x bits of the input if its length is $\geq x$, and returns the input otherwise. For two sets $\mathcal{X}$ and $\mathcal{Y}$, a uniform random function $\rho : \mathcal{X} \to \mathcal{Y}$ which maps inputs $X \in \mathcal{X}$ independently from other inputs and uniformly at random to outputs $Y \in \mathcal{Y}$. For an event E, we denote by $\Pr[E]$ the probability of E. For positive integers n and k, we denote the falling factorial as $(n)_k := \frac{n!}{k!}$.

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 $\Delta_{\mathbf{A}}\left(\mathcal{O}^1; \mathcal{O}^2\right) := |\Pr[\mathbf{A}^{\mathcal{O}^1} \Rightarrow 1] - \Pr[\mathbf{A}^{\mathcal{O}^2} \Rightarrow 1]|$ 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 and those of the adversary, if any. W.l.o.g., we assume that $\mathbf{A}$ never asks queries to which it already knows the answer.

A block cipher E with associated key space $\mathcal{K}$ and message space $\mathcal{M}$ is a mapping $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ such that for every key $K \in \mathcal{K}$, it holds that $E(K, \cdot)$ is a permutation over $\mathcal{M}$. We define $\mathsf{Block}(\mathcal{K}, \mathcal{M})$ as the set of all block ciphers with key space $\mathcal{K}$ and message space $\mathcal{M}$. A tweakable block cipher $\widetilde{E}$ with associated key space $\mathcal{K}$, tweak space $\mathcal{T}$, and message space $\mathcal{M}$ is a mapping $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ such that 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 $\mathcal{M}$. We also write $\widetilde{E}_K^T(\cdot)$ as short form in the remainder.

The STPRP security of $\widetilde{E}$ is defined via upper bounding the advantage of a distinguishing adversary $\mathbf{A}$ in a game, where we consider the ideal-cipher model throughout this work. There, $\mathbf{A}$ has access to oracles $(\mathcal{O}, E^{\pm})$, where $E^{\pm}$ is the usual notation for access to the encryption oracle E and to the decryption oracle E^{-1}. $\mathcal{O}$ is called construction oracle, and is either the real construction $\widetilde{E}_K^{\pm}(\cdot, \cdot)$, or $\widetilde{\pi} \twoheadleftarrow \mathsf{TPerm}(\mathcal{T}, \mathcal{M})$. $E^{\pm} \twoheadleftarrow \mathsf{Perm}(\mathcal{M})$ is an ideal block cipher underneath $\widetilde{E}$. The STPRP advantage of $\mathbf{A}$ is defined as $\Delta_{\mathbf{A}}(\widetilde{E}_K^{\pm}(\cdot, \cdot), E^{\pm}(\cdot, \cdot); \widetilde{\pi}^{\pm}(\cdot, \cdot), E^{\pm}(\cdot, \cdot))$, where the probabilities are taken over random and independent choice of K, E, $\widetilde{\pi}$, and the coins of $\mathbf{A}$ if any. For the remainder, we say that $\mathbf{A}$ is a (q_C, q_P)-distinguisher if it asks at most q_C queries to its construction oracle and at most q_P queries to its primitive oracle.

Definition 1 (Almost-Uniform Hash Function). Let $\mathcal{H} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of keyed hash functions. We call $\mathcal{H}$ ϵ-almost-uniform (ϵ-$\mathsf{AUniform}$) if, for $K \twoheadleftarrow \mathcal{K}$ and all $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, it holds that $\Pr_{K \twoheadleftarrow \mathcal{K}}[\mathcal{H}(K, X) = Y] \leq \epsilon$.

Definition 2 (Almost-XOR-Universal Hash Function). Let $\mathcal{H} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of keyed hash functions with $\mathcal{Y} \subseteq \{0,1\}^*$. We say that $\mathcal{H}$ is ϵ-almost-XOR-universal (ϵ-AXU) if, for $K \twoheadleftarrow \mathcal{K}$, and for all distinct $X, X' \in \mathcal{X}$ and any $\Delta \in \mathcal{Y}$, it holds that $\Pr_{K \twoheadleftarrow \mathcal{K}} [\mathcal{H}(K, X) \oplus \mathcal{H}(K, X') = \Delta] \leq \epsilon$.

Minematsu and Iwata [14] defined partial-almost-XOR-universality to capture the probability of partial output collisions.

Definition 3 (Partial-AXU Hash Function). Let $\mathcal{H} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^n \times \{0,1\}^k$ be a family of hash functions. We say that $\mathcal{H}$ is (n, k, ϵ)-partial-AXU ((n, k, ϵ)-pAXU) if, for $K \twoheadleftarrow \mathcal{K}$, and for all distinct $X, X' \in \mathcal{X}$ and all $\Delta \in \{0,1\}^n$, it holds that $\Pr_{K \twoheadleftarrow \mathcal{K}} \left[\mathcal{H}(K, X) \oplus \mathcal{H}(K, X') = (\Delta, 0^k)\right] \leq \epsilon$.

The H-Coefficient Technique. The H-coefficients technique is a method due to Patarin [5,16]. It assumes 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 $\mathcal{O}_{\text{ideal}}$. A transcript τ is called *attainable* if the probability to obtain τ in the ideal world is non-zero. One assumes that $\mathbf{A}$ does not ask duplicate queries or queries prohibited by the game or to which it already knows the answer. Denote by Θ_{real} and Θ_{ideal} the distribution of transcripts in the real and the ideal world, respectively. Then, the fundamental Lemma of the H-coefficients technique states:

Lemma 1 (Fundamental Lemma of the H-coefficient Technique [16]). Assume, the set of attainable transcripts is partitioned into two disjoint sets $\textsc{GoodT}$ and $\textsc{BadT}$. Further assume, there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in \textsc{GoodT}$, it holds that

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

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

The proof is given in [5,16].

3 The Generic GXHX Construction

Let $n, k, \ell \geq 1$ be integers and $\mathcal{K} = \{0,1\}^k$, $\mathcal{L} = \{0,1\}^\ell$, and $\mathcal{T} \subseteq \{0,1\}^*$. Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and $\mathcal{H} : \mathcal{L} \times \mathcal{T} \to \{0,1\}^n \times \mathcal{K} \times \{0,1\}^n$ be a family of hash functions. Then, we define by $\text{GXHX}[E, \mathcal{H}] : \mathcal{L} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ the tweakable block cipher instantiated with E and $\mathcal{H}$ that, for given key $L \in \mathcal{L}$, tweak $T \in \mathcal{T}$, and message $M \in \{0,1\}^n$, computes the ciphertext C, as shown on the left side of Algorithm 1. Likewise, given key $L \in \mathcal{L}$, tweak $T \in \mathcal{T}$, and ciphertext $C \in \{0,1\}^n$, the plaintext M is computed by $M \leftarrow \text{GXHX}[E, \mathcal{H}]_L^{-1}(T, C)$, as shown on the right side of Algorithm 1. Clearly, $\text{GXHX}[E, \mathcal{H}]$ is a correct and tidy tweakable permutation, i.e., for all

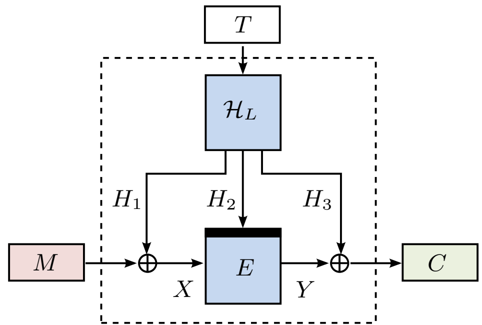

Fig. 1: Schematic illustration of the encryption process of a message M and a tweak T with the general GXHX$[E, \mathcal{H}]$ tweakable block cipher. $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a keyed permutation and $\mathcal{H} : \mathcal{L} \times \mathcal{T} \to \{0,1\}^n \times \mathcal{K} \times \{0,1\}^n$ a keyed universal hash function.

Algorithm 1 Encryption and decryption algorithms of the general GXHX$[E, \mathcal{H}]$ construction.

function $\mathrm{GXHX}[E,\mathcal{H}]_L(T, M)$	**function** $\mathrm{GXHX}[E,\mathcal{H}]_L^{-1}(T, C)$
$\quad (H_1, H_2, H_3) \leftarrow \mathcal{H}(L, T)$	$\quad (H_1, H_2, H_3) \leftarrow \mathcal{H}(L, T)$
$\quad C \leftarrow E_{H_2}(M \oplus H_1) \oplus H_3$	$\quad M \leftarrow E^{-1}_{H_2}(C \oplus H_3) \oplus H_1$
$\quad$ **return** C	$\quad$ **return** M

keys $L \in \mathcal{L}$, all tweak-plaintext inputs $(T, M) \in \mathcal{T} \times \{0,1\}^n$, and all tweak-ciphertext inputs $(T, C) \in \mathcal{T} \times \{0,1\}^n$, it holds that

$$\mathrm{GXHX}[E,\mathcal{H}]_L^{-1}(T, \mathrm{GXHX}[E,\mathcal{H}]_L(T, M)) = M \text{ and}$$
$$\mathrm{GXHX}[E,\mathcal{H}]_L(T, \mathrm{GXHX}[E,\mathcal{H}]_L^{-1}(T, C)) = C.$$

Figure 1 illustrates the encryption process schematically.

