Improving Upon the TET Mode of Operation

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

Abstract. Naor and Reingold had proposed the construction of a strong pseudo-random permutation (SPRP) by using a layer of ECB encryption between two layers of invertible block-wise universal hash functions. At Crypto 2007, Halevi presented constructions of invertible block-wise universal hash functions and a new mode of operation (called TET) based on them. In this paper, we present a new mode of operation called HEH using the Naor-Reingold approach. This is built using a new construction of invertible block-wise universal hash function. The new construction improves over Halevi's construction by removing restrictions on the hashing key. This in turn, leads to HEH improving over TET by allowing more efficient encryption and decryption of variable length messages as well as supporting better key agility. For the important application of disk encryption, we present a variant called HEHfp which has better key agility than TET.
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) [8]. 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 introduced by Liskov-Rivest-Wagner [7]. This notion was extended to variable input length tweakable SPRP by Halevi-Rogaway [5]. Earlier, a method for constructing SPRPs was given by Naor-Reingold. An important application of tweakable SPRP is that of disk encryption as has been pointed out in [5]. Currently, the literature contains several constructions of tweakable SPRPs. These constructions can be classified into three main groups.

The first type of construction consists of using a layer of ECB encryption between two invertible block-wise universal hashing layers. This method was introduced by Naor-Reingold [11, 10]. Recently, there has been an interest in this type of constructions and the proposals PEP [2] and TET [4] are of this type. The second type consists of using a counter mode of encryption between two layers of universal hash function computation. This idea was introduced in XCB [9] and later constructions are HCTR [13] and HCH [1]. The third type of construction is to use a mixing layer between two layers of encryption. This technique was introduced by Halevi-Rogaway [5] and the constructions CMC [5], EME [6] and EME* [3] are of this type.

Our Contributions: TET is a very recent construction which follows the Naor-Reingold hash-ECB-hash approach. For fixed length messages, TET has good performance. It, however, has two drawbacks. First, it is not suited for variable length messages and second, the key agility of TET is not good, in the sense that a lot of computation needs to be done for every key change.

The purpose of the current work is to propose a new construction of tweakable SPRP following the Naor-Reingold approach. We call this HEH. The new construction removes the above mentioned drawbacks of TET, while retaining its performance. HEH is well suited for the special application of disk encryption.

As mentioned earlier, the Naor-Reingold approach is to use a layer of ECB encryption between two layers of invertible block-wise universal hash functions. TET uses this approach. The main novelty in TET is to design an invertible block-wise universal hash function. It is shown that both the hash function *and* its inverse are block-wise universal. The universal hash function defined in [4] has a drawback which in turn leads to the earlier mentioned drawbacks of TET. When m blocks are to be hashed, the hashing key τ has to satisfy the condition that $1 + \tau + \cdots + \tau^{m-1} \neq 0$.

In this paper, we design a new invertible block-wise universal hash function. But, in our case, the inverse is not block-wise invertible. Importantly, this does not matter in the design of SPRP. It is sufficient to place the ECB layer between the hash function and its inverse. In fact, this has already been done by Naor-Reingold [10]. An important advantage of the new hash function over the one in [4] is that there are no restrictions on the hashing key. It is this feature which ultimately allows HEH to improve over TET.

2 Invertible Block-Wise Universal Hash Function

Let $\mathbb{F}$ be a finite field. Additions and multiplications are done over this field.

The notion of block-wise universal hash function is defined for a keyed family of functions. Fix a positive integer m. Let $\mathcal{F} : \mathcal{K} \times \mathbb{F}^m \to \mathbb{F}^m$ be a keyed family of functions where $\mathcal{K}$ is the key space. The family $\mathcal{F}$ is said to be ϵ-block-wise universal if $\Pr_K[Y_i = Y'_{i'}] \leq \epsilon$, where $(Y_1, \ldots, Y_m) = \mathcal{F}_K(\mathbf{x})$, $(Y'_1, \ldots, Y'_1) = \mathcal{F}_K(\mathbf{x}')$, $1 \leq i, i' \leq m$ and $(\mathbf{x}, i) \neq (\mathbf{x}', i')$. We are interested in invertible block-wise universal hash functions, i.e., $\mathcal{F}_K()$ should be invertible for each $K \in \mathcal{K}$.

