Efficient Tweakable Enciphering Schemes from (Block-Wise) Universal Hash Functions

Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
email: palash@isical.ac.in

Abstract. We present several constructions of tweakable enciphering schemes which use a single encryption layer between two layers of universal hash function computation. The earliest known construction of this type is due to Naor and Reingold, where the encryption layer is the electronic codebook mode. A more recent work of this type is TET and is due to Halevi at Crypto 2007. We present a new construction Ψ of an invertible block-wise almost universal hash function. Using this we construct a tweakable enciphering scheme HEH. For variable length messages HEH has better efficiency than TET, while for fixed length messages HEH provides better key agility. HEH can only handle messages whose lengths are multiples of the block length. To tackle this, we define variants of Ψ and present a construction HEH* which can handle partial blocks. We show that the basic universal hash function can be combined with the counter mode of operation and the output feedback (OFB) mode to obtain new tweakable enciphering schemes of the hash-Ctr-hash and the hash-OFB-hash type. The hash-Ctr-hash type construction improves upon previous work, while the hash-OFB-hash construction is the first proposal using the OFB mode. An important feature of our work is to show that a new class of polynomials defined by Bernstein can be used to construct the universal hash function. This results in an improvement of efficiency of the hashing layers by almost a factor of two. From a practical point of view, our constructions provide the currently best known algorithms for disk encryption protocols.[1]
Keywords: modes of operations, tweakable encryption, strong pseudo-random permutation, disk encryption.

1 Introduction

A block cipher is a fundamental primitive in cryptography. The formal model of a block cipher is that of a pseudo-random permutation (PRP) or a strong PRP (SPRP) [10]. By itself, a block cipher can encrypt fixed length strings. A mode of operation extends the domain of a block cipher to longer and variable length strings.

A variable input length SPRP can be considered to be a mode of operation of a block cipher. The notion of tweakable block cipher was formalized by Liskov, Rivest and Wagner [9]. This notion was extended to variable input length tweakable SPRP (also called tweakable enciphering scheme) by Halevi and Rogaway [7]. Earlier, a method for constructing SPRPs was given by Naor and Reingold [14]. An important application of tweakable SPRP is that of disk encryption as has been pointed out in [7].

1.1 Previous Work

At a top level, the currently known constructions can be divided into two types: the encrypt-mix-encrypt approach and the hash-encrypt-hash approach. The electronic codebook (ECB) mode and

[1] An earlier version of the work has appeared as [16]. This version substantially expands [16] and provides several new constructions.

the counter mode has been suggested in the literature for instantiating the encryption layer in the second approach. We briefly review the main features of the known approaches with focus on the hash-ECB-hash approach.

Hash-ECB-Hash. Naor-Reingold was the first to propose the hash-ECB-hash approach for the construction of strong pseudorandom permutation. A sketch of a mode of operation based on their more theoretical paper [14] was given in [13]. The construction is the following.

$$h_2^{-1} \circ \mathrm{ECB} \circ h_1 \tag{1}$$

where h_1 and h_2 are invertible block-wise almost XOR universal hash functions and ECB denotes a layer of electronic codebook encryption. (See Definition 1 for the definitions of almost universal hash functions and almost XOR universal hash functions.) One can choose $h_1 = h_2$, so it is sufficient to specify one hash function h. As given in [13], the function h is the following map.

$$(X_1, \ldots, X_m) \mapsto (X_1 \oplus Y, \ldots, X_m \oplus Y) \oplus (u_2(1), \ldots, u_2(m)), \tag{2}$$

where $Y = u_1(X_m \oplus g(X_1, \ldots, X_{m-1}))$. Here g, u_1 and u_2 are almost XOR universal hash functions. Exact specifications for g, u_1 and u_2 were not provided. In fact, the description was really meant to be a sketch. Concrete security bounds, methods for tackling partial blocks and variable length messages were also not discussed.

A construction which follows and develops upon the Naor-Reingold approach is TET [6]. This construction is a complete specification with concrete security bounds, and methods for tackling partial blocks. The hashing functions h_1 and h_2 are chosen to be different (though similar). Below we describe the first hashing layer for the encryption function of TET. For ease of comparison to the Naor-Reingold approach, we describe the situation only for full blocks.

$$(X_1, \ldots, X_m) \mapsto (X_1 \oplus Y, \ldots, X_m \oplus Y) \oplus (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta) \tag{3}$$

where $Y = \sigma^{-1}(X_1\tau^m \oplus \cdots \oplus X_{m-1}\tau^2 \oplus X_m\tau)$ and $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$. The computation is over the finite field $GF(2^n)$. The hashing key is (τ, β) and α is a primitive element of $GF(2^n)$.

For fixed length messages, TET has good performance. It, however, has two drawbacks. First, it is not suited for variable length messages and second, for fixed length messages, the key agility of TET is not good, in the sense that a lot of computation needs to be done for every key change. In fact, the drawback of TET for variable length messages has been mentioned in [6, Page 423] itself: "Hence, TET is not very appealing as a variable-input-length mode". Key agility was not considered in [6] at all.

PEP [4] is another construction (which appeared earlier to TET) which is also of the hash-ECB-hash type. We do not discuss PEP, since it is slower than TET.

Hash-Ctr-Hash. Here the counter mode of operation is used to instantiate the encryption layer. The first construction of this type is XCB [11]. Later constructions are HCTR [18] and HCH [3]. When they appeared, XCB did not have a concrete security bound and HCTR had a cubic bound. Recently, however, quadratic bounds have been proved for both – in [12] for XCB and in [2] for HCTR. In terms of efficiency, all three constructions have similar efficiency with XCB being slightly slower than the other two.

Encrypt-Mix-Encrypt. This approach was introduced by Halevi and Rogaway [7]. In this work, they used the cipher block chaining (CBC) mode for the encryption layers. Later work by Halevi and Rogaway [8] and Halevi [5] showed how to use ECB for the encryption layers. We do not discuss the details of this approach, since we do not present any construction of this type.

1.2 Our Contributions

In this work, we present several constructions of tweakable enciphering schemes using the hash-encrypt-hash approach.

Electronic Codebook Mode. Our first construction is HEH and falls within the hash-ECB-hash approach. This requires a new construction of invertible block-wise almost universal hash function. The construction is of the type

$$(X_1, \ldots, X_m) \mapsto (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus (\alpha\beta, \ldots, \alpha^{m-1}\beta, \beta) \tag{4}$$

where

$$Y = X_m \oplus \tau\psi_\tau(X_1, \ldots, X_{m-1}) \tag{5}$$

and ψ is a suitable universal hash function.

Note that the structure of the last component in (4) is different from that of (2) and (3). Basically, in (4), the last component is Y while in (2) and (3) it is $X_m \oplus Y$. Though this may appear minor, it is due to this difference that we do not have any restriction on the hashing key τ. On the other hand, (3) has the restriction that the hashing key τ must be chosen such that $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$ is non-zero and the inverse of the hash function requires the inverse of σ.

For the construction of the strong pseudorandom permutation, both the NRmode [13] and TET [6] require XOR universal hash function. In contrast, our construction is only universal and not XOR universal. We show that this is sufficient to obtained the desired property. Thus, while our construction indeed falls within the hash-ECB-hash approach, it is different from the Naor-Reingold construction. TET, on the other hand, follows the Naor-Reingold approach more faithfully, as can be seen from (2) and (3).

The hash function given in (4) cannot handle partial blocks and so the SPRP based on it also cannot handle partial blocks. We obtain two variants of (4) to tackle partial blocks and define the construction HEH* based on one of the variants.

In (5) a universal hash function $\psi_\tau(X_1, \ldots, X_{m-1})$ is used. Typically, this is instantiated using a polynomial, i.e., $\psi_\tau(X_1, \ldots, X_{m-1}) = X_1\tau^{m-2} \oplus X_2\tau^{m-3} \oplus \cdots \oplus X_{m-1}$. Evaluating this polynomial using Horner's rule requires $(m-2)$ multiplications over $GF(2^n)$. We let $\mathsf{Poly}_\tau(X_1, \ldots, X_m)$ be the polynomial in τ of degree $(m-1)$ with coefficients (from highest power to lowest power) $X_1, \ldots, X_m$.

Recently, Bernstein [1] has introduced another class of polynomials by building upon previous work due to Rabin and Winograd [15]. We call these the BRW polynomials. By $\mathsf{BRW}_\tau(X_1, \ldots, X_m)$ we denote the polynomial in τ obtained from $X_1, \ldots, X_m$. (See Section 2.2 later for the exact definition of BRW.) Following [1], the map BRW_τ is an injective embedding from $GF(2^n)^m$ to the polynomial ring $GF(2^n)[\tau]$.

The main feature of $\mathsf{BRW}_\tau(X_1, \ldots, X_m)$ is that it can be evaluated using approximately $m/2$ multiplications over $GF(2^n)$. (This was pointed out to us by Daniel J. Bernstein.) On the other hand, the degree of $\mathsf{BRW}_\tau(X_1, \ldots, X_m)$ is approximately twice the degree of $\mathsf{Poly}_\tau(X_1, \ldots, X_m)$.

We show that the $\psi_\tau(X_1, \ldots, X_{m-1})$ used in (5) can be instantiated using $\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1})$. This is significant since it almost halves the number of multiplications required in evaluating ψ. The trade-off is an insignificant degradation in the security bound by a factor of 2 (due to the degree of BRW_τ being about twice the degree of Poly_τ).

There is another issue regarding Poly versus BRW. If τ is fixed, then by pre-computing a multiplication table for τ, it is possible to significantly speed up the computation of $\mathsf{Poly}_\tau(X_1, \ldots, X_m)$. This

is not possible for $\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1})$. The issue of pre-computation, however, is contentious. For one thing, using large pre-computed tables leads to caching problems. Secondly, in our applications, the pre-computed table will be secret and hence will require secure storage. Managing a large number of pre-computed tables in a multi-key situation can be a problem. So, overall, though pre-computation can be an advantage, it can also be a mixed blessing. One needs to carefully weigh the options at hand before deciding on using pre-computed multiplication tables.

The application of BRW to the construction of HEH is more or less straightforward. However, when we consider partial blocks in HEH*, there is a problem. In the security proof, we need to consider the expression

$$\begin{aligned} Z(\tau) \;=\; & (X_m \oplus Y_m) \oplus \tau\psi_\tau(X_1 \oplus Y_1, \ldots, X_{m-1} \oplus Y_{m-1}, (X_{m+1}||0^{n-r}) \oplus (Y_{m+1}||0^{n-r})) \\ & \oplus (U_m \oplus V_m) \oplus \tau\psi_\tau(U_1 \oplus V_1, \ldots, U_{m-1} \oplus V_{m-1}, (U_{m+1}||0^{n-r}) \oplus (V_{m+1}||0^{n-r})) \end{aligned}$$

where the blocks indexed by $(m+1)$ are of length r bits with $1 \le r \le n-1$ and it is known that $(X_1 \oplus Y_1, \ldots, X_m \oplus Y_m, X_{m+1} \oplus Y_{m+1})$ and $(U_1 \oplus V_1, \ldots, U_m \oplus V_m, U_{m+1} \oplus V_{m+1})$ are independent and random strings. From this we need to bound the probability that a randomly chosen τ is a root of $Z(x)$. For $\psi = \mathsf{Poly}$, this is easy to do. On the other hand, for $\psi = \mathsf{BRW}$, this is little more complicated. In the proof we show how to handle this situation.

Three variants each of HEH and HEH* are defined. These variants are obtained by suitably defining the hashing keys. The definition of the hashing keys are general in nature and also work when used with other modes of encryption.

Counter and Output Feedback (OFB) Modes. The definition of ψ and the hashing keys are combined with the counter and the OFB modes to obtain several other tweakable enciphering schemes. We call these iHCH and HOH. In iHCH, we use a somewhat non-conventional definition of the counter mode. This allows us to improve upon the previous construction HCH by reducing the number of block cipher calls by one. Currently, iHCH and its variants provide the most efficient hash-counter-hash construction.

The generality of our approach is highlighted by showing that the OFB mode can be easily used to replace the counter mode. For a sequential implementation, both the OFB and the counter modes have the same efficiency. But, for a parallel implementation, one would prefer the counter mode. CBC and the cipher feedback (CFB) modes can also be used with our approach. We do not provide these details, since from the viewpoint of tweakable enciphering schemes, we do not see any advantage of using these modes.

Disk encryption. Perhaps the most important application of tweakable enciphering schemes is disk encryption. Here the disk sectors are separately encrypted and the sector addresses are taken to be the tweaks. Since sectors are of fixed length, for disk encryption one considers only fixed length messages.

We present a definition of hashing keys which is suited for fixed length messages. Using $\psi = \mathsf{BRW}$ leads to a scheme which is at least as efficient as the encrypt-mix-encrypt approach. Using $\psi = \mathsf{Poly}$ and a pre-computed table can lead to a faster implementation. If one wants to avoid pre-computation, then using $\psi = \mathsf{BRW}$ with an appropriate variant of HEH is the currently best construction for disk encryption application.

2 Background

2.1 Definitions

Let $\mathbb{F}$ be a finite field. Additions and multiplications are done over this field. We will be interested in keyed families of hash functions where the domain for the hash functions consists of tuples over $\mathbb{F}$.

Definition 1. *Fix a positive integer m and let $\mathcal{F} : \mathcal{K} \times \mathbb{F}^m \to \mathbb{F}$ be a keyed family of functions where $\mathcal{K}$ is the key space.*

Almost universal. *The family $\mathcal{F}$ is said to be ϵ-almost universal (AU) if for every $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^m$, with $\mathbf{x} \neq \mathbf{x}'$;*

$$\Pr_K[Y = Y'] \leq \epsilon,$$

where $Y = \mathcal{F}_K(\mathbf{x})$ and $Y' = \mathcal{F}_K(\mathbf{x}')$. If $\epsilon = 1/|\mathbb{F}|$, then $\mathcal{F}$ is said to be universal.

Almost XOR universal. *The family $\mathcal{F}$ is said to be ϵ-almost XOR universal (AXU) if for every $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^m$, with $\mathbf{x} \neq \mathbf{x}'$; and for every fixed $\delta \in \mathbb{F}$,*

$$\Pr_K[Y - Y' = \delta] \leq \epsilon,$$

where $Y = \mathcal{F}_K(\mathbf{x})$ and $Y' = \mathcal{F}_K(\mathbf{x}')$.

Probabilities of the type $\Pr_K[Y = Y']$ are called collision probabilities and probabilities of the type $\Pr_K[Y - Y' = \delta]$ are called differential probabilities.

In Definition 1, the range is $\mathbb{F}$. We will work with hash functions whose range also consists of tuples over $\mathbb{F}$. The next definition is a modified version of block-wise universal hash function given in [6].

Definition 2. *Fix positive integers m and l. Let $\mathcal{F} : \mathcal{K} \times \mathbb{F}^m \to \mathbb{F}^l$ be a keyed family of functions where $\mathcal{K}$ is the key space. The family $\mathcal{F}$ is said to be (ϵ_1, ϵ_2)-block-wise almost universal (BAU) if for every $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^m$, $1 \leq i, i' \leq m$ with $(\mathbf{x}, i) \neq (\mathbf{x}', i')$;*

$$\begin{aligned} \Pr_K[Y_i = Y'_{i'}] &\leq \epsilon_1 \quad \text{if } i \neq i'; \\ &\leq \epsilon_2 \quad \text{if } i = i'; \end{aligned}$$

where $(Y_1, \ldots, Y_l) = \mathcal{F}_K(\mathbf{x})$ and $(Y'_1, \ldots, Y'_l) = \mathcal{F}_K(\mathbf{x}')$.

- *If $m = l$ and $\mathcal{F}_K$ is* invertible *for each $K \in \mathcal{K}$, then $\mathcal{F}$ is said to be* invertible *(ϵ_1, ϵ_2)-block-wise almost universal.*
- *One can similarly define the notion of (invertible) block-wise almost XOR universal (BAXU) hash functions.*

2.2 Polynomial Hashing

Let $\mathsf{Poly}_\tau(X_1, \ldots, X_m) = X_1\tau^{m-1} + X_2\tau^{m-2} + \cdots + X_{m-1}\tau + X_m$.

Suppose we define $\psi_\tau(X_1, \ldots, X_m) = \mathsf{Poly}_\tau(X_1, \ldots, X_m)$. Then it is well known and easy to prove that ψ is $(m-1)/|\mathbb{F}|$-AU. Using Horner's rule, it is possible to compute ψ using $(m-1)$ multiplications over $\mathbb{F}$.

If, on the other hand, we define $\psi_\tau(X_1, \ldots, X_m) = \tau\mathsf{Poly}_\tau(X_1, \ldots, X_m)$ then ψ is $m/|\mathbb{F}|$-AXU (again this is well known and easy to show). Also, using Horner's rule, it is possible to compute ψ using m multiplications over $\mathbb{F}$.

Bernstein-Rabin-Winograd Polynomials: In a recent work, Bernstein [1] builds upon earlier work due to Rabin and Winograd [15] to define a sequence of polynomials, which we will call the BRW polynomials. These are elements of $\mathbb{F}[\tau]$ and are defined as follows.

- $\mathsf{BRW}_\tau() = 0$.
- $\mathsf{BRW}_\tau(X_1) = X_1$.
- $\mathsf{BRW}_\tau(X_1, X_2) = X_1\tau + X_2$.
- $\mathsf{BRW}_\tau(X_1, X_2, X_3) = (\tau + X_1)(\tau^2 + X_2) + X_3$.
- $\mathsf{BRW}_\tau(X_1, X_2, \ldots, X_m) = \mathsf{BRW}_\tau(X_1, \ldots, X_{t-1})(\tau^t + X_t) + \mathsf{BRW}_\tau(X_{t+1}, \ldots, X_m)$ if $t \in \{4, 8, 16, 32, \ldots\}$ and $t \leq m < 2t$.