4 XHX: Deriving the Hash Keys from the Block Cipher

In the following, we adapt the general GXHX construction to XHX. which differs from the former in two aspects: first, XHX splits the hash function into three functions $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$; second, since we need at least $n + k$ bit of key material for the hash functions, it derives the hash-function key from a key K using the block cipher E. We denote by $s \geq 0$ the number of derived hash-function keys L_i and collect them together with the user-given key $K \in \{0,1\}^k$ into a vector $L := (K, L_1, \ldots, L_s)$. Moreover, we define a set of variables I_i and K_i, for $1 \leq i \leq s$, which denote input and key to the block cipher E for computing: $L_i := E_{K_i}(I_i)$. We allow flexible, usecase-specific definitions for the values I_i and K_i as long as they fulfill certain properties that will be listed in Section 4.1. We redefine the key space of the hash functions to $\mathcal{L} \subseteq \{0,1\}^k \times$

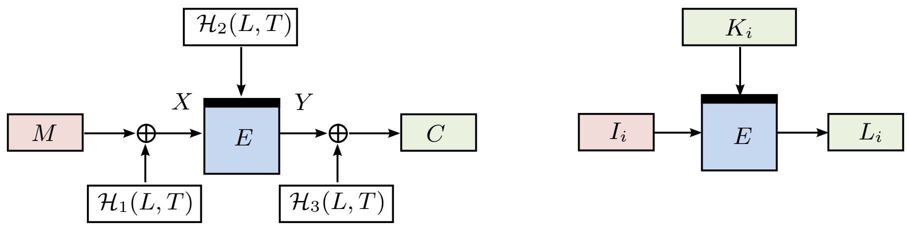

Fig. 2: Schematic illustration of the XHX$[E,\mathcal{H}]$ construction where we derive the hash-function keys L_i from the block cipher E.

Algorithm 2 Encryption and decryption algorithms of XHX where the keys are derived from the block cipher. We define $\mathcal{H} := (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$. Note that the exact definitions of I_i and K_i are usecase-specific.

function XHX$[E,\mathcal{H}]$.KeySetup(K)
  for $i \leftarrow 1$ to s **do**
    $L_i \leftarrow E_{K_i}(I_i)$
  $L \leftarrow (K, L_1, \ldots, L_s)$
  return L

function $\mathcal{H}(L,T)$
  $H_1 \leftarrow \mathcal{H}_1(L,T)$
  $H_2 \leftarrow \mathcal{H}_2(L,T)$
  $H_3 \leftarrow \mathcal{H}_3(L,T)$
  return (H_1, H_2, H_3)

function XHX$[E,\mathcal{H}]_K(T,M)$
  $L \leftarrow$ XHX$[E,\mathcal{H}]$.KeySetup(K)
  $(H_1, H_2, H_3) \leftarrow \mathcal{H}(L,T)$
  $C \leftarrow E_{H_2}(M \oplus H_1) \oplus H_3$
  return C

function XHX$[E,\mathcal{H}]_K^{-1}(T,C)$
  $L \leftarrow$ XHX$[E,\mathcal{H}]$.KeySetup(K)
  $(H_1, H_2, H_3) \leftarrow \mathcal{H}(L,T)$
  $M \leftarrow E_{H_2}^{-1}(C \oplus H_3) \oplus H_1$
  return M

$(\{0,1\}^n)^s$. Note, the values L_i are equal for all encryptions and decryptions and hence, can be precomputed and stored for all encryptions under the same key.

The Constructions by Wang et al. The 32 constructions $\widetilde{\mathbb{E}}$[2] by Wang et al. are a special case of our construction with the parameters $s = 1$, key length $k = n$, with the inputs $I_i, K_i \in \{0^n, K\}$, and the option $(I_i, K_i) = (0^n, 0^n)$ excluded. Their constructions compute exactly one value L_1 by $L_1 := E_{K_1}(I_1)$. One can easily describe their constructions in the terms of the XHX framework, with three variables $X_1, X_2, X_3 \in \{K, L_1, K \oplus L_1\}$ for which holds that $X_1 \neq X_2$ and $X_3 \neq X_2$, and which are used in XHX as follows:

$$\mathcal{H}_1(L,T) := X_1,$$
$$\mathcal{H}_2(L,T) := X_2 \oplus T,$$
$$\mathcal{H}_3(L,T) := X_3.$$

4.1 Security Proof of XHX

This section concerns the security of the XHX construction in the ideal-cipher model where the hash-function keys are derived by the (ideal) block cipher E.

Properties of $\mathcal{H}$. For our security analysis, we list a set of properties that we require for $\mathcal{H}$. We assume that L is sampled uniformly at random from $\mathcal{L}$. To address parts of the output of $\mathcal{H}$, we also use the notion $\mathcal{H}_i : \mathcal{L} \times \mathcal{T} \to \{0,1\}^{o_i}$ to refer to the function that computes the i-th output of $\mathcal{H}(L,T)$, for $1 \leq i \leq 3$, with $o_1 := n$, $o_2 := k$, and $o_3 := n$. Moreover, we define $\mathcal{H}_{1,2}(T) := (\mathcal{H}_1(L,T), \mathcal{H}_2(L,T))$, and $\mathcal{H}_{3,2}(T) := (\mathcal{H}_3(L,T), \mathcal{H}_2(L,T))$.

Property P1. For all distinct $T, T' \in \mathcal{T}$ and all $\Delta \in \{0,1\}^n$, it holds that

$$\max_{i \in \{1,3\}} \Pr_{L \twoheadleftarrow \mathcal{L}} \left[\mathcal{H}_{i,2}(T) \oplus \mathcal{H}_{i,2}(T') = \left(\Delta, 0^k\right) \right] \leq \epsilon_1.$$

Property P2. For all $T \in \mathcal{T}$ and all $(c_1, c_2) \in \{0,1\}^n \times \{0,1\}^k$, it holds that

$$\max_{i \in \{1,3\}} \Pr_{L \twoheadleftarrow \mathcal{L}} \left[\mathcal{H}_{i,2}(T) = (c_1, c_2) \right] \leq \epsilon_2.$$

Note that Property P1 is equivalent to saying $\mathcal{H}_{1,2}$ and $\mathcal{H}_{3,2}$ are (n, k, ϵ_1)-pAXU; Property P2 is equivalent to the statement that $\mathcal{H}_{1,2}$ and $\mathcal{H}_{3,2}$ are ϵ_2-$\mathsf{AUniform}$. Clearly, it must hold that $\epsilon_1, \epsilon_2 \geq 2^{-(n+k)}$.

Property P3. For all $T \in \mathcal{T}$, all chosen I_i, K_i, for $1 \leq i \leq s$, and all $\Delta \in \{0,1\}^n$, it holds that

$$\Pr_{L \twoheadleftarrow \mathcal{L}} \left[\mathcal{H}_{1,2}(T) \oplus (I_i, K_i) = \left(\Delta, 0^k\right) \right] \leq \epsilon_3.$$

Property P4. For all $T \in \mathcal{T}$, all chosen K_i, L_i, for $1 \leq i \leq s$, and all $\Delta \in \{0,1\}^n$, it holds that

$$\Pr_{L \twoheadleftarrow \mathcal{L}} \left[\mathcal{H}_{3,2}(T) \oplus (L_i, K_i) = \left(\Delta, 0^k\right) \right] \leq \epsilon_4.$$

Properties P3 and P4 represent the probabilities that an adversary's query hits the inputs that have been chosen for computing a hash-function key. We list a further property which gives the probability that a set of constants chosen by the adversary can hit the values I_i and K_i from generating the keys L_i:

Property P5. For $1 \leq i \leq s$, and all $(c_1, c_2) \in \{0,1\}^n \times \{0,1\}^k$, it holds that

$$\Pr_{K \twoheadleftarrow \mathcal{K}} \left[(I_i, K_i) = (c_1, c_2) \right] \leq \epsilon_5.$$

In other words, the tuples (I_i, K_i) contain a sufficient amount of close to n bit entropy, and cannot be predicted by an adversary with greater probability, i.e., ϵ_5 should not be larger than a small multiple of $1/2^n$. From Property 5 and the fact that the values L_i are computed from $E_{K_i}(I_i)$ with an ideal permutation E, it follows that for $1 \leq i \leq s$ and all $(c_1, c_2) \in \{0,1\}^n \times \{0,1\}^k$

$$\Pr_{K \twoheadleftarrow \mathcal{K}} \left[(L_i, K_i) = (c_1, c_2) \right] \leq \epsilon_5.$$

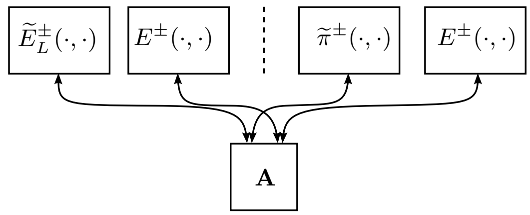

Fig. 3: Schematic illustration of the oracles available to **A**.