2.1 Block-wise Polynomial Evaluation [4]

In this section, we describe the constructions given in [4]. 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 $\sigma = 1 + \tau + \tau^2 + \cdots + \tau^{m-1}$. 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:

$$(X_1, \ldots, X_m) \mapsto (X_1 + R, \ldots, X_m + R) \tag{1}$$

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 $\mathbb{F}^n$ to $\mathbb{F}^n$ are defined in the following manner.

$$\begin{aligned} \mathrm{BPE}_{\tau,\beta}(\mathbf{x}) &= M_\tau \mathbf{x}^T + \mathbf{b} \quad \text{and} \quad \mathrm{BPE}^{-1}_{\tau,\beta}(\mathbf{x}) = M_\tau^{-1}(\mathbf{x} - \mathbf{b})^T \\ \widetilde{\mathrm{BPE}}_{\tau,\beta}(\mathbf{x}) &= M_\tau(\mathbf{x} - \mathbf{b})^T \quad \text{and} \quad \widetilde{\mathrm{BPE}}^{-1}_{\tau,\beta}(\mathbf{x}) = M_\tau^{-1}\mathbf{x}^T + \mathbf{b} \end{aligned} \tag{2}$$

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 [12].

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 [4], that the functions defined by (2) 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 $\sigma = \sum_{i=1}^{m} \tau^{i-1}$ is non-zero. This means that τ cannot be an arbitrary element of $\mathbb{F}$. Further, computing the inverse of BPE and $\widetilde{\mathrm{BPE}}$ requires the inverse of σ.

2.2 A New Construction

Fix a positive integer m and a primitive element α of $\mathbb{F}$. Let τ and β be independent and random elements of $\mathbb{F}$. Define $\mathbf{e} = (\alpha\beta, \alpha^2\beta, \ldots, \alpha^{m-1}\beta, \beta)$. As mentioned earlier, for a proper choice of the primitive element α, multiplication by α is very fast and the cost is negligible compared to a general multiplication over $\mathbb{F}$. We define the map $\Psi_{\tau,\beta} : \mathbb{F}^m \to \mathbb{F}^m$ in the following manner.

$$\Psi_{\tau,\beta}(X_1, \ldots, X_m) = (X_1 + Y, \ldots, X_{m-1} + Y, Y) + \mathbf{e} \tag{3}$$

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

Invertibility is easily seen as follows. Let $(Y_1, \ldots, Y_m) = \Psi_{\tau,\beta}(X_1, \ldots, X_m)$. Set $(U_1, \ldots, U_m) = (Y_1, \ldots, Y_m) - \mathbf{e}$. Then $X_i = U_i - U_m$ for $1 \leq i \leq m-1$ and $X_m = U_m - \tau(\sum_{i=1}^{m-1} X_i \tau^{m-1-i})$. Using Horner's rule, computing either $\Psi_{\tau,\beta}$ or its inverse $\Psi_{\tau,\beta}^{-1}$ requires $(m-1)$ multiplications.

Examples: For $m = 4$, we provide the outputs of $\mathrm{BPE}_{\tau,\beta}$ and $\Psi_{\tau,\beta}$ to illustrate the difference between the two functions.

$$\begin{aligned}
\mathrm{BPE}_{\tau,\beta}(X_1, X_2, X_3, X_4) = (&X_1 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \beta,\\
&X_2 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha\beta,\\
&X_3 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha^2\beta,\\
&X_4 + X_1\tau + X_2\tau^2 + X_3\tau^3 + X_4\tau^4 + \alpha^3\beta)\\
\Psi_{\tau,\beta}(X_1, X_2, X_3, X_4) = (&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_1 + \alpha\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_2 + \alpha^2\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + X_3 + \alpha^3\beta,\\
&X_1\tau^3 + X_2\tau^2 + X_3\tau + X_4 + \beta)
\end{aligned}$$

The order of evaluation in BPE and Ψ 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 [4] 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 following result establishes the block-wise universality of Ψ and its proof is based on the standard argument over roots of polynomials.

Theorem 1. *Fix a positive integer m and let τ and β be independent and random elements of $\mathbb{F}$. Let $(Y_1, \ldots, Y_m) = \Psi_{\tau,\beta}(X_1, \ldots, X_m)$, $(Y'_1, \ldots, Y'_m) = \Psi_{\tau,\beta}(X'_1, \ldots, X'_m)$, $1 \leq i, i' \leq m$ and $((X_1, \ldots, X_m), i) \neq ((X'_1, \ldots, X'_m), i')$.*

1. *If $i \neq i'$, then $\Pr_{\tau,\beta}[Y_i = Y'_{i'}] = \frac{1}{|\mathbb{F}|}$.*
2. *If $i = i'$, then $\Pr_{\tau,\beta}[Y_i = Y'_i] \leq \frac{m-1}{|\mathbb{F}|}$.*

Consequently, the function $\Psi_{\tau,\beta}$ is $\left(\frac{m-1}{|\mathbb{F}|}\right)$-block-wise universal.

Proof : The two cases are proved separately.

