Deck-Based Wide Block Cipher Modes and an Exposition of the Blinded Keyed Hashing Model

Aldo Gunsing, Joan Daemen and Bart Mennink

Digital Security Group, Radboud University, Nijmegen, The Netherlands
aldo.gunsing@ru.nl, joan@cs.ru.nl, b.mennink@cs.ru.nl

Abstract. We present two tweakable wide block cipher modes from doubly-extendable cryptographic keyed (deck) functions and a keyed hash function: double-decker and docked-double-decker. Double-decker is a direct generalization of Farfalle-WBC of Bertoni et al. (ToSC 2017(4)), and is a four-round Feistel network on two arbitrarily large branches, where the middle two rounds call deck functions and the first and last rounds call the keyed hash function. Docked-double-decker is a variant of double-decker where the bulk of the input to the deck functions is moved to the keyed hash functions. We prove that the distinguishing advantage of the resulting wide block ciphers is simply two times the sum of the pseudorandom function distinguishing advantage of the deck function and the blinded keyed hashing distinguishing advantage of the keyed hash functions. We demonstrate that blinded keyed hashing is more general than the conventional notion of XOR-universality, and that it allows us to instantiate our constructions with keyed hash functions that have a very strong claim on bkh security but not necessarily on XOR-universality, such as Xoofffie (ePrint 2018/767). The bounds of double-decker and docked-double-decker are moreover reduced tweak-dependent, informally meaning that collisions on the keyed hash function for different tweaks only have a limited impact. We describe two use cases that can exploit this property opportunistically to get stronger security than what would be achieved with prior solutions: SSD encryption, where each sector can only be written to a limited number of times, and incremental tweaks, where one includes the state of the system in the variable-length tweak and appends new data incrementally.

Keywords: wide block cipher · tweakable · deck function · double-decker · docked-double-decker · disk encryption · incremental tweak

1 Introduction

Block ciphers have long been the main building block for symmetric cryptography. However, block ciphers operate on data of fixed and predetermined length. For example, DES had a block size of 64 bits and AES has a block size of 128 bits. One can encrypt data of variable length by using a block cipher in a mode of operation, such as counter mode, CBC or OFB.

Security of such modes typically depends on a nonce. This might be a random value that should never be repeated or a counter. In some applications it is desirable to have security even in the absence of a nonce or in the case of accidental nonce violation. For example, with full disk encryption the nonce serves as the sector index. When the content of a sector changes, this is re-encrypted with the same sector index as nonce. Consider an adversary that may have access to the encrypted sector before and after the change. In the case of stream encryption it can derive from this the bitwise difference of the plaintexts: $P \oplus P' = C \oplus C'$. When CBC would be used, it still leaks equality of first blocks. A solution would be to store a counter in the sector that would increment with

each re-encryption, but often it is undesirable to have a difference between the sector size with or without encryption. Another example is the Tor protocol for anonymity. It enciphers the payload in network packets iteratively with different keys. The encryption should be length-preserving and there is no place for a nonce. For these cases, one can use a tweakable *wide block cipher*. This encrypts arbitrarily large strings in such a way that each bit of the ciphertext depends on each bit of the plaintext and vice versa. If the plaintext changes, the ciphertext will look completely random, even if the change is just a single bitflip.

The idea of wide block ciphers is gaining traction: the latest contribution to the field is Adiantum [CB18], a mode that is specifically developed for disk encryption but still fares very well as a general wide block cipher mode. Adiantum follows a structure that reminds of Feistel but it is asymmetric. It uses two branches: an arbitrary-length branch for bulk encryption, and a fixed-length branch that is processed using AES and that is used as a seed for encryption of the arbitrary-length branch. A description of Adiantum is given in Appendix A. Adopting arbitrary-length branches appears like a fruitful direction, but decryption requires evaluation of the inverse of AES. Other notable constructions are HEH [NR97], EME [HR04], HCTR [WFW05], AEZ [HKR15], FAST [CGLS17] and Tweakable HCTR [DN18]. We refer to Table 1 for an overview of these modes.

1.1 Deck-Based Wide Block Cipher Modes

In this work, we formalize and analyze two similar deck-based tweakable wide block cipher modes: *double-decker* and *docked-double-decker*. Double-decker is an immediate generalization of Farfalle-WBC, a wide block cipher construction proposed by Bertoni et al. [BDH+17]. Double-decker follows the well-established Feistel design, but unlike all previous instances, it is based on two arbitrary-length branches and is built upon two arbitrary-length primitives called *doubly-extendable cryptographic keyed (deck)* functions. Deck functions, a notion of Daemen et al. [DHVV18], are cryptographic primitives that have arbitrary-length input and output. In this light, they are *the* natural building block in our wide block cipher mode whose two branches are both arbitrary-length. Double-decker, as such, consists of two Feistel rounds of deck functions surrounded by two Feistel rounds of a keyed hash function.

We also introduce docked-double-decker, a variant of double-decker. It moves the bulk of the input of the deck functions to the keyed hash functions. This illustrates the flexibility that the Feistel structure provides: it does not matter which function absorbs inputs, as long as all input is absorbed by a cryptographic function. Moreover, the input to the deck functions becomes fixed length, as long as the tweak has a fixed length as well. This allows one to conceptually view the deck functions as stream ciphers in this case.

Nonetheless, there is more to the deck-based modes. As deck functions support multiple strings as input, we feed a tweak to the inner rounds. This leads to the actual constructions of Section 4: double-decker depicted in Figure 2 and docked-double-decker depicted in Figure 3.

1.2 Blinded Keyed Hashing Model

Security of cryptographic constructions based on keyed hash functions often relies on the *universality* of the hash function. For example, in the analyses of the schemes listed before, the keyed hash functions are assumed to be ε-XOR-universal for some small value of ε. However, not all keyed hash functions fit well in this model. An example of such function is Xoofffie, whose security claim [DHP+18, Claim 2] uses a different security notion. Based on this claim, we introduce a more general security model for keyed hash functions: *blinded keyed hashes (bkh)*.

In a nutshell, bkh does not look at single pairs of inputs at a time, as XOR-universality does, but instead looks at a whole list of queries at the same time. As not every pair of inputs is as bad as the worst case, this formalization allows for a more fine-grained security analysis and allows one to improve the security claim in some cases.

We show that an ε-XOR-universal function has a bkh advantage of at most $\binom{q}{2}\varepsilon$, but the bound is not necessarily tight. We demonstrate the power of bkh based on Xoofffie. In detail, in Section 3.2 we show that the security claim for constructions based on Xoofffie increases from 64 bits to 128 bits in the common case, just by changing the security model from XOR-universality to bkh. This gain propagates through the mode, and henceforth our analyses of double-decker and docked-double-decker will be based on the assumption that the keyed hash functions are sufficiently bkh secure.

1.3 Security

In Section 5 we prove that a simplified security bound for the deck-based wide block cipher modes is of the form

$$2\mathbf{Adv}^{\mathrm{prf}}(\sigma) + \sum_{W} 2\mathbf{Adv}^{\mathrm{bkh}}(q_W, \sigma_W)\,, \tag{1}$$

where σ is the total data complexity, σ_W is the data complexity with tweak W, q_W the number of queries with tweak W, $\mathbf{Adv}^{\mathrm{prf}}$ is the prf-advantage of the deck function and $\mathbf{Adv}^{\mathrm{bkh}}$ is the bkh-advantage of the keyed hash function. Our proof is based on ideas of Iwata and Kurosawa [IK02]. However, a difference is that we base our analysis on the bkh-security of the keyed hash functions rather than their XOR-universality.

1.4 Reduced Tweak-Dependence

Noting that the deck-based modes have a n-bit keyed hash function, a naive security bound would have been of the form

$$2\mathbf{Adv}^{\mathrm{prf}}(\sigma) + 2\mathbf{Adv}^{\mathrm{bkh}}(q, \sigma)\,. \tag{2}$$

In particular, the loss of the keyed hash function is $\mathbf{Adv}^{\mathrm{bkh}}(q, \sigma)$ directly. This is indeed the case for most existing tweakable wide block ciphers, including Adiantum. However, the deck-based modes are *tweakable* wide block ciphers, and the tweak turns out to allow for notable improvement of the bound. Different tweaks separate the domain, hence the underlying deck function should ideally produce independent outputs resulting in independent permutations. As given in Section 1.3, the security bound is of the form in (1).