Theorem 1. Let $E \twoheadleftarrow \mathsf{Block}(\mathcal{K}, \{0,1\}^n)$ be an ideal cipher. Further, let $\mathcal{H}_i : \mathcal{L} \times \mathcal{T} \to \{0,1\}^{o_i}$, for $1 \leq i \leq 3$ be families of hash functions for which Properties P1 through P4 hold, and let $K \twoheadleftarrow \mathcal{K}$. Moreover, let Property P5 hold for the choice of all I_i and K_i. Let s denote the number of keys L_i, $1 \leq i \leq s$. Let **A** be a (q_C, q_P)-distinguisher on $\mathrm{XHX}[E, \mathcal{H}]_K$. Then

$$\underset{\mathbf{A}}{\Delta}\left(\mathrm{XHX}[E, \mathcal{H}], E^{\pm}; \widetilde{\pi}^{\pm}, E^{\pm}\right) \leq q_C^2 \epsilon_1 + 2 q_P q_C \epsilon_2 + q_C s(\epsilon_3 + \epsilon_4) + 2 q_P s \epsilon_5 + \frac{s^2}{2^{n+1}}.$$

Proof Idea. The proof of Theorem 1 follows from Lemmas 1, 2, and 3. Those can be found in Appendix A. Let $\widetilde{E}$ denote the $\mathrm{XHX}[E, \mathcal{H}]$ construction in the remainder. Figure 3 illustrates the oracles available to **A**. The queries by **A** are collected in a transcript τ. We will define a series of bad events that can happen during the interaction of **A** with its oracles:

- Collisions between two construction queries,
- Collisions between a construction and a primitive query,
- Collisions between two primitive queries,
- The case that the adversary finds an input-key tuple in either a primitive or construction query that was used to derive a key L_i.

The proof will bound the probability of these events to occur in the transcript in Lemma 2. We define a transcript as bad if it satisfies at least one such bad event, and define BADT as the set of all attainable bad transcripts.

Lemma 2. It holds that

$$\Pr\left[\Theta_{\mathrm{ideal}} \in \mathrm{BADT}\right] \leq q_C^2 \epsilon_1 + 2 q_P q_C \epsilon_2 + q_C s(\epsilon_3 + \epsilon_4) + 2 q_P s \epsilon_5 + \frac{s^2}{2^{n+1}}.$$

The proof is given in Appendix A.1.

Good Transcripts. Above, we have considered bad events. In contrast, we define GOODT as the set of all good transcripts, i.e., all attainable transcripts that are *not* bad.

Lemma 3. Let $\tau \in \mathrm{GOODT}$ be a good transcript. Then

$$\frac{\Pr\left[\Theta_{\mathrm{real}} = \tau\right]}{\Pr\left[\Theta_{\mathrm{ideal}} = \tau\right]} \geq 1.$$

The full proof can be found in Appendix A.2.

Algorithm 3 The universal hash function $\mathcal{H}^*$.

function $\mathcal{H}^*_L(T)$
  $(K, L_1, \ldots, L_s) \leftarrow L$
  $K' \leftarrow \textsc{trunc}_n(K)$
  $H_1 \leftarrow \mathcal{F}_{K'}(T)$
  $H_2 \leftarrow \textsc{trunc}_k\left(\mathcal{F}_{L_1}(T) \parallel \cdots \parallel \mathcal{F}_{L_s}(T)\right)$
  $H_3 \leftarrow \mathcal{F}_{K'}(T)$
  return (H_1, H_2, H_3)

function $\mathcal{F}_K(T)$
  $p \leftarrow |T| \bmod n$
  if $p \neq 0$ **then**
    $T \leftarrow T \parallel 0^{n-p}$
  Parse $T_1, \ldots, T_m \xleftarrow{n} T$
  $T_{m+1} \leftarrow \langle |T| \rangle_n$
  $Y \leftarrow 0$
  for $i \leftarrow 1$ **to** $m+1$ **do**
    $Y \leftarrow (Y \oplus T_i) \cdot K$
  return $(Y \cdot K) \oplus K$

5 Efficient Instantiations

The hash function for XHX needs to satisfy multiple conditions for the construction to be secure. This section provides concrete instantiations of hash functions which satisfy those conditions. While it is rather straight-forward to design hash functions in the case of independent keys, by using two independent n-bit AXU and AUniform hash functions, the additional conditions for XHX require more analysis. We present two instantiations depending on the maximum tweak length. While the case of n-bit tweaks has already been covered by Wang et al., there is the general important case of having a variable-length tweak still open, that we address with an instantiation. Additionally, we also present a second hash function that is more efficient for the special case of having a $2n$-bit tweak.
Our proposals use field multiplications over $\mathbb{GF}(2^n)$ and need $(k+n)$ bits of key material, where the ideal cipher is used for key derivation. We define $K_i := K$ and $I_i := \langle i \rangle$, for $1 \leq i \leq s$, i.e., we compute the subkeys L_i as $L_i \leftarrow E_K(\langle i \rangle)$.

$\mathcal{H}^*$ – ***A Hash Function for Variable-Length Tweaks.*** We propose a first instantiation $\mathcal{H}^*$ for variable-length tweaks. $\mathcal{H}^*$ uses two universal hash functions keyed by K and L_1, and takes T as input. Assume $k \geq n$ be positive integers and $s \leq 2^{k-1}$. More specifically, let $\mathcal{F} := \{F \mid F : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n\}$ denote an $\epsilon(m)$-AXU and $\rho(m)$-AUniform family of hash functions. Here, $\epsilon(m)$ and $\rho(m)$ denote the maximum AXU and AUniform biases for any input pair whose length is at most m in n-bit blocks. Then, $\mathcal{H}^* : \mathcal{L} \times \{0,1\}^* \to \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^n$ is defined in Algorithm 3. We suggest a polynomial hash for $\mathcal{F}_K(\cdot)$ with a minimum degree of one; this means, it holds that $\mathcal{F}_K(\varepsilon) = K$ for the empty string ε to avoid fixed points. For simplicity, $\mathcal{H}^*$ conducts all computations in the same field $\mathbb{GF}(2^n)$ in all calls to $\mathcal{F}$. In general, we have to consider three potential cases for the relation of state size and key lengths:

- **Case $\mathbf{k = n}$**. In this case, the hash values H_1, H_2, and H_3 are the results of polynomial hash functions $\mathcal{F}$. In this case, $\mathcal{H}^*$ employs K directly as hashing key to generate H_1 and H_3, and a derived key L_1 to compute H_2. Hence, it holds that $s = 1$ in this case.

- **Case $\mathbf{k} < \mathbf{n}$**. In this case, we could simply truncate H_2 from n to k bits. Theoretically, we could derive a longer key from K for the computation of H_1 and H_3; however, we disregard this case since ciphers with smaller key than state length are very uncommon.
- **Case $\mathbf{k} > \mathbf{n}$**. In the third case, we truncate the hash key K for the computation of H_1 and H_3 to n bits. Moreover, we derive s hashing keys $L_1, \ldots, L_s$ from the block cipher E. For H_2, we concatenate the output of s instances of $\mathcal{F}$, and truncate the result to k bits if necessary. This construction is well-known to be $\epsilon^s(m)$-pAXU if $\mathcal{F}$ is $\epsilon(m)$-pAXU.

Lemma 4. $\mathcal{H}^*$ is $2^{sn-k}\epsilon^{s+1}(m)$-$\mathsf{pAXU}$ and $2^{sn-k}\rho^{s+1}(m)$-Uniform. Moreover, it satisfies Properties P3 and P4 with probability $2^{sn-k}\rho^{s+1}(m)$ each, and Property P5 with $\epsilon_5 \leq 2/2^k$ for our choice of the values I_i and K_i.

Remark 1. The term 2^{sn-k} results from the potential truncation of H_2 if the key length k of the block cipher is no multiple of the state size n. H_2 is computed by concatenating the results of multiple independent invocations of a polynomial hashing function $\mathcal{F}$ in $\mathbb{GF}(2^n)$ under assumed independent keys. Clearly, if $\mathcal{F}$ is ϵ-AXU, then their sn-bit concatenation is ϵ^s-AXU. However, after truncating sn to k bits, we may lose information, which results in the factor of 2^{sn-k}. For the case $k = n$, it follows that $s = 1$, and the terms $2^{sn-k}\epsilon^{s+1}(m)$ and $2^{sn-k}\rho^{s+1}(m)$ simplify to $\epsilon^2(m)$ and $\rho^2(m)$, respectively.

Our instantiation of $\mathcal{F}$ has $\epsilon(m) = \rho(m) = (m+2)/2^n$. Before we prove Lemma 4, we derive from it the following corollary for XHX when instantiated with $\mathcal{H}^*$.

Corollary 1. Let E and $\mathrm{XHX}[E, \mathcal{H}^*]$ be defined as in Theorem 1, where the maximum length of any tweak is limited by at most m n-bit blocks. Moreover, let $K \twoheadleftarrow \mathcal{K}$. Let $\mathbf{A}$ be a (q_C, q_P)-distinguisher on $\mathrm{XHX}[E, \mathcal{H}^*]$. Then

$$\underset{\mathbf{A}}{\Delta}\left(\mathrm{XHX}[E, \mathcal{H}^*], E^{\pm}; \widetilde{\pi}^{\pm}, E^{\pm}\right) \leq \frac{(q_C^2 + 2q_C q_P + 2q_C s)(m+2)^{s+1}}{2^{n+k}} + \frac{4q_P s}{2^k} + \frac{s^2}{2^{n+1}}.$$

The proof of the corollary stems from the combination of Lemma 4 with Theorem 1 and can be omitted.

Proof of Lemma 4. In the following, we assume that $T, T' \in \{0,1\}^*$ are distinct tweaks of at most m blocks each. Again, we consider the pAXU property first.