Case $i \neq i'$: Without loss of generality, we assume that $1 \leq i < i' \leq m$. First suppose $i' < m$. From (3), $Y_i - Y'_{i'} = (\alpha^i - \alpha^{i'})\beta + R$, where R is a quantity which depends on τ and not on β. Since α is a primitive element of $\mathbb{F}$, we have $\alpha^i \neq \alpha^{i'}$ (for $m \leq 2^n - 2$). The event $Y_i = Y'_{i'}$ translates into the event $\beta = (\alpha^{i'} - \alpha^i)^{-1}R$. Since β is a random element of $GF(2^n)$ and is independent of the right hand side, the probability that this happens is $1/2^n$. If $i' = m$, then $Y_i - Y'_{i'} = (\alpha^i - 1)\beta + R$ and a similar argument holds.

Case $i = i'$: In this case, we necessarily have $(X_1, \ldots, X_m) \neq (X'_1, \ldots, X'_m)$. First suppose $i < m$. Then $Y_i - Y'_i = (X_i + Y) - (X'_i + Y')$ where $Y = \sum_{i=1}^{m} X_i\tau^{m-i}$ and $Y' = \sum_{i=1}^{m} X'_i\tau^{m-i}$. We have

$$X_i + Y = X_1\tau^{m-1} + X_2\tau^{m-2} + \cdots + X_{m-1}\tau + (X_m + X_i).$$

Let $(V_1, \ldots, V_m) = (X_1, \ldots, X_{m-1}, X_m + X_i)$. The map $(X_1, \ldots, X_m) \mapsto (V_1, \ldots, V_m)$ is a bijection and so $(X_1, \ldots, X_m) \neq (X'_1, \ldots, X'_m)$ implies $(V_1, \ldots, V_m) \neq (V'_1, \ldots, V'_m)$. As a consequence $(V_1 - V'_1, \ldots, V_m - V'_m) \neq (0, \ldots, 0)$. Now,

$$\begin{aligned} Y_i - Y'_i &= (X_i + Y) - (X'_i + Y') \\ &= (V_1 - V'_1)\tau^{m-1} + \cdots + (V_{m-1} - V'_{m-1})\tau + (V_m - V'_m). \end{aligned}$$

The last expression is a non-zero polynomial in τ and is zero if and only if τ is a root of this polynomial. Since τ is a random element of $\mathbb{F}$ and a polynomial of degree $(m-1)$ has at most $(m-1)$ distinct roots, we have $\Pr[Y_i = Y'_i] \leq (m-1)/|\mathbb{F}|$.

If $i = m$, then a similar argument holds. □

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. 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. We have the following extension of Theorem 1.

Theorem 2. *Fix an integer $m > 1$ and choose τ, β independently and uniformly at random from $\mathbb{F}$. Let $\mathbf{x}_i = (X_{i,1}, \ldots, X_{i,m})$, for $1 \leq i \leq q$ be a set of distinct tuples from $\mathbb{F}^m$. For $1 \leq i \leq q$, let $\mathbf{y}_i = \Psi_{\tau,\beta}(\mathbf{x}_i)$, where $\mathbf{y}_i = (Y_{i,1}, \ldots, Y_{i,m})$. The probability (over τ, β) that for any choice of $(i_1, j_1) \neq (i_2, j_2)$, Y_{i_1,j_1} is equal to Y_{i_2,j_2} is at most $\frac{2(qm)^2}{|\mathbb{F}|}$.*

Proof : Suppose $(i_1, j_1) \neq (i_2, j_2)$. There are two cases. First suppose that $j_1 = j_2 = j$. There are $m \times \binom{q}{2}$ such pairs. For any such pair, the probability that $Y_{i_1,j} = Y_{i_2,j}$ is at most $(m-1)/2^n$ (from Theorem 1(2)). Thus, the total probability of collisions among such pairs is at most $m \times \binom{q}{2} \times (m-1)/|\mathbb{F}| \leq (m^2q^2)/|\mathbb{F}|$. On the other hand, if $j_1 \neq j_2$, then the probability of $Y_{i_1,j_1} = Y_{i_2,j_2}$ is at most $1/|\mathbb{F}|$ (from Theorem 1(1)). There are $\binom{qm}{2} - m \times \binom{q}{2} \leq (qm)^2$ such pairs and the probability of a collision among such pairs is at most $(qm)^2/|\mathbb{F}|$. Thus, the total probability of a collision among the Ys is at most $2(qm)^2/|\mathbb{F}|$. □

3 The HEH Construction

For the description of the tweakable SPRP, we will consider 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 and the primitive element α is simply taken to be x. The polynomial $\rho(x)$ can be chosen to be a trinomial or a pentanomial, and hence multiplication by x modulo $\rho(x)$ can be done very efficiently. 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.