The bound of (1) significantly improves over the naive bound of (2) if the maximum number of tweak repetitions is limited. For example, if the cipher is called for q/ℓ tweaks W, each tweak is used ℓ times and the hash is a traditional ε-XOR-universal function with $\mathbf{Adv}^{\mathrm{bkh}}(q, \sigma) = \binom{q}{2}\varepsilon$ (see Proposition 1), the loss on the keyed hash function in (1) is of the form

$$(q/\ell)\binom{\ell}{2}\varepsilon = (\ell - 1)q\varepsilon \ll \binom{q}{2}\varepsilon\,, \tag{3}$$

provided that $\ell \ll q$. Examples of such functions that would fulfill this include GHASH [NIS07] and Poly1305 [Ber05]. Moreover, if every tweak is used at most once, so $\ell = 1$, we see that the mode has no security loss to the keyed hash functions. In this way our modes compare favorably with most other tweakable wide block ciphers (see also Table 1). Of course, queries for different tweaks still influence the first term of (1).

Some applications can take advantage of (3) and use different tweaks in the deck-based modes. We demonstrate this in detail in Section 6 via two use cases. One use case is in the context of disk encryption, in Section 6.1. The use case sets physical SSD sector numbers as tweaks, and relies on the fact that the number of write operations to particular SSD sectors is physically limited [BD10]. In the use case detailed in Section 6.1, an adversary has an advantage of at most $2^{46}\varepsilon$ on the hash functions with the deck-based modes whereas it would be of the order $2^{74}\varepsilon$ for the typical security bound that is met by, for example, Adiantum. Here we assume that the differentially uniform function is a traditional ε-XOR-universal function. The second use cases is on incremental tweaks, in Section 6.2. Deck functions have the pleasant property that inputs consist of a sequence of an arbitrary number of strings, and in addition, that the inputs are incremental: appending a string to the sequence costs as much as just processing the new string. One can henceforth use our deck-based modes in a stateful manner, where the tweak is dependent on all or most of prior messages exchanged. This way, each new evaluation of the mode is performed for a new tweak, collisions among hash function evaluations for the same tweak cannot occur (as one needs at least two evaluations), and the loss on the keyed hash function, the second term in (1), vanishes.

1.5 Comparison with Prior Solutions

Various tweakable block ciphers have appeared over the last years. The most notable examples include HCTR [WFW05], HEH [NR97], FAST [CGLS17], Tweakable HCTR [DN18] and most recently Adiantum [CB18]. A comparison among these ciphers, double-decker and docked-double-decker can be found in Table 1. Most of these constructions resemble a Feistel network. However, (Tweakable) HCTR and Adiantum use a block cipher on a branch. The size of this branch is bounded by the size of a typical block cipher: 128 bits. As the birthday bound applies in some use cases, this limits their security to 64 bits. Furthermore, they require the use of the inverse block cipher, incurring additional implementation cost for most block ciphers. Double-decker and docked-double-decker do not have this limitation, without requiring extra passes over the bulk of the data.

Furthermore, double-decker and docked-double-decker have the nice security property of reduced tweak-dependence (see Section 1.4). This means that collisions in the keyed hash functions do not reduce the security as long as different tweaks were used. A feature that is not fulfilled by many earlier schemes; only by Tweakable HCTR.

To evaluate the efficiency of the deck-based modes, we need to look into instantiations with concrete deck functions and keyed hash functions. Recently, such functions have been proposed in [BDH$^+$17], Kravatte and Short-Kravatte, and in [DHVV18, DHP$^+$18], Xoofff and Xoofffie. The former are built on the 1600-bit permutation Keccak-p with 6 rounds and the latter on the 384-bit permutation Xoodoo. For the latter, [DHVV18, Table 5] gives performance numbers. On ARM Cortex M0 and M3, representative for low-end CPU, Xoofff and Xoofffie process (long) input or output about 4 times faster than AES. So a Xoofff call with N bits of input and M bits of output is about four times as fast as applying AES-128 CBC-MAC to an N-bit input and then AES-128 in counter mode generating an M-bit keystream.

On high-end CPUs like Intel Skylake the presence of dedicated AES instructions makes the difference between Xoofff and AES smaller, but still on the recent Intel SkylakeX, Xoofff beats AES in speed.

Of course Xoofff has not gone through the amount of scrutiny that Rijndael/AES has, but these performance numbers indicate that there is great potential should Xoofff and Xoofffie turn out to stand up to their security claims, that are actually much more ambitious than those of AES.

Table 1: Comparison of double-decker and docked-double-decker with state of the art. N is the length of the message, n is a security parameter (i.e., 128 or 256). The cost of keyed hash functions, stream ciphers (SC) and deck functions are displayed as the number of processed bits plus the number of generated bits. The cost of (tweakable) block ciphers ((T)BC) is displayed as only the number of processed bits (or equivalent in the case of AEZ). Refer to Section 1.5 for more information about the properties "Inverse free" and "Reduced tweak-independence".

Mode	Keyed hash	SC	Deck	(T)BC	Inverse free	Reduced tweak-dependence	Reference
HEH	$4N$	—	—	N	no	no	[NR97]
EME	—	—	—	$2N+2$	no	no	[HR04]
HCTR	$2N$	—	—	N	no	no	[WFW05]
AEZ	—	—	—	$N+4$	yes	no	[HKR15]
FAST	$2N+4n$	—	—	N	yes	no	[CGLS17]
Tweakable HCTR	$2N$	—	—	N	no	yes	[DN18]
Adiantum	$2N$	N	—	n	no	no	[CB18]
Double-decker	$N+2n$	—	$2N$	—	yes	yes	Figure 2
Docked-double-decker	$2N$	—	$N+2n$	—	yes	yes	Figure 3

1.6 Related Work

Luby and Rackoff considered the Feistel construction with fixed-length independent pseudorandom function in each round [LR88]. They proved that three rounds are sufficient to obtain a pseudorandom permutation that can be used in forward direction, and four rounds are sufficient to obtain a strong pseudorandom permutation that can be used in both forward and inverse direction. A large amount of research has been aimed at reducing the requirements for the underlying primitives. One way to do so is by reducing the number of independent primitives [Pie90, Pat92]. Another avenue is in replacing cryptographically strong pseudorandom functions by (weaker) universal hash functions. The idea was first suggested by Lucks [Luc96] and was further investigated in [PRS02, IK02]. Particularly relevant to our work is the analysis of Iwata and Kurosawa [IK02] that considered a four-round Feistel construction where the two internal rounds are based on a single pseudorandom function and the two outer rounds on two independent universal hash functions with certain conditions.

Above-mentioned results so far achieve $n/2$-bit security, where n is the size of the branches. Patarin proved security beyond $n/2$ of the Feistel construction with 5 or more rounds [Pat98, Pat03, Pat04]. We refer to Nachef et al. [NPV17] for a detailed discussion of the security of multiple-round Feistel networks.

One can consider length doublers as a specific type of wide block ciphers. Length doublers use a block cipher with block size n, and transform it to an encryption primitive that allows encryption of strings of length between n and $2n-1$. Due to their flexibility, they suit well as building blocks for arbitrary-length (authenticated) encryption. Ristenpart and Rogaway [RR07] introduced the concept in 2007 and presented XLS, a construction later broken by Nandi [Nan14]. Other length doublers are DE by Nandi [Nan09] and HEM by Zhang [Zha12], both based on block ciphers and both birthday-bound secure in the block size of the primitive, and LDT by Chen et al. [CLMP17, CMN18], beyond $n/2$ secure but based on a tweakable block cipher.

2 Preliminaries

2.1 Security Model

Our schemes will be parameterized by a security parameter n. This security parameter will restrict messages to be at least $2n$ bits long in our mode. For technical reasons we also limit the maximum message size to some natural number L. This allows us to more easily define a security model, as it is simpler to randomly draw from a finite set. We define message space

$$\mathcal{M} = \bigcup_{i=2n}^{L} \{0,1\}^i \,,$$

key space $\mathcal{K}$, and tweak space $\mathcal{W}$, and consider a tweakable wide block cipher

$$\mathcal{E} : \mathcal{K} \times \mathcal{W} \times \mathcal{M} \to \mathcal{M} \,.$$

For fixed key K, we write $\mathcal{E}_K = \mathcal{E}(K, \cdot, \cdot)$. We require it to preserve length (i.e., $|\mathcal{E}_K(W,P)| = |P|$ for any $(K,W,P) \in \mathcal{K} \times \mathcal{W} \times \mathcal{M}$). We also require it to be invertible for fixed $(K,W) \in \mathcal{K} \times \mathcal{W}$ and denote its inverse with abuse of notation by $\mathcal{E}_K^{-1}$ (i.e., $\mathcal{E}_K^{-1}(W, \mathcal{E}_K(W,P)) = P$ for any $(K,W,P) \in \mathcal{K} \times \mathcal{W} \times \mathcal{M}$).