Partial Almost-XOR-Universality. This is the probability that for any $\Delta \in \{0,1\}^n$:

$$\begin{aligned}
&\Pr_{L \twoheadleftarrow \mathcal{L}}\left[(\mathcal{F}_{K'}(T), \mathcal{F}_{L_1,\ldots,L_s}(T)) \oplus (\mathcal{F}_{K'}(T'), \mathcal{F}_{L_1,\ldots,L_s}(T')) = (\Delta, 0^n)\right] \\
&= \Pr_{L \twoheadleftarrow \mathcal{L}}\left[\mathcal{F}_{K'}(T) \oplus \mathcal{F}_{K'}(T') = \Delta, \mathcal{F}_{L_1,\ldots,L_s}(T) \oplus \mathcal{F}_{L_1,\ldots,L_s}(T') = 0^n\right] \\
&\leq 2^{sn-k} \cdot \epsilon^{s+1}(m).
\end{aligned}$$

We assume independent hashing keys K', $L_1, \ldots, L_s$ here. When $k = n$, this probability is upper bounded by $\epsilon^2(m)$ since $\mathcal{F}$ is $\epsilon(m)$-AXU. Note that $s = 1$ in this case. In the case $k > n$, we compute s words of H_2 that are concatenated and truncated to k bits. Hence, $\mathcal{F}_{L_1,\ldots,L_s}$ is $2^{sn-k} \cdot \epsilon^s(m)$-$\mathsf{AXU}$. In combination with the AXU bound for $\mathcal{F}_{K'}$, we obtain the pAXU bound for $\mathcal{H}^*$ above.

Almost-Uniformity. This is the probability that for any $\Delta_1, \Delta_2 \in \{0,1\}^n$

$$\Pr_{L \twoheadleftarrow \mathcal{L}}[(\mathcal{F}_{K'}(T), \mathcal{F}_{L_1,\ldots,L_s}(T)) = (\Delta_1, \Delta_2)] = \Pr_{L \twoheadleftarrow \mathcal{L}}[\mathcal{F}_{K'}(T) = \Delta_1, \mathcal{F}_{L_1,\ldots,L_s}(T) = \Delta_2] \leq 2^{sn-k} \cdot \rho^{s+1}(m)$$

since $\mathcal{F}$ is $\rho(m)$-$\mathsf{AUniform}$, and using a similar argumentation for the cases $k = n$ and $k > n$ as for partial-almost-XOR universality.

Property P3. For all $T \in \mathcal{T}$ and $\Delta \in \{0,1\}^n$, Property P3 is equivalent to

$$\Pr_{L \twoheadleftarrow \mathcal{L}}[\mathcal{F}_{K'}(T) = (\Delta \oplus I_i), \mathcal{F}_{L_1,\ldots,L_s}(T) = K]$$

for a fixed $1 \leq i \leq s$. Here, this property is equivalent to almost uniformity; hence, the probability for the latter equality is at most $2^{sn-k} \cdot \rho^s(m)$. The probability for the former equality is at most $\rho(m)$ since the property considers a fixed i. Since we assume independence of K and $L_1, \ldots, L_s$, it holds that $\epsilon_3 \leq 2^{sn-k} \cdot \rho^{s+1}(m)$.

Property P4. For all $T \in \mathcal{T}$ and $\Delta \in \{0,1\}^n$, Property P4 is equivalent to

$$\Pr_{L \twoheadleftarrow \mathcal{L}}[\mathcal{F}_{K'}(T) = (\Delta \oplus L_i), \mathcal{F}_{L_1,\ldots,L_s}(T) = K]$$

for a fixed $1 \leq i \leq s$. Using a similar argumentation as for Property P3, the probability is upper bounded by $\epsilon_4 \leq 2^{sn-k} \cdot \rho^{s+1}(m)$.

Property P5. We derive the hashing keys L_i with the help of E and the secret key K. So, in the simple case that $s = 1$, the probability that the adversary can guess any tuple (I_i, K_i), for $1 \leq i \leq s$, that is used to derive the hashing keys L_i, or guess any tuple (L_i, K_i) is at most $1/2^k$. Under the reasonable assumption $s < 2^{k-1}$, the probability becomes for fixed i in the general case:

$$\Pr_{K \twoheadleftarrow \mathcal{K}}[(I_i, K_i) = (c_1, c_2)] \leq \frac{1}{2^k - s} \leq \frac{2}{2^k}.$$

A similar argument holds that the adversary can guess any tuple (L_i, K_i), for $1 \leq i \leq s$. Hence, it holds for $\mathcal{H}^*$ that $\epsilon_5 \leq 2/2^k$.

$\epsilon(m)$ ***and*** $\rho(m)$***.*** It remains to determine $\epsilon(m)$ and $\rho(m)$ for our instantiation of $\mathcal{F}_K(\cdot)$. It maps tweaks $T = T_1, \ldots, T_m$ to the result of

$$\left(\bigoplus_{i=1}^{m} T_i \cdot K^{m+3-i}\right) \oplus \langle |T| \rangle_n \cdot K \oplus K.$$

This is a polynomial of degree at most $m+2$, which is $(m+2)/2^n$-AXU. Moreover, over $L \subset \mathcal{L}$, it lacks fixed points but for every $\Delta \in \{0,1\}^n$, and any fixed subset of m blocks of $T_1, \ldots, T_m$, there are at most $m+2$ out of 2^n values for the block T_{m+1} that fulfill $\mathcal{F}_K(T) = \Delta$. Hence, $\mathcal{F}$ is also $(m+2)/2^n$-$\mathsf{AUniform}$. $\square$

Algorithm 4 The universal hash function $\mathcal{H}^2$.

function $\mathcal{H}^2_L(T)$
  $(K, L_1, \ldots, L_s) \leftarrow L$
  $(T_1, T_2) \xleftarrow{n} T$
  $K' \leftarrow \textsc{trunc}_n(K)$
  $H_1 \leftarrow T_1 \boxdot K'$
  $H_2 \leftarrow \textsc{trunc}_k\left(\mathcal{F}_{L_1}(T) \parallel \cdots \parallel \mathcal{F}_{L_s}(T)\right)$
  $H_3 \leftarrow T_1 \boxdot K'$
  return (H_1, H_2, H_3)

function $\mathcal{F}_{L_i}(T_1 \parallel T_2)$
  return $(T_1 \boxdot L_i) \oplus T_2$

$\mathcal{H}^*$ is a general construction which supports arbitrary tweak lengths. Though, if we used $\mathcal{H}^*$ for $2n$-bit tweaks, we would need four Galois-Field multiplications. However, we can hash more efficiently, even optimal in terms of the number of multiplications in this case. For this purpose, we define $\mathcal{H}^2$.

$\mathcal{H}^2$ – *A Hash Function for $2n$-bit Tweaks.* Naively, for two-block tweaks $|T| = 2n$, an ϵ-pAXU construction with $\epsilon \approx 1/2^{2n}$ could be achieved by simply multiplying the tweak with some key $L \in \mathbb{GF}(2^{2n})$ sampled uniformly over $\mathbb{GF}(2^{2n})$. We can perform this even more efficiently by using two multiplications over the smaller field $\mathbb{GF}(2^n)$. Additional conditions, such as uniformity, are satisfied by introducing squaring in the field to avoid fixed points in multiplication-based universal hash function. Following the notations from the previous sections, let $L = (K, L_1)$ be the $2n$-bit key of our hash function. For $X, Y \in \mathbb{GF}(2^n)$, we define the operation $\boxdot : \mathbb{GF}(2^n) \times \mathbb{GF}(2^n) \to \mathbb{GF}(2^n)$ as

$$X \boxdot Y := \begin{cases} X \cdot Y & \text{if } X \neq 0 \\ Y^2 & \text{otherwise.} \end{cases}$$

We assume the standard encoding between the bit space and $\mathbb{GF}(2^n)$, i.e. a polynomial in the field is represented as its coefficient vector, e. g., the all-zero vector denotes the zero element 0, and the bit string $(0 \ldots 01)$ denotes the identity element. Hereafter, we write X interchangeably as an element of $\mathbb{GF}(2^n)$ or of $\{0,1\}^n$. For $\mathcal{L} = (\{0,1\}^n)^2$, $\mathcal{X} = (\{0,1\}^n)^2$ and $\mathcal{Y} = \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^n$, the construction $\mathcal{H}^2 : \mathcal{L} \times \mathcal{X} \to \mathcal{Y}$ is defined in Algorithm 4. We stress that the usage of keys has been chosen carefully, e.g., a swap of K and L_1 in $\mathcal{H}^2$ would invalidate Property P4.

Lemma 5. $\mathcal{H}^2$ is $2^{s+1}/2^{n+k}$-pAXU, $2^s/2^{n+k}$-AUniform, satisfies Properties P3 and P4 with probability $2/2^{n+k}$ each, and Property P5 with $\epsilon_5 = s/2^n$ for our choices of I_i and K_i, for $1 \leq i \leq s$.

Before proving Lemma 5, we derive from it the following corollary for XHX when instantiated with $\mathcal{H}^2$.

Corollary 2. Let E and $\mathrm{XHX}[E, \mathcal{H}^2]$ be defined as in Theorem 1. Moreover, let $K \twoheadleftarrow \mathcal{K}$. Let $\mathbf{A}$ be a (q_C, q_P)-distinguisher on $\mathrm{XHX}[E, \mathcal{H}^2]_K$. Then

$$\underset{\mathbf{A}}{\Delta}\left(\mathrm{XHX}[E, \mathcal{H}^2], E^{\pm}; \widetilde{\pi}^{\pm}, E^{\pm}\right) \leq \frac{2^{s+2} q_C^2 + 2^{s+1} q_C q_P + 4 q_C s}{2^{n+k}} + \frac{2 q_P s^2}{2^n} + \frac{s^2}{2^{n+1}}.$$

Again, the proof of the corollary stems from the combination of Lemma 5 with Theorem 1 and can be omitted.