The basic structure of the HEH construction is shown in Figure 1. Pseudo-codes are given in Figure 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 Figure 3.

In HEH, the number of blocks m can vary; n-bit tweaks (associated data) are supported and only 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. If such 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.

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. Consequently, it is sufficient to take the tweak to be an n-bit string. 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.

3.1 Other Issues

We briefly consider several other issues in the design of a possible tweakable SPRP.

Arbitrary length messages. HEH and its variants defined so far can only handle messages which are multiples of block length n. It is possible to define a variant which can handle messages of any length greater than or equal to n. The technique for doing this is based on the technique used for EME* and has been used for TET. Actually, the inner layer of ECB mode is not particularly suited for handling partial blocks. This is better tackled using a counter mode of encryption, as for

Fig. 1. Encryption and decryption using HEH. The block cipher key is K; and the hash key is (τ, β_1, β_2). See Figure 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** $\mathbf{E}_{K,\tau,\beta_1,\beta_2}(P_1, \ldots, P_m)$
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** $\mathbf{D}_{K,\tau,\beta_1,\beta_2}(C_1, \ldots, C_m)$
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.     $(P_1, \ldots, P_m) = \Psi^{-1}_{\tau,\beta_1}(PP_1, \ldots, PP_m)$.
```

Fig. 2. Detailed pseudo-code of encryption and decryption using HEH.

```
**Algorithm** $\mathbf{E}_{K,\tau,\beta_1,\beta_2}(P_1, \ldots, P_m)$
1.  $U = P_1$;
2.  for $i = 2$ to $m$ do $U = U\tau \oplus P_i$;
3.  $Q = \beta_1$;
4.  for $i = 1$ to $m - 1$ do
5.      $Q = xQ$;
6.      $PP_i = P_i \oplus U \oplus Q$;
7.      $CC_i = E_K(PP_i)$;
8.  end do;
9.  $PP_m = U \oplus \beta_1$; $CC_m = E_K(PP_m)$;
10. $V = CC_m \oplus \beta_2$; $Q = \alpha\beta_2$;
11. $C_1 = CC_1 \oplus Q \oplus V$; $W = C_1$;
12. for $i = 2$ to $m - 1$ do
13.     $Q = \alpha Q$;
14.     $C_i = CC_i \oplus Q \oplus V$;
15.     $W = W\tau \oplus C_i$;
16. end do;
17. $C_m = V \oplus W\tau$;
**end.**
```