For a tweakable wide block cipher $\mathcal{E}$ with security parameter n and maximum message size L, the distinguishing advantage of an adversary $\mathcal{D}$ is

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{tprp}}(\mathcal{D}) = \left| \mathbb{P}\Big[K \xleftarrow{\$} \mathcal{K} : \mathcal{D}^{\mathcal{E}_K, \mathcal{E}_K^{-1}} = 1\Big] - \mathbb{P}\Big[\pi \xleftarrow{\$} \mathrm{TPerm}(\mathcal{M}) : \mathcal{D}^{\pi, \pi^{-1}} = 1\Big] \right| ,$$

where $\mathrm{TPerm}(\mathcal{M})$ is the set of all length-preserving tweakable permutations over the message space $\mathcal{M}$, i.e., the set of all $\pi : \mathcal{W} \times \mathcal{M} \to \mathcal{M}$ such that $|\pi(W,P)| = |P|$ for all $(W,P) \in \mathcal{W} \times \mathcal{M}$ and $\pi(W, \cdot)$ invertible for any fixed $W \in \mathcal{W}$. Again, we denote its inverse with abuse of notation by π^{-1}.

2.2 Deck Functions

We adopt a simplification of the definition of a deck function of Daemen et al. [DHVV18]. A deck (doubly-extendable cryptographic keyed) function F takes as input a secret key $K \in \mathcal{K}_F$, two arbitrarily long strings $W, X \in \{0,1\}^*$, produces a potentially infinite string of bits and takes from it the range starting from a specified offset $q \in \mathbb{N}$ and for a specified length $n \in \mathbb{N}$. We denote this as

$$0^n \oplus F_K(W,X) \ll q \,.$$

As a deck function has an arbitrary long input and a potentially infinite output, a powerful security definition is that it should behave like a pseudo-random function (prf). This leads to the following security definition for a deck function F:

$$\mathbf{Adv}_F^{\mathrm{prf}}(L, \mathcal{D}) = \left| \mathbb{P}\left[K \xleftarrow{\$} \mathcal{K}_F : \mathcal{D}^{F_K} = 1\right] - \mathbb{P}\left[G \xleftarrow{\$} \mathrm{PRF}[L] : \mathcal{D}^{G} = 1\right] \right| ,$$

where $\mathrm{PRF}[L]$ is the set of all pseudo-random functions with two arguments with maximum size L, i.e., the set of all $G : \mathcal{X} \times \mathcal{X} \to \{0,1\}^L$, where $\mathcal{X} = \cup_{i=0}^{L} \{0,1\}^i$. We define

$$\mathbf{Adv}_F^{\mathrm{prf}}(\sigma) = \sup_{\mathcal{D} \in \mathcal{D}(\sigma)} \mathbf{Adv}_F^{\mathrm{prf}}(L, \mathcal{D}) \,,$$

where $\mathcal{D}(\sigma)$ is the set of all distinguishers such that $\sum_i \sigma^{(i)} \leqslant \sigma$, where $\sigma^{(i)} = |W^{(i)}| + |X^{(i)}| + n^{(i)}$ is the data complexity of query i and where $L = \max_i \sigma^{(i)}$ is the maximum size of the inputs and outputs.

2.3 Differentially Uniform Hash Functions

Consider a keyed hash $H = \{H_K : \{0,1\}^* \to \{0,1\}^n \mid K \in \mathcal{K}_H\}$. It is called ε-*XOR-universal* if for any two distinct elements $X, X' \in \{0,1\}^*$ and any element $Y \in \{0,1\}^n$

$$\mathbb{P}\Big[K \xleftarrow{\$} \mathcal{K}_H : H_K(X) \oplus H_K(X') = Y\Big] \leqslant \varepsilon\,.$$

The value ε is typically of the form 2^{a-n} for a small value a.

2.4 H-Coefficient Technique

Our proof will rely on the H-coefficient technique by Patarin [Pat91, Pat08], but we will follow the modernization of Chen and Steinberger [CS14].

Consider two oracles $\mathcal{O}$ and $\mathcal{P}$, and any distinguisher $\mathcal{D}$ that has query access to either of these oracles. The distinguisher's goal is to distinguish both worlds and we denote by

$$\mathbf{Adv}(\mathcal{D}) = \big|\mathbb{P}[\mathcal{D}^{\mathcal{O}} = 1] - \mathbb{P}[\mathcal{D}^{\mathcal{P}} = 1]\big|$$

its advantage. If we denote the maximum amount of queries by q, we can define a transcript τ which summarizes all query-response tuples seen by the distinguisher during its interaction with its oracle $\mathcal{O}$ or $\mathcal{P}$. Denote by $X_{\mathcal{O}}$ (resp., $X_{\mathcal{P}}$) the probability distribution of transcripts when interacting with $\mathcal{O}$ (resp., $\mathcal{P}$). We call a transcript $\tau \in \mathcal{T}$ attainable if $\mathbb{P}[X_{\mathcal{P}} = \tau] > 0$, or in other words if the transcript τ can be obtained from an interaction with $\mathcal{P}$.

Lemma 1 (H-coefficient technique)**.** *Consider a fixed deterministic distinguisher $\mathcal{D}$. Define a partition $\mathcal{T} = \mathcal{T}_{\text{good}} \cup \mathcal{T}_{\text{bad}}$, where $\mathcal{T}_{\text{good}}$ is the subset of $\mathcal{T}$ which contains all the "good" transcripts and $\mathcal{T}_{\text{bad}}$ is the subset with all the "bad" transcripts. Let $0 \leqslant \varepsilon \leqslant 1$ be such that for all $\tau \in \mathcal{T}_{\text{good}}$:*

$$\frac{\mathbb{P}[X_{\mathcal{O}} = \tau]}{\mathbb{P}[X_{\mathcal{P}} = \tau]} \geqslant 1 - \varepsilon\,.$$

Then, we have $\mathbf{Adv}(\mathcal{D}) \leqslant \varepsilon + \mathbb{P}[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}]$.

Conventionally, $\mathcal{O}$ corresponds to the real world (in our case $\mathcal{E}_K, \mathcal{E}_K^{-1}$) and $\mathcal{P}$ to the ideal world (in our case π, π^{-1}). With this lemma, we can prove an upper bound by defining good and bad transcripts. It tells us that the advantage will be small, as long as the good transcripts are almost as likely to occur in the real world as in the ideal world, and the probability that a bad transcript happens in the ideal world is small.

3 Blinded Keyed Hashes

Consider a keyed hash $H = \{H_K : \{0,1\}^* \to \{0,1\}^n \mid K \in \mathcal{K}_H\}$. We do not look at its output directly, instead we first blind it through a random oracle: a function that returns an independent random value for every new input. The keyed hash is called a *blinded keyed hash (bkh)* if the distinguishing advantage between the following two worlds, with inputs (X, Δ), is small:

- Real world $\mathcal{O}$: $\mathcal{RO}_1(H_K(X) \oplus \Delta)$ with $K \xleftarrow{\$} \mathcal{K}_H$ and a secret random oracle $\mathcal{RO}_1$;
- Ideal world $\mathcal{P}$: $\mathcal{RO}_2(X, \Delta)$ with a secret random oracle $\mathcal{RO}_2$.

We denote its advantage by

$$\mathbf{Adv}_H^{\text{bkh}}(\mathcal{D}) = \big|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}} = 1\right] - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}} = 1\right]\big|\,.$$

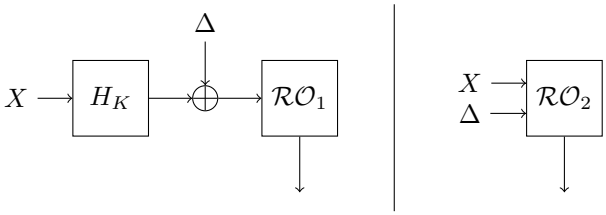


Figure 1: The distinguishability setup for bkh. Left is the real world $\mathcal{O}$ and right is the ideal world $\mathcal{P}$.

For a fixed list of queries Q, we define

$$\mathbf{Adv}_H^{\mathrm{bkh}}(Q) = \sup_{\mathcal{D}\in\mathcal{D}(Q)} \mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D})\,,$$

where $\mathcal{D}(Q)$ is the set of all distinguishers that make the fixed list of queries Q. And for a total number of q queries with data complexity $\sigma = \sum_i |X^{(i)}| + |\Delta^{(i)}|$, we define

$$\mathbf{Adv}_H^{\mathrm{bkh}}(q,\sigma) = \sup_{\mathcal{D}\in\mathcal{D}(q,\sigma)} \mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D})\,,$$

where $\mathcal{D}(q,\sigma)$ is the set of all distinguishers that make at most q queries with data complexity $\sigma = \sum_i |X^{(i)}| + |\Delta^{(i)}|$.

3.1 Characterization of the Advantage

The bkh security notion makes use of a random oracle to blind the output of the hash function. This prevents the distinguisher from seeing the output of the hash function. It can only know whether there was a collision in the input to the random oracle, and nothing more. We formally show that this is indeed the case in the following lemma.