Proof of Lemma 5. Since H_1 and H_3 are equal, we can restrict the analysis of the properties of Lemma 5 to only the outputs (H_1, H_2). Note that K and L_1 are independent. In the following, we denote the hash-function results for some tweak T as H_1, H_2, H_3, and those for some tweak $T' \neq T$ as H_1', H_2', H_3'. Moreover, we denote the n-bit words of H_2 as $(H_2^1, \ldots, H_2^s)$, and those of H_2' as $({H_2'}^1, \ldots, {H_2'}^s)$.

Partial Almost-XOR-Universality. First, let us consider the pAXU property. It holds that $H_1 := T_1 \boxdot K$ and $H_2 := \textsc{trunc}_k(\mathcal{F}_{L_1}(T), \ldots, \mathcal{F}_{L_s}(T))$. Considering H_1, it must hold that $H_1' = H_1 \oplus \Delta$, with

$$\Delta = (T_1' \boxdot K) \oplus (T_1 \boxdot K).$$

For any $X \neq 0^n$, it is well-known that $X \boxdot Y$ is $1/2^n$-AXU. So, for any fixed T_1 and fixed $\Delta \in \{0,1\}^n$, there is exactly one value T_1' that fulfills the equation if $H_1' \neq K \boxdot K$, and exactly two values if $H_1' = K \boxdot K$, namely $T_1' \in \{0^n, K\}$. So

$$\Pr_{K \twoheadleftarrow \mathcal{L}}\left[(T_1 \boxdot K) \oplus (T_1' \boxdot K) = \Delta\right] \leq 2/2^n.$$

The argumentation for H_2 is similar. The probability that any $L_i = 0^n$, for $1 \leq i \leq s$, is at most $s/2^n$. In the remainder, we can then assume that all $L_i \neq 0^n$. W.l.o.g., we focus for now on the first word of H_2, H_2^1, in the following. For fixed (T_1, T_2), H_2^1, and T_2', there is exactly one value T_1' s.t. ${H_2'}^1 = H_2^1$ if ${H_2'}^1 \neq L_1 \boxdot (L_1 \oplus T_2')$, namely $T_1' := T_1 \oplus (T_2 \oplus T_2') \boxdot L_1^{-1}$. There exist exactly two values T_1' if ${H_2'}^1 = L_1 \boxdot L_1 \oplus T_2'$, namely $T_1' \in \{0^n, L_1\}$. Hence, it holds that

$$\Pr_{L_1 \twoheadleftarrow \mathcal{L}}\left[H_2^1 = {H_2'}^1\right] \leq 2/2^n.$$

The same argumentation follows for $2 \leq i \leq s$ since the keys L_i are pairwise independent. Since $ns - k$ bits of H_2^s and ${H_2'}^s$ are truncated if k is not a multiple of n, the bound has to be multiplied with 2^{sn-k}. With the factor of $2/2^n$ for H_1, it follows for fixed $\Delta \in \{0,1\}^n$ that $\mathcal{H}^2$ is ϵ-pAXU for ϵ upper bounded by

$$\frac{2}{2^n} \cdot 2^{sn-k} \cdot \frac{2^s}{2^{sn}} = \frac{2^{s+1}}{2^{n+k}}.$$

Almost-Uniformity. Here, we concern the probability for any H_1 and H_2:

$$\Pr_{L \twoheadleftarrow \mathcal{L}}\left[T_1 \boxdot K = H_1, \textsc{trunc}_k(\mathcal{F}_{L_1}(T), \ldots, \mathcal{F}_{L_s}(T)) = H_2\right].$$

If $K = 0^n$ and $H_1 = 0^n$, then the first equation may be fulfilled for any T_1. Though, the probability for $K = 0^n$ is $1/2^n$. So, we can assume $K \neq 0^n$ in the remainder. Again, we focus on the first word of H_2 next. For fixed L_1 and H_2^1, there exist at most two values (T_1, T_2) to fulfill $(T_1 \boxdot L_1) \oplus T_2 = H_2^1$. In the case $H_1 \neq K \boxdot K$, there is exactly one value $T_1 := H_1 \boxdot K^{-1}$ that yields H_1. Then, T_1, L_1, and H_2^1 determine $T_2 := H_2^1 \oplus (T_1 \boxdot L_1)$ uniquely. In the opposite case that $H_1 = K \boxdot K$, there exist exactly two values (T_1, T_1') that yield H_1, namely 0^n and K. Each of those determines T_2 uniquely. The probability that the so-fixed values T_1, T_2 yield also $H_2^2, \ldots, H_2^s$ is at most $(2/2^n)^{s-1}$ if k is a multiple of n since the keys L_i are pairwise independent; if k is not a multiple of n, we have again an additional factor of 2^{sn-k} from the truncation. Hence, $\mathcal{H}^2$ is $2^{sn-k} \cdot 2^s / 2^{n+sn} = 2^s/2^{n+k}$-AUniform.

Property P3. Given $I_i = \langle i-1 \rangle$ and $K_i = K$, for $1 \leq i \leq s$, ϵ_3 is equivalent to the probability that a chosen (T_1, T_2) yields $\Pr[T_1 \boxdot K = \Delta \oplus \langle i-1 \rangle$, $\textsc{trunc}_k(\mathcal{F}_{L_1}(T), \ldots, \mathcal{F}_{L_s}(T)) = K]$, for some i. This can be rewritten to

$$\begin{aligned} &\Pr\left[T_1 \boxdot K = \Delta \oplus \langle i-1 \rangle\right] \\ &\cdot \Pr\left[\textsc{trunc}_k(\mathcal{F}_{L_1}(T), \ldots, \mathcal{F}_{L_s}(T)) = K \,|\, T_1 \boxdot K = \Delta \oplus \langle i-1 \rangle\right]. \end{aligned}$$

For fixed $\Delta \neq K \boxdot K$, there is exactly one value T_1 that satisfies the first part of the equation; otherwise, there are exactly two values T_1 if $\Delta = K \boxdot K$. Moreover, K is secret; so, the values T_1 require that the adversary guesses K correctly. Given fixed T_1, Δ, and K, there is exactly one value T_2 that matches the first n bits of K; $T_2 := (T_1 \boxdot L_1) \oplus K[k-1..k-n]$. The remaining bits of K are matched with probability $2^{sn-k}/2^{(s-1)n}$, assuming that the keys L_i are independent. Hence, it holds that $\epsilon_3 \leq 2/2^n \cdot 2^{sn-k}/2^{sn} = 2/2^{n+k}$.

Property P4. This argument follows from a similar argumentation as Property P3. Hence, it holds that $\epsilon_4 \leq 2/2^{n+k}$. $\square$

Acknowledgments. This work was initiated during the group sessions of the 6th Asian Workshop on Symmetric Cryptography (ASK 2016) held in Nagoya. We thank the anonymous reviewers of the ToSC 2017 and Latincrypt 2017 for their fruitful comments.

References

1. 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 *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

2. Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
3. John Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2006.
4. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.
5. 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 *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.
6. Peter Gazi and Ueli M. Maurer. Cascade Encryption Revisited. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2009.
7. 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.
8. Rodolphe Lampe and Yannick Seurin. Tweakable Blockciphers with Asymptotically Optimal Security. In Shiho Moriai, editor, *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2013.
9. Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable blockciphers with beyond birthday-bound security. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2012.
10. Jooyoung Lee. Towards Key-Length Extension with Optimal Security: Cascade Encryption and Xor-cascade Encryption. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 405–425. Springer, 2013.
11. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
12. Bart Mennink. Optimally Secure Tweakable Blockciphers. In Gregor Leander, editor, *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 428–448. Springer, 2015.
13. 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.
14. Kazuhiko Minematsu and Tetsu Iwata. Tweak-Length Extension for Tweakable Blockciphers. In Jens Groth, editor, *IMA Int. Conf.*, volume 9496 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 2015.
15. Yusuke Naito. Tweakable Blockciphers for Efficient Authenticated Encryptions with Beyond the Birthday-Bound Security. *IACR Transactions on Symmetric Cryptology*, 2017(2):1–26, 2017.
16. Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008.

17. Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
18. Richard Schroeppel and Hilarie Orman. The Hasty Pudding Cipher. *AES candidate submitted to NIST*, 1998.
19. 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.
20. Lei Wang, Jian Guo, Guoyan Zhang, Jingyuan Zhao, and Dawu Gu. How to Build Fully Secure Tweakable Blockciphers from Classical Blockciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 455–483, 2016.