```
**Algorithm** $\mathbf{D}_{K,\tau,\beta_1,\beta_2}(C_1, \ldots, C_m)$
1.  $U = C_1$;
2.  for $i = 2$ to $m$ do $U = U\tau \oplus C_i$;
3.  $Q = \beta_2$;
4.  for $i = 1$ to $m - 1$ do
5.      $Q = xQ$;
6.      $CC_i = C_i \oplus U \oplus Q$;
7.      $PP_i = E_K^{-1}(CC_i)$;
8.  end do;
9.  $CC_m = U \oplus \beta_1$; $PP_m = E_K(CC_m)$;
10. $V = PP_m \oplus \beta_1$; $Q = \alpha\beta_1$;
11. $P_1 = PP_1 \oplus Q \oplus V$; $W = P_1$;
12. for $i = 2$ to $m - 1$ do
13.     $Q = \alpha Q$;
14.     $P_i = PP_i \oplus Q \oplus V$;
15.     $W = W\tau \oplus P_i$;
16. end do;
17. $P_m = V \oplus W\tau$;
**end.**
```

Fig. 3. HEH and its variants obtained by suitably defining the hashing keys τ, β_1 and β_2. T is an n-bit tweak and m is the number of blocks. K is a randomly chosen block cipher key and in HEHp and HEHfp, τ is a randomly chosen n-bit string.

HEH	HEHp	HEHfp
1. $\gamma = E_K(T)$; 2. $\beta_1 = E_K(\gamma \oplus \mathsf{bin}_n(m))$; 3. $\beta_2 = x\beta_1$; 4. $\tau = \gamma$.	1. $\gamma = E_K(T)$; 2. $\beta_1 = E_K(\gamma \oplus \mathsf{bin}_n(m))$; 3. $\beta_2 = x\beta_1$.	1. $\beta_1 = E_K(T)$; 2. $\beta_2 = x\beta_1$.

example in HCH. The technique for tackling partial blocks in EME* is essentially to have a counter mode type technique for the last block.

Arbitrary length tweaks. For applications which require arbitrary length tweaks, one can use a pseudo-random function (PRF) with a separate and independent key to produce an n-bit tweak which can then be used in HEH. The hashing key τ is chosen independently of the PRF key, which allows for the hashing and the processing of tweak to proceed in parallel. This approach has been used in TET and if desired, a similar approach can also be used with HEH.

4 Discussion and Comparison

HEH uses the hash-ECB-hash approach introduced by Naor-Reingold. An earlier construction using the same approach is TET. (We do not consider PEP, since it is slower than TET.) The difference between the two is mainly in the definition of the universal hash functions. Both BPE (used in TET) and Ψ (used in HEH) are invertible block-wise universal hash functions. The efficiencies of computing both functions as well as their inverses are the same. The differences are the following.

1. One main difference is that the inverse of BPE is block-wise universal while the inverse of Ψ is not. Importantly however, it is possible to construct a tweakable SPRP without requiring the inverse to be block-wise universal.
2. In BPE, the hash key τ has the restriction that $\sigma = \sum_{i=1}^{m} \tau^{i-1}$ must not be zero. This creates problems of key agility. For one thing, τ cannot be an arbitrary random element of $GF(2^n)$. It has been suggested in [4], 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. The other approach, and the one used in the construction of TET, is to repeatedly apply a PRF with a separate key K_1 to $(0, i)$, starting from $i = 1$, to obtain candidate values of τ, until one is obtained for which $\sigma \neq 0$. Since σ depends on m (and K_1), this procedure needs to be repeated for each m, which makes TET rather unsuitable for variable number of blocks. When m is fixed, this computation can be done off-line. However, the problem is only reduced and not eliminated. Whenever, a key change is required (i.e., the PRF key K_1 is changed), the entire procedure to generate τ needs to be performed. This adversely affects the key agility of TET.
3. The inversion of BPE requires the value of σ^{-1}. Computing this online is rather inefficient. When τ is computed off-line, this value can be pre-computed and stored. Also, even if it is computed off-line, it still has to be computed for every key change, again adversely affecting key agility. For hardware only implementation, this means that an inversion circuit has to be implemented which is substantially more expensive compared to a multiplication circuit.

Points (2) and (3) above are major drawbacks of TET. For one thing, it makes TET unsuitable for variable number of blocks and secondly, even for fixed number of blocks, the key agility is not good. The new construction, HEH, improves over TET in both these aspects.

There are three efficient constructions using the hash-counter-hash approach – XCB, HCTR and HCH. Among these constructions, HCH has a quadratic security bound; HCTR has a cubic bound while no bound is known for XCB. HEH has efficiency similar to these constructions; a quadratic security bound; and also has similar key agility. Thus, HEH shows that it is possible to use the hash-ECB-hash approach to obtain a construction which is as good as hash-counter-hash approach.

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. If one block cipher call takes more time than two multiplications, then the hash-encrypt-hash approach is faster, otherwise the encrypt-mix-encrypt approach is faster. A related issue is whether a pre-computed table can be used to speed up multiplication by the hashing key τ. For the single key HEH, this is not possible. On the other hand, for the simple variant HEHp, this is possible and it is also possible for HEHfp.

A detailed comparison among the different tweakable SPRPs is given in [4]. In Table 1, we provide a comparison among some of the more important features of four previous constructions with HEH. The four constructions are CMC, EME* (of the encrypt-mix-encrypt type), HCH (of the hash-Ctr-hash type) and TET (of the hash-ECB-hash type). These four constructions are representative of the currently best known constructions of each type.