Lemma 2. *Let $Q = (X^{(1)},\Delta^{(1)}),\ldots,(X^{(q)},\Delta^{(q)})$ be a list of q distinct queries and let $bad(Q)$ be the event that there exist $1 \leqslant i < j \leqslant q$ such that $H_K(X^{(i)}) \oplus \Delta^{(i)} = H_K(X^{(j)}) \oplus \Delta^{(j)}$. Then,*

$$\mathbf{Adv}_H^{\mathrm{bkh}}(Q) = \mathbb{P}[bad(Q)]\,.$$

Proof. Let Q be a list of queries. We prove the equality by proving both corresponding inequalities. Let $\mathcal{D}$ be a distinguisher that makes the queries Q. Then

$$\begin{aligned}\mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}) &= \left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1\right] - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\right| \\ &= \left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid bad(Q)\right]\cdot\mathbb{P}[bad(Q)] + \mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid \neg bad(Q)\right]\cdot\mathbb{P}[\neg bad(Q)]\right. \\ &\quad \left. - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\right|\,.\end{aligned}$$

As long as $bad(Q)$ does not occur, there are no internal collisions. This means that all outputs are independent outputs from a random oracle, in both the real and ideal world. In other words, they are indistinguishable in this case, hence $\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid \neg bad(Q)\right] = \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]$. From this we get

$$\begin{aligned}\mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}) &= \left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid bad(Q)\right]\cdot\mathbb{P}[bad(Q)] + \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\cdot\mathbb{P}[\neg bad(Q)] - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\right| \\ &= \left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid bad(Q)\right]\cdot\mathbb{P}[bad(Q)] - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\cdot\mathbb{P}[bad(Q)]\right| \\ &= \mathbb{P}[bad(Q)]\cdot\left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid bad(Q)\right] - \mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\right|\,.\end{aligned}$$

This gives the first inequality of the lemma, as $\left|\mathbb{P}\left[\mathcal{D}^{\mathcal{O}}=1 \mid bad(Q)\right]-\mathbb{P}\left[\mathcal{D}^{\mathcal{P}}=1\right]\right| \leqslant 1$ implies

$$\mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}) \leqslant \mathbb{P}[bad(Q)]\,.$$

As this holds for all distinguishers $\mathcal{D}$ that makes the queries Q, we find that

$$\mathbf{Adv}_H^{\mathrm{bkh}}(Q) \leqslant \mathbb{P}[bad(Q)]\,.$$

We now look at the other inequality. For every $m \in \mathbb{N}$, we define the distinguisher $\mathcal{D}_m$ as follows. It makes the queries Q and asks for the first m output bits of the random oracle. Then it returns 1 if there is a collision in these outputs, and 0 otherwise. For this distinguisher, we have the following two properties. First, it always finds a collision in the real world when $bad(Q)$ occurs, so $\mathbb{P}\left[\mathcal{D}_m^{\mathcal{O}}=1 \mid bad(Q)\right]=1$. Second, it only falsely finds a collision in the ideal world when there is a collision in the first m output bits of the random oracle. This happens with probability 2^{-m} for all pairs, so $\mathbb{P}\left[\mathcal{D}_m^{\mathcal{P}}=1\right] \leqslant \binom{q}{2}2^{-m}$, where q is the length of the list. We find that

$$\begin{aligned}
\mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}_m) &= \mathbb{P}[bad(Q)] \cdot \left|\mathbb{P}\left[\mathcal{D}_m^{\mathcal{O}}=1 \mid bad(Q)\right]-\mathbb{P}\left[\mathcal{D}_m^{\mathcal{P}}=1\right]\right| \\
&\geqslant \mathbb{P}[bad(Q)] \cdot \left(1-\binom{q}{2}2^{-m}\right) \\
&= \mathbb{P}[bad(Q)] - \binom{q}{2}2^{-m} \cdot \mathbb{P}[bad(Q)] \\
&\geqslant \mathbb{P}[bad(Q)] - \binom{q}{2}2^{-m}\,.
\end{aligned}$$

As $\lim_{m\to\infty}\binom{q}{2}2^{-m}=0$, this means that

$$\mathbf{Adv}_H^{\mathrm{bkh}}(Q) = \sup_{\mathcal{D}\in\mathcal{D}(q,Q)} \mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}) \geqslant \sup_{m\in\mathbb{N}} \mathbf{Adv}_H^{\mathrm{bkh}}(\mathcal{D}_m) \geqslant \mathbb{P}[bad(Q)]\,,$$

which completes the second inequality of the lemma. □

3.2 Relation to Differentially Uniform Functions

The more common definition of differentially uniform functions given in Section 2.3 is roughly equivalent to the definition of blinded keyed hashes. Proposition 1 shows the relation between ε-XOR-universal and bkh functions. We can reduce any ε to a bkh advantage bound and vice-versa, but the ε-XOR-universal definition is more strict. It requires an upper bound ε on the probability of collisions, but not all inputs have to be near this ε as the probabilities do not have to be uniform. On the other hand, the bkh definition allows for more flexibility as it looks at a list of queries and makes a claim about an attack as a whole. Indeed, some functions have much better guarantees with the bkh definition than the ε-XOR-universal one. For example Xoofffie has the following security claim [DHP+18, Claim 2]:

$$\mathbf{Adv}_{\text{Xoofffie}}^{\mathrm{bkh}}(q,\sigma) \leqslant \frac{M}{2^{128}} + \frac{M^2}{2^{n-4}}\,,$$

where $M = \lceil \sigma/384 \rceil$ is the data complexity expressed in the number of input blocks and where n can be chosen variably. If $n-4$ is larger than 256, the quadratic term is negligible and Xoofffie claims to have around 128 bits of security. We would not get this level of security in the claim when we look at the traditional ε-XOR-universal definition. By

Proposition 1, Xoofffie is ε-XOR-universal with

$$\begin{aligned}\varepsilon &= \mathbf{Adv}^{\mathrm{bkh}}_{\mathrm{Xoofffie}}(2, 2 \cdot 384) \\ &\leqslant \frac{2}{2^{128}} + \frac{2^2}{2^{n-4}} \\ &\approx 2^{-127}\,.\end{aligned}$$

Here, we assume that Xoofffie is only called with one block inputs. If it is called with more input, its ε-XOR-universal security becomes even worse. However, with this definition we approximate the advantage of Xoofffie as $\binom{q}{2}/2^{127} \leqslant \binom{M}{2}/2^{127}$, which would suggest that Xoofffie only has around 64 bits of security. By using the bkh definition, we can make use of these better security properties.

To our knowledge, the only currently known cryptographic primitive with a bkh-model security claim is Xoofffie [DHP+18, Claim 2]. As a bkh claim is more general than a differential-uniformity one, proofs using the bkh-model are also valid for the latter. However, conversely, a bkh-model security claim can give a better security bound for high data complexity. In particular, in Farfalle-like keyed hash functions, similar claims can be made when using permutations such as the Ascon permutation [DEMS16], Gimli [BKL+17] or Keccak-$p[r = 6]$ as in Kravatte [BDH+17].

Proposition 1. *The following two properties hold.*

1. *If H is ε-XOR-universal, then H is bkh with advantage $\mathbf{Adv}^{\mathrm{bkh}}_H(q, \sigma) \leqslant \varepsilon\binom{q}{2}$.*
2. *If H is bkh, then H is ε-XOR-universal with $\varepsilon = \mathbf{Adv}^{\mathrm{bkh}}_H(2, 2l)$, where l is the maximum allowed query length.*

Proof. We prove the two properties separately.

1. As the bkh-definition uses a distinguisher between two worlds, we can apply the H-coefficient technique on it. As the distinguishers are computationally unbounded, we can assume without loss of generality that they are deterministic. The transcripts are the queries the distinguisher makes. Let Q be a list of q queries with data complexity σ. It is called *bad* if $bad(Q)$ holds as in Lemma 2. Otherwise, it is called *good*. First, we analyze the bad queries. For every $1 \leqslant i < j \leqslant q$ we have that

$$\begin{aligned}&\mathbb{P}[H_K(X^{(i)}) \oplus \Delta^{(i)} = H_K(X^{(j)}) \oplus \Delta^{(j)}] \\ &= \mathbb{P}[H_K(X^{(i)}) \oplus H_K(X^{(j)}) = \Delta^{(i)} \oplus \Delta^{(j)}] \\ &\leqslant \varepsilon\,.\end{aligned}$$

   Since there are $\binom{q}{2}$ of these pairs, we find that $\mathbb{P}[bad(Q)] \leqslant \varepsilon\binom{q}{2}$. Now we look at the *good* queries. If a list of queries Q is *good*, then there are no internal collisions. This means that all outputs are independent outputs from a random oracle, in both the real and ideal world. In other words, they are indistinguishable in this case, hence

$$\frac{\mathbb{P}[X_{\mathcal{O}} = Q]}{\mathbb{P}[X_{\mathcal{P}} = Q]} = 1\,.$$