The BRW polynomials have several interesting properties. If $m \geq 3$ and $t \leq m < 2t$, then BRW is a monic polynomial of degree $2t-1$. For a fixed non-negative integer m and a fixed $\tau \in \mathbb{F}$, BRW_τ injectively maps $\mathbb{F}^m$ into the polynomial ring $\mathbb{F}[\tau]$. Another way of viewing this is the following. For $m \geq 3$, let $t \leq m < 2t$ and

$$\mathsf{BRW}_\tau(X_1, \ldots, X_m) = W_1\tau^{2t-1} + W_2\tau^{2t-2} + \cdots + W_{2t-1}\tau + W_{2t} \tag{6}$$

where $W_1 = 1$ (since BRW is monic). Then the map $(X_1, \ldots, X_m) \mapsto (W_1, \ldots, W_{2t})$ from $\mathbb{F}^m$ to $\mathbb{F}^{2t}$ is injective.

This injective property ensures the following. Let $\psi_\tau(X_1, \ldots, X_m) = \mathsf{BRW}_\tau(X_1, \ldots, X_m)$. Then if $m = 1$, ψ is the identity map; if $m = 2$, then ψ is $1/|\mathbb{F}|$-AU; and if $m \geq 3$ with $t \leq m < 2t-1$, then ψ is $(2t-1)/|\mathbb{F}|$-AU. Since $t \leq m$, we have that ψ is $(2m-1)/|\mathbb{F}|$-AU.

Also, if $\psi_\tau(X_1, \ldots, X_m) = \tau\mathsf{BRW}_\tau(X_1, \ldots, X_m)$, then if $m = 1$, ψ is $1/|\mathbb{F}|$-AXU; if $m = 2$, ψ is $2/|\mathbb{F}|$-AXU; and if $m \geq 3$ with $t \leq m < 2t-1$, then ψ is $2t/|\mathbb{F}|$-AXU and hence also $2m/|\mathbb{F}|$-AXU.

The main advantage of the BRW polynomials is that it can be computed using less number of multiplications. It is not too difficult to see from the definition that for $m \geq 2$, $\mathsf{BRW}_\tau(X_1, \ldots, X_m)$ can be computed using $\lfloor m/2 \rfloor$ multiplications and $\lg m$ squarings (to compute the powers $\tau^2, \tau^4, \ldots$). In contrast, computing $X_1\tau^{m-1} + X_2\tau^{m-2} + \cdots + X_{m-1}\tau + X_m$ using Horner's rule requires $(m-1)$ multiplications. Thus, even though $\mathsf{BRW}_\tau(X_1, \ldots, X_m)$ has almost twice the degree of $\mathsf{Poly}_\tau(X_1, \ldots, X_m)$, it can be computed using half the number of multiplications. This along with the fact that BRW_τ is AU (with a slightly weaker bound) is a very important observation made by Bernstein [1].

2.3 Block-Wise Polynomial Evaluation [6]

In this section, we describe the constructions given in [6]. For $\tau \in \mathbb{F}$ and a positive integer m, let A_τ be the following matrix.

$$A_\tau = \begin{bmatrix} \tau & \tau^2 & & \tau^m \\ \tau & \tau^2 & & \tau^m \\ & & \ddots & \\ \tau & \tau^2 & & \tau^m \end{bmatrix}$$

Define $M_\tau = A_\tau + I$ and let $\sigma = 1 + \tau + \tau^2 + \cdots + \tau^m$. The matrix M_τ is invertible if and only if $\sigma \neq 0$ and then $M_\tau^{-1} = I - (A_\tau/\sigma)$. Let $\mathbf{x} = (X_1, \ldots, X_m)$. The map $\mathbf{x} \mapsto M_\tau\mathbf{x}^T$ is the following:

$$(X_1, \ldots, X_m) \mapsto (X_1 + R, \ldots, X_m + R) \tag{7}$$

where $R = \sum_{i=1}^m X_i\tau^i$.

Let $\beta \in \mathbb{F}$ and α be a fixed primitive element of $\mathbb{F}$. Define $\mathbf{b} = (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)$. Two functions (and their inverses) from $\mathbb{F}^m$ to $\mathbb{F}^m$ are defined in the following manner.

$$\begin{array}{lll} \mathrm{BPE}_{\tau,\beta}(\mathbf{x}) = M_\tau \mathbf{x}^T + \mathbf{b} & \text{and} & \mathrm{BPE}^{-1}_{\tau,\beta}(\mathbf{x}) = M_\tau^{-1}(\mathbf{x} - \mathbf{b})^T \\ \widetilde{\mathrm{BPE}}_{\tau,\beta}(\mathbf{x}) = M_\tau(\mathbf{x} - \mathbf{b})^T & \text{and} & \widetilde{\mathrm{BPE}}^{-1}_{\tau,\beta}(\mathbf{x}) = M_\tau^{-1}\mathbf{x}^T + \mathbf{b} \end{array} \tag{8}$$

The matrix-vector product $M_\tau \mathbf{x}$ and $M_\tau^{-1}\mathbf{x}$ can be computed as efficiently as polynomial evaluation. Using a suitable representation for $\mathbb{F}$ ensure that it is very efficient to multiply by the primitive element α. Thus, the cost of evaluating BPE is essentially the cost of polynomial evaluation. Using Horner's rule, computing BPE requires m multiplications over $\mathbb{F}$. If τ is fixed, then a pre-computed table can be used to speed up the polynomial computation [17].

For a fixed value of m and random and independent choices of τ (subject to the fact that $\sigma \neq 0$) and β from $\mathbb{F}$, it has been shown in [6], that the functions defined by (8) are block-wise universal.

Note: It has been remarked that the same proof also holds when m is allowed to vary. We note that this is incorrect. To see this consider the two distinct messages $\mathbf{x}_1 = (0, 0)$ and $\mathbf{x}_2 = (0, 0, 0)$. Then $\mathrm{BPE}_{\tau,\beta}(\mathbf{x}_1) = (\beta, \alpha\beta)$ and $\mathrm{BPE}_{\tau,\beta}(\mathbf{x}_2) = (\beta, \alpha\beta, \alpha^2\beta)$. The first two components of $\mathrm{BPE}_{\tau,\beta}(\mathbf{x}_1)$ and $\mathrm{BPE}_{\tau,\beta}(\mathbf{x}_2)$ are equal which violates the block-wise universality condition.

Drawbacks: The key τ has to be chosen such that $\sigma = \sum_{i=0}^{m} \tau^i$ is non-zero. This means that τ cannot be an arbitrary element of $\mathbb{F}$. The probability that $\sigma = 0$ for a randomly chosen τ is small, so this may not be a major problem in practice. It has been suggested in [6], that one can choose τ to be a random primitive element of $GF(2^n)$. This approach has practical difficulties. Often, the entity providing the new value of the key will not have access to the internal implementation of the algorithm. Without such access, in particular, without knowing the primitive polynomial realizing the field $GF(2^n)$, it is not possible to determine whether a particular element is a primitive element of the concrete realization of the field. Further, determination of primitive element requires substantial computation and the knowledge of the prime factors of $2^n - 1$. Thus, the idea of using τ to be a random primitive element of $GF(2^n)$ is quite impractical in practice.

More importantly, if the hash function needs to be evaluated for different values of m, then computing the inverse of BPE and $\widetilde{\mathrm{BPE}}$ will require the computation of σ and τ/σ. This requires $(m - 1)$ multiplications and one inverse over $\mathbb{F}$.

3 New Constructions

Fix a positive integer m and let α be a primitive element of $\mathbb{F}$. Let $\psi : \mathbb{F}^m \to \mathbb{F}$ be an ϵ-AU hash function defined using either Poly_τ or BRW_τ, i.e., either $\psi_\tau(X_1, \ldots, X_{m-1}) = \mathsf{Poly}_\tau(X_1, \ldots, X_{m-1})$ or $\psi_\tau(X_1, \ldots, X_{m-1}) = \mathsf{BRW}_\tau(X_1, \ldots, X_{m-1})$.

3.1 Block-Wise Almost Universal

We give a simple construction of an invertible BAU function Ψ. The key space of Ψ is $\mathbb{F} \times \mathbb{F}$ and the domain and range are $\mathbb{F}^m$. Given $(\tau, \beta) \in \mathbb{F} \times \mathbb{F}$, we define

$$\Psi_{\tau,\beta}(X_1, \ldots, X_m) = (X_1 + Y, \ldots, X_{m-1} + Y, Y) + (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta) \tag{9}$$

where $Y = X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1})$. This means that

– if $\psi = \mathsf{Poly}$, then $Y = X_m + \tau\mathsf{Poly}_\tau(X_1, \ldots, X_{m-1})$ and
– if $\psi = \mathsf{BRW}$, then $Y = X_m + \tau\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1})$.

Inverting Ψ is easy. Let $(Y_1, \ldots, Y_m) = \Psi_{\tau,\beta}(X_1, \ldots, X_m)$. The tuple $(X_1, \ldots, X_m)$ is obtained from $(Y_1, \ldots, Y_m)$ using the following steps.

1. Set $(U_1, \ldots, U_m) = (Y_1, \ldots, Y_m) - (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta)$.
2. Set $(X_1, \ldots, X_{m-1}) = (U_1 - U_m, \ldots, U_{m-1} - U_m)$.
3. Set $X_m = U_m - \psi_\tau(X_1, \ldots, X_{m-1})$.

Formally, we write $\Psi[\psi]$ to denote the BAU family obtained using the AU family ψ.

Examples: For $m = 4$, we provide the outputs of $\mathrm{BPE}_{\tau,\beta}$, $\Psi[\mathsf{Poly}]_{\tau,\beta}$ and $\Psi[\mathsf{BRW}]_{\tau,\beta}$ to illustrate the difference between these functions.

$$\begin{aligned}
\mathrm{BPE}_{\tau,\beta}(X_1, X_2, X_3, X_4) = (&X_1 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \beta,\\
&X_2 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha\beta,\\
&X_3 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha^2\beta,\\
&X_4 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha^3\beta)\\
\Psi[\mathsf{Poly}]_{\tau,\beta}(X_1, X_2, X_3, X_4) = (&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_1 + \alpha\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_2 + \alpha^2\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_3 + \alpha^3\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + \beta)\\
\Psi[\mathsf{BRW}]_{\tau,\beta}(X_1, X_2, X_3, X_4) = (&\tau((\tau + X_1)(\tau^2 + X_2) + X_3) + X_4 + X_1 + \alpha\beta,\\
&\tau((\tau + X_1)(\tau^2 + X_2) + X_3) + X_4 + X_2 + \alpha^2\beta,\\
&\tau((\tau + X_1)(\tau^2 + X_2) + X_3) + X_4 + X_3 + \alpha^3\beta,\\
&\tau((\tau + X_1)(\tau^2 + X_2) + X_3) + X_4 + \beta)
\end{aligned}$$

The order of evaluation in BPE and $\Psi[\mathsf{Poly}]$ are in reverse order. This difference is, however, not significant. One can define BPE to evaluate in the reverse order (i.e., $X_1\tau^4 + X_2\tau^3 + X_3\tau^2 + X_4\tau$) as has indeed been done in [6] while defining TET. The significant differences between the two maps are in the degrees of the polynomials (in τ) and the treatment of the last component. The structure of the map $\Psi[\mathsf{BRW}]$ is quite different from the other two.

Theorem 1. *Fix a positive integer m. Then $\Psi[\psi]$ is $(1/|\mathbb{F}|, \epsilon)$-BAU, where $\epsilon = (m-1)/|\mathbb{F}|$ if $\psi = \mathsf{Poly}$ and $\epsilon = (2m-1)/|\mathbb{F}|$ if $\psi = \mathsf{BRW}$.*

Proof : Let $\mathbf{x} = (X_1, \ldots, X_m)$, $\mathbf{x}' = (X'_1, \ldots, X'_m)$ and $1 \le i, i' \le m$ with $(\mathbf{x}, i) \neq (\mathbf{x}', i')$. Further, let $(Y_1, \ldots, Y_m) = \Psi_{\tau,\beta}(X_1, \ldots, X_m)$ and $(Y'_1, \ldots, Y'_m) = \Psi_{\tau,\beta}(X'_1, \ldots, X'_m)$. Then

$$\begin{aligned}
Y_i &= \alpha^i\beta + X_i + X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1}) && \text{if } 1 \le i < m;\\
&= \beta + X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1}) && \text{if } i = m.
\end{aligned}$$

Similar equations hold for the primed variables.

First suppose $i \neq i'$. Without loss of generality assume $1 \le i < i' \le m$. Then

$$\begin{aligned}
Y_i - Y'_{i'} &= \alpha^i(1 - \alpha^{i'-i})\beta + \tau\psi_\tau(X_1, \ldots, X_m) - \tau\psi_\tau(X'_1, \ldots, X'_m) && \text{if } i' < m;\\
&= (\alpha^i - 1)\beta + \tau\psi_\tau(X_1, \ldots, X_m) - \tau\psi_\tau(X'_1, \ldots, X'_m) && \text{if } i' = m.
\end{aligned}$$

In both cases, the coefficient of β is non-zero. Since β is independent of τ, it follows that $\Pr[Y_i = Y'_{i'}] = 1/|\mathbb{F}|$.

Now consider $i = i'$. Then $\mathbf{x} \neq \mathbf{x}'$. The value of i (and hence i') may be equal to m or it may be between 1 and $(m-1)$. If $i = m$, then the condition $Y_m = Y'_{m'}$ is equivalent to

$$X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1}) = X'_m + \tau\psi_\tau(X'_1, \ldots, X'_m). \tag{10}$$

On the other hand, if $1 \leq i \leq m-1$, then the condition $Y_i = Y'_{i'}$ is equivalent to

$$\psi_\tau(X_1, \ldots, X_{m-1}) + X_m + X_i = \psi_\tau(X'_1, \ldots, X'_{m-1}) + X'_m + X'_i \tag{11}$$

For both these situations, the cases $\psi = \mathsf{Poly}$ and $\psi = \mathsf{BRW}$ are treated separately.

Case $\psi = \mathsf{Poly}$: First, let $i = m$. Set $(W_1, \ldots, W_m) = (X_1 - X'_1, \ldots, X_m - X'_m)$. Then (10) is equivalent to

$$W_1\tau^{m-1} + \cdots + W_{m-1}\tau + W_m = 0. \tag{12}$$

Since $(X_1, \ldots, X_m) \neq (X'_1, \ldots, X'_m)$, $W_1\tau^{m-1} + \cdots + W_{m-1}\tau + W_m$ is a non-zero polynomial of degree at most $(m-1)$ and (12) holds if and only if τ is a root of this polynomial. Since τ is a random element of $\mathbb{F}$, the probability of the last event is at most $(m-1)/\mathbb{F}$.

Now, consider $1 \leq i \leq m-1$. Let f be the map $f(X_1, \ldots, X_m) = (U_1, \ldots, U_m)$ where $U_j = X_j$, $1 \leq j \leq m-1$ and $U_m = X_i + X_m$. It is easy to verify that f is an injective map.

Now, $X_i + X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1}) = \mathsf{Poly}_\tau(f(X_1, \ldots, X_m)) = \mathsf{Poly}_\tau(U_1, \ldots, U_m)$ and similarly for the primed variables. The injectivity of f implies that if $(X_1, \ldots, X_m) \neq (X'_1, \ldots, X'_m)$, then $(U_1, \ldots, U_m) \neq (U'_1, \ldots, U'_1)$. Let $V_i = U_i - U'_i$ for $1 \leq i \leq m$ and so, $(V_1, \ldots, V_m) \neq (0, \ldots, 0)$. Then, the probability that (11) holds is equal to the probability that $\mathsf{Poly}_\tau(V_1, \ldots, V_m) = V_1\tau^{m-1} + \cdots + V_{m-1}\tau + V_m$ is equal to zero. Since $\mathsf{Poly}_\tau(V_1, \ldots, V_m)$ is a non-zero polynomial of degree at most $m-1$, it has at most $m-1$ roots over $\mathbb{F}$. The probability that the randomly chosen τ is one of these roots is therefore at most $(m-1)/|\mathbb{F}|$.

Case $\psi = \mathsf{BRW}$: If $m \leq 2$, then this reduces to the case $\psi = \mathsf{Poly}$. So we consider only $m \geq 3$. Let $t \in \{4, 8, 16, 32, \ldots\}$ be such that $t \leq m-1 < 2t$. Then $\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1})$ is a monic polynomial of degree $2t-1$. Let

$$\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1}) = W_1\tau^{2t-1} + W_2\tau^{2t-2} + \cdots + W_{2t-1}\tau + W_{2t}.$$

Then $\mathsf{coe}(\mathsf{BRW}_\tau(X_1, \ldots, X_m)) = (W_1, \ldots, W_{2t})$ with $W_1 = 1$. By the properties of BRW polynomials, the map $(X_1, \ldots, X_{m-1}, X_m) \mapsto (W_1, \ldots, W_{2t}, X_m)$ is injective. Also

$$\begin{aligned} Y &= X_m + \tau\mathsf{BRW}_\tau(X_1, \ldots, X_{m-1}) \\ &= W_1\tau^{2t} + W_2\tau^{2t-1} + \cdots + W_{2t-1}\tau^2 + W_{2t}\tau + X_m \\ &= \mathsf{Poly}_\tau(W_1, \ldots, W_{2t-1}, X_m). \end{aligned}$$

Thus, Y is a polynomial of degree $2t$. Since $t \leq m-1 < 2t$, we have $t < t+1 \leq m$ and so $2t \leq 2m-1$. The rest of the argument is similar to that of the previous case. □

Variable m: Theorem 1 holds for a fixed value of m. If m is allowed to vary, then the result does not hold. This can be seen as in the case of BPE by considering the two distinct messages $(0, 0, 0)$ and $(0, 0)$.