For variable number of blocks (Table 1), 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 hashing key τ such that $\sigma = 1 + \tau + \cdots + \tau^{m-1} \neq 0$ and to invert σ. Thus, the restriction on the hashing key directly reflects on the performance of TET. In comparison, HEH does not require these computations. HEH uses only a single block cipher key and cannot utilize pre-computation. If pre-computation is desired, then one can use HEHp.

When m is fixed (Table 2), as in the case for disk encryption, 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. HEHfp is the variant of HEH which is suited for fixed values of m and can utilize pre-computation. The efficiency of HEHfp for encryption and decryption is similar (actually slightly better) to that of TET and HCHfp. The improvement over TET is to offer better key agility. Currently, if one wishes to use the Naor-Reingold approach, then HEHfp is the construction of choice to implement disk encryption schemes.

Table 1. Comparison of tweakable SPRPs where the number of blocks m can vary and n-bit tweaks are used. For TET, $\imath$ is a value which depends on m (and K). [BC]: block cipher invocation; [M]: $GF(2^n)$ multiplication; [I]: $GF(2^n)$ inversion; [BCK]: block cipher key; [AK]: auxilliary n-bit key material;

Mode	type	comp. cost	keys	passes	enc. layers
CMC [5]	enc-mix-enc	$(2m+1)$[BC]	1[BCK]	2	2
EME* [3]	enc-mix-enc	$(2m+\frac{m}{n}+1)$[BC]	1[BCK]+2[AK]	2	2
HCH [1]	hash-Ctr-hash	$(m+3)$[BC] +2$(m-1)$[M]	1[BCK]	2	1
TET [4]	hash-ECB-hash	$\imath((m-1)$[M]+2[BC]) +$(m+2)$[BC] +2m[M]+1[I]	2[BCK]	3	1
HEH	hash-ECB-hash	$(m+2)$[BC] +2$(m-1)$[M]	1[BCK]	3	1

Table 2. 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	CMC [5]	EME* [3]	HCHfp [1]	TET [4]	HEHfp
[BC]	$2m+1$	$2m+1+m/n$	$m+2$	$m+1$	$m+1$
[M]	–	–	$2(m-1)$	$2m$	$2(m-1)$
[BCK]	1	1	1	2	1
[AK]	–	2	1	3	1

Table 3. Efficiency of key change. For TET, the value of $\imath$ depends only on K (since the number of blocks m is fixed). [I]: $GF(2^n)$ inversion.

Mode	CMC	EME*	HCHfp	TET	HEHfp
comp. cost	–	–	–	$\imath((m-1)$[M]+2[BC])+1[BC]+1[I]	–
key sch.	1	1	1	2	1
mult. tab.	–	–	1	1	1

5 Security of HEH

5.1 Definitions and Notation

The discussion in this section is based on earlier work [5]. 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 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 $\mathcal{M}$. For HEH we have $\mathcal{M} = \cup_{m\geq 1}\{0,1\}^{nm}$. 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. 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 $\mathbf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\mathrm{HEH}[\mathrm{Perm}(n)]}(\sigma_n)$. 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.

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

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

The task of the security proof is to upper bound $\mathbf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\mathrm{HEH}[E]}(\sigma_n, t)$. For this it is sufficient to upper bound $\mathbf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\mathrm{HEH}[\mathrm{Perm}(n)]}(\sigma_n)$. Again, to upper bound the last quantity, it is sufficient to upper bound $\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH}[\mathrm{Perm}(n)]}(\sigma_n)$. 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 [5, 6, 2]. 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 $\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH}[\mathrm{Perm}(n)]}(\sigma_n)$.

Theorem 3. *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*

$$\mathbf{Adv}^{\pm\mathrm{rnd}}_{\mathrm{HEH}[\mathrm{Perm}(n)]}(\sigma_n) \leq \frac{4\sigma_n^2}{2^n} \tag{4}$$

The same bound holds when HEH *is replaced by* HEHp *or* HEHfp.

The proof is given in Section A.

6 Conclusion

In this paper, we have proposed a new tweakable SPRP called HEH following the hash-ECB-hash approach introduced by Naor-Reingold [11]. This is done by designing a new invertible block-wise universal hash function. The new hash function improves over the invertible block-wise universal hash function defined in [4] by removing restrictions on the hashing key. This in turn results in HEH being able to remove the drawbacks of the tweakable SPRP called TET which was also proposed in [4]. An important special application of tweakable SPRP is disk encryption. For this application, we suggest a variant called HEHfp. Currently, HEHfp is the best candidate for implementing a disk encryption scheme using the Naor-Reingold approach.

References

1. 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`.
2. 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.
3. 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.
4. Shai Halevi. Invertible universal hashing and the TET encryption mode. Cryptology ePrint Archive, Report 2007/014, 2007. `http://eprint.iacr.org/`, to appear in the proceedings of Crypto 2007.
5. 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.
6. 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.
7. 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.
8. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
9. 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/`.
10. Moni Naor and Omer Reingold. A pseudo-random encryption mode. Manuscript available from `www.wisdom.weizmann.ac.il/~naor`.
11. Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *J. Cryptology*, 12(1):29–66, 1999.
12. 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.
13. 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 Theorem 3

We first consider the case for HEH and later show the necessary modifications for HEHp and HEHfp.

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 HEH with the block cipher substituted by a random permutation from 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 $T_s, (C_{s,1}, \ldots, C_{s,m_s})$. 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 $\sigma_n = \sum_{i=1}^{q}(1 + m_s)$.