   By the H-coefficient technique (Lemma 1) this means that

$$\mathbf{Adv}^{\mathrm{bkh}}_H(q, \sigma) \leqslant \varepsilon\binom{q}{2}\,.$$

2. Let H be bkh and let l be the maximum allowed query length. For any distinct elements $X, X' \in \{0,1\}^*$ with $|X| + n, |X'| + |Y| \leqslant l$ and any element $Y \in \{0,1\}^n$

we can consider a list of queries $Q = \{(X, 0^n), (X', Y)\}$. From Lemma 2 we get

$$\begin{aligned}
&\mathbb{P}\left[H_K(X) \oplus H_K(X') = Y\right] \\
&= \mathbb{P}\left[H_K(X) \oplus 0^n = H_K(X') \oplus Y\right] \\
&= \mathbb{P}[bad(Q)] \\
&= \mathbf{Adv}_H^{\mathrm{bkh}}(Q) \\
&\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(2, |X| + |0^n| + |X'| + |Y|) \\
&\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(2, 2l)\,.
\end{aligned}$$

This means that H is ε-XOR-universal with $\varepsilon = \mathbf{Adv}_H^{\mathrm{bkh}}(2, 2l)$. □

4 Deck-Based Wide Block Cipher Modes

We introduce two modes: double-decker, directly based on Farfalle-WBC [BDH+17], and docked-double-decker, a slight modification of double-decker that moves the bulk of the input from the deck functions to the hash functions.

Double-decker and docked-double-decker are tweakable wide block ciphers that take as input three keys $(K, K_1, K_2) \in \mathcal{K} = \mathcal{K}_H \times \mathcal{K}_F \times \mathcal{K}_F$, a tweak $W \in \mathcal{W}$, and a message $P \in \mathcal{M}$, and transform it to a ciphertext $C \in \mathcal{M}$. Instead of using two keys $K_1, K_2 \in \mathcal{K}_F$, it is also possible to use a single key and apply domain separation between the functions, as that gives two independent functions as well. Both double-decker and docked-double-decker are a generalized four-round Feistel construction based on two independent deck functions F_{K_1} and F_{K_2} and two evaluations of a bkh function H_K.

Double-decker is illustrated in Figure 2. Here, the message P is first split into the inputs to the two branches U and V using a split function *split*. This function *split* takes as input the length of the message P and outputs the length of the left branch, $|U|$. The states U and V are further split into U_L, U_R, V_L and V_R with the outside branches length n, so $|U_L| = |V_R| = n$. This means that we have a condition that $n \leqslant split(|P|) \leqslant |P| - n$ for all P. In other words, *split* can be an arbitrary function, as long as it is defined for input strings of at least $2n$ bits, and its corresponding branches are of length at least n.

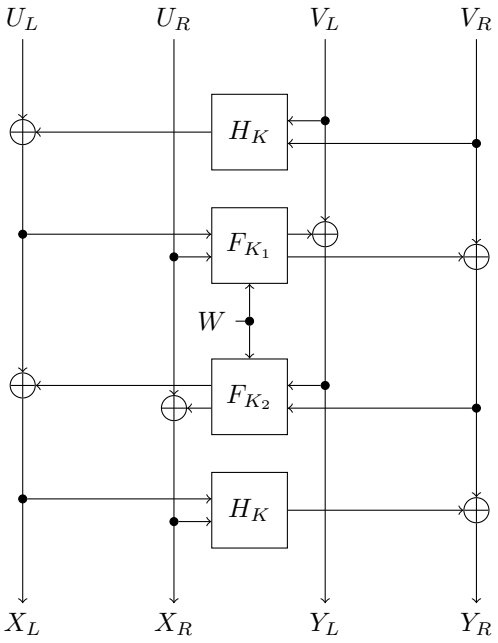
Docked-double-decker is illustrated in Figure 3. It is similar to double-decker, but with two differences. First, the branch V_L is removed, i.e. $split(|P|) = |P| - n$. Second, the bulk of the input to F_{K_1}, on branch U_R, is moved from the deck function F_{K_1} to the first bkh function H_K. This illustrates the flexibility that the Feistel structure provides: it does not matter which function absorbs inputs, as long as all input is absorbed by a cryptographic function. Moreover, the input to the deck functions F_{K_1} and K_{K_2} becomes fixed length, as long as the tweak has a fixed length as well. This allows one to conceptually view the deck functions as stream ciphers in this case.

5 Analysis

We prove security of both the double-decker and the docked-double-decker construction of Section 4.

Theorem 1. *Let $\mathcal{E}$ be either the double-decker construction or the docked-double-decker construction of Section 4. For any distinguisher $\mathcal{D}$ making at most q queries with a length of at most L, we have*

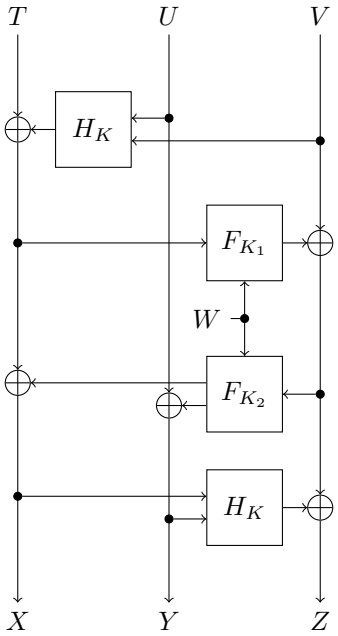
$$\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{tprp}}(\mathcal{D}) &\leqslant \mathbf{Adv}_F^{\mathrm{prf}}(\sigma_{F_1}) + \mathbf{Adv}_F^{\mathrm{prf}}(\sigma_{F_2}) \\
&\quad + \sum_{W \in \mathcal{W}} \left(\mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W}) + \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_2,W}) + \binom{q_W}{2} 2^{-2n} \right),
\end{aligned}$$



function ENCRYPT(W, P)
$U\|V \leftarrow P$, with $|U| = split(|P|)$
$L \leftarrow U \oplus (H_K(V) \parallel 0^{|U|-n})$
$R \leftarrow V \oplus F_{K_1}(W, L)$
$X \leftarrow L \oplus F_{K_2}(W, Y)$
$Y \leftarrow R \oplus (0^{|R|-n} \parallel H_K(X))$
$C \leftarrow X\|Y$
return C

function DECRYPT(W, C)
$X\|Y \leftarrow C$, with $|X| = split(|C|)$
$R \leftarrow Y \oplus (0^{|Y|-n} \parallel H_K(X))$
$L \leftarrow X \oplus F_{K_2}(W, R)$
$V \leftarrow R \oplus F_{K_1}(W, L)$
$U \leftarrow L \oplus (H_K(V) \parallel 0^{|L|-n})$
$P \leftarrow U\|V$
return P

Figure 2: Double-decker tweakable wide block cipher. The parsing of P into states U_L, U_R, V_L, V_R, where $|U_L| = |V_R| = n$ and $|U| = split(|P|)$ is not depicted.



function ENCRYPT(W, P)
$T\|U\|V \leftarrow P$, with $|T| = |V| = n$
$L \leftarrow T \oplus H_K(U\|V)$
$R \leftarrow V \oplus F_{K_1}(W, L)$
$X \leftarrow L \oplus F_{K_2}(W, R)$
$Y \leftarrow U \oplus F_{K_2}(W, R) \lll n$
$Z \leftarrow R \oplus H_K(X\|Y)$
$C \leftarrow X\|Y\|Z$
return C

function DECRYPT(W, C)
$X\|Y\|Z \leftarrow C$, with $|X| = |Z| = n$
$R \leftarrow Z \oplus (H_K(X\|Y)$
$L \leftarrow X \oplus F_{K_2}(W, R)$
$U \leftarrow Y \oplus F_{K_2}(W, R) \lll n$
$V \leftarrow R \oplus F_{K_1}(W, L)$
$T \leftarrow L \oplus (H_K(U\|V)$
$P \leftarrow T\|U\|V$
return P

Figure 3: Docked-double-decker tweakable wide block cipher. The parsing of P into states T, U, V, where $|T| = |V| = n$ is not depicted.

where σ_{F_1}, σ_{F_2} are the total data complexities on each F and where $\sigma_{H_1,W}$, $\sigma_{H_2,W}$ are the data complexities on each H and q_W is the number of queries with tweak W made by

distinguisher $\mathcal{D}$. For double-decker we have

$$\begin{aligned}\sigma_{F_1} = \sigma_{F_2} &= \sum_{i=1}^{q} |W^{(i)}| + |P^{(i)}|\,,\\ \sigma_{H_1,W} &= \sum_{\substack{i=1\\ W^{(i)}=W}}^{q} |P^{(i)}| - \mathit{split}(|P^{(i)}|) + n\,,\\ \sigma_{H_2,W} &= \sum_{\substack{i=1\\ W^{(i)}=W}}^{q} \mathit{split}(|P^{(i)}|) + n\,.\end{aligned}$$