$\Psi^{-1}_{\tau,\beta}$ **is not block-wise universal.** We show this only for $\Psi[\mathsf{Poly}]$. This is seen by considering

1. $\Psi^{-1}_{\tau,\beta}(\alpha\beta, \alpha^2\beta, \alpha^3\beta, \beta) = (0, 0, 0, 0)$ and
2. $\Psi^{-1}_{\tau,\beta}(A\tau^3 + A + \alpha\beta, A\tau^3 + \alpha^2\beta, A\tau^3 + \alpha^3\beta, A\tau^3 + \beta) = (A, 0, 0, 0)$ for a non-zero A.

The last three components are equal, violating the block-wise universal property. Thus, we have an example of a function which is invertible and block-wise universal but its inverse is not block-wise universal.

3.2 Block-Wise Almost XOR Universal

The constructions in [6] are proved to be BAXU which the new construction is not. On the other hand, it is easy to modify the new construction to yield a BAXU family.

As before, fix a positive integer m and a primitive element α of $\mathbb{F}$. Let τ and β be independent and random elements of $\mathbb{F}$. We define the map $\Psi^{\mathrm{XOR}}_{\tau,\beta} : \mathbb{F}^m \to \mathbb{F}^m$ in the following manner.

$$\Psi^{\mathrm{XOR}}_{\tau,\beta}(X_1, \ldots, X_m) = (X_1 + Y, \ldots, X_{m-1} + Y, Y) + (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta) \tag{13}$$

where $Y = \tau(X_m + \tau\psi_\tau(X_1, \ldots, X_{m-1}))$.

The only difference from the definition of Ψ is in the definition of Y, which is now a polynomial of one higher degree and whose constant term is 0. The proof that Ψ^{XOR} is BAXU is similar to the proof of Theorem 1. Also, it is not difficult to see that Ψ^{XOR} is invertible if $\tau \neq 0$. Computing the inverse of Ψ^{XOR} requires the inverse of τ. As a BAXU family, Ψ^{XOR} improves upon the constructions in [6] in the following way. Ψ^{XOR} requires that $\tau \neq 0$ and the inverse of τ, while the constructions in [6] require $\sigma = 1 + \tau + \cdots + \tau^m \neq 0$ and the inverse of σ. This difference, though small, can be significant in practice. We note that the inverse of Ψ^{XOR} is not BAU, while the inverses of the constructions in [6] are BAXU. In this paper, we do not work with Ψ^{XOR}; instead we work only with Ψ, which does not require the inverse of τ for computing ψ^{-1}.

3.3 Discussion

The salient features of the new constructions as well as BPE are shown in Table 1. The other variant $\widetilde{\mathrm{BPE}}$ of BPE defined in [6] has the same properties as that of BPE.

First, note that both variants of Ψ are BAU, while BPE and both variants of Ψ^{XOR} are BAXU. Second, the inverse of BPE is BAXU, while the inverses of the other constructions are not BAU. Importantly, however, for the construction of tweakable enciphering schemes, it is sufficient to use a BAU family without requiring the inverse to be BAU. Thus, for the purpose of constructing tweakable enciphering schemes, BPE is an overkill in two ways – it is BAXU and its inverse is also BAXU. However, there may be other applications where the full power of BPE may be desirable, though at this point, we do not know of any such applications.

Computing BPE or $\Psi[\mathsf{Poly}]$ roughly m multiplications, whereas computing $\Psi[\mathsf{BRW}]$ and its inverse requires roughly $m/2$ multiplications. There are two trade-offs. First, the bound for AU property is slightly weaker for $\Psi[\mathsf{BRW}]$. Second, the multiplications for computing BPE or $\Psi[\mathsf{Poly}]$ have one operand fixed, so that one can use a pre-computed table [17] to speed up each multiplication. This is not possible for $\Psi[\mathsf{BRW}]$ (of course, one can store the power $\tau, \tau^2, \tau^4, \ldots$; but this is something different). The issue of pre-computed table is contentious and the actual performance in a multiple key situation is not clear. Nevertheless, we think it worthwhile to point out the difference.

Computing the inverse of BPE requires $2(m-1)$ multiplications. This is because $\sigma = 1 + \tau + \cdots + \tau^m$ also needs to be computed requiring an extra $(m-1)$ multiplications. If τ is fixed, then σ can be pre-computed. However, since τ is a key, σ will require to be pre-computed for every key change.

There is a restriction on the hashing key τ for BPE. This directly impacts the performance of the resulting tweakable enciphering scheme TET. On the other hand, Ψ does not have any restriction on τ. There is a restriction on τ for Ψ^{XOR} which is weaker than the restriction on τ for BPE. But, since it is enough to use Ψ to construct a tweakable enciphering scheme, we do not consider Ψ^{XOR} any further in this paper.

Table 1. Comparison among different hash families. There are m blocks and the hashing keys in all cases are (τ, β). Here [M] and [I] denotes a multiplication and an inversion over $\mathbb{F}$. For BPE, $\sigma = 1 + \tau + \cdots + \tau^m$ and precomputing σ^{-1} brings down the cost of inversion to m[M]. Similarly, for Ψ^{XOR}, precomputing τ^{-1}, eliminates the requirement of performing an online inversion.

	forward		inverse			
Family	type	efficiency	type	efficiency	restriction	pre-comp tab?
BPE	$(1/\lvert\mathbb{F}\rvert, m/\lvert\mathbb{F}\rvert)$-BAXU	m[M]	BAXU	$2(m-1)$[M]+1[I]	$\sigma \neq 0$	yes
Ψ[Poly]	$(1/\lvert\mathbb{F}\rvert, (m-1)/\lvert\mathbb{F}\rvert)$-BAU	$(m-1)$[M]	not BAU	$(m-1)$[M]	none	yes
Ψ[BRW]	$(1/\lvert\mathbb{F}\rvert, (2m-1)/\lvert\mathbb{F}\rvert)$-BAU	$\lceil m/2 \rceil$[M]	not BAU	$\lceil m/2 \rceil$[M]	none	no
Ψ^{XOR}[Poly]	$(1/\lvert\mathbb{F}\rvert, m/\lvert\mathbb{F}\rvert)$-BAXU	m[M]	not BAU	m[M]+1[I]	$\tau \neq 0$	yes
Ψ^{XOR}[BRW]	$(1/\lvert\mathbb{F}\rvert, (2m)/\lvert\mathbb{F}\rvert)$-BAXU	$1 + \lceil m/2 \rceil$[M]	not BAU	$1 + \lceil m/2 \rceil$[M]+1[I]	$\tau \neq 0$	no

4 The HEH Construction

For the description of the tweakable SPRP, we will take the finite field $\mathbb{F}$ to be $GF(2^n)$ and use the operator $\oplus$ to denote addition over this field. The field $GF(2^n)$ is realized using a primitive polynomial $\rho(x)$ of degree n. Usually, it is possible to choose the polynomial $\rho(x)$ to be a trinomial or a pentanomial. As is standard, the elements of $GF(2^n)$ can be interchangeably considered to be either as polynomials over $GF(2)$ of degree at most $n-1$ or as n-bit strings. For $0 \leq i \leq 2^n - 1$, by $\mathsf{bin}_n(i)$ we denote the n-bit binary representation of i.

Choice of α: a standard and efficient choice of the primitive element α is to take it to be equal to x. (It has been mentioned in [6] that depending on the endianness, it may be appropriate to choose α to be equal to $1/x$.) Multiplication by x modulo $\rho(x)$ is very efficient.

Choice of block cipher: A block cipher $E_K : \{0,1\}^n \to \{0,1\}^n$ will be used. We do not make any assumption on the length of the key K. Tweaks are assumed to be n bits long.

Choice of Ψ: The description is in terms of a BAU family Ψ which can be instantiated as either Ψ[Poly] or as Ψ[BRW].

The basic structure of the HEH construction is shown in Table 2. In this construction, there are one block cipher key K and three hashing keys τ, β_1 and β_2. By suitably defining the hashing keys, it is possible to obtain several variants of the basic construction. This is shown in Table 3. If KeyDef1 is used we call the construction HEH; if KeyDef2 is used we call the construction HEHp; and if KeyDef3 is used we call the construction HEHfp.

In Table 2, the BAU family Ψ is used. This, in turn, is built using the AU family ψ. Formally, the family $\Psi[\psi]$ is used, where $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$. To reflect this dependence, formally we will write HEH[ψ] and instantiating ψ gives rise to either HEH[Poly] or HEH[BRW]. A similar notation is used for HEHp and HEHfp.

- **HEH:** The message space consists of $\{0,1\}^{mn}$ with $m \geq 1$ and so the length ℓ of a particular message is $\ell = mn$ for some positive integer m. The construction is length preserving, i.e., the length of the ciphertext is equal to the length of the plaintext. A single block cipher key is used. The hashing key τ depends on the tweak T. Hence, it is not possible to speed up multiplication by τ using a pre-computed table.
- **HEHp:** If pre-computation is desired, then it is easy to modify HEH, to obtain a variant supporting pre-computation. Instead of setting $\tau = \gamma$, simply choose τ to be a random element of $GF(2^n)$. We call this variant HEHp. The message space remains the same as that of HEH.
- **HEHfp:** An important special application of tweakable SPRP is that of disk encryption. In this application, the number of blocks m is fixed and the tweak is the sector address. Since m is fixed, it is possible to eliminate one block cipher call while deriving the hashing keys. Also, the hashing key τ is chosen to be a random element of $GF(2^n)$ so that pre-computation can be utilized. We call this variant HEHfp. In this variant, the hashing key is τ and the block cipher key is K. The message space is $\{0,1\}^{mn}$ for some fixed positive integer m.

In effect, Tables 2 and 3 define three algorithms HEH[ψ], HEHp[ψ] and HEHfp[ψ]. We will refer to the encryption and decryption algorithms of HEH[ψ] as HEH[ψ].Encrypt and HEH[ψ].Decrypt respectively and similarly for the other two variants. In Table 3, we have used ℓ to derive β_1. Since $m = \ell/n$, one can also use m in place of ℓ. The use of ℓ is for compatibility with the more general version given later where n does not necessarily divide ℓ.

Pre-computation and Ψ[BRW]: HEH and HEHp make the same number of block cipher calls. The difference is that in the latter case one uses a separate random τ as the hashing key which can be used to prepare a pre-computed table to possible speed up Horner's rule computation. Ψ[BRW], however, cannot profit from this. Hence, for Ψ[BRW], it is not meaningful to use HEHp[BRW] and it is better to stick to the single key HEH[BRW]. For the case of HEHfp[BRW], however, the number of block cipher calls is reduced by one. Thus, if the number of blocks is fixed, then one should use HEHfp[BRW] with Ψ[BRW], even though the resulting algorithm cannot profit from pre-computation.

Table 2. Encryption and decryption using HEH. The block cipher key is K; and the hash key is (τ, β_1, β_2). See Table 3 for details of how the hashing keys are derived in HEH and its variants. $\mathrm{ECB}_K(X_1, \ldots, X_m)$ returns $(E_K(X_1), \ldots, E_K(X_m))$ and $\mathrm{ECB}_K^{-1}(Y_1, \ldots, Y_m)$ returns $(E_K^{-1}(Y_1), \ldots, E_K^{-1}(Y_m))$.

Algorithm $\mathsf{Encrypt}_{K,\tau,\beta_1,\beta_2}(P_1, \ldots, P_m)$ 1. $(PP_1, \ldots, PP_m) = \Psi_{\tau,\beta_1}(P_1, \ldots, P_m)$; 2. $(CC_1, \ldots, CC_m) = \mathrm{ECB}_K(PP_1, \ldots, PP_m)$; 3. $(C_1, \ldots, C_m) = \Psi^{-1}_{\tau,\beta_2}(CC_1, \ldots, CC_m)$.	**Algorithm** $\mathsf{Decrypt}_{K,\tau,\beta_1,\beta_2}(C_1, \ldots, C_m)$ 1. $(CC_1, \ldots, CC_m) = \Psi_{\tau,\beta_2}(C_1, \ldots, C_m)$; 2. $(PP_1, \ldots, PP_m) = \mathrm{ECB}_K^{-1}(CC_1, \ldots, CC_m)$; 3. $(P_1, \ldots, P_m) = \Psi^{-1}_{\tau,\beta_1}(PP_1, \ldots, PP_m)$.

Table 3. Different definitions of the hashing keys τ, β_1 and β_2. Here T is an n-bit tweak and ℓ is the length of the message (or ciphertext) in bits.

KeyDef1	KeyDef2	KeyDef3
1. $\gamma = E_K(T)$; 2. $\beta_1 = E_K(\gamma \oplus \mathsf{bin}_n(\ell))$; 3. $\beta_2 = \alpha\beta_1$; 4. $\tau = \gamma$.	1. $\gamma = E_K(T)$; 2. $\beta_1 = E_K(\gamma \oplus \mathsf{bin}_n(\ell))$; 3. $\beta_2 = \alpha\beta_1$; 4. choose τ randomly from $GF(2^n)$.	1. $\beta_1 = E_K(T)$; 2. $\beta_2 = \alpha\beta_1$; 3. choose τ randomly from $GF(2^n)$.

5 An Extension of HEH

HEH and its variants can handle messages whose lengths are multiples of the block length n. In this section, we define an extension HEH* of HEH and its variants. HEH* can handle partial blocks. For most applications, n-bit tweaks and messages without partial blocks are probably sufficient. But, this may not be true for *all* possible applications. For this reason we describe the more general construction.

Arbitrary length tweaks. The length of tweaks in HEH* is n bits, which means that HEH* cannot handle arbitrary length tweaks. An easy technique for handling arbitrary length tweaks is the following. Use a pseudorandom function (PRF) with an independent key to map the arbitrary length tweak into an n-bit tweak; and then use HEH* (or HEH or one of the other variants). The security of this construction is generic. This strategy has been suggested for HCH and a slightly modified variant of this strategy has been used in TET to handle arbitrary length tweaks.

5.1 Variants of Ψ

To handle partial blocks, we need to change the defintion of Ψ. Two modifications are given. The first one is more appealing from a theoretical point of view, while the second one is more practical.

The general definition of Ψ is over an arbitrary finite field. But, in the construction of HEH, we work over $GF(2^n)$. Actually, for the mode of operation, we only need $GF(2^n)$. So, the variants of Ψ are also defined only over $GF(2^n)$.

The Construction Γ. Fix a positive integer m and an integer r with $1 \le r \le n-1$. We consider m full blocks and one r-bit partial block. For the definition of Γ, the first block is taken to be the partial block. Fix a primitive element α of $GF(2^n)$. The hash key consists of two random elements β and τ of $GF(2^n)$.

Define $\mathbf{g} = (0^r, \alpha\beta, \ldots, \alpha^{m-1}\beta, \beta)$, where $\alpha^i\beta$ is considered to be an n-bit string and 0^r denotes a string of r zeros. Let $(W, X_1, \ldots, X_m)$ be the message, where $|W| = r$ and $|X_1| = \cdots = |X_m| = n$. We define

$$\Gamma_{\tau,\beta}(W, X_1, \ldots, X_m) = (W, X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus \mathbf{g} \tag{14}$$

where

$$Y = X_m \oplus \tau\psi_\tau(W||0^{n-r}, X_1, \ldots, X_{m-1}). \tag{15}$$

It is not difficult to see that for a fixed τ and β, the map Γ is invertible. Basically, the first component remains unchanged in the map. Obtaining the last m components of the input from last m components of the output and W is almost the same as the technique used for inverting Ψ.

As in the case of Ψ, there are two variants of Γ depending on whether $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$. We will write these as $\Gamma[\mathsf{Poly}]$ and $\Gamma[\mathsf{BRW}]$.

The first block is not a full block, so we cannot really talk about blockwise universality for the whole of Γ. On the other hand, the other components of Γ do satisfy the blockwise universality property as stated in the following result. The proof of the result is very similar to that of Theorem 1 and is omitted. The only thing to note is that ψ in this case has m arguments as against $(m-1)$ for the BAU construction. Consequently, the value of ϵ in the result below is slightly more than that in Theorem 1.

Theorem 2. *Fix* $\mathbf{x} = (W, X_1, \ldots, X_m)$ *and* $\mathbf{x}' = (W', X'_1, \ldots, X'_m)$ *and* $1 \le i, j \le m$ *such that* $(\mathbf{x}, i) \neq (\mathbf{x}', i')$. *Then*

$$\begin{array}{rl} \Pr_{\tau,\beta}[Y_i = Y'_{i'}] = \frac{1}{2^n} & \text{if } i \neq i'; \\ \le \epsilon & \text{if } i = i'. \end{array}$$

Here

$$\begin{aligned} (W, Y_1, \ldots, Y_m) &= \Gamma[\psi]_{\tau,\beta}(W, X_1, \ldots, X_m), \\ (W', Y'_1, \ldots, Y'_m) &= \Gamma[\psi]_{\tau,\beta}(W', X'_1, \ldots, X'_m). \end{aligned}$$

Further, $\epsilon = m/2^n$ *if* $\psi = \mathsf{Poly}$*; and* $\epsilon = 2m/2^n$ *if* $\psi = \mathsf{BRW}$.