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

Game HEH*:* In HEH, the adversary interacts with $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$. For each query, the permutation π is invoked on $PP_{s,i}$ to obtain $CC_{s,i}$; or π^{-1} is invoked $CC_{s,i}$ to obtain $PP_{s,i}$. The permutation π is applied to the tweak T_s to obtain the hashing key τ_s and then $\beta_{1,s}$ is obtained by applying π to $\tau_s \oplus \mathsf{bin}_n(m_s)$.

Instead of initially choosing 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. Tweaks can be repeated by the adversary as also the number of blocks can be same for different queries. In this and the subsequent games, if T_s is equal to T_r for some $r < s$, then τ_s is set to be equal to τ_r. If $(T_s, m_s) = (T_r, m_s)$, for $r < s$, then $(\tau_s, \beta_{1,s})$ is set to be equal to $(\tau_r, \beta_{1,r})$. In other words, we never redefine T_s and $\tau_s \oplus \mathsf{bin}_n(m_s)$.

Game RAND1*:* We modify Game HEH in the following manner. Let $\mathcal{D}$ (resp. $\mathcal{R}$) be a (multi)set which consists of all n-bit strings in the domain (resp. range) of π which have been defined up to now. Initially, both $\mathcal{D}$ and $\mathcal{R}$ are empty. Whenever $\pi(X)$ needs to be defined, we simply choose a random element Y from $\{0,1\}^n$ and set $\pi(X) = Y$. The value X is added to the domain and Y is added to the range. A similar action is performed when π^{-1} needs to be defined. Since we do not check whether X has already occurred in $\mathcal{D}$, this does not guarantee that π is a permutation and hence $\mathcal{D}$ and $\mathcal{R}$ can in principle be multisets. Due to the way tweaks and number of blocks are handled in the games, tweaks are never repeated in $\mathcal{D}$ and for the same value of tweak and number of blocks, $\tau_s \oplus \mathsf{bin}_n(m_s)$ is never repeated in $\mathcal{D}$.

Thus, if the s-th query is an encryption query, we have to define the value of $CC_{s,i}$ to be equal to $\pi(PP_{s,i})$. In the previous game, we would check to see whether π has already been defined for this value. In this game, we do not perform this check and define $CC_{s,i}$ to be a random n-bit string. As a result, $(CC_{s,1}, \ldots, CC_{s,m})$ consists of random n-bit blocks. Since $(C_{s,1}, \ldots, C_{s,m})$ is obtained from $(CC_{s,1}, \ldots, CC_{s,m})$ by the application of the bijection $\Psi_{\tau_s,\beta_{2,s}}()$, the strings $(C_{s,1}, \ldots, C_{s,m})$ are also random n-bit blocks. As a result, on any encryption query, the adversary receives random strings as answer. Similarly, on any decryption query also, the adversary receives random strings as answer.

Let bad be a flag which is set to true if there is a collision in either $\mathcal{D}$ or $\mathcal{R}$. The Games HEH and RAND1 are identical as long as the flag bad is not set to true. So,

$$\Pr[\mathcal{A}^{\mathrm{HEH}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathrm{RAND1}} \Rightarrow 1] \le \Pr[A^{\mathrm{RAND1}} \text{ sets } \mathsf{bad}] \tag{5}$$

Game RAND2*:* In this game, for an encryption query $T_s, (P_{s,1}, \ldots, P_{s,m_s})$, we handle T_s and $\tau_s \oplus \mathsf{bin}_n(m_s)$ as in the earlier games. We set $(PP_{s,1}, \ldots, PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s})$. Next we choose the ciphertext blocks $(C_{s,1}, \ldots, C_{s,m_s})$ to be random n-bit strings and return these to the adversary. Then we set $(CC_{s,1}, \ldots, CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s})$ and enter the values $CC_{s,1}, \ldots, CC_{s,m_s}$ into $\mathcal{R}$ and the values $PP_{s,1}, \ldots, PP_{s,m_s}$ into $\mathcal{D}$.

Similarly, for a decryption query on $T_s, (C_{s,1}, \ldots, C_{s,m_s})$, we first handle the tweak and number of blocks as in the previous games. We set $(CC_{s,1}, \ldots, CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1}, \ldots, C_{s,m_s})$. We then choose the plaintext blocks $(P_{s,1}, \ldots, P_{s,m_s})$ to be random n-bit strings and return these to the adversary. Then we set $(PP_{s,1}, \ldots, PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1}, \ldots, P_{s,m_s})$ and enter the values $PP_{s,1}, \ldots, PP_{s,m_s}$ into $\mathcal{D}$ and the values $CC_{s,1}, \ldots, CC_{s,m_s}$ into $\mathcal{R}$.

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,

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

In RAND2, the adversary is supplied with random bits as response to queries to both the encrypt and the decrypt oracles. Hence,

$$\Pr[A^{\text{RAND2}} \Rightarrow 1] = \Pr[A^{\$(.,.),\$(.,.)} \Rightarrow 1] \tag{6}$$

We get

$$\begin{aligned}
\mathbf{Adv}^{\pm\text{rnd}}_{\text{HEH}[\text{Perm}(n)]}(\mathcal{A}) &= \Pr[\mathcal{A}^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(.,.),\$(.,.)} \Rightarrow 1] && (7)\\
&= \Pr[\mathcal{A}^{\text{HEH}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{RAND2}} \Rightarrow 1]\\
&= \Pr[\mathcal{A}^{\text{HEH}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{RAND1}} \Rightarrow 1]\\
&\le \Pr[\mathcal{A}^{\text{RAND1}} \text{ sets } \mathsf{bad}]\\
&= \Pr[\mathcal{A}^{\text{RAND2}} \text{ sets } \mathsf{bad}] && (8)
\end{aligned}$$

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 $\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]$. 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 $\Pr[A^{\text{RAND2}} \text{ sets } \mathsf{bad}]$ 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.

For the s-th query (encryption or decryption) with tweak T_s; plaintext blocks $(P_{s,1},\ldots,P_{s,m_s})$; and ciphertext blocks $(C_{s,1},\ldots,C_{s,m_s})$ we perform the following. We assume that the adversary does not repeat a query or make a pointless query.