A Proof Details

The proof of Theorem 1 follows from Lemmas 1, 2, and 3. Let $\widetilde{E}$ denote the $\mathrm{XHX}[E, \mathcal{H}]$ construction in the remainder. W.l.o.g., we assume, $\mathbf{A}$ does not ask duplicated queries nor trivial queries to which it already knows the answer, e.g., feeds the result of an encryption query to the corresponding decryption oracle or vice versa. The queries by $\mathbf{A}$ are collected in a transcript τ. We define that τ is composed of two disjoint sets of queries τ_C and τ_P and L, $\tau = \tau_C \cup \tau_P \cup \{L\}$, where $\tau_C := \{(M^i, C^i, T^i, H_1^i, H_2^i, H_3^i, X^i, Y^i, d^i)\}_{1 \leq i \leq q_C}$ denotes the queries by $\mathbf{A}$ to the construction oracle plus internal variables H_1^i, H_2^i, H_3^i (i.e., the outputs of $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, respectively), X^i and Y^i (where $X^i \leftarrow H_1^i \oplus M^i$ and $Y^i \leftarrow H_3^i \oplus C^i$, respectively); and $\tau_P := \{(\widehat{K}^i, \widehat{X}^i, \widehat{Y}^i, d^i)\}_{1 \leq i \leq q_P}$ the queries to the primitive oracle; both sets store also binary variables d^i that indicate the direction of the i-th query, where $d^i = 1$ represents the fact that the i-th query is an encryption query, and $d^i = 0$ that it is a decryption query. The internal variables for one call to XHX are as given in Algorithm 2 and Figure 2.
We apply a common strategy for handling bad events from both worlds: in the real world, all secrets (i.e., the hash-function key L) are revealed to the $\mathbf{A}$ *after* it finished its interaction with the available oracles, but before it has output its decision bit regarding which world it interacted with. Similarly, in the ideal world, the oracle samples the hash-function key independently from the choice of E and $\widetilde{\pi}$ uniformly at random, $L \twoheadleftarrow \mathcal{L}$, and also reveals L to $\mathbf{A}$ *after* the adversary finished its interaction and before has output its decision bit. The internal variables in construction queries – H_1^i, H_2^i, H_3^i, X^i, Y^i – can then be computed and added to the transcript also in the ideal world using the oracle inputs and outputs T^i, M^i, C^i, H_1^i, H_2^i, and H_3^i.
Let $1 \leq i \neq j \leq q$. We define that an attainable transcript τ is bad, i.e., $\tau \in$ BADT, if one of the following conditions is met:

- bad_1: There exist $i \neq j$ s.t. $(H_2^i, X^i) = (H_2^j, X^j)$.
- bad_2: There exist $i \neq j$ s.t. $(H_2^i, Y^i) = (H_2^j, Y^j)$.
- bad_3: There exist $i \neq j$ s.t. $(H_2^i, X^i) = (\widehat{K}^j, \widehat{X}^j)$.

– $\mathsf{bad}_{\mathbf{4}}$: There exist $i \neq j$ s.t. $(H_2^i, Y^i) = (\widehat{K}^j, \widehat{Y}^j)$.
– $\mathsf{bad}_{\mathbf{5}}$: There exist $i \neq j$ s.t. $(\widehat{K}^i, \widehat{X}^i) = (\widehat{K}^j, \widehat{X}^j)$.
– $\mathsf{bad}_{\mathbf{6}}$: There exist $i \neq j$ s.t. $(\widehat{K}^i, \widehat{Y}^i) = (\widehat{K}^j, \widehat{Y}^j)$.
– $\mathsf{bad}_{\mathbf{7}}$: There exist $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, q_C\}$ s.t. $(X^j, H_2^j) = (I_i, K_i)$ and $d^j = 1$.
– $\mathsf{bad}_{\mathbf{8}}$: There exist $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, q_C\}$ s.t. $(Y^j, H_2^j) = (L_i, K_i)$ and $d^j = 0$.
– $\mathsf{bad}_{\mathbf{9}}$: There exist $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, q_P\}$ s.t. $(\widehat{X}^j, \widehat{K}^j) = (I_i, K_i)$.
– $\mathsf{bad}_{\mathbf{10}}$: There exist $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, q_P\}$ s.t. $(\widehat{Y}^j, \widehat{K}^j) = (L_i, K_i)$.
– $\mathsf{bad}_{\mathbf{11}}$: There exist $i, j \in \{1, \ldots, s\}$ and $i \neq j$ s.t. $(K_i, L_i) = (K_j, L_j)$ but $I_i \neq I_j$.

The events

– bad_1 and bad_2 consider collisions between two construction queries,
– bad_3 and bad_4 consider collisions between primitive and construction queries,
– bad_5 and bad_6 consider collisions between two primitive queries, and
– bad_7 through bad_{10} address the case that the adversary may could find an input-key tuple in either a primitive or construction query that has been used to derive some of the subkeys L_i.
– bad_{11} addresses the event that the ideal oracle produces a collision while sampling the hash-function keys independently uniformly at random.

Note that the events bad_5 and bad_6 are listed here only for the sake of completeness. We will show briefly that these events can never occur.

A.1 Proof of Lemma 2

Proof. In the following, we upper bound the probabilities of each bad event.

***bad*$_1$ *and* *bad*$_2$.** Events bad_1 and bad_2 represent the cases that two distinct construction queries would feed the same tuple of key and input to the underlying primitive E if the construction would be the real $\widetilde{E}$; bad_1 considers the case when the values $H_2^i = H_2^j$ and $X^i = X^j$ collide. In the real world, it follows that $Y^i = Y^j$, while this holds only with small probability in the ideal world. The event bad_2 concerns the case when the values $H_2^i = H_2^j$ and $Y^i = Y^j$ collide. Again, in the real world, it follows then that $X^i = X^j$, whereas this holds only with small probability in the ideal world. So, both events would allow $\mathbf{A}$ to distinguish both worlds. Let us consider bad_1 first, and let us start in the real world. Since $\mathbf{A}$ asks no duplicate queries, it must hold that two distinct queries (M^i, T^i) and (M^j, T^j) yielded

$$X^i = \left(M^i \oplus H_1^i\right) = \left(M^j \oplus H_1^j\right) = X^j \quad \text{and} \quad H_2^i = H_2^j.$$

We define $\Delta := M^i \oplus M^j$ and consider two subcases: in the subcase that $T^i = T^j$, it automatically holds that $H_2^i = H_2^j$ and $H_1^i = H_1^j$. However, this also implies

that $M^i = M^j$, i.e., **A** would have asked a duplicate query, which is prohibited. So, it must hold that $T^i \neq T^j$ in the real world.
If $T^i = T^j$ in the ideal world, it must hold that the plaintexts are disjoint, $M^i \neq M^j$, since we assumed that **A** does not make duplicate queries. Since $\widetilde{\pi}(T^i, \cdot)$ is a permutation, the resulting plaintexts are also disjoint: $M^i \neq M^j$. From $T^i = T^j$ follows that $H_1^i = H_1^j$ and thus, X^i and X^j cannot be equal:

$$X^i = M^i \oplus H_1^i \neq M^j \oplus H_1^j = X^j,$$

which contradicts with our definition of bad_1. So, it must hold that $T^i \neq T^j$ also in the ideal world. From Property P1 and over $L \twoheadleftarrow \mathcal{L}$, it holds then

$$\begin{aligned}\Pr[\mathsf{bad}_1] &= \Pr\left[\exists i \neq j; 1 \leq i, j \leq q_C : \left(X^i, H_2^i\right) = \left(X^j, H_2^j\right)\right] \\ &= \Pr\left[\exists i \neq j; 1 \leq i, j \leq q_C : \mathcal{H}_{1,2}\left(T^i\right) \oplus \mathcal{H}_{1,2}\left(T^j\right) = \left(\Delta, 0^k\right)\right] \leq \binom{q_C}{2}\epsilon_1.\end{aligned}$$

Using a similar argumentation, it follows also from Property P1 that for $T^i \neq T^j$

$$\begin{aligned}\Pr[\mathsf{bad}_2] &= \Pr\left[\exists i \neq j; 1 \leq i, j \leq q_C : \left(Y^i, H_2^i\right) = \left(Y^j, H_2^j\right)\right] \\ &= \Pr\left[\exists i \neq j; 1 \leq i, j \leq q_C : \mathcal{H}_{3,2}\left(T^i\right) \oplus \mathcal{H}_{3,2}\left(T^j\right) = \left(\Delta, 0^k\right)\right] \leq \binom{q_C}{2}\epsilon_1.\end{aligned}$$

bad$_3$ ***and bad***$_4$***.*** Events bad_3 and bad_4 represent the cases that a construction query to the *real construction* $\widetilde{E}$ would feed the same key and input (H_2^i, X^i) to the underlying primitive E in the real construction as a primitive query $(\widehat{K}^j, \widehat{X}^j)$. This is equivalent to guessing the hash-function output for the i-th query. Let us consider bad_3 first. Over $L \twoheadleftarrow \mathcal{L}$ and for all $(\widehat{K}^j, \widehat{X}^j)$, the probability of bad_3 is upper bounded by

$$\begin{aligned}\Pr[\mathsf{bad}_3] &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : (X^i, H_2^i) = (\widehat{X}^j, \widehat{K}^j)\right] \\ &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : (H_1^i = M^i \oplus \widehat{X}^j) \wedge (H_2^i = \widehat{K}^j)\right] \\ &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : \mathcal{H}_{1,2}(T^i) = (M^i \oplus \widehat{X}^j, \widehat{K}^j)\right] \\ &\leq q_C \cdot q_P \cdot \epsilon_2\end{aligned}$$

due to Property P2. Using a similar argumentation, it holds that

$$\begin{aligned}\Pr[\mathsf{bad}_4] &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : (X^i, H_2^i) = (\widehat{Y}^j, \widehat{K}^j)\right] \\ &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : (H_3^i = C^i \oplus \widehat{Y}^j) \wedge (H_2^i = \widehat{K}^j)\right] \\ &= \Pr\left[\exists i, j; 1 \leq i \leq q_C, 1 \leq j \leq q_P : \mathcal{H}_{3,2}(T^i) = (C^i \oplus \widehat{Y}^j, \widehat{K}^j)\right] \\ &\leq q_C \cdot q_P \cdot \epsilon_2.\end{aligned}$$

***bad*$_5$ *and* *bad*$_6$.** Events bad_5 and bad_6 represent the cases that two distinct primitive queries feed the same key and the same input to the primitive E. Clearly, in both worlds, this implies that **A** either has asked a duplicate primitive query or has fed the result of an earlier primitive query to the primitive's inverse oracle. Both types of queries are forbidden; so, they will not occur.