Note. It is easy to see that if r is taken to be zero, i.e., there is no partial block, then the definition of Γ is exactly the same as that of Ψ. So, it would be natural to work with Γ for tackling partial blocks. However, the problem is that the partial block is the first block. This may not be convenient for many applications. For this reason, we provide another variant of Ψ, where the partial block is the last block.

The Construction Φ. As in the definition of Γ, fix a positive integer m and an integer r with $1 \le r \le n-1$. Consider $(X_1, \ldots, X_m, X_{m+1})$, where $X_1, \ldots, X_m \in \{0,1\}^n$ and $X_{m+1} \in \{0,1\}^r$. In other words, the first m blocks are full blocks and the last block is a partial block.

Given an element $\beta \in GF(2^n)$, define $\mathbf{f} = (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta, 0^r)$ where, as before $\alpha^i\beta$ is considered to be an n-bit string and 0^r denotes the string of r zeros. The hashing key consists of two random elements β and τ in $GF(2^n)$. We define

$$\Phi_{\tau,\beta}(X_1, \ldots, X_m, X_{m+1}) = (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y, X_{m+1}) \oplus \mathbf{f} \tag{16}$$

where

$$Y = X_m \oplus \tau\psi_\tau(X_1, \ldots, X_{m-1}, X_{m+1} || 0^{n-r}). \tag{17}$$

Note that in the definitions of Y for Ψ, Γ and Φ, the last full block forms the constant term of the polynomial in τ. As before, we will formally denote the above family by $\Phi[\psi]$. Similar to the result for Γ, we have the following result for Φ.

Theorem 3. *Fix* $\mathbf{x} = (X_1, \ldots, X_m, X_{m+1})$ *and* $\mathbf{x}' = (X'_1, \ldots, X'_m, X'_{m+1})$ *and* $1 \le i, j \le m$ *such that* $(\mathbf{x}, i) \neq (\mathbf{x}', i')$. *Then*

$$\begin{array}{rl} \Pr_{\tau,\beta}[Y_i = Y'_{i'}] = \frac{1}{2^n} & \text{if } i \neq i'; \\ \le \epsilon & \text{if } i = i'. \end{array}$$

Here

$$(Y_1, \ldots, Y_m, X_{m+1}) = \Phi[\psi]_{\tau,\beta}(X_1, \ldots, X_m, X_{m+1}),$$
$$(Y'_1, \ldots, Y'_m, Y'_{m+1}) = \Phi[\psi]_{\tau,\beta}(X'_1, \ldots, X'_m, X'_{m+1}).$$

Further, $\epsilon = m/2^n$ *if* $\psi = \mathsf{Poly}$ *and* $\epsilon = 2m/2^n$ *if* $\psi = \mathsf{BRW}$.

Discussion The construction TET given in [6] handles partial blocks. This also requires a polynomial evaluation involving the partial block. A partial block in TET is padded with 1 followed by required number of zeros. (In contrast, we pad only with zeros; in fact, we do not see any reason to use the technique of "one followed by zeros" padding in the current context.)

Suppose the message is $(P_1, \ldots, P_m, P_{m+1})$, where P_{m+1} is of length r bits. For $1 \leq i \leq m$, the main hashing step for the ith step consists of the following evaluation.

$$P_i \oplus \sigma^{-1}(P_1\tau^m \oplus P_2\tau^{m-1} \oplus \cdots \oplus P_{m-1}\tau^2 \oplus P_m\tau) \oplus (P_{m+1}||10^{n-r-1})$$

where $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$.

This is to be compared to our definition of Y for Γ and Φ. One important aspect is that in the above definition, the partial block forms the constant term of the polynomial evaluation. In comparison, for our constructions, the last full block is always taken to be the constant term of the similar polynomial.

5.2 Handling Arbitrary Length Messages

Here we define an extension with the capability of handling arbitrary length messages. This extension is called HEH* and is shown in Table 4. As in the case of HEH, the variants HEHp* and HEHfp* are obtained by suitably defining the keys as in Table 3. Note that the key definitions for the starred and the unstarred variants are the same.

As before, we will use the notation HEH*[Poly] or HEH*[BRW] to denote whether Poly or BRW is used for instantiating ψ.

The description in Table 4 uses Φ. This has been done since we expect most applications to have partial blocks at the end. On the other hand, it is also possible to handle partial blocks using Γ. This construction is shown in Table 5. The definition of the keys for the construction in Table 5 is also the same as that given in Table 3. We do not provide a separate security proof for the construction in Table 5 since such a proof is essentially the same as the proof for the construction in Table 4.

The length of the message is $\ell = mn + r$, where $m \geq 1$ and $0 \leq r \leq n-1$, i.e., there are m full blocks $P_1, \ldots, P_m$ and a possible partial block P_{m+1}. If $r = 0$, then there is no partial block and we essentially revert to the encryption and decryption algorithms given in Table 2. While defining the keys β_1 and β_2 in Table 3, we had remarked that one can use m instead of ℓ. But, in the present case, using ℓ is required since n may not divide ℓ.

HEH* handles arbitrary length strings and uses a single n-bit key, which is the block cipher key. It, however, cannot utilize pre-computation. HEHp*, on the other hand, can utilize pre-computation. For this, we require an extra n-bit hashing key in addition to the block cipher key. HEHp* can also handle arbitrary length strings. The variant HEHfp* handles fixed length strings and can utilize pre-computation. Since the length of the input is fixed, it might appear that HEHfp* and HEHfp are identical. This is not so, because for HEHfp, the input must be a multiple of the block length,

Table 4. Encryption and decryption using HEH*. The block cipher key is K; and the hash key is (τ, β_1, β_2). Definitions of hashing keys are given in Table 3. If KeyDef1 is used we call the construction HEH*; if KeyDef2 is used we call the construction HEHp*; and if KeyDef3 is used we call the construction HEHfp*.

Algorithm $\mathsf{Encrypt}_{K,\tau,\beta_1,\beta_2}(P_1, \ldots, P_m, P_{m+1})$

Case $(|P_{m+1}| = 0)$:

1. $(PP_1, \ldots, PP_m) = \Psi_{\tau,\beta_1}(P_1, \ldots, P_m)$;
2. $(CC_1, \ldots, CC_m) = \mathrm{ECB}_K(PP_1, \ldots, PP_m)$;
3. $(C_1, \ldots, C_m) = \Psi^{-1}_{\tau,\beta_2}(CC_1, \ldots, CC_m)$.

Case $(1 \le |P_{m+1}| \le n-1)$:

1. $(PP_1, \ldots, PP_m, P_{m+1}) = \Phi_{\tau,\beta_1}(P_1, \ldots, P_m, P_{m+1})$;
2. $(CC_1, \ldots, CC_m) = \mathrm{ECB}_K(PP_1, \ldots, PP_m)$;
3. $Z = E_K(PP_m \oplus CC_m)$;
4. $C_{m+1} = P_{m+1} \oplus \mathsf{First}_r(Z)$;
5. $(C_1, \ldots, C_m, C_{m+1}) = \Phi^{-1}_{\tau,\beta_2}(CC_1, \ldots, CC_m, C_{m+1})$.

Algorithm $\mathsf{Decrypt}_{K,\tau,\beta_1,\beta_2}(C_1, \ldots, C_m, C_{m+1})$

Case $(|C_{m+1}| = 0)$:

1. $(CC_1, \ldots, CC_m) = \Psi_{\tau,\beta_2}(C_1, \ldots, C_m)$;
2. $(PP_1, \ldots, PP_m) = \mathrm{ECB}^{-1}_K(CC_1, \ldots, CC_m)$;
3. $(P_1, \ldots, P_m) = \Psi^{-1}_{\tau,\beta_1}(PP_1, \ldots, PP_m)$.

Case $(1 \le |C_{m+1}| \le n-1)$:

1. $(CC_1, \ldots, CC_m, C_{m+1}) = \Phi_{\tau,\beta_1}(C_1, \ldots, C_m, C_{m+1})$;
2. $(PP_1, \ldots, PP_m) = \mathrm{ECB}^{-1}_K(CC_1, \ldots, CC_m)$;
3. $Z = E_K(PP_m \oplus CC_m)$;
4. $P_{m+1} = C_{m+1} \oplus \mathsf{First}_r(Z)$;
5. $(P_1, \ldots, P_m, P_{m+1}) = \Phi^{-1}_{\tau,\beta_2}(PP_1, \ldots, PP_m, P_{m+1})$.

whereas for HEHfp* the input length should only be fixed; there is no need for it to be a multiple of the block length.

The notation $\mathsf{First}_r(Z)$ denotes the first r bits of the n-bit binary string Z. The partial block PP_{m+1} is encrypted in a stream cipher like manner, i.e., by XORing with a pseudorandom string of appropriate length. The pseudorandom string is obtained by encrypting the XOR of PP_m and CC_m. This technique is adopted from HCTR and has also been used in HCH. Note that this is not present in the other hash-counter-hash type construction XCB.

Table 5. Handling partial blocks using Γ. The block cipher key is K; and the hash key is (τ, β_1, β_2). See Table 3 for details of how the hashing keys are derived in HEH* and its variants.

Algorithm $\mathsf{Encrypt}_{K,\tau,\beta_1,\beta_2}(Q, P_1, \ldots, P_m)$

1. $(Q, PP_1, \ldots, PP_m) = \Gamma_{\tau,\beta_1}(Q, P_1, \ldots, P_m)$;
2. $(CC_1, \ldots, CC_m) = \mathrm{ECB}_K(PP_1, \ldots, PP_m)$;
3. if $|Q| > 0$ then
4. $Z = E_K(PP_m \oplus CC_m)$;
5. $R = Q \oplus \mathsf{First}_r(Z)$;
6. end if;
7. $(R, C_1, \ldots, C_m) = \Gamma^{-1}_{\tau,\beta_2}(R, CC_1, \ldots, CC_m)$.

Algorithm $\mathsf{Decrypt}_{K,\tau,\beta_1,\beta_2}(R, C_1, \ldots, C_m)$

1. $(R, CC_1, \ldots, CC_m) = \Gamma_{\tau,\beta_2}(R, C_1, \ldots, C_m)$;
2. $(PP_1, \ldots, PP_m) = \mathrm{ECB}^{-1}_K(CC_1, \ldots, CC_m)$;
3. if $|R| > 0$ then
4. $Z = E_K(PP_m \oplus CC_m)$;
5. $Q = R \oplus \mathsf{First}_r(Z)$;
6. end if;
7. $(Q, P_1, \ldots, P_m) = \Gamma^{-1}_{\tau,\beta_1}(Q, PP_1, \ldots, PP_m)$.

6 Combining With Other Modes of Operations

The AU family ψ has been used to build the BAU family $\Psi[\psi]$. This in turn has been combined with the ECB mode of operation to construct HEH$[\psi]$, HEH$^*[\psi]$ and their variants. On the other hand, it is possible to directly combine ψ with other modes of operations to obtain different tweakable enciphering schemes.

6.1 Counter Mode of Operation

For the case of $\psi = \mathsf{Poly}$, one example is HCH. Below, we present a more general description using ψ and show that it can also be instantiated with BRW. In our description, we use a different version of the counter mode. In this version, an additional key β is required. The role of β is similar to that played in the definition of HEH. As a result of the new form of the counter mode, we are able to eliminate one block cipher call from HCH. The new version is called iHCH (for improved HCH).

Fix a positive integer m and define a keyed family of hash functions $H_{\tau,\beta} : GF(2^n)^m \to GF(2^n)$ as follows.

$$H_{\tau,\beta} = \beta \oplus X_1 \oplus \tau\psi_\tau(X_2, \ldots, X_{m-1}). \tag{18}$$

Here ψ is either Poly or BRW. In both cases, X_1 is the constant term of the corresponding polynomial.

The encryption and decryption algorithms for iHCH are shown in Table 6. The definition of the hashing keys are as in Table 3 giving rise to the variants iHCHp and iHCHfp.

Note: For HEH we had starred and unstarred versions, while for HCH we have only the unstarred version. In case of HEH, the starred version can handle arbitrary length messages, while the unstarred version can only handle message lengths which are multiples of the block length n. In contrast, HCH itself can handle arbitrary length messages. This difference is essentially due to the difference in the encryption layer. When ECB is used, it is a little more complicated to handle arbitrary length messages and so we have presented two versions (unstarred and starred) which should help in understanding the construction. On the other hand, when the counter mode is used, the encryption layer is essentially a stream cipher, and hence it is easy to directly handle arbitrary length messages. That is why we have presented only one version of HCH.

The message length ℓ is greater than n. (In [3], the case $\ell = n$ is tackled separately and the same technique also works in the present case). The number of blocks is ≥ 2; blocks $P_1, \ldots, P_m$ are full blocks and P_{m+1} is a possible partial block of length r, so that $\ell = mn + r$, with $1 \leq r \leq n$.

Given an n-bit string S and a key K we define the counter mode as follows.

$$\mathsf{Ctr}_{K,\beta,S}(A_1, \ldots, A_m) = (A_1 \oplus E_K(S \oplus \beta), A_2 \oplus E_K(S \oplus \alpha\beta), \ldots, A_m \oplus E_K(S \oplus \alpha^{m-1}\beta)). \tag{19}$$

Depending on whether $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$, we obtain iHCH[Poly] or iHCH[BRW] and its variants.

6.2 Output Feedback Mode of Operation

The output feedback mode, OFB, is defined in the following manner.

$$\mathsf{OFB}_{K,S}(X_1, \ldots, X_m) = (X_1 \oplus S_1, \ldots, X_m \oplus S_m) \tag{20}$$

Table 6. Encryption and decryption using iHCH. The block cipher key is K; and the hash key is (τ, β_1, β_2). Definitions of hashing keys are given in Table 3. If KeyDef1 is used we call the construction iHCH; if KeyDef2 is used we call the construction iHCHp; and if KeyDef3 is used we call the construction iHCHfp.

Algorithm $\mathsf{Encrypt}^T_{K,\beta_1,\beta_2}(P_1,\ldots,P_m)$	**Algorithm** $\mathsf{Decrypt}^T_K(C_1,\ldots,C_m)$
2. $M_m = \mathsf{pad}_{n-r}(P_m)$;	2. $U_m = \mathsf{pad}_{n-r}(C_m)$;
3. $M_1 = H_{\tau,\beta_1}(P_1,\ldots,P_{m-1},M_m)$;	3. $U_1 = H_{\tau,\beta_2}(C_1,\ldots,C_{m-1},U_m)$;
4. $U_1 = E_K(M_1)$; $S = M_1 \oplus U_1 \oplus (\beta_1 \oplus \beta_2)$;	4. $M_1 = E_K^{-1}(U_1)$; $S = M_1 \oplus U_1 \oplus (\beta_1 \oplus \beta_2)$;
5. $(C_2,\ldots,C_{m-1},D_m)$ $= \mathsf{Ctr}_{K,\beta_1,S}(P_2,\ldots,P_{m-1},M_m)$;	5. $(P_2,\ldots,P_{m-1},V_m)$ $= \mathsf{Ctr}_{K,\beta_1,S}(C_2,\ldots,C_{m-1},U_m)$;
6. $C_m = \mathsf{drop}_{n-r}(D_m)$; $U_m = \mathsf{pad}_{n-r}(C_m)$;	6. $P_m = \mathsf{drop}_{n-r}(V_m)$; $M_m = \mathsf{pad}_{n-r}(P_m)$;
7. $C_1 = H_{\tau,\beta_2}(U_1,C_2,\ldots,C_{m-1},U_m)$;	7. $P_1 = H_{\tau,\beta_1}(M_1,P_2,\ldots,P_{m-1},M_m)$;
8. return $(C_1,\ldots,C_m)$.	8. return $(P_1,\ldots,P_m)$.

where $S_1 = E_K(S)$ and for $i > 1$, $S_i = E_K(S_{i-1})$.

Basically, both counter and OFB modes turn a block cipher into a stream cipher. In the counter mode of operation, the encryptions can be done in parallel, whereas in the OFB mode of operation, the encryptions are necessarily sequential.

Encryption and decryption algorithms are given in Table 7. This gives rise to HOH[ψ] where ψ is either Poly or BRW. The variants HOHp and HOHfp of HOH are obtained by defining the keys τ and β_1, β_2 as in Table 3.

For hardware implementation, if one wishes to implement only one encryption block, then both iHCH and HOH are expected to have the same efficiency. There does not appear to be any compelling technical reason to prefer one over the other. On the other hand, if more than one encryption blocks are to be implemented, or there are other opportunities for exploiting parallelism, then one would prefer iHCH over HOH.

Table 7. Encryption and decryption using HOH. The block cipher key is K; and the hash key is (τ, β_1, β_2). Definitions of hashing keys are given in Table 3. If KeyDef1 is used we call the construction HOH; if KeyDef2 is used we call the construction HOHp; and if KeyDef3 is used we call the construction HOHfp.