For docked-double-decker we have

$$\begin{aligned}\sigma_{F_1} &= \sum_{i=1}^{q} |W^{(i)}| + 2n\,,\\ \sigma_{F_2} &= \sum_{i=1}^{q} |W^{(i)}| + |P^{(i)}|\,,\\ \sigma_{H_1,W} = \sigma_{H_2,W} &= \sum_{\substack{i=1\\ W^{(i)}=W}}^{q} |P^{(i)}|\,.\end{aligned}$$

The proof of Theorem 1 will be given in the remainder of this section. We first use the triangle inequality to transform the deck functions to random oracles G_1, G_2. This happens at a cost of $\mathbf{Adv}^{\mathrm{prf}}_F(\sigma_{F_1}) + \mathbf{Adv}^{\mathrm{prf}}_F(\sigma_{F_2})$, as in query i the first deck function is called with complexity $\sigma^{(i)}_{F_1}$ and the second one with complexity $\sigma^{(i)}_{F_2}$. These lengths have the following values:

$$\sigma^{(i)}_{F_1} = \begin{cases} |W^{(i)}| + |U^{(i)}| + |V^{(i)}| = |W^{(i)}| + |P^{(i)}| & \text{for double-decker}\,,\\ |W^{(i)}| + |T^{(i)}| + |V^{(i)}| = |W^{(i)}| + 2n & \text{for docked-double-decker}\,,\end{cases}$$

$$\sigma^{(i)}_{F_2} = \begin{cases} |W^{(i)}| + |V^{(i)}| + |U^{(i)}| = |W^{(i)}| + |P^{(i)}| & \text{for double-decker}\,,\\ |W^{(i)}| + |T^{(i)}| + |U^{(i)}| + |V^{(i)}| = |W^{(i)}| + |P^{(i)}| & \text{for docked-double-decker}\,.\end{cases}$$

The rest of the proof is based on ideas of Iwata and Kurosawa [IK02]. It is performed using the H-coefficient technique, which we will first outline in Section 2.4. For this technique, we have to define our transcripts and the way we partition them in *bad* and *good* ones. We do this in Section 5.1. The analysis of the bad transcripts is done in Section 5.2, and the analysis of the good ones is done in Section 5.3, which completes the proof.

5.1 Transcripts

5.1.1 Double-Decker

Distinguisher $\mathcal{D}$ makes q queries to its oracle ($\mathcal{O}$ or $\mathcal{P}$) and these queries are summarized in a transcript of the form

$$\tau = \left(U^{(1)}, V^{(1)}, W^{(1)}, X^{(1)}, Y^{(1)}\right), \ldots, \left(U^{(q)}, V^{(q)}, W^{(q)}, X^{(q)}, Y^{(q)}\right)\,.$$

Note that in the transcript, $P^{(i)}$ is already parsed into $U^{(1)}$ and $V^{(1)}$ implicitly. Additionally, after all the queries are made, the key K is added to the transcript in order to be able to define the bad transcripts. Furthermore, since both worlds are consistent, we can assume

without loss of generality that $\mathcal{D}$ does not make any pointless queries that encrypt a plaintext or decrypt a ciphertext already queried before.

For every $1 \leqslant i \leqslant q$ we define $\ell^{(i)} = \left|U^{(i)}\right| + \left|V^{(i)}\right|$ as the length of the query. We also define $\mathcal{L} = \left\{\ell^{(i)} \mid 1 \leqslant i \leqslant q\right\}$ as the set of used lengths, $q_W = \#\left\{i \mid 1 \leqslant i \leqslant q, W^{(i)} = W\right\}$ as the number of queries with tweak W, and finally $q_{\ell,W} = \#\left\{i \mid 1 \leqslant i \leqslant q, \ell^{(i)} = \ell, W^{(i)} = W\right\}$ as the number of queries with length ℓ and tweak W.

A transcript is called *bad* if there exist $1 \leqslant i < j \leqslant q$ with $W^{(i)} = W^{(j)}$ such that at least one of the following holds:

$$\begin{aligned} bad_1 &: H_K\left(V^{(i)}\right) \oplus U_L^{(i)} = H_K\left(V^{(j)}\right) \oplus U_L^{(j)} \text{ and } U_R^{(i)} = U_R^{(j)}\,, \\ bad_2 &: H_K\left(X^{(i)}\right) \oplus Y_R^{(i)} = H_K\left(X^{(j)}\right) \oplus Y_R^{(j)} \text{ and } Y_L^{(i)} = Y_L^{(j)}\,. \end{aligned}$$

Absence of these events implies that a good transcript has no collision in an input to G_1 or G_2.

5.1.2 Docked-Double-Decker

Distinguisher $\mathcal{D}$ makes q queries to its oracle ($\mathcal{O}$ or $\mathcal{P}$) and these queries are summarized in a transcript of the form

$$\tau = \left(T^{(1)}, U^{(1)}, V^{(1)}, W^{(1)}, X^{(1)}, Y^{(1)}, Z^{(1)}\right), \ldots, \left(T^{(q)}, U^{(q)}, V^{(q)}, W^{(q)}, X^{(q)}, Y^{(q)}, Z^{(q)}\right)\,.$$

Note that in the transcript, $P^{(i)}$ is already parsed into $T^{(1)}$, $U^{(1)}$ and $V^{(1)}$ implicitly. Additionally, after all the queries are made, the key K is added to the transcript in order to be able to define the bad transcripts. Furthermore, since both worlds are consistent, we can assume without loss of generality that $\mathcal{D}$ does not make any pointless queries that encrypt a plaintext or decrypt a ciphertext already queried before.

For every $1 \leqslant i \leqslant q$ we define $\ell^{(i)} = \left|T^{(i)}\right| + \left|U^{(i)}\right| + \left|V^{(i)}\right|$ as the length of the query. We also define $\mathcal{L} = \left\{\ell^{(i)} \mid 1 \leqslant i \leqslant q\right\}$ as the set of used lengths, $q_W = \#\left\{i \mid 1 \leqslant i \leqslant q, W^{(i)} = W\right\}$ as the number of queries with tweak W, and finally $q_{\ell,W} = \#\left\{i \mid 1 \leqslant i \leqslant q, \ell^{(i)} = \ell, W^{(i)} = W\right\}$ as the number of queries with length ℓ and tweak W.

A transcript is called *bad* if there exist $1 \leqslant i < j \leqslant q$ with $W^{(i)} = W^{(j)}$ such that at least one of the following holds:

$$\begin{aligned} bad_1 &: H_K\left(U^{(i)} \| V^{(i)}\right) \oplus T^{(i)} = H_K\left(U^{(j)} \| V^{(j)}\right) \oplus T^{(j)}\,, \\ bad_2 &: H_K\left(X^{(i)} \| Y^{(i)}\right) \oplus Z^{(i)} = H_K\left(X^{(j)} \| Y^{(j)}\right) \oplus Z^{(j)}\,. \end{aligned}$$

Absence of these events implies that a good transcript has no collision in an input to G_1 or G_2.

5.2 Analysis of Bad Transcripts

5.2.1 Double-Decker

Let $W \in \mathcal{W}$ be a tweak and let $Q_W = \{(V^{(i)}, U_L^{(i)}) \mid 1 \leqslant i \leqslant q, W^{(i)} = W\}$ be the set of input queries with tweak W. If event bad_1 occurs with tweak W, that means that

$$H_K\left(V^{(i)}\right) \oplus U_L^{(i)} = H_K\left(V^{(j)}\right) \oplus U_L^{(j)}\,,$$

for some $1 \leqslant i < j \leqslant q$ with $W^{(i)} = W^{(j)} = W$ and $U_R^{(i)} = U_R^{(j)}$. It cannot be the case that $V^{(i)} = V^{(j)}$ and $U_L^{(i)} = U_L^{(j)}$ as well, as that would mean that j is a pointless query. In

other words, all elements of Q_W are distinct, so $bad(Q_W)$ occurred. By Lemma 2 we get

$$\begin{aligned}\mathbb{P}[bad(Q_W)] &\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(Q_W) \\ &\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W})\,.\end{aligned}$$

Since bad_1 can only occur between queries with the same tweak, we can estimate the final probability $\mathbb{P}[bad_1]$ by summing over all tweaks. This means that

$$\mathbb{P}[bad_1] \leqslant \sum_{W \in \mathcal{W}} \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W})\,.$$

Bounding the probability of bad_2 is similar, but with queries $Q'_W = \{(X^{(i)}, Y_R^{(i)}) \mid 1 \leqslant i \leqslant q, W^{(i)} = W\}$ instead of Q_W. Hence

$$\mathbb{P}[X_{\mathcal{P}} \in \mathcal{V}_{\mathrm{bad}}] \leqslant \sum_{W \in \mathcal{W}} \left(\mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W}) + \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_2,W})\right)\,.$$