***bad*$_7$ *and* *bad*$_8$.** Let us consider bad_7 first, which considers the case that the j-th construction query in encryption direction matches the inputs to E used for generating a hash function subkeys L_i, for some $j \in [1..q]$ and $i \in [1..s]$. bad_8 considers the equivalent case in decryption direction. We define $\Delta := M^j \oplus \mathcal{H}_1(L, T^j)$. For this bad event, it must hold that $M^j \oplus \mathcal{H}_1(L, T^j) = I_i$ and $\mathcal{H}_2(L, T^j) = K_i$. Concerning the tuples I_i, K_i, we cannot exclude in general that all values $K_1(K) = \ldots = K_s(K)$ are equal and therefore, L_i are outputs of the same permutation. From Property P3 and the fact that there have been j queries and the adversary can hit one out of s values, and over $L \twoheadleftarrow \mathcal{L}$, it follows that the probability for this event can be upper bounded by

$$\begin{aligned}
\Pr[\mathsf{bad}_7] &= \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_C : (X^j, H_2^j) \oplus (I_i, K_i) = (\Delta, 0^k)\right] \\
&= \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_C : \mathcal{H}_{1,2}(T^j) \oplus (I_i, K_i) = (\Delta, 0^k)\right] \\
&\le q_C \cdot s \cdot \epsilon_3.
\end{aligned}$$

Using a similar argument, it follows from Property P4 that

$$\begin{aligned}
\Pr[\mathsf{bad}_8] &= \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_C : (Y^j, H_2^j) \oplus (L_i, K_i) = (\Delta, 0^k)\right] \\
&= \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_C : \mathcal{H}_{3,2}(T^j) \oplus (L_i, K_i) = (\Delta, 0^k)\right] \\
&\le q_C \cdot s \cdot \epsilon_4.
\end{aligned}$$

***bad*$_9$ *and* *bad*$_{10}$.** The event bad_9 models the case that a primitive query in encryption direction matches key and input used for generating L_i, for some $i \in [1..s]$: $(\widehat{X}^j, \widehat{K}^j) = (I_i, K_i)$. The event bad_{10} considers the equivalent case in decryption direction. From our assumption that Property P5 holds and the fact that the adversary can hit one out of s values, and over $K \twoheadleftarrow \mathcal{K}$, the probability for this event can be upper bounded by

$$\Pr[\mathsf{bad}_9] = \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_P : \left(\widehat{X}^j, \widehat{K}^j\right) = (I_i, K_i)\right] \le q_P \cdot s \cdot \epsilon_7 5$$

We can use a similar argument and Property P5 to upper bound the probability that the j-th query of **A** hits L_i, K_i by

$$\Pr[\mathsf{bad}_{10}] = \Pr\left[\exists i, j; 1 \le i \le s, 1 \le j \le q_P : \left(\widehat{Y}^j, \widehat{K}^j\right) = (L_i, K_i)\right] \le q_P \cdot s \cdot \epsilon_5.$$

***bad*$_{11}$.** It is possible that a number of key inputs $K_i = K_j$, for some $i, j \in \{1, \ldots, s\}$, $i \ne j$, are equal. The event bad_{11} models the case that the ideal oracle produces a collision $(K_i, L_i) = (K_j, L_j)$, although it holds that $I_i \ne I_j$, which indicates that the hash-function keys cannot be result of computing them from the block cipher E. In the worst case, all keys K_i, for $1 \leq i \leq s$, are equal. So, the probability for this event can be upper bounded by

$$\Pr[\mathsf{bad}_{11}] = \Pr\left[\exists i, j \in \{1, \ldots, s\}, i \neq j : (K_i, L_i) = (K_j, L_j), I_i \neq I_j\right] \leq \frac{s^2}{2^{n+1}}.$$

Our claim in Lemma 2 follows from summing up the probabilities of all bad events. □

Before proceeding with the proof of good transcripts, we formulate a short fact that will serve useful later on. In the remainder, we denote the falling factorial as $(n)_k := \frac{n!}{k!}$.

Fact 1. Let $u_1, \ldots, u_r$ and $v_1, \ldots, v_s$ be positive integers such that it holds

$$\sum_{i=1}^{r} u_i = \sum_{j=1}^{s} v_j, \tag{1}$$

$$r \leq s, \quad \text{and} \tag{2}$$

$$v_i \leq u_i, \quad \text{for all } 1 \leq i \leq r. \tag{3}$$

Then, it holds for any positive integer $N \geq \sum_{i=1}^{r} u_i$ that

$$\prod_{i=1}^{r} (N)_{u_i} \leq \prod_{i=1}^{s} (N)_{v_i} \quad \text{and thus} \quad \prod_{i=1}^{r} \frac{1}{(N)_{u_i}} \geq \prod_{i=1}^{s} \frac{1}{(N)_{v_i}}.$$

The proof follows from simple arithmetics and is therefore omitted.

A.2 Proof of Lemma 3

Proof. Fix a good transcript τ. In the ideal world, the probability to obtain τ is

$$\Pr[\Theta_{\text{ideal}} = \tau] = \Pr_{\forall i}\left[\widetilde{\pi}(T^i, M^i) = C^i\right] \cdot \Pr_{\forall j}\left[E(\widehat{K}^j, \widehat{X}^j) = Y^j\right] \cdot \Pr_{\forall g}\left[L_g\right]$$
$$\cdot \Pr\left[K \twoheadleftarrow \mathcal{K} : K\right].$$

In the real world, the probability to obtain a transcript τ is given by

$$\Pr[\Theta_{\text{real}} = \tau] = \Pr_{\forall i, \forall j, \forall g}\left[\widetilde{E}_L(T^i, M^i) = C^i,\ E(\widehat{K}^j, \widehat{X}^j) = Y^j, E(K_g, I_g) = L_g\right]$$
$$\cdot \Pr\left[K \twoheadleftarrow \mathcal{K} : K\right].$$

First, we consider the distribution of keys. In the ideal world, all components of $L = (K, L_1, \ldots, L_s)$ are sampled uniformly and independently at random; the real world employs the block cipher E for generating $L_1, \ldots, L_s$. Let us focus on K, which is sampled uniformly in both worlds:

$$\Pr\left[K \twoheadleftarrow \mathcal{K} : K\right] = \frac{1}{|\mathcal{K}|}.$$

The remaining hash-function key $L_1, \ldots, L_s$ will be considered in turn. To prove the remainder of our claim in Lemma 3, we have to show that

$$\Pr_{\forall i, \forall j, \forall g}\left[\widetilde{E}_L(T^i, M^i) = C^i\,,\, E(\widehat{K}^j, \widehat{X}^j) = Y^j, E(K_g, I_g) = L_g\right] \tag{4}$$
$$\geq \Pr_{\forall i}\left[\widetilde{\pi}(T^i, M^i) = C^i\right] \cdot \Pr_{\forall j}\left[E(\widehat{K}^j, \widehat{X}^j) = Y^j\right] \cdot \prod_{g=1}^{s} \Pr\left[L_g \twoheadleftarrow \{0,1\}^n : L_g\right].$$

We reindex the keys used in primitive queries to $\widehat{\mathsf{K}}^1, \ldots, \widehat{\mathsf{K}}^\ell$ to eliminate duplicates and group all primitive queries into sets $\widehat{\mathcal{K}}^j$, for $1 \leq j \leq \ell$, s.t. all sets are distinct and each set $\widehat{\mathcal{K}}^j$ contains exactly only the primitive queries with key $\widehat{\mathsf{K}}^j$:

$$\widehat{\mathcal{K}}^j := \left\{ \left(\widehat{\mathsf{K}}^i, \widehat{X}^i, \widehat{Y}^i\right) : \widehat{\mathsf{K}}^i = \widehat{\mathsf{K}}^j \right\}.$$

We denote by $\widehat{k}^j = |\widehat{\mathcal{K}}^j|$ the number of queries with key $\widehat{\mathsf{K}}^j$. Clearly, it holds that $\ell \leq q_P$ and $\sum_{j=1}^{\ell} \widehat{k}^j = q_P$.

Moreover, we also re-index the tweaks of the construction queries to $\mathsf{T}^1, \ldots, \mathsf{T}^r$ for the purpose of eliminating duplicates. Given these new indices, we group all construction queries into sets $\mathcal{T}^j$, for $1 \leq j \leq r$, s.t. all sets are distinct and each set $\mathcal{T}^j$ contains exactly only all construction queries with the tweak T^j:

$$\mathcal{T}^j := \left\{ \left(\mathsf{T}^i, M^i, C^i\right) : \mathsf{T}^i = \mathsf{T}^j \right\}$$

We denote by $t^j = |\mathcal{T}^j|$ the number of queries with tweak T^j. It holds that $r \leq q_C$ and $\sum_{j=1}^{r} t^j = q_C$.