Algorithm $\mathsf{Encrypt}^T_{K,\beta_1,\beta_2}(P_1,\ldots,P_m)$	**Algorithm** $\mathsf{Decrypt}^T_K(C_1,\ldots,C_m)$
2. $M_m = \mathsf{pad}_{n-r}(P_m)$;	2. $U_m = \mathsf{pad}_{n-r}(C_m)$;
3. $M_1 = H_{\tau,\beta_1}(P_1,\ldots,P_{m-1},M_m)$;	3. $U_1 = H_{\tau,\beta_2}(C_1,\ldots,C_{m-1},U_m)$;
4. $U_1 = E_K(M_1)$; $S = M_1 \oplus U_1$;	4. $M_1 = E_K^{-1}(U_1)$; $S = M_1 \oplus U_1$;
5. $(C_2,\ldots,C_{m-1},D_m)$ $= \mathsf{OFB}_{K,S}(P_2,\ldots,P_{m-1},M_m)$;	5. $(P_2,\ldots,P_{m-1},V_m)$ $= \mathsf{OFB}_{K,S}(C_2,\ldots,C_{m-1},U_m)$;
6. $C_m = \mathsf{drop}_{n-r}(D_m)$; $U_m = \mathsf{pad}_{n-r}(C_m)$;	6. $P_m = \mathsf{drop}_{n-r}(V_m)$; $M_m = \mathsf{pad}_{n-r}(P_m)$;
7. $C_1 = H_{\tau,\beta_2}(U_1,C_2,\ldots,C_{m-1},U_m)$;	7. $P_1 = H_{\tau,\beta_1}(M_1,P_2,\ldots,P_{m-1},M_m)$;
8. return $(C_1,\ldots,C_m)$.	8. return $(P_1,\ldots,P_m)$.

6.3 CBC and CFB

It is possible to obtain tweakable enciphering schemes using CBC and CFB as the encryption layers. The constructions are similar to that used for the OFB mode and the key definitions of Table 3 can be used. We have chosen not to present these details, since, from the point of view of tweakable enciphering modes, we could not see any particular advantage of using CBC or CFB. It is possible that such constructions may be advantageous for other applications.

7 Discussion and Comparison

Efficiency of the different schemes (HEH, HEH*, iHCH and HOH) and their variants are determined by the two things: whether $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$; and which of the key definitions in Table 3 is used. Formally, the constructions given in this paper are given in Table 8. For convenience of

Table 8. The constructions defined in this work.

HEH[KeyDef1,Poly]	HEH[KeyDef2,Poly]	HEH[KeyDef3,Poly]
HEH[KeyDef1,BRW]	HEH[KeyDef2,BRW]	HEH[KeyDef3,BRW]
HEH*[KeyDef1,Poly]	HEH*[KeyDef2,Poly]	HEH*[KeyDef3,Poly]
HEH*[KeyDef1,BRW]	HEH*[KeyDef2,BRW]	HEH*[KeyDef3,BRW]
iHCH[KeyDef1,Poly]	iHCH[KeyDef2,Poly]	iHCH[KeyDef3,Poly]
iHCH[KeyDef1,BRW]	iHCH[KeyDef2,BRW]	iHCH[KeyDef3,BRW]
HOH[KeyDef1,Poly]	HOH[KeyDef2,Poly]	HOH[KeyDef3,Poly]
HOH[KeyDef1,BRW]	HOH[KeyDef2,BRW]	HOH[KeyDef3,BRW]

notation, we have not used any suffix when KeyDef1 is used; have used the suffix "p" when KeyDef2 is used; and have used the suffix "fp" when KeyDef3 is used. For example, HEH[Poly] is actually HEH[KeyDef1,Poly] and HEHfp[BRW] is actually HEH[KeyDef3,BRW].

The number of keys and the number of block cipher calls are determined by the key definition. This is shown in Table 9. On the other hand, the number of $GF(2^n)$ multiplications is determined by whether $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$. This dependence is shown in Table 10. We note the following

Table 9. Number of keys and the number of block cipher calls for the different modes used with the key definitions given in Table 3. We assume that there are m (full or partial) blocks.

	KeyDef1	KeyDef2	KeyDef3
# keys	1[BK]	1[BK]+1[AK]	1[BK]+1[AK]
# BC calls	$m+2$	$m+2$	$m+1$

Table 10. Number of $GF(2^n)$ multiplications for $\psi = \mathsf{Poly}$ versus $\psi = \mathsf{BRW}$. This number includes multiplications for both the hashing layers. We assume that there are m (full or partial) blocks.

	$\psi = \mathsf{Poly}$	$\psi = \mathsf{BRW}$
# mults.	$2(m-1)$	m

points for the schemes described in this paper.

1. HEH and its variants HEHp and HEHfp can only handle messages which are multiples of the block length. All other modes can handle messages with partial blocks.
2. If KeyDef1 or KeyDef2 is used, then variable length messages can be handled. If KeyDef3 is used, then the length of the message has to be fixed.
3. If KeyDef2 or KeyDef3 is used, then the hashing key τ is independent of the block cipher key. (For KeyDef1 this is derived from the block cipher key.) Consequently, if $\psi = \mathsf{Poly}$, one can use a pre-computed table to speed up the $GF(2^n)$ multiplication for these two cases. On the other hand, if $\psi = \mathsf{BRW}$, then this is not possible because BRW polynomials cannot utilize pre-computation.

4. If KeyDef3 is used, the length of the messages is fixed. Due to this reason it is possible to reduce the number of block cipher calls by one.

7.1 Comparison Between TET and HEH*

HEH* uses the hash-ECB-hash approach introduced by Naor-Reingold. An earlier construction using the same approach is TET. Both HEH* and TET use a hashing key τ, with the difference that for TET, τ must satsify the restriction that $\sigma = \sum_{i=0}^{m} \tau^i \neq 0$, while for HEH* there is no restriction on τ.

Variable Length Messages. For TET, since m varies, for each message, σ and σ^{-1}, has to be computed. Computing σ requires m multiplications over $GF(2^n)$. This together with the requirement of computing an inverse makes TET unsuitable for variable length messages. Such computation is not required at all for HEH*.

Key Agility. If m is fixed, then for TET, σ and σ^{-1} can be pre-computed. The hash key τ is obtained in TET by applying a PRF having key K_1 to a fixed input. Now, suppose the key K_1 is changed. Then τ also changes and hence σ and σ^{-1} also need to be re-computed. Thus, for TET, key change is computationally expensive. For HEH, key change does not require any field operation.

Computing a Field Inverse. TET requires computation of a field inverse, either in the online phase, or during a key change. For hardware only implementation, this means that an inversion circuit has to be implemented. HEH does not require field inversion for any operation.

For variable number of blocks, a comparison between TET and HEH* is given in Table 11. TET requires some extra block cipher invocations and multiplications; basically the term multiplied by $\imath$. The value of $\imath$ itself depends on the number of blocks and the PRF key. Also, TET requires a $GF(2^n)$ inversion. These computations are required to obtain a hash key τ such that $\sigma = 1 + \tau + \cdots + \tau^m \neq 0$ and to invert σ (if $\sigma \neq 0$). Thus, the restriction on the hashing key directly reflects on the performance of TET. We note that the value of $\imath$ is at least 1, which makes the performance of TET clearly inferior to that of HEH*.

Table 11. Comparison between TET and HEH* when the number of blocks m can vary and n-bit tweaks are used. For TET, $\imath$ is a value (at least 1) which depends on m (and K). [BC]: block cipher invocation; [M]: $GF(2^n)$ multiplication; [I]: $GF(2^n)$ inversion.

Mode	comp. cost	# keys
TET [6]	$\imath((m-1)$[M]$+2$[BC]$)+(m+2)$[BC]$+2m$[M]$+1$[I]	2
HEH*[Poly]	$(m+2)$[BC]$+2(m-1)$[M]	1
HEH*[BRW]	$(m+2)$[BC]$+m$[M]	1

7.2 Comparison Among Different Constructions for Fixed Length Messages

Disk encryption is perhaps the most important application of tweakable enciphering schemes. For this application, encryption is done sector-wise and the sector address is the tweak. Since the length of a sector is fixed, the length of the messages is also fixed and is usually a multiple of the block length. For example, a typical value of sector length is 512 bytes. If $n = 128$, then 512 bytes correspond to 32 n-bit blocks.

We make a comparison among different constructions under the condition that messages are of fixed length. This is shown in Table 12. For some of the constructions, it is possible to take advantage of fixed length inputs and simplify encryption and decryption by pre-computing some quantities. The values in Table 12 reflect this approach. Though pre-computing quantities improve the efficiency of encryption and decryption, there is a disadvantage. The pre-computed quantities depend on the key and hence, a larger amount of secure memory is required for storing them. Perhaps more importantly, these quantities are required to be computed at every key change which negatively impacts key agility. Efficiencies of key changes for the different modes are shown in Table 13.

Table 12. Comparison of different tweakable SPRPs where the number of blocks m is fixed and n-bit tweaks are used. [BC]: number of block cipher invocations; [M]: $GF(2^n)$ multiplication; [BCK]: block cipher key; [AK]: auxiliary n-bit string (including polynomial hash keys).

Mode	[BC]	[M]	[BCK]	[AK]	pre-comp.
CMC [7]	$2m+1$	–	1	–	–
EME* [5]	$2m+1+m/n$	–	1	2	–
XCB [11]	$m+1$	$2(m+3)$	3	2	yes
HCTR [18]	m	$2(m+1)$	1	1	yes
HCHfp [3]	$m+2$	$2(m-1)$	1	1	yes
TET [6]	$m+1$	$2m$	2	3	yes
HEHfp[Poly]	$m+1$	$2(m-1)$	1	1	yes
HEHfp[BRW]	$m+1$	m	1	1	no
iHCHfp[Poly]	$m+1$	$2(m-1)$	1	1	yes
iHCHfp[BRW]	$m+1$	m	1	1	no
HOHfp[Poly]	$m+1$	$2(m-1)$	1	1	yes
HOHfp[BRW]	$m+1$	m	1	1	no

Table 13. Efficiency of key change. For TET, the value of $\imath$ depends only on the block cipher key K (since the number of blocks m is fixed) and is at least 1. [I]: $GF(2^n)$ inversion.

Mode	comp. cost	key sch.	mult. tab.
CMC [7]	–	1	–
EME* [5]	–	1	–
XCB [11]	5[BC]	3	2
HCTR [18]	–	1	1
HCHfp [3]	–	1	1
TET [6]	$\imath((m-1)$[M]+2[BC])+1[BC]+1[I]	2	1
HEHfp[Poly]	–	1	1
HEHfp[BRW]	–	1	–
iHCHfp[Poly]	–	1	1
iHCHfp[BRW]	–	1	–
HOHfp[Poly]	–	1	1
HOHfp[BRW]	–	1	–

The comparison to the encrypt-mix-encrypt approach is based on the relative efficiency of a block cipher call and a $GF(2^n)$ multiplication and whether $\psi = \mathsf{Poly}$ or $\psi = \mathsf{BRW}$.

1. Case $\psi = \mathsf{Poly}$. In this case, the hash-encrypt-hash approach is faster than the encrypt-mix-encrypt approach, provided one block cipher call takes more time than two multiplications. If

feasible, then it is possible to use pre-computed tables to speed up multiplications and ensure this condition.
2. Case $\psi = \mathsf{BRW}$. In this case, the hash-encrypt-hash approach is faster than the encrypt-mix-encrypt approach, provided one block cipher call takes more time than one multiplication. Pre-computed multiplication tables cannot be used in this case. Even without pre-computed tables, a $GF(2^{128})$ multiplication will not be less efficient than one AES-128 invocation. Hence, for AES-128, using $\psi = \mathsf{BRW}$ will ensure that the hash-encrypt-hash approach is at least as efficient as the encrypt-mix-encrypt approach.

Key agility of TET. When m is fixed, the value of τ and σ^{-1} can be pre-computed in TET. This makes the actual encryption and decryption in TET quite efficient. However, the problem of generating τ and σ^{-1} is still present whenever the key needs to be changed. That is, even though m is fixed, whenever the PRF key changes, the value of τ and σ^{-1} has to be computed afresh. This adversely affects the key agility of TET.

Which mode is the best for disk encryption? The answer to this question comes in parts.

1. From the discussion above, there does not seem to be any reason to prefer the encrypt-mix-encrypt approach over the hash-encrypt-hash approach, especially since the constructions in the encrypt-mix-encrypt approach are covered by IP claims.
2. From Tables 12 and 13, the proposals for the hash-encrypt-hash approach given in this paper are more efficient than previous proposals even with $\psi = \mathsf{Poly}$. This includes both the cases when the encryption layer is ECB mode and when the encryption layer is the counter mode. There has no previous proposal using the OFB mode.
3. If pre-computation of multiplication tables is considered undesirable (because of the problems of caching and key agility, especially in a multi-key environment), then HEHfp[BRW] is the construction of choice for implementing disk encryption protocols. To the best of our knowledge, there are no IP claims associated with this construction.

8 Security

The definitions and notation are based on earlier work [7]. An n-bit block cipher is a function $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$, where $\mathcal{K} \neq \emptyset$ is the key space and for any $K \in \mathcal{K}$, $E(K,.)$ is a permutation. We write $E_K(\,)$ instead of $E(K,.)$.

An adversary $\mathcal{A}$ is a probabilistic algorithm which has access to some oracles and which outputs either 0 or 1. Oracles are written as superscripts. The notation $\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2} \Rightarrow 1$ denotes the event that the adversary $\mathcal{A}$, interacts with the oracles $\mathcal{O}_1, \mathcal{O}_2$, and finally outputs the bit 1. Let $\mathrm{Perm}(n)$ denote the set of all permutations on $\{0,1\}^n$. Formally, a tweakable enciphering scheme is a function $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$, where $\mathcal{K} \neq \emptyset$ and $\mathcal{T} \neq \emptyset$ are the key space and the tweak space respectively. The message and the cipher spaces are $\mathcal{M}$. We shall write $\mathbf{E}_K^T(.)$ instead of $\mathbf{E}(K,T,.)$. The inverse of an enciphering scheme is $\mathbf{D} = \mathbf{E}^{-1}$ where $X = \mathbf{D}_K^T(Y)$ if and only if $\mathbf{E}_K^T(X) = Y$.

Let $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ denote the set of all functions $\boldsymbol{\pi} : \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ where $\boldsymbol{\pi}(\mathcal{T},.)$ is a length preserving permutation. Such a $\boldsymbol{\pi} \in \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ is called a tweak indexed permutation. For a tweakable enciphering scheme $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$, we define the advantage an adversary $\mathcal{A}$ has in distinguishing $\mathbf{E}$ and its inverse from a random tweak indexed permutation and its inverse in the following manner.

$$\mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{prp}}}(\mathcal{A}) = \left| \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : \mathcal{A}^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\right|.$$

Pointless queries: We assume that an adversary never repeats a query, i.e., it does not ask the encryption oracle with a particular value of (T, P) more than once and neither does it ask the decryption oracle with a particular value of (T, C) more than once. Furthermore, an adversary never queries its deciphering oracle with (T, C) if it got C in response to an encipher query (T, P) for some P. Similarly, the adversary never queries its enciphering oracle with (T, P) if it got P as a response to a decipher query of (T, C) for some C. These queries are called *pointless* as the adversary knows what it would get as responses for such queries.

We define the query complexity σ_n of an adversary to be the total number of n-bit blocks it provides in all its encryption and decryption queries. This includes the plaintext and ciphertext blocks as well as the n-bit tweak. By $\mathbf{Adv}(\sigma_n)$ (with suitable sub and super-scripts) we denote the maximum advantage of any adversary with query complexity σ_n. The notation $\mathbf{Adv}(\sigma_n, t)$ denotes the maximum advantage of any adversary with query complexity σ_n and running time t.

The notation HEH[E] denotes a tweakable enciphering scheme, where the block cipher E is used in the manner specified by HEH. (Similar notations hold for the other constructions.) The notation HEH[Perm(n)] denotes a tweakable enciphering scheme obtained by plugging in a random permutation from Perm(n) into the structure of HEH. For an adversary attacking HEH[Perm(n)], we do not put any bound on the running time of the adversary, though we still put a bound on the query complexity σ_n. This advantage is denoted by $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{HEH[Perm}(n)]}(\sigma_n)$. We need to consider an adversary's advantage in distinguishing a tweakable enciphering scheme $\mathbf{E}$ from an oracle which simply returns random bit strings. This advantage is defined in the following manner.

$$\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH[Perm}(n)]}(A) = \left|\Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\mathbf{E}_\pi, \mathbf{D}_\pi} \Rightarrow 1\right] - \Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right]\right|$$

where $\$(., M)$ returns random bits of length $|M|$.

The task of the security proof is to upper bound $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{HEH}[E]}(\sigma_n, t)$. For this it is sufficient to upper bound $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{HEH[Perm}(n)]}(\sigma_n)$. Again, to upper bound the last quantity, it is sufficient to upper bound $\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH[Perm}(n)]}(\sigma_n)$. This approach has been used in previous works and for details of how these three advantages are related we refer the reader to previous work [7, 8, 4]. The relationships between these three advantages are independent of the particular tweakable SPRP being considered. Hence, we do not repeat the details here. The main task of the proof is to obtain a upper bound on $\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH[Perm}(n)]}(\sigma_n)$.

Theorem 4. *Fix n and σ_n to be positive integers. Suppose that an adversary uses a total of σ_n blocks in all its queries, where each block is an n-bit string. Then*

$$\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{XXX[Perm}(n)]}(\sigma_n) \leq \frac{20\sigma_n^2}{2^n} \tag{21}$$

Here XXX *is one of the tweakable enciphering schemes* HEH, HEH*, iHCH *or* HOH *used with either* KeyDef1, KeyDef2 *or* KeyDef3 *and* $\psi =$ Poly *or* BRW.