5.2.2 Docked-Double-Decker

Let $W \in \mathcal{W}$ be a tweak and let $Q_W = \{(U^{(i)} \| V^{(i)}, T^{(i)}) \mid 1 \leqslant i \leqslant q, W^{(i)} = W\}$ be the set of input queries with tweak W. If event bad_1 occurs with tweak W, that means that

$$H_K\left(U^{(i)} \| V^{(i)}\right) \oplus T^{(i)} = H_K\left(U^{(j)} \| V^{(j)}\right) \oplus T^{(j)}\,,$$

for some $1 \leqslant i < j \leqslant q$ with $W^{(i)} = W^{(j)} = W$. As the whole plaintext is contained in the queries, all elements of Q_W have to be distinct as pointless queries are forbidden, so $bad(Q_W)$ occurred. By Lemma 2 we get

$$\begin{aligned}\mathbb{P}[bad(Q_W)] &\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(Q_W) \\ &\leqslant \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W})\,.\end{aligned}$$

Since bad_1 can only occur between queries with the same tweak, we can estimate the final probability $\mathbb{P}[bad_1]$ by summing over all tweaks. This means that

$$\mathbb{P}[bad_1] \leqslant \sum_{W \in \mathcal{W}} \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W})\,.$$

Bounding the probability of bad_2 is similar, but with queries $Q'_W = \{(X^{(i)} \| Y^{(i)}, Z^{(i)}) \mid 1 \leqslant i \leqslant q, W^{(i)} = W\}$ instead of Q_W. Hence

$$\mathbb{P}[X_{\mathcal{P}} \in \mathcal{V}_{\mathrm{bad}}] \leqslant \sum_{W \in \mathcal{W}} \left(\mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_1,W}) + \mathbf{Adv}_H^{\mathrm{bkh}}(q_W, \sigma_{H_2,W})\right)\,.$$

5.3 Analysis of Good Transcripts

Let τ be a good transcript.

Lemma 3. *G_1 and G_2 never receive the same input with the same tweak twice.*

Proof (Proof (for double-decker)). Suppose that G_1 receives the same input with the same tweak for $1 \leqslant i < j \leqslant q$. This immediately implies that $W^{(i)} = W^{(j)}$ and $U_R^{(i)} = U_R^{(j)}$. Furthermore, the first n bits of the input have to be the same, which means that

$$H_K\left(V^{(i)}\right) \oplus U_L^{(i)} = H_K\left(V^{(j)}\right) \oplus U_L^{(j)}\,.$$

But this means that bad_1 occurred, which is impossible as τ is a good transcript.

The case for G_2 is similar, but with bad_2. □

Proof (Proof (for docked-double-decker)). Suppose that G_1 receives the same input with the same tweak for $1 \leqslant i < j \leqslant q$. This immediately implies that $W^{(i)} = W^{(j)}$. Furthermore, the inputs to G_1 have to be the same, which means that

$$H_K\left(U^{(i)} \| V^{(i)}\right) \oplus T^{(i)} = H_K\left(U^{(j)} \| V^{(j)}\right) \oplus T^{(j)} .$$

But this means that bad_1 occurred, which is impossible as τ is a good transcript.

The case for G_2 is similar, but with bad_2. □

By Lemma 3, there are no collisions in the inputs to G_1 and G_2. This means that for every query every output of these functions is independent and equally likely. By choosing these outputs, every ciphertext is possible in an unambiguous way in the case of a forwards query, and similar for the plaintext for a backwards query. This means that the probability that the specific output in query i occurs is equal to $2^{-\ell^{(i)}}$. As these probabilities are independent, the probability that transcript τ occurs is equal to

$$\mathbb{P}[X_{\mathcal{O}} = \tau] = \prod_{i=1}^{q} \frac{1}{2^{\ell^{(i)}}} = \prod_{\substack{\ell \in \mathcal{L} \\ W \in \mathcal{W}}} \frac{1}{2^{\ell \cdot q_{\ell,W}}} .$$

In the ideal world, a random permutation is chosen for every length and tweak. For every length ℓ and tweak W, there are $q_{\ell,W}$ values fixed. This means that there are $\left(2^\ell - q_{\ell,W}\right)!$ possible permutations out of a total of $\left(2^\ell\right)!$. Hence the probability that the transcript τ occurs, equals

$$\mathbb{P}[X_{\mathcal{P}} = \tau] = \prod_{\substack{\ell \in \mathcal{L} \\ W \in \mathcal{W}}} \frac{\left(2^\ell - q_{\ell,W}\right)!}{\left(2^\ell\right)!} .$$

This means that

$$\begin{aligned} \frac{\mathbb{P}[X_{\mathcal{O}} = \tau]}{\mathbb{P}[X_{\mathcal{P}} = \tau]} &= \prod_{\substack{\ell \in \mathcal{L} \\ W \in \mathcal{W}}} \frac{\left(2^\ell\right)!}{2^{\ell \cdot q_{\ell,W}} \left(2^\ell - q_{\ell,W}\right)!} \\ &= \prod_{\substack{\ell \in \mathcal{L} \\ W \in \mathcal{W}}} \frac{1}{2^{\ell \cdot q_{\ell,W}}} \prod_{i=0}^{q_{\ell,W}-1} \left(2^\ell - i\right) \\ &= \prod_{\substack{\ell \in \mathcal{L} \\ W \in \mathcal{W}}} \prod_{i=0}^{q_{\ell,W}-1} \left(1 - \frac{i}{2^\ell}\right) . \end{aligned} \tag{4}$$

We have that

$$\prod_{i=1}^{k} (1 - x_i) \geqslant 1 - \sum_{i=1}^{k} x_i$$

for non-negative x_i, which can be shown by induction to k. Hence

$$\begin{aligned}(4) &\geqslant \prod_{\substack{\ell\in\mathcal{L}\\W\in\mathcal{W}}}\left(1-\sum_{i=0}^{q_{\ell,W}-1}\frac{i}{2^\ell}\right)\\ &= \prod_{\substack{\ell\in\mathcal{L}\\W\in\mathcal{W}}}\left(1-\frac{1}{2^\ell}\binom{q_{\ell,W}}{2}\right)\\ &\geqslant 1-\sum_{\substack{\ell\in\mathcal{L}\\W\in\mathcal{W}}}\frac{1}{2^\ell}\binom{q_{\ell,W}}{2}\\ &\geqslant 1-\frac{1}{2^{2n}}\sum_{\substack{\ell\in\mathcal{L}\\W\in\mathcal{W}}}\binom{q_{\ell,W}}{2}\\ &\geqslant 1-\frac{1}{2^{2n}}\sum_{W\in\mathcal{W}}\binom{q_W}{2}.\end{aligned}$$

6 Application

6.1 Disk Encryption

In our deck-based modes, a naive security loss to the hash functions would be of the form $2\mathbf{Adv}_H^{\mathrm{bkh}}(q,\sigma)$, where q is the total number of queries and σ the total data complexity. However, the more fine-grained security loss given in Theorem 1 is of the form $\sum_{W\in\mathcal{W}} 2\mathbf{Adv}_H^{\mathrm{bkh}}(q_W,\sigma_W)$, where q_W is the number of queries and σ_W the data complexity with tweak W. This is a significant improvement over the naive security loss for some functions if the tweak varies a lot. It does not give a real improvement for hash functions with a more linear security loss in the bkh model. However, some hash functions have a quadratic security loss, even in the bkh model. Examples of such functions include GHASH [NIS07] and Poly1305 [Ber05]. In these cases, the naive security loss to the hash functions would be of the form $2\binom{q}{2}\varepsilon$ and the more fine-grained one given in Theorem 1 would be of the form $\sum_{W\in\mathcal{W}} 2\binom{q_W}{2}\varepsilon$.

This turns out to give a significant gain in the context of disk encryption, if the deck-based modes are called with the physical sector number as tweak. In this case, every sector basically gets its own dedicated wide block cipher. This is especially useful in the context of SSDs. As SSDs get damaged every time data is written to a sector, the firmware of the SSD tries to distribute the data uniformly over its sectors [GT05]. If a sector is written to too many times, the sector cannot be used anymore [BD10]. In the context of the bound of Theorem 1, this means that the distinguisher is limited in its attack: every tweak can only be used at most a constant number of times.

We outline the gain with a concrete example. The Kingston UV500 960 GB [Kin18] has a Total Bytes Written specification of 480 TB. This means that every sector can be written at most 480TB/960GB = 500 times. For a sector size of 4 KiB, which implies $N = 960\mathrm{GB}/4\mathrm{KiB} \approx 2^{28}$ sectors, this concretely means a security loss on the hash functions of

$$2N\binom{500}{2}\varepsilon \approx 2^{46}\varepsilon\,.$$

This is an improvement over the typical birthday bound of

$$2\binom{500N}{2}\varepsilon \approx 2^{74}\varepsilon$$

that would we achieved by, for example, Adiantum [CB18]. Concretely, the security level goes up $\log_2 N$ bits, and the gain would become particularly meaningful if ε is not very small.