First, we consider the probability of an obtained good transcript in the ideal world. Therein, all components $L_1, \ldots, L_s$ are sampled independently uniformly at random from $\{0,1\}^n$. So, in the ideal world, it holds that

$$\prod_{g=1}^{s} \Pr\left[L_g \twoheadleftarrow \{0,1\}^n : L_g\right] = \frac{1}{(2^n)^s}.$$

Recall that every $\widetilde{\pi}(\mathsf{T}^j, \cdot)$ and $\widetilde{\pi}^{-1}(\mathsf{T}^j, \cdot)$ is a permutation, and the assumption that **A** does not ask duplicate queries or such to which it already knows the answer. So, all queries are pairwise distinct. The probability to obtain the outputs of our transcript for some fixed tweak T^j is given by

$$\frac{1}{2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - t^j + 1)} = \frac{1}{(2^n)_{t^j}}.$$

The same applies for the outputs of the primitive queries in our transcript for some fixed key $\widehat{\mathsf{K}}^j$:

$$\frac{1}{(2^n)_{\widehat{k}^j}}.$$

The outputs of construction and primitive queries are independent from each other in the ideal world. Over all disjoint key and tweak sets, the probability for obtaining τ in the ideal world is given by

$$\Pr\left[\Theta_{\text{ideal}} = \tau\right] = \left(\prod_{i=1}^{r} \frac{1}{(2^n)_{t^j}}\right) \cdot \left(\prod_{j=1}^{\ell} \frac{1}{(2^n)_{\widehat{k}^j}}\right) \cdot \frac{1}{(2^n)^s} \cdot \frac{1}{|\mathcal{K}|}. \tag{5}$$

It remains to upper bound the probability τ in the real world. We observe that for every pair of queries i and j with $T^i = T^j$, it holds that $H_2^i = H_2^j$, i.e., both queries always target the same underlying permutation. Moreover, in the real world, two distinct tweaks $T^i \neq T^j$ can still collide in their hash-function outputs $H_2^i = H_2^j$. In this case, the queries with tweaks T^i and T^j also use the same permutation. Furthermore, there may be hash-function outputs H_2^i from construction queries that are identical to keys $\widehat{K}^j$ that were used in primitive queries. In this case, both queries also employ the same permutation and so, the outputs from primitive and from construction queries are not independent as in the ideal world. Moreover, the derived keys L_i are also constructed from the same block cipher E; hence, the inputs K_i may also use the same permutation as primitive and construction queries.
For our purpose, we also reindex the keys in all primitive queries into sets to $\widehat{\mathsf{K}}^1, \ldots, \widehat{\mathsf{K}}^\ell$, and also reindex the tweaks in construction queries to $\mathsf{T}^1, \ldots \mathsf{T}^r$ to eliminate duplicates. We define key sets $\mathcal{K}^j$, for $1 \leq j \leq \ell$, and tweak sets $\mathcal{T}^j$, for $1 \leq j \leq r$, analogously as we did for the ideal world. Moreover, for every so-indexed tweak T^i, we compute its corresponding value H_2^i. We also reindex the hash values H_2^j to $\mathsf{H}_2^1, \ldots, \mathsf{H}_2^u$ for duplicate elimination, and group the construction queries into sets

$$\mathcal{H}_2^j := \left\{ \left(T^i, M^i, C^i\right) : \mathcal{H}_2(L, T^i) = \mathsf{H}_2^j \right\}.$$

We denote by $h_2^j = |\mathcal{H}_2^j|$ the number of queries whose tweak maps to H_2^j. Clearly, it still holds that $\sum_{i=1}^{u} h_2^i = q_C$. We can define an ordering s.t. for all $1 \leq i \leq u$, T^i is mapped to H_2^i. Since for all $1 \leq i \leq r$, all queries of tweak T^j are contained in exactly one set $\mathcal{H}_2^j$, there exists some $j \in \{1, \ldots, u\}$, s.t. it holds

$$\sum_{j=1}^{u} h_2^j = \sum_{i=1}^{r} t^i = q_C, \quad u \leq r, \quad \text{and} \quad h_2^i \geq t^i, \text{ for all } 1 \leq i \leq r.$$

Hence, it follows from Fact 1 that

$$\prod_{j=1}^{u} \frac{1}{(2^n)_{h_2^j}} \geq \prod_{i=1}^{r} \frac{1}{(2^n)_{t^i}}.$$

In addition, we reindex the key inputs K_i that are used for generating the keys $L_1, \ldots, L_s$ to $\mathsf{K}_1, \ldots, \mathsf{K}_w$ to eliminate duplicates, and group all tuples (I_i, K_i)

into sets $\mathcal{K}^j$, for $1 \leq j \leq w$, s.t. all sets are distinct and each set contains exactly those key-generating tuples with the key K_j:

$$\mathcal{K}^j := \left\{(I_i, K_i) : K_i = \mathsf{K}^j.\right\}.$$

On this base, we unify and reindex the values H_2^j, $\widehat{\mathsf{K}}^j$, and K^j to values $\mathsf{P}^1, \ldots, \mathsf{P}^v$ (using P for permutation). We group all queries into sets $\mathcal{P}^j$, for $1 \leq j \leq v$, s.t. all sets are distinct and each set $\mathcal{P}^j$ consists of exactly the union of all construction queries with the hash value $\mathsf{H}_2 = \mathsf{P}^j$, all primitive queries with $\widehat{\mathsf{K}} = \mathsf{P}^j$, and all key-generating tuples with $\mathsf{K} = \mathsf{P}^j$:

$$\mathcal{P}^j := \left\{\mathcal{H}_2^i : \mathsf{H}_2^i = \mathsf{P}^j\right\} \cup \left\{\widehat{\mathcal{K}}^i : \widehat{\mathsf{K}}^i = \mathsf{P}^j\right\} \cup \left\{\mathcal{K}^i : \mathsf{K}^i = \mathsf{P}^j\right\}.$$

We denote by $p^j = |\mathcal{P}^j|$ the number of queries that use the same permutation. Clearly, it holds that $\sum_{j=1}^{v} p^j = q_P + q_C + s$. Recall that $\mathsf{Block}(k,n)$ denotes the set of all k-bit key, n-bit block ciphers. In the following, we call a block cipher E *compatible* with τ iff

1. For all $1 \leq i \leq q_C$, it holds that $C^i = E_{H_2^i}\left(M^i \oplus H_1^i\right) \oplus H_3^i$, where $H_1^i = \mathcal{H}_1(L, \mathsf{T}^i)$, $H_2^i = \mathcal{H}_2(L, \mathsf{T}^i)$, and $H_3^i = \mathcal{H}_3(L, \mathsf{T}^i)$, and
2. for all $1 \leq j \leq q_P$, it holds that $\widehat{Y}^j = E_{\widehat{K}^j}(\widehat{X}^j)$,
3. and for all $1 \leq g \leq s$, it holds that $L_i = E_{K_i}(I_i)$.

Let $\mathsf{Comp}(\tau)$ denote the set of all block ciphers E compatible with τ. Then,

$$\Pr\left[\Theta_{\text{real}} = \tau\right] = \Pr\left[E \twoheadleftarrow \mathsf{Block}\left(k,n\right) : E \in \mathsf{Comp}(\tau)\right] \cdot \Pr\left[K \middle| \Theta_{\text{real}} = \tau\right]. \quad (6)$$

We focus on the first factor on the right-hand side. Since we assume that no bad events have occurred, the fraction of compatible block ciphers is given by

$$\Pr\left[E \twoheadleftarrow \mathsf{Block}\left(k,n\right) : E \in \mathsf{Comp}(\tau)\right] = \prod_{i=1}^{v} \frac{1}{\left(2^n\right)_{p^i}}.$$

It holds that

$$\sum_{i=1}^{v} p^i = q_P + q_C + s = \sum_{j=1}^{\ell} \widehat{k}^j + \sum_{j=1}^{r} t^j + \sum_{j=1}^{w} k^j = \sum_{j=1}^{\ell} \widehat{k}^j + \sum_{j=1}^{u} h_2^j + \sum_{j=1}^{w} k^j.$$

We can substitute the variables $\widehat{k}^j$, h_2^j, and k^j on the right-hand side by auxiliary variables z^j

$$\sum_{i=1}^{v} p^i = \sum_{j=1}^{\ell+u+w} z^j \quad \text{where} \quad z^j = \begin{cases} \widehat{k}^j & \text{if } j \leq \ell, \\ h_2^j & \text{if } \ell < j \leq \ell + u, \\ k^j & \text{otherwise.} \end{cases}$$

It holds that $v \leq \ell + u + w \leq \ell + r + w$. Since each permutation set $\mathcal{P}^i$ consists of all queries in τ that use a certain key $\widehat{\mathsf{K}}^j$, and/or all queries in τ that use one hash H_2^j, and/or all tuples (I_i, K_i) that use one value K^j, it further holds that for all $1 \leq i \leq v$, there exists some $j \in \{1, \ldots, \ell + u + w\}$ s.t.

$$p^i \geq z^j.$$

So, we can directly apply Fact 1, from which it follows that

$$\begin{aligned}
\prod_{i=1}^{v} \frac{1}{(2^n)_{p^i}} &\geq \left(\prod_{j=1}^{\ell} \frac{1}{(2^n)_{\widehat{k}^j}} \right) \cdot \left(\prod_{j=1}^{u} \frac{1}{(2^n)_{h_2^j}} \right) \cdot \left(\prod_{j=1}^{w} \frac{1}{(2^n)_{k^j}} \right) \\
&\geq \left(\prod_{j=1}^{\ell} \frac{1}{(2^n)_{\widehat{k}^j}} \right) \cdot \left(\prod_{j=1}^{r} \frac{1}{(2^n)_{t^j}} \right) \cdot \left(\prod_{j=1}^{w} \frac{1}{(2^n)_{k^j}} \right) \\
&\geq \left(\prod_{j=1}^{\ell} \frac{1}{(2^n)_{\widehat{k}^j}} \right) \cdot \left(\prod_{j=1}^{r} \frac{1}{(2^n)_{t^j}} \right) \cdot \frac{1}{(2^n)^s}.
\end{aligned} \tag{7}$$

Using the combined knowledge from Equations (4) through (7), we can derive that the probability for obtaining the construction and primitive outputs in the transcript is at least as high as the probability in the ideal world:

$$\Pr\left[\Theta_{\text{real}} = \tau\right] \geq \Pr\left[\Theta_{\text{ideal}} = \tau\right].$$

So, we obtain our claim in Lemma 3. $\square$