Proofs for HEH, HEHp and HEHfp are similar and is given in Section A. Proofs of HEH* and its variants are a little different. We do not describe the entire proof. The reason for the difference and how this is handled is described in Section B. Sketches of proofs for iHCH and HOH are given in Sections C and D.

The constant 20 holds for all the modes. But, for specific modes, the constant is lower. It depends on whether $\psi =$ Poly or whether $\psi =$ BRW. For example, for HEH, HEHp and HEHfp,

with $\psi = \mathsf{Poly}$ the constant is 4 whereas with $\psi = \mathsf{BRW}$, the constant is 10. We have stated the result with a single constant so as to simplify the statement. Also, 20 by itself is a small enough constant.

9 Conclusion

In this paper, we have proposed new tweakable enciphering schemes which use a single layer of encryption between two layers of universal hash function computations. This is achieved by defining new (block-wise) universal hash functions and suitably combining with ECB, Counter and OFB modes of operations of a block cipher. An important aspect of our work is to use a new class of polynomials introduced by Bernstein. This improves the efficiency of the hashing layers by almost a factor of two. In the current state of the art, our work provides the best known algorithms for the important practical problem of disk encryption protocols.

Acknowledgement

We would like to thank Daniel J. Bernstein for pointing out reference [1] to us. The work [1] is significant, since it allows to improve the efficiency of the hashing layers in our constructions by almost a factor of two.

References

1. Daniel J. Bernstein. Polynomial evaluation and message authentication, 2007. `http://cr.yp.to/papers.html#pema`.
2. Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR, 2008. to appear.
3. Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006. full version available at `http://eprint.iacr.org/2007/028`.
4. Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2006.
5. Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004.
6. Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.
7. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003.
8. Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.
9. 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.
10. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
11. David A. McGrew and Scott R. Fluhrer. The extended codebook (XCB) mode of operation. Cryptology ePrint Archive, Report 2004/278, 2004. `http://eprint.iacr.org/`.
12. David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (XCB) mode of operation. In *Selected Areas in Cryptography*, Lecture Notes in Computer Science. Springer, 2007. To appear.
13. Moni Naor and Omer Reingold. A pseudo-random encryption mode. Manuscript available from `www.wisdom.weizmann.ac.il/~naor`.
14. Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *J. Cryptology*, 12(1):29–66, 1999.

15. Michael O. Rabin and Shmuel Winograd. Fast evaluation of polynomials by rational preparation. *Communications on Pure and Applied Mathematics*, 25:433–458, 1972.
16. Palash Sarkar. Improving upon the TET mode of operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2007.
17. Victor Shoup. On fast and provably secure message authentication based on universal hashing. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328. Springer, 1996.
18. Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *CISC*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.

A Proof of HEH and its variants

The adversary has to distinguish between two kinds of oracles. In the first kind, the adversary is given encryption and decryption oracles for the mode of operation with the block cipher substituted by a random permutation from $\mathrm{Perm}(n)$. In the second kind, the adversary is given two oracles which simply returns random strings of length equal to its input. The statement of the result upper bounds an adversary's advantage in distinguishing between these two kinds of oracles. The proof is via a sequence of games.

Notation: In this and the following games, we will use the subscript s to denote quantities related to the s-th query. For example, if the s-the query is an encryption query, then it will be of the form $T_s, (P_{s,1}, \ldots, P_{s,m_s})$. This means that the tweak is T_s, the number of blocks is m_s and the plaintext blocks are $(P_{s,1}, \ldots, P_{s,m_s})$. A similar interpretation holds when the s-th query is a decryption query of the form $T_s, (C_{s,1}, \ldots, C_{s,m_s})$. By τ_s, $\beta_{1,s}$ and $\beta_{2,s}$ we denote the τ, β_1 and β_2 associated with the s^{th} query. In HEHp and HEHfp, τ does not change with the message. It is randomly chosen at the start of the game. We assume that the adversary $\mathcal{A}$ makes a total of q queries. In the s-th query, the adversary specifies $m_s + 1$ blocks, i.e., the m_s plaintext or ciphertext blocks along with the n-bit tweak. Thus, we have $\sigma_n = \sum_{s=1}^{q}(1 + m_s)$.

Denote by $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$ respectively the encryption and decryption oracles of HEH instantiated by a random permutation from $\mathrm{Perm}(n)$. Let $\mathcal{A}$ be an adversary. The notation $\mathcal{A}^{\mathrm{HEH1}} \Rightarrow 1$ denotes the fact that $\mathcal{A}$ outputs 1 in Game HEH1. Similar notation holds for the other games.

Game HEH1*:* In HEH1, the adversary interacts with $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$. Instead of choosing a random permutation π, we build up π in the following manner. Initially π is assumed to be undefined everywhere. When $\pi(X)$ is needed, but the value of π is not yet defined at X, then a random value is chosen among the available range values. Similarly when $\pi^{-1}(Y)$ is required and there is no X yet defined for which $\pi(X) = Y$, we choose a random value for $\pi^{-1}(Y)$ from the available domain values.

Table 14 shows how the s^{th} query is tackled in Game HEH1. The sets *Domain* and *Range* are subsets of $\{0,1\}^n$ and keep track of the elements in the domain and range of π which have been defined. Initially, these are chosen to be empty sets and at any point of time $\overline{Domain} = \{0,1\}^n \setminus Domain$ and $\overline{Range} = \{0,1\}^n \setminus Range$. Subroutines Ch-$\pi$ and Ch-π^{-1} used in the game are shown in Table 15.

HEH and its variants differ in the way the hashing keys are defined. This difference is reflected in Game HEH1. The different ways of defining the hashing keys for this game are shown in Table 16. If KeyDef2 or KeyDef3 are used, then τ is randomly chosen once for all at the beginning of the game.

Table 14. Game HEH1

Encrypt query:$(T_s; P_{s,1}, P_{s,2}, \ldots P_{s,m_s})$	Decrypt query:$(T_s; C_{s,1}, C_{s,2}, \ldots C_{s,m_s})$
If KeyDef1 is used, then do Sub1(T_s, ℓ_s) in Table 16.	If KeyDef1 is used, then do Sub1(T_s, ℓ_s) in Table 16.
If KeyDef2 is used, then do Sub2(T_s, ℓ_s) in Table 16.	If KeyDef2 is used, then do Sub2(T_s, ℓ_s) in Table 16.
If KeyDef3 is used, then do Sub3(ℓ_s) in Table 16.	If KeyDef3 is used, then do Sub3(ℓ_s) in Table 16.
$(PP_{s,1}, \ldots, PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s})$;	$(CC_{s,1}, \ldots, CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s})$;
for $i = 1$ **to** m_s **do**	**for** $i = 1$ **to** m_s **do**
$CC_{s,i} = \text{Ch-}\pi(PP_{s,i})$;	$PP_{s,i} = \text{Ch-}\pi^{-1}(CC_{s,i})$;
end do;	**end do**;
$(C_{s,1}, \ldots, C_{s,m_s}) = \Psi^{-1}_{\tau_s,\beta_{2,s}}(CC_{s,1}, \ldots, CC_{s,m_s})$;	$(P_{s,1}, \ldots, P_{s,m_s}) = \Psi^{-1}_{\tau_s,\beta_{1,s}}(PP_{s,1}, \ldots, PP_{s,m_s})$;
return $(C_{s,1}, \ldots, C_{s,m_s})$.	**return** $(P_{s,1}, \ldots, P_{s,m_s})$.

Table 15. Subroutines Ch-$\pi(X)$ and Ch-$\pi^{-1}(Y)$. The boxed portions are not part of game RAND1.

Subroutine Ch-$\pi(X)$	**Subroutine Ch-**$\pi^{-1}(Y)$
$Y \xleftarrow{\$} \{0,1\}^n$;	$X \xleftarrow{\$} \{0,1\}^n$;
if $Y \in \mathit{Range}$ **then**	**if** $X \in \mathit{Domain}$, **then**
bad $\leftarrow$ true; $\boxed{Y \xleftarrow{\$} \overline{\mathit{Range}};}$	bad $\leftarrow$ true; $\boxed{X \xleftarrow{\$} \overline{\mathit{Domain}};}$
if $X \in \mathit{Domain}$ **then**	**if** $Y \in \mathit{Range}$ **then**
bad $\leftarrow$ true; $\boxed{Y \leftarrow \pi(X);}$	bad $\leftarrow$ true; $\boxed{X \leftarrow \pi^{-1}(Y);}$
$\pi(X) \leftarrow Y$;	$\pi(X) \leftarrow Y$;
$\mathit{Domain} \leftarrow \mathit{Domain} \cup \{X\}$;	$\mathit{Domain} \leftarrow \mathit{Domain} \cup \{X\}$;
$\mathit{Range} \leftarrow \mathit{Range} \cup \{Y\}$;	$\mathit{Range} \leftarrow \mathit{Range} \cup \{Y\}$;
return(Y);	**return**(X);

Table 16. Sub1 shows how to obtain τ_s and $\beta_{1,s}$ when KeyDef1 is used. Sub2 shows how to choose $\beta_{1,s}$ when KeyDef2 is used. Sub3 shows how to choose $\beta_{1,s}$ when KeyDef3 is used. For Sub2 and Sub3, τ does not change with the message. It is randomly chosen once for all at the start of the game. Also, for KeyDef3 (and hence for Sub3), the length ℓ is fixed for all messages.

Sub1(T_s, ℓ_s)	Sub2(T_s, ℓ_s)	Sub3(T_s)
if $T_s = T_t$ for some $t < s$	**if** $T_s = T_t$ for some $t < s$	
then	**then**	
$\tau_s = \tau_t$;	$\gamma_s = \gamma_t$;	
if $\ell_s = \ell_t$	**if** $\ell_s = \ell_t$	
then $\beta_{1,s} = \beta_{1,t}$;	**then** $\beta_{1,s} = \beta_{1,t}$;	**if** $T_s = T_t$ for some $t < s$
else $\beta_{1,s} \leftarrow \text{Ch-}\pi(\tau_s \oplus \mathsf{bin}(\ell_s))$;	**else** $\beta_{1,s} \leftarrow \text{Ch-}\pi(\gamma_s \oplus \mathsf{bin}(\ell_s))$;	**then** $\beta_{1,s} = \beta_{1,t}$;
else	**else**	**else** $\beta_{1,s} \leftarrow \text{Ch-}\pi(T_s)$;
$\tau_s \leftarrow \text{Ch-}\pi(T_s)$;	$\gamma_s \leftarrow \text{Ch-}\pi(T_s)$;	$\beta_{2,s} = \alpha\beta_{1,s}$.
$\beta_{1,s} \leftarrow \text{Ch-}\pi(\tau_s \oplus \mathsf{bin}(\ell_s))$;	$\beta_{1,s} \leftarrow \text{Ch-}\pi(\gamma_s \oplus \mathsf{bin}(\ell_s))$;	
endif;	**endif**;	
$\beta_{2,s} = \alpha\beta_{1,s}$.	$\beta_{2,s} = \alpha\beta_{1,s}$.	

Game RAND1*:* We modify Game HEH1 by dropping the boxed portions in subroutines Ch-π and Ch-π^{-1}. As a result, it is no longer ensured that π is a permutation. This happens if bad is set to true for some invocation to one of these subroutines. On the other hand, if bad is not set to true, then Games HEH1 and RAND1 are identical. So,

$$\Pr[\mathcal{A}^{\text{HEH1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{RAND1}} \Rightarrow 1] \leq \Pr[A^{\text{RAND1}} \text{ sets } \mathsf{bad}]. \tag{22}$$

Game RAND2*:* Consider an encryption query in Game RAND1. Since there are no checks on π, each $CC_{s,i}$ is an independently and uniformly chosen random n-bit string. Further, since

$$(C_{s,1}, \ldots, C_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(CC_{s,1}, \ldots, CC_{s,m_s})$$

and Ψ is an invertible map, $(C_{s,1}, \ldots, C_{s,m_s})$ is distributed uniformly over $\{0,1\}^{nm_s}$. Thus, the adversary gets back random strings in response to any encryption query. A similar reasoning shows that the adversary also gets back random strings in response to any decryption query.

Table 17. Game RAND2.

Respond to the s^{th} adversarial query as follows: ENCRYPT QUERY $\mathsf{Enc}(T_s; P_{s,1}, P_{s,2}, \ldots, P_{s,m_s})$ $ty_s = \mathsf{Enc}$; $(C_{s,1}, C_{s,2}, \ldots, C_{s,m_s}) \xleftarrow{\$} \{0,1\}^{nm_s}$; **return** $(C_{s,1}, C_{s,2}, \ldots, C_{s,m_s})$; DECRYPT QUERY $\mathsf{Dec}(T_s; C_{s,1}, C_{s,2}, \ldots, C_{s,m_s})$ $ty^s = \mathsf{Dec}$; $(P_{s,1}, P_{s,2}, \ldots, P_{s,m_s}) \xleftarrow{\$} \{0,1\}^{nm_s}$; **return** $(P_{s,1}, P_{s,2}, \ldots, P_{s,m_s})$;
Finalization
First Phase: For the s^{th} query do the following. 1. Let T_s be the tweak for the s^{th} query. 2. If KeyDef1 is used, then do RAND2-Sub1(T_s, ℓ_s) in Table 18. 3. If KeyDef2 is used, then do RAND2-Sub2(T_s, ℓ_s) in Table 18. 4. If KeyDef3 is used, then do RAND2-Sub3(T_s) in Table 18. 5. $(PP_{s,1}, \ldots, PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s})$; 6. $(CC_{s,1}, \ldots, CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s})$; 7. **for** $i = 1$ **to** m_s **do** 8. $\mathcal{D} = \mathcal{D} \cup \{PP_{s,i}\}$; $\mathcal{R} = \mathcal{R} \cup \{CC_{s,i}\}$; 9. **end for**; *Second Phase:* **if** (some value occurs more than once in $\mathcal{D}$) **then** bad = true; **endif**; **if** (some value occurs more than once in $\mathcal{R}$) **then** bad = true; **endif**.

In Game RAND2, we make this explicit by rewriting the structure of the game. In response to any query (encryption or decryption), the adversary is given random strings. After the query phase is over, a finalization phase is executed. In this phase, the $PP_{s,i}$s and $CC_{s,i}$s are obtained by applying $\Psi_{\tau_s,\beta_{1,s}}$ and $\Psi_{\tau_s,\beta_{2,s}}$ respectively to the $P_{s,i}$s and $C_{s,i}$s.

There is an additional change in this game. In Game RAND1 and the key defining subroutines, calls are made to Ch-π and Ch-π^{-1}. Such calls can result in the variable bad being set to true. This happens when a repetition occurs in one of the sets *Domain* or *Range*. In Game RAND2, we make

Table 18. Obtaining hashing keys for the game RAND2. The definitions are essentially the same as that in Table 16. The only difference is that in this case there are no checks and values are appropriately inserted into $\mathcal{D}$ and $\mathcal{R}$.

RAND2-Sub1(T_s, ℓ_s)	RAND2-Sub2(T_s, ℓ_s)	RAND3-Sub3(T_s)

if $T_s = T_t$ for some $t < s$
then
$\tau_s = \tau_t$;
if $\ell_s = \ell_t$
then $\beta_{1,s} = \beta_{1,t}$;
else
$\beta_{1,s} \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} = \mathcal{D} \cup \{\tau_s \oplus \mathsf{bin}(\ell_s)\}$;
$\mathcal{R} = \mathcal{R} \cup \{\beta_{1,s}\}$;
endif
else
$\tau_s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} = \mathcal{D} \cup \{T_s\}$; $\mathcal{R} = \mathcal{R} \cup \{\tau_s\}$;
$\beta_{1,s} \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} = \mathcal{D} \cup \{\tau_s \oplus \mathsf{bin}(\ell_s)\}$;
$\mathcal{R} = \mathcal{R} \cup \{\beta_{1,s}\}$;
endif;
$\beta_{2,s} = \alpha\beta_{1,s}$.
	if $T_s = T_t$ for some $t < s$
	then
	$\gamma_s = \gamma_t$;
	if $\ell_s = \ell_t$
	then $\beta_{1,s} = \beta_{1,t}$;
	else
	$\beta_{1,s} \xleftarrow{\$} \{0,1\}^n$;
	$\mathcal{D} = \mathcal{D} \cup \{\gamma_s \oplus \mathsf{bin}(\ell_s)\}$;
	$\mathcal{R} = \mathcal{R} \cup \{\beta_{1,s}\}$;
	endif
	else
	$\gamma_s \xleftarrow{\$} \{0,1\}^n$;
	$\mathcal{D} = \mathcal{D} \cup \{T_s\}$; $\mathcal{R} = \mathcal{R} \cup \{\gamma_s\}$;
	$\beta_{1,s} \xleftarrow{\$} \{0,1\}^n$;
	$\mathcal{D} = \mathcal{D} \cup \{\gamma_s \oplus \mathsf{bin}(\ell_s)\}$;
	$\mathcal{R} = \mathcal{R} \cup \{\beta_{1,s}\}$;
	endif;
	$\beta_{2,s} = \alpha\beta_{1,s}$.
		if $T_s = T_t$ for some $t < s$
		then $\beta_{1,s} = \beta_{1,t}$;
		else
		$\beta_{1,s} \xleftarrow{\$} \{0,1\}^n$;
		$\mathcal{D} = \mathcal{D} \cup \{T_s\}$; $\mathcal{R} = \mathcal{R} \cup \{\beta_{1,s}\}$;
		endif;
		$\beta_{2,s} = \alpha\beta_{1,s}$.

this explicit by replacing the sets *Domain* and *Range* by the multi-sets $\mathcal{D}$ and $\mathcal{R}$ respectively. In the first phase of initialization, values are inserted (possibly more than once) into the sets $\mathcal{D}$ and $\mathcal{R}$. In the second phase, the variable bad is set to true if there is a repetition in either of these two sets. With this approach, the subroutines Ch-π and Ch-π^{-1} are no longer required. Instead the key definitions are made and values are directly inserted into $\mathcal{D}$ and $\mathcal{R}$ as shown in Table 18.