6.2 Incremental Tweak

Following Daemen et al. [DHVV18], deck functions have two features:

(i) their inputs consist of a sequence of an arbitrary number of strings, each of arbitrary length;

(ii) their inputs are *incremental*: appending a string to the input sequence costs only the processing of the new string.

These features, along with the reduced tweak-dependence of the bounds of our constructions, leads to pleasant use cases for double-decker and docked-double-decker.

In more detail, in our constructions we already make use of feature (i) by presenting the tweak and intermediate result as a sequence of two strings. We do not yet use feature (ii). However, suppose we make the tweak dependent on prior messages exchanged, or in general on the history of the protocol. It suffices to clone the state of the deck function after absorbing the new tweak string and in the next construction call start from this state. The tweak used in the computation of a construction call will consist of the sequence of the tweak strings that were absorbed in all previous construction calls. This tweak is *different* for each evaluation of double-decker/docked-double-decker, which, in the context of the security bound, implies $q_W = 1$ for each tweak W. Note that this use case comes at modest cost only: by incrementality (feature (ii)), appending data to the earlier sequence costs only the processing of the new string.

We remark that this use case is not far-fetched: similar ideas may be found in several modern protocols. One example is the TLS handshake [TD08], where a hash is computed over all steps and where only this hash is signed later on. Another example is the STROBE protocol framework of Hamburg [Ham17]. STROBE makes use of stateful objects in such a way that all strings absorbed by the object impact its state. Hence, any output generated by the stateful object depends on the concatenation of all that came before.

Acknowledgments. The authors would like to thank Seth Hoffert, Gilles Van Assche and the anonymous reviewers of ToSC for their valuable feedback. Aldo Gunsing is supported by the Netherlands Organisation for Scientific Research (NWO) under TOP grant TOP1.18.002 SCALAR. Joan Daemen is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Bart Mennink is supported by a postdoctoral fellowship from the Netherlands Organisation for Scientific Research (NWO) under Veni grant 016.Veni.173.017.

References

[BD10] Simona Boboila and Peter Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In Randal C. Burns and Kimberly Keeton, editors, *8th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, February 23-26, 2010*, pages 115–128. USENIX, 2010.

[BDH+17] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Farfalle: parallel permutation-based cryptography. *IACR Trans. Symmetric Cryptol.*, 2017(4):1–38, 2017.

[Ber05] Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.

[BKL+17] Daniel J. Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke Todo, and Benoît Viguier. Gimli : A Cross-Platform Permutation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 299–320. Springer, 2017.

[CB18] Paul Crowley and Eric Biggers. Adiantum: length-preserving encryption for entry-level processors. *IACR Trans. Symmetric Cryptol.*, 2018(4):39–61, 2018.

[CGLS17] Debrup Chakraborty, Sebati Ghosh, Cuauhtemoc Mancillas Lopez, and Palash Sarkar. Fast: Disk encryption and beyond. Cryptology ePrint Archive, Report 2017/849, 2017.

[CLMP17] Yu Long Chen, Atul Luykx, Bart Mennink, and Bart Preneel. Efficient Length Doubling From Tweakable Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(3):253–270, 2017.

[CMN18] Yu Long Chen, Bart Mennink, and Mridul Nandi. Short Variable Length Domain Extenders with Beyond Birthday Bound Security. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 244–274. Springer, 2018.

[CS14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to Round 3 of the CAESAR competition, 2016.

[DHP+18] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodoo cookbook. Cryptology ePrint Archive, Report 2018/767, 2018.

[DHVV18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of Xoodoo and Xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

[DN18] Avijit Dutta and Mridul Nandi. Tweakable HCTR: A BBB Secure Tweakable Enciphering Scheme. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, volume 11356 of *Lecture Notes in Computer Science*, pages 47–69. Springer, 2018.

[GT05] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories. *ACM Comput. Surv.*, 37(2):138–163, 2005.

[Ham17] Mike Hamburg. The STROBE protocol framework. Cryptology ePrint Archive, Report 2017/003, 2017.

[HKR15] 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, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44. Springer, 2015.

[HR04] Shai Halevi and Phillip Rogaway. A Parallelizable Enciphering Mode. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.

[IK02] Tetsu Iwata and Kaoru Kurosawa. On the Universal Hash Functions in Luby-Rackoff Cipher. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 226–236. Springer, 2002.

[Kin18] Kingston. UV500 Encrypted SSD. https://www.kingston.com/datasheets/SUV500_us.pdf, 2018.

[LR88] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[Luc96] Stefan Lucks. Faster Luby-Rackoff Ciphers. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 1996.

[Nan09] Mridul Nandi. A Generic Method to Extend Message Space of a Strong Pseudorandom Permutation. *Computación y Sistemas*, 12(3), 2009.

[Nan14] Mridul Nandi. XLS is Not a Strong Pseudorandom Permutation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 478–490. Springer, 2014.

[NIS07] NIST. NIST special publication 800-38D, recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC, November 2007.

[NPV17] Valérie Nachef, Jacques Patarin, and Emmanuel Volte. *Feistel Ciphers - Security Proofs and Cryptanalysis*. Springer, 2017.

[NR97] Moni Naor and Omer Reingold. A pseudo-random encryption mode, 1997.

[Pat91] Jacques Patarin. *Étude des Générateurs de Permutations Basés sur le Schéma du D.E.S.* PhD thesis, Université Paris 6, Paris, France, 1991.

[Pat92] Jacques Patarin. How to Construct Pseudorandom and Super Pseudorandom Permutations from one Single Pseudorandom Function. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1992.

[Pat98] Jacques Patarin. About Feistel Schemes with Six (or More) Rounds. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 103–121. Springer, 1998.

[Pat03] Jacques Patarin. Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 513–529. Springer, 2003.

[Pat04] Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2004.

[Pat08] Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008.

[Pie90] Josef Pieprzyk. How to Construct Pseudorandom Permutations from Single Pseudorandom Functions. In Ivan Damgård, editor, *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 140–150. Springer, 1990.

[PRS02] Sarvar Patel, Zulfikar Ramzan, and Ganapathy S. Sundaram. Luby-Rackoff Ciphers: Why XOR Is Not So Exclusive. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2002.

[RR07] Thomas Ristenpart and Phillip Rogaway. How to Enrich the Message Space of a Cipher. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 101–118. Springer, 2007.

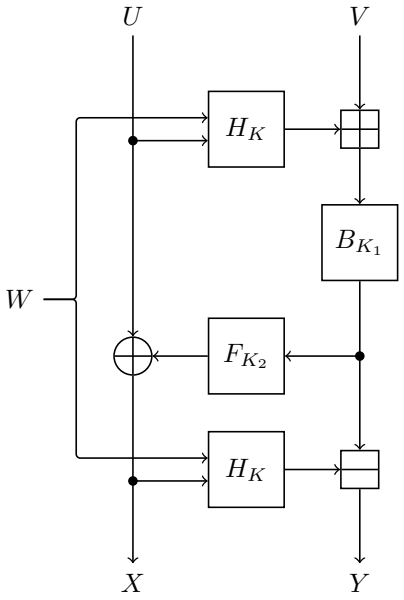
[TD08] E. Rescorla T. Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. Network Working Group of the IETF, RFC 5246, August 2008.

[WFW05] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.

[Zha12] Haibin Zhang. Length-Doubling Ciphers and Tweakable Ciphers. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, volume 7341 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2012.

A Adiantum

Adiantum is a (tweakable) wide block cipher design of Crowley and Biggers [CB18]. We describe its mode, in our notation and terminology, in Figure 4. Here, H is a ε-almost-Δ-universal hash function, F a stream cipher, and B a block cipher.



function ENCRYPT(W, P)
  $U \| V \leftarrow P$, with $|V| = n$
  $R \leftarrow V \boxplus H_K(W, U)$
  $R \leftarrow B_{K_1}(R)$
  $X \leftarrow U \oplus F_{K_2}(R, |U|)$
  $Y \leftarrow R \boxminus H_K(W, X)$
  $C \leftarrow X \| Y$
  return C

function DECRYPT(W, C)
  $X \| Y \leftarrow C$, with $|Y| = n$
  $R \leftarrow Y \boxplus H_K(W, X)$
  $U \leftarrow X \oplus F_{K_2}(R, |X|)$
  $R \leftarrow B_{K_2}^{-1}(R)$
  $V \leftarrow R \boxminus H_K(W, U)$
  $P \leftarrow U \| V$
  return P

Figure 4: Adiantum mode (originally called HBSH in [CB18]).