Doing the above does not alter the adversary's view of the game since for each such change the adversary obtains random n-bit strings both before and after the change. Thus,

$$\Pr[\mathcal{A}^{\text{RAND1}} \Rightarrow 1] = \Pr[\mathcal{A}^{\text{RAND2}} \Rightarrow 1] \quad \text{and} \quad \Pr[\mathcal{A}^{\text{RAND1}} \text{ sets } \mathsf{bad}] = \Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}].$$

In RAND2, the adversary is supplied with random bits as response to queries to both the encrypt and the decrypt oracles. Hence, $\Pr[\mathcal{A}^{\text{RAND2}} \Rightarrow 1] = \Pr[\mathcal{A}^{\$(.,.),\$(.,.)} \Rightarrow 1]$. We get

$$\begin{aligned}
\mathbf{Adv}^{\pm\text{rnd}}_{\text{HEH}[\text{Perm}(n)]}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathbf{E}_\pi, \mathbf{D}_\pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(.,.),\$(.,.)} \Rightarrow 1] \qquad (23)\\
&= \Pr[\mathcal{A}^{\text{HEH1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{RAND2}} \Rightarrow 1]\\
&= \Pr[\mathcal{A}^{\text{HEH1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{RAND1}} \Rightarrow 1]\\
&\leq \Pr[\mathcal{A}^{\text{RAND1}} \text{ sets } \mathsf{bad}]\\
&= \Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}] \qquad (24)
\end{aligned}$$

Our task is thus to bound $\Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}]$.

Game NON*:* We want to bound the maximum value of $\Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}]$. This probability extends over the random coins of the adversary. However, since the adversary gets back random strings in response to all its queries, it achieves nothing by the interaction with the oracles. We assume that the adversary fixes its queries a priori in a manner such that $\Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}]$ is maximized. Now we can forget about the adversary and work only with the fixed queries.

In the previous games, for an encrypt query, the adversary specified the tweak and the plaintext blocks; and for a decrypt query, the adversary specified the tweak and the ciphertext blocks. We now consider the stronger condition, whereby the adversary specifies the tweak, the plaintext blocks and the ciphertext blocks in both the encryption and the decryption queries. Even under this condition, we show that the flag bad is rarely set to true.

Transcript: A sequence of the following form:

$$\begin{array}{l}\mathsf{ty}_1, T_1, (P_{1,1}, \ldots, P_{1,m_1}), (C_{1,1}, \ldots, C_{1,m_1}),\\ \mathsf{ty}_2, T_2, (P_{2,1}, \ldots, P_{2,m_2}), (C_{2,1}, \ldots, C_{2,m_2}),\\ \vdots\\ \mathsf{ty}_q, T_q, (P_{q,1}, \ldots, P_{q,m_q}), (C_{q,1}, \ldots, C_{q,m_q}).\end{array}$$

The tys denote whether the query is encryption or decryption; tweaks are $T_1, T_2, \ldots$; plaintext blocks are $P_{s,i}$ and the corresponding ciphertext blocks are $C_{s,i}$. Number of blocks can vary and the number of blocks in the s^{th} query is m_s. The query complexity is $\sigma_n = \sum_{s=1}^{q}(1 + m_s)$.

Allowed Transcript: We say that a transcript is allowed if no query (either encryption or decryption) is repeated and if a ciphertext is obtained on a plaintext query with a certain tweak, then it is not allowed to make a decryption query on the ciphertext with the same tweak and vice versa for decryption queries. Queries of the above type are called pointless. Formally, in an allowed transcript the following condition has to hold. For $s \neq t$,

$$(T_s, (P_{s,1}, \ldots, P_{s,m_s})) \neq (T_t, (P_{t,1}, \ldots, P_{t,m_t})) \text{ and } (T_s, (C_{s,1}, \ldots, C_{s,m_s})) \neq (T_t, (C_{t,1}, \ldots, C_{t,m_t})).$$

Given an (allowed) transcript, for $1 \leq s \leq q$, we set

$$\begin{aligned}(PP_{s,1}, \ldots, PP_{s,m_s}) &= \Psi_{\tau,\beta_1}(P_{s,1}, \ldots, P_{s,m_s})\\ (CC_{s,1}, \ldots, CC_{s,m_s}) &= \Psi_{\tau,\beta_2}(C_{s,1}, \ldots, C_{s,m_s})\end{aligned}$$

It is helpful to think of the PPs (resp. CCs) as being organised as a two dimensional list $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) having q rows, with m_s entries in row s. The (s, i) entry of $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) is $PP_{s,i}$ (resp. $CC_{s,i}$).

Source of randomness: An (allowed) transcript fixes the tweaks and the plaintext and ciphertext blocks. The source of randomness comes from the internal choice of τs and βs. Recall that these are chosen as in Table 18.

Consider a subset A of $\{1, \ldots, q\}$, where s is in A if T_s is "new", i.e., there is no $t < s$, such that $T_s = T_t$. Similarly, consider a subset B of $\{1, \ldots, q\}$, where s is in B if (T_s, m_s) is "new", i.e., there is no $t < s$, such that $(T_s, m_s) = (T_t, m_t)$.

Note. For the rest of the analysis, we need to separately consider the three ways of defining the keys. This is because, the actual elements in $\mathcal{D}$ and $\mathcal{R}$ depend on the particular key definition that has been used.

Case of HEH**.** The multi-sets $\mathcal{D}$ and $\mathcal{R}$ are as follows. $\mathcal{D}$ consists of all elements (with possible repetitions) in $\mathcal{L}_1$ along with the elements (also with possible repetitions) in the set

$$\{T_s, \tau_s \oplus (\mathsf{bin}_n(\ell_s)) : s \in A\} \cup \{\tau_s \oplus (\mathsf{bin}_n(\ell_s)) : s \in B \setminus A\}.$$

$\mathcal{R}$ consists of all elements (with possible repetitions) in $\mathcal{L}_2$ along with the elements (also with possible repetitions) in the set

$$\{\tau_s, \beta_{1,s} : s \in A\} \cup \{\beta_{1,s} : s \in B \setminus A\}.$$

It might be helpful to have an idea of how the elements in $\mathcal{D}$ and $\mathcal{R}$ look like.

Elements in $\mathcal{D}$: T_s, $\tau_s \oplus \mathsf{bin}(\ell_s)$, $PP_{s,1}, \ldots, PP_{s,m_s}$.
Elements in $\mathcal{R}$: τ_s, $\beta_{1,s}$, $CC_{s,1}, \ldots, CC_{s,m_s}$.

For the ensuing collision analysis, we consider the elements in $\mathcal{D}$ and $\mathcal{R}$ to be formal variables or names which are assigned values in the Game NON. Then, we bound the probability that two different names are assigned the same value in the game.

Collision Analysis: The purpose of this analysis is to show that the probability of a repetition in $\mathcal{D}$ or $\mathcal{R}$ is small.

We first consider the probability that two elements in $\mathcal{D}$ have the same value. Let $1 \le s, t \le q$ and $1 \le i \le m_s$, $1 \le j \le m_t$ with $(s, i) \ne (t, j)$. Further, let $\delta = \Pr[PP_{s,i} = PP_{t,j}]$. We bound the value of δ. For the analysis, we recall that by Theorem 1, $\Psi[\psi]$ is $(1/2^n, \epsilon)$-BAU where $\epsilon = (m-1)/2^n$ if $\psi = \mathsf{Poly}$ and $\epsilon = 2m/2^n$ if $\psi = \mathsf{BRW}$. Here m is the number of blocks and $\ell = nm$. Since queries may have different number of blocks, we will write ϵ_s corresponding to m_s.

There are several cases.

$s = t$: Then $i \ne j$ and from Theorem 1, $\delta = 1/2^n$.

$s \ne t$, $(T_s, \ell_s) \ne (T_t, \ell_t)$: Then the values of $\beta_{1,s}$ and $\beta_{1,t}$ are chosen randomly and the probability that they are equal is $1/2^n$. Using this we obtain $\delta = 1/2^n$.

$s \ne t$, $(T_s, \ell_s) = (T_t, \ell_t)$, $i \ne j$: In this case, the hashing keys τ_s and τ_t are equal and so are $\beta_{1,s}$ and $\beta_{1,t}$. However, since $i \ne j$, using Theorem 1, we have $\delta = 1/2^n$.

$s \ne t$, $(T_s, \ell_s) = (T_t, \ell_t)$, $i = j$: Again, the hashing keys are equal, and using Theorem 1, we have $\delta \le \epsilon_s = \epsilon_t$. We call a pair $(PP_{s,i}, PP_{t,j})$ corresponding to such a case to be a special pair.

Total probability of a collision in $\mathcal{D}$: To perform this analysis, we divide the queries into groups where all queries in one group have the same length and the same tweak. Suppose there are p such groups with group k ($1 \le k \le p$) having n_k queries each of length l_k. (Note that two l_ks can be equal, since queries with same length but different tweak are placed in different groups.) Then $\sum_{s=1}^{q} m_s = \sum_{k=1}^{p} l_k n_k$.

It is helpful to consider the PPs in the k^{th} query group to be organised as an $n_k \times l_k$ matrix. Special pairs are formed by choosing two elements from the *same* column of one of these matrices. So, the number of special pairs is $\sum_{i=1}^{p} \binom{n_i}{2} l_i$. For finding the total probability of a collision, we will require the specific value of ϵ_s, i.e., whether it is $(m_s - 1)/2^n$ (if $\psi = \mathsf{Poly}$) or whether it is $2m_s/2^n$ (if $\psi = \mathsf{BRW}$).

First suppose $\psi = \mathsf{Poly}$. The total probability of a collision due to a special pair is at most

$$\sum_{k=1}^{p} \binom{n_k}{2} l_k \epsilon_k = \sum_{k=1}^{p} \binom{n_k}{2} l_k (l_k - 1) \le \sum_{k=1}^{p} \binom{n_k}{2} l_k^2 \le \sum_{k=1}^{p} (n_k l_k)^2 \le \left(\sum_{k=1}^{p} n_k l_k \right)^2 = \left(\sum_{s=1}^{q} m_s \right)^2 \le \sigma_n^2.$$

If $\psi = \mathsf{BRW}$, then this bound is $4\sigma_n^2/2^n$. The total number of all other pairs (including pairs with one or both element as T_s or $\tau_s \oplus \mathsf{bin}_n(\ell_s)$) is clearly bounded above by $\binom{|\mathcal{D}|}{2}$. It is easy to verify that the probability of a collision for any such pair is $1/2^n$ and hence the total collision probability of non-special pairs is at most $\binom{|\mathcal{D}|}{2}/2^n \leq \sigma_n^2/2^n$.

Combining the two probabilities, the total probability of a domain collision is at most $2\sigma_n^2/2^n$ if $\psi = \mathsf{Poly}$ and is at most $5\sigma_n^2/2^n$ if $\psi = \mathsf{BRW}$.

A similar argument shows the same bound for range collisions, so that the total probability of a domain or a range collision is at most $4\sigma_n^2/2^n$ if $\psi = \mathsf{Poly}$ and is at most $10\sigma_n^2/2^n$ if $\psi = \mathsf{BRW}$.

Case of HEHp**:** In this case, the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ are same as before. Also, the definition of the sets A and B remain the same. The difference is in the definition of the (multi-)sets $\mathcal{D}$ and $\mathcal{R}$. Here, $\mathcal{D}$ consists of elements (with repetition) of $\mathcal{L}_1$ along with the elements (with repetition) in the following (multi-)set: $\{T_s, \gamma_s \oplus (\mathsf{bin}_n(\ell_s)) : s \in A\} \cup \{\gamma_s \oplus (\mathsf{bin}_n(\ell_s)) : s \in B \setminus A\}$.

Similarly, $\mathcal{R}$ consists of elements (with repetition) of $\mathcal{L}_2$ along with the elements (with repetition) in the following (multi-)set: $\{\gamma_s, \beta_{1,s} : s \in A\} \cup \{\beta_{1,s} : s \in B \setminus A\}$. The element τ does not depend on the message and is randomly chosen once for all at the start of the game. The rest of the collision analysis remains the same.

Case of HEHfp**:** In this case also, the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ are same as for HEH. The definition of the set A is also the same and since all messages have the same length, the set B is not required. Multi-sets $\mathcal{D}$ and $\mathcal{R}$ are defined as follows. $\mathcal{D}$ consists of elements (with repetition) of $\mathcal{L}_1$ along with the elements (with repetition) in the following (multi-)set: $\{T_s : s \in A\}$.

$\mathcal{R}$ consists of elements (with repetition) of $\mathcal{L}_2$ along with the elements (with repetition) in the following (multi-)set: $\{\beta_{1,s} : s \in A\}$. Also, as in HEHp, the element τ is randomly chosen once for all at the start of the game. Again, the rest of the collision analysis remains the same. □

B Proof of HEH*

The basic structure of the proof of HEH* is the same as that of HEH. After a few initial games, we consider a game NON and then perform an analysis of the probability of a repetition in the multi-sets $\mathcal{D}$ and $\mathcal{R}$. The actual games for HEH* are slightly different from that of HEH. The main difference occurs in obtaining NON from RAND2. To explain this, we start from Game RAND2 for HEH* which is shown in Table 19. Key definition remains the same as that for HEH. For HEH*, obtaining Game RAND2 itself from previous games is quite similar to that for HEH and hence we do not present the details.

In NON, we would like to do away with the adversary and concentrate only on a fixed transcript which specifies both the plaintext and ciphertext blocks. For HEH, we defined *allowed* transcripts to be those which did not contain any *pointless* queries. However, for the present case, the notion of allowed transcripts needs to be defined differently. This is because preventing pointless queries is not enough for the probability analysis to be successful. The notion of allowed transcript has to be more restrictive for a proper probability analysis. This complication is due to the presence of the elements W_s in $\mathcal{D}$ and the presence of the elements V_s in $\mathcal{R}$.

For HEH, the randomness of the τs and βs were enough to ensure that collisions were rare. The randomness of the $C_{s,i}$s (for encryption queries) and of the $P_{s,i}$s (for decryption queries) were not required at all. In the present case, this is not enough to ensure that the Ws and Vs do not collide. We require to use the randomness of the $C_{s,i}$s and $P_{s,i}$s to ensure this. This is taken care of by carefully defining the notion of *allowed* transcripts.

Table 19. Game RAND2 for HEH*. In this case, for the s^{th} query $\ell_s = nm_s + r_s$, with $0 \leq r_s \leq n-1$. The length of the last block is r_s bits.

Respond to the s^{th} adversarial query as follows:
ENCRYPT QUERY $\mathsf{Enc}(T_s; P_{s,1}, P_{s,2}, \ldots, P_{s,m_s}, P_{s,m_s+1})$
$ty_s = \mathsf{Enc}$;
$(C_{s,1}, C_{s,2}, \ldots, C_{s,m_s}, C_{s,m_s+1}) \xleftarrow{\$} \{0,1\}^{\ell_s}$;
return $(C_{s,1}, C_{s,2}, \ldots, C_{s,m_s}, C_{s,m_s+1})$;
DECRYPT QUERY $\mathsf{Dec}(T_s; C_{s,1}, C_{s,2}, \ldots, C_{s,m_s}, C_{s,m_s+1})$
$ty^s = \mathsf{Dec}$;
$(P_{s,1}, P_{s,2}, \ldots, P_{s,m_s}, P_{s,m_s+1}) \xleftarrow{\$} \{0,1\}^{\ell_s}$;
return $(P_{s,1}, P_{s,2}, \ldots, P_{s,m_s}, P_{s,m_s+1})$;

Finalization

First Phase:
For the s^{th} query do the following.
1. Let T_s be the tweak for the s^{th} query.
2. If KeyDef1 is used, then do $\mathsf{RAND2-Sub1}(T_s, \ell_s)$ in Table 18.
3. If KeyDef2 is used, then do $\mathsf{RAND2-Sub2}(T_s, \ell_s)$ in Table 18.
4. If KeyDef3 is used, then do $\mathsf{RAND2-Sub3}(T_s)$ in Table 18.
5. Case ($|P_{s,m_s+1}| = 0$):
6. $(PP_{s,1}, \ldots, PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s})$;
7. $(CC_{s,1}, \ldots, CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s})$;
8. **for** $i = 1$ **to** m_s **do**
9. $\mathcal{D} = \mathcal{D} \cup \{PP_{s,i}\}$; $\mathcal{R} = \mathcal{R} \cup \{CC_{s,i}\}$;
10. **end for**;
11. Case ($1 \leq |P_{s,m_s+1}| < n$):
12. $(PP_{s,1}, \ldots, PP_{s,m_s}, P_{s,m_s+1}) = \Phi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s}, P_{s,m_s+1})$;
13. $(CC_{s,1}, \ldots, CC_{s,m_s}, C_{s,m_s+1}) = \Phi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s}, C_{s,m_s+1})$;
14. **for** $i = 1$ **to** m_s **do**
15. $\mathcal{D} = \mathcal{D} \cup \{PP_{s,i}\}$; $\mathcal{R} = \mathcal{R} \cup \{CC_{s,i}\}$;
16. **end for**;
17. Let R_s be a random string of length $(n - r_s)$;
18. Let $W_s = PP_{s,m_s} \oplus CC_{s,m_s}$ and $V_s = (C_{s,m_s+1} \oplus P_{s,m_s+1})||R_s$;
19. $\mathcal{D} = \mathcal{D} \cup \{W_s\}$; $\mathcal{R} = \mathcal{R} \cup \{V_s\}$.

Second Phase:
if (some value occurs more than once in $\mathcal{D}$) **then** bad = true **endif**;
if (some value occurs more than once in $\mathcal{R}$) **then** bad = true **endif**.

Each of the elements V_s in Game RAND2 is a random n-bit string. This is because R_s is a random $(n - r_s)$-bit string and in an encrypt query C_{s,m_s+1} is a random r_s-bit string while in a decrypt query P_{s,m_s+1} is a random r_s-bit string. Let $\mathbf{E}_1$ be the event that for some $s \neq t$, $V_s = V_t$. Then $\mathsf{Pr}[\mathbf{E}_1] = \binom{q}{2}/2^n$.

Now, we recall the definition of Φ. From (16),

$$\Phi_{\tau,\beta}(X_1, \ldots, X_m, X_{m+1}) = (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y, X_{m+1}) \oplus \mathbf{f}$$

where $\mathbf{f} = (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta, 0^r)$ and $Y = X_m \oplus \tau\psi_\tau(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r})$. So,

$$PP_{s,m_s} = \beta_{1,s} \oplus P_{s,m_s} \oplus \tau_s\psi_{\tau_s}(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s+1}||0^{n-r_s})$$

and

$$CC_{s,m_s} = \beta_{2,s} \oplus C_{s,m_s} \oplus \tau_s\psi_{\tau_s}(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s+1}||0^{n-r_s}).$$

Hence,

$$\begin{aligned} W_s = (\beta_{1,s} \oplus \beta_{2,s}) \oplus (P_{s,m_s} \oplus C_{s,m_s}) \oplus \\ \tau_s(\psi_{\tau_s}(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s+1}||0^{n-r_s}) \oplus \psi_{\tau_s}(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s+1}||0^{n-r_s})). \end{aligned}$$

The map $\psi_\tau(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r})$ is either

$$\mathsf{Poly}_\tau(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r}) \text{ or } \mathsf{BRW}_\tau(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r}).$$

Both of these are polynomials in τ though of different degrees. The common feature, however, is that both of these define an injective map from $(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r})$ to the coefficients of the corresponding polynomial. By, $\mu = \mathsf{coe} \circ \psi$ we denote the map from $(X_1, \ldots, X_{m-1}, X_{m+1}||0^{n-r})$ to these coefficients. Then, μ is an injective map and we write

$$\begin{aligned} \mu(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s+1}||0^{n-r_s}) &= (U_{s,1}, \ldots, U_{s,k_s}) \\ \mu(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s+1}||0^{n-r_s}) &= (V_{s,1}, \ldots, V_{s,k_s}) \end{aligned}$$

If $\psi = \mathsf{Poly}$, then $k_s = m_s$ and if $\psi = \mathsf{BRW}$, then k_s is obtained from (6). For $\psi = \mathsf{BRW}$, we will not worry about the actual value of k_s; it is enough to know that $k_s < 2m_s$. For $1 \leq i \leq k_s$, let $W_{s,i} = U_{s,i} \oplus V_{s,i}$ and $Z_{s,k_s+1} = P_{s,m_s} \oplus C_{s,m_s}$. Now we can write

$$W_s = (\beta_{1,s} \oplus \beta_{2,s}) \oplus \mathsf{Poly}_{\tau_s}(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1}).$$

Since for each query, one of $(P_{s,1}, \ldots, P_{s,m_s}, P_{s,m_s+1})$ or $(C_{s,1}, \ldots, C_{s,m_s}, C_{s,m_s+1})$ is a random ℓ_s-bit string, and μ is an injective map, it follows that the entropy of $(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1})$ is ℓ_s. (Consequently, the entropy of the polynomial $\mathsf{Poly}_{\tau_s}(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1})$ is also ℓ_s.)

We divide the queries into groups where queries in each group have the same length and the same tweak, so that $\ell_s = \ell_t$, $m_s = m_t$ and consequently $k_s = k_t$. (Also, $\tau_s = \tau_t$ and $\beta_{1,s} = \beta_{1,t}$.) Let $\mathbf{E}_2$ be the event that $(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1}) = (W_{t,1}, \ldots, W_{t,k_t}, Z_{t,k_t+1})$ for some s, t corresponding to one query group. If we fix one such pair s, t, we have the probability $(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1}) = (W_{t,1}, \ldots, W_{t,k_t}, Z_{t,k_t+1})$ to be $1/2^{\ell_s}$. Suppose the distinct lengths of the queries are $l_1, \ldots, l_p$ and that there are n_k queries of length l_k. Then

$$\mathsf{Pr}[\mathbf{E}_2] \leq \binom{n_1}{2}\frac{1}{2^{l_1}} + \cdots + \binom{n_p}{2}\frac{1}{2^{l_p}}$$

$$\leq \frac{1}{2^n}\left(\binom{n_1}{2}+\cdots+\binom{n_p}{2}\right)$$
$$\leq \frac{1}{2^n}\binom{n_1+\cdots+n_p}{2}.$$

Let us consider the event $\overline{\mathbf{E}_2}$. For any transcript falling withing $\overline{\mathbf{E}_2}$, the following holds. If s and t are query indexes for one query group, then $(W_{s,1},\ldots,W_{s,k_s},Z_{s,k_s+1}) \neq (W_{t,1},\ldots,W_{t,k_t},Z_{t,k_t+1})$ and so the polynomial

$$\begin{aligned}&\mathsf{Poly}_{\tau_s}(W_{s,1}\oplus W_{t,1},\ldots,W_{s,k_s}\oplus W_{t,k_t},Z_{s,k_s+1}\oplus Z_{t,k_t+1})\\ &\quad=\mathsf{Poly}_{\tau_s}(W_{s,1},\ldots,W_{s,k_s},Z_{s,k_s+1})\oplus\mathsf{Poly}_{\tau_t}(W_{t,1},\ldots,W_{t,k_t},Z_{t,k_t+1})\end{aligned}$$

is a non-zero polynomial in τ_s. One can then talk about the roots of this polynomial and upper bound the probability that the randomly chosen τ_s is one of the roots. This is required in Game NON.

Let $\mathbf{E}$ be the event that either $\mathbf{E}_1$ or $\mathbf{E}_2$ occurs. Then

$$\begin{aligned}\mathsf{Pr}[\mathbf{E}] &\leq \mathsf{Pr}[\mathbf{E}_1]+\mathsf{Pr}[\mathbf{E}_2]\\ &\leq \frac{1}{2^n}\left(\binom{q}{2}+\binom{n_1+\cdots+n_p}{2}\right)\\ &\leq \sigma_n^2/2^n.\end{aligned}$$

Now we have

$$\begin{aligned}\Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}] &= \Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}\wedge(\mathbf{E}\vee\overline{\mathbf{E}})]\\ &= \Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}\wedge\mathbf{E}]+\Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}\wedge\overline{\mathbf{E}}]\\ &= \Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}|\mathbf{E}]\Pr[\mathbf{E}]+\Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}|\overline{\mathbf{E}}]\Pr[\overline{\mathbf{E}}]\\ &\leq \Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}|\overline{\mathbf{E}}]\Pr[\overline{\mathbf{E}}]+\Pr[\mathbf{E}]\\ &\leq \Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}|\overline{\mathbf{E}}]+\frac{\sigma_n^2}{2^n}.\end{aligned}$$

So, we have to analyze $\Pr[A^{\mathrm{RAND2}}\text{ sets }\mathsf{bad}|\overline{\mathbf{E}}]$.

At this point, we move from Game RAND2 to Game NON by defining allowed transcripts. The definition of allowed transcripts is that there should not be any pointless queries and that none of $\mathbf{E}_1$ or $\mathbf{E}_2$ occur (i.e., $\overline{\mathbf{E}}$ should occur). The rest of the combinatorial analysis is similar to that of HEH. We provide a sketch of the rest of the argument. The forms of the elements in $\mathcal{D}$ and $\mathcal{R}$ are the following.

Elements in $\mathcal{D}$:	T_s, $\tau_s\oplus\mathsf{bin}(\ell_s)$, $PP_{s,1},\ldots,PP_{s,m_s}$, if $\lvert P_{s,m_s+1}\rvert=0$; $PP_{s,1},\ldots,PP_{s,m_s}$, $PP_{s,m_s}\oplus CC_{s,m_s}$, if $1\leq\lvert P_{s,m_s+1}\rvert\leq n-1$.
Elements in $\mathcal{R}$:	τ_s, $\beta_{1,s}$, $CC_{s,1},\ldots,CC_{s,m_s}$, if $\lvert P_{s,m_s+1}\rvert=0$; $CC_{s,1},\ldots,CC_{s,m_s}$, $(P_{s,m_s+1}\oplus C_{s,m_s+1})\lvert\rvert R_s$, if $1\leq\lvert P_{s,m_s+1}\rvert\leq n-1$.

As in the case of HEH, we divide queries into groups, where queries in one group have the same length and the same tweak. Again, as before, two elements in $\mathcal{D}$ (or $\mathcal{R}$) which come from two

different groups are equal with probability $1/2^n$. Suppose s and t are query indexes corresponding to two different query groups. If $i \neq j$, then $PP_{s,i}$ is equal to $PP_{t,j}$ with probability $1/2^n$ and if $i = j$, then $PP_{s,i}$ is equal to $PP_{t,j}$ with probability ϵ_s, where $\epsilon_s \leq m_s/2^n$ if $\psi = \mathsf{Poly}$ and $\epsilon_s \leq 2m_s/2^n$ if $\psi = \mathsf{BRW}$. This much has already been done for HEH.

The new element is W_s. We need to consider possible collision of W_s with another W_t or with another type of element. For the second case, it is easy to see that the collision probability is $1/2^n$. For the first case, if s and t belong to different query groups, then again the collision probability is $1/2^n$ (due to the independent randomness of $\beta_{1,s}$ and $\beta_{1,t}$). So, suppose that s and t correspond to queries in the same query group. Recall that in Game NON, we are working with allowed transcripts and so event $\overline{\mathbf{E}_2}$ holds. This means that $\mathsf{Poly}_{\tau_s}(W_{s,1} \oplus W_{t,1}, \ldots, W_{s,k_s} \oplus W_{t,k_t}, Z_{s,k_s+1} \oplus Z_{t,k_t+1})$ is a non-zero polynomial in τ_s. Also, note

$$\begin{aligned} W_s \oplus W_t &= (\beta_{1,s} \oplus \beta_{2,s}) \oplus \mathsf{Poly}_{\tau_s}(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1}) \\ &\quad \oplus \\ &\quad (\beta_{1,t} \oplus \beta_{2,t}) \oplus \mathsf{Poly}_{\tau_t}(W_{t,1}, \ldots, W_{t,k_t}, Z_{t,k_t+1}) \\ &= \mathsf{Poly}_{\tau_s}(W_{s,1}, \ldots, W_{s,k_s}, Z_{s,k_s+1}) \oplus \mathsf{Poly}_{\tau_t}(W_{t,1}, \ldots, W_{t,k_t}, Z_{t,k_t+1}) \\ &= \mathsf{Poly}_{\tau_s}(W_{s,1} \oplus W_{t,1}, \ldots, W_{s,k_s} \oplus W_{t,k_t}, Z_{s,k_s+1} \oplus Z_{t,k_t+1}). \end{aligned}$$

Thus, the probability that W_s is equal to W_t is equal to the probability that τ_s is a root of this non-zero polynomial. The rest of the argument is similar to the case of HEH. □

C Security of iHCH

The security reduction for iHCH is similar to that of HCH with two important differences. The first concerns elimination of a block cipher call before initializing the counter mode of operation, while the second concerns the use of BRW polynomial. We elaborate on these aspects below.

Initialization of the counter mode: In HCH, the counter mode is initialized with the encryption of $M_1 \oplus U_1$. It has been mentioned in HCH that this is required to obtain a quadratic security bound. We briefly revisit that argument. In the final probability analysis of collisions, the quantity $M_1 \oplus U_1$ is expressed as a polynomial in the $(P_i \oplus C_i)$s. Without the encryption, for each query small variations of these polynomials will be encrypted. If all queries are of same number of blocks m and have the same tweak, then in the collision analysis, this will mean that we will have to consider approximately XORs of σ_n^2 pairs of polynomials where the probability that τ is a root of one of these XORs is approximately m. This leads approximately to a bound of $m\sigma_n^2/2^n$ for the collision analysis. The way to avoid this in HCH was to encrypt $U_1 \oplus M_1$ before initializing the counter mode.

In the present case, however, we do away with the encryption. We still claim that the quadratic security bound holds. The reason is the manner in which the counter mode itself has been defined. In the game sequence analysis, we move to Game NON where we finally have to consider collisions in possible (multi-)sets $\mathcal{D}$ and $\mathcal{R}$. The form of these are as follows.

$$\begin{aligned} \text{Elements in } \mathcal{D}: \quad & T_s,\ \tau_s \oplus \mathsf{bin}(\ell_s), \\ & M_{1,s} = H_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s}||0^{n-r_s}), \\ & S_s \oplus \beta_{1,s}, S_s \oplus \alpha\beta_{1,s}, \ldots, S_s \oplus \alpha^{m_s-2}\beta_{1,s}. \\ \text{Elements in } \mathcal{R}: \quad & \tau_s,\ \beta_{1,s}, \\ & U_{1,s} = H_{\tau_s,\beta_{1,s}}(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s}||0^{n-r_s}), \\ & P_{s,2} \oplus C_{s,2}, \ldots, P_{s,m_s-1} \oplus C_{s,m_s-1}, (P_{s,m_s} \oplus C_{s,m_s})||-^{n-r_s}. \end{aligned}$$

Here

$$\begin{aligned}S_s &= M_{1,s} \oplus U_{1,s} \oplus (\beta_{1,s} \oplus \beta_{2,s}) \\ &= H_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s}||0^{n-r_s}) \oplus H_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s}||0^{n-r_s}) \\ &= (P_{s,1} \oplus C_{s,1}) \oplus \tau_s(\psi_{\tau_s}(P_{s,2}, \ldots, P_{s,m_s-1}, P_{s,m_s}||0^{n-r_s}) \oplus \psi_{\tau_s}(C_{s,2}, \ldots, C_{s,m_s-1}, C_{s,m_s}||0^{n-r_s})).\end{aligned}$$

Consider the s^{th} and the t^{th} queries and the i^{th} and j^{th} blocks in these queries respectively. Then we have to consider $Z_s = W_s \oplus \alpha^i\beta_{1,s}$ and $Z_t = W_t \oplus \alpha^j\beta_{1,t}$, where W is a polynomial in τ defined from the XORs of the Ps and Cs. If either the length or the tweak is different, then the independent randomness of the $\beta_{1,s}$ and $\beta_{1,t}$ ensure that $Z_s = Z_t$ occurs with probability $1/2^n$. Even if the lengths and tweaks are equal, if $i \neq j$, then $Z_s \oplus Z_t = (\alpha^i \oplus \alpha^j)\beta_{1,s} \oplus W_s \oplus W_t$ and again the probability that $Z_s = Z_t$ occur with probability $1/2^n$. Thus, the only situation when we need to use the "root of polynomial" argument is when the lengths and tweaks are equal and $i = j$. This makes it possible to bound the total probability of collision by a constant multiple of $\sigma_n^2/2^n$.

Use of BRW polynomials. This was not present in the analysis of HCH. These polynomials make the security analysis a little more complicated. In the proof of HCH, an *allowed* transcript was defined by ensuring that the $(P_i \oplus C_i)$ were all distinct. This, however, is not sufficient when we are considering BRW polynomials. The notion of allowed transcript needs to be defined differently. In fact, this notion is similar to that used in the proof of HEH*. We define the event $\mathbf{E}_2$ as in the proof of HEH*. The event $\mathbf{E}_1$ is different. Here we require all the n-bit strings $P_{s,i} \oplus C_{s,i}$s (including the partial block, if present) to be distinct. Event $\mathbf{E}$ is again the union of $\mathbf{E}_1$ and $\mathbf{E}_2$. The rest of the argument and analysis is similar to that of HEH*. The proofs of iHCHp and iHCHfp are also similar.

D Security of HOH

In this case, the form of the elements in $\mathcal{D}$ and $\mathcal{R}$ are as follows.

$$\begin{array}{ll}\text{Elements in } \mathcal{D}: & T_s,\ \tau_s \oplus \mathsf{bin}(\ell_s), \\ & M_{1,s} = H_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s-1}, P_{s,m_s}||0^{n-r_s}), \\ & S_s = M_{1,s} \oplus U_{1,s}, S_{s,1}, \ldots, S_{s,m_s-2}. \\ \text{Elements in } \mathcal{R}: & \tau_s,\ \beta_{1,s}, \\ & U_{1,s} = H_{\tau_s,\beta_{1,s}}(C_{s,1}, \ldots, C_{s,m_s-1}, C_{s,m_s}||0^{n-r_s}), \\ & S_{s,1}, \ldots, S_{s,m_s-1}.\end{array}$$

The notion of allowed transcript in this case is simpler. We only need to consider event $\mathbf{E}_2$, i.e., $\mathbf{E} = \mathbf{E}_2$. The collision analysis for HOH, HOHp and HOHfp are similar to that of the other modes of operations.