```
1.  If $T_s$ is "new", then
2.      choose random values for $\tau_s$ and $\beta_{1,s}$;
3.      add $T_s$ and $\tau_s \oplus \mathsf{bin}_n(m_s)$ to $\mathcal{D}$;
4.      add $\tau_s$ and $\beta_{1,s}$ to $\mathcal{R}$;
5.  else
6.      use the previous corresponding value of $\tau_s$;
7.      if $(T_s, m_s)$ is "new", then
8.          choose $\beta_{1,s}$ randomly;
9.          add $\tau_s \oplus \mathsf{bin}_n(m_s)$ to $\mathcal{D}$;
10.         add $\beta_{1,s}$ to $\mathcal{R}$;
11.     else
12.         use the previous corresponding value of $\beta_{1,s}$;
13.     end if;
14. end if;
15. Set $(PP_{s,1},\ldots,PP_{s,m_s}) = \Psi_{\tau_s,\beta_{1,s}}(P_{s,1},\ldots,P_{s,m_s})$;
16. Add $(PP_{s,1},\ldots,PP_{s,m_s})$ to $\mathcal{D}$.
17. Set $(CC_{s,1},\ldots,CC_{s,m_s}) = \Psi_{\tau_s,\beta_{2,s}}(C_{s,1},\ldots,C_{s,m_s})$;
18. Add $(CC_{s,1},\ldots,CC_{s,m_s})$ to $\mathcal{R}$.
```

Collision Analysis: We upper bound the probability of collision in $\mathcal{D}$ and $\mathcal{R}$. For this it will be useful to list the forms of the elements in these two multisets.

Elements in $\mathcal{D}$: T_s, $\tau_s \oplus \mathsf{bin}_n(m_s)$;
$PP_{s,1} = U_s \oplus P_{s,1} \oplus x\beta_{1,s}, \ldots, PP_{s,m_s-1} = U_s \oplus P_{s,m_s-1} \oplus x^{m_s-1}\beta_{1,s}$,
$PP_{s,m_s} = U_s \oplus \beta_{1,s}$.

Here $U_s = \bigoplus_{i=1}^{m_s} P_{s,i}\tau_s^{m_s-i}$.

Elements in $\mathcal{R}$: τ_s, $\beta_{1,s}$;
$CC_{s,1} = V_s \oplus C_{s,1} \oplus x\beta_{2,s}, \ldots, CC_{s,m_s-1} = V_s \oplus C_{s,m_s-1} \oplus x^{m_s-1}\beta_{2,s}$,
$CC_{s,m_s} = V_s \oplus \beta_{2,s}$.

Here $V_s = \bigoplus_{i=1}^{m_s} C_{s,i}\tau_s^{m_s-i}$ and also recall that $\beta_{2,s} = x\beta_{1,s}$.

We first consider the probability of a collision in $\mathcal{D}$. First note the values T_s in $\mathcal{D}$ are all distinct, i.e., we never redefine the Ts. Hence, there cannot be any collision among these values. Suppose the distinct lengths of the queries are $n_1, \ldots, n_p$ and there are l_k queries of length n_k, so that $\sum_{k=1}^{p} l_k = q$ and $\sum_{k=1}^{p} l_k n_k = \sigma_n - q$. We say that a pair of elements $(PP_{s,i}, PP_{r,i})$ is special if $(T_s, m_s) = (T_r, m_r)$. The number of special pairs is at most $\sum_{k=1}^{p} \binom{l_k}{2} m_k$. For a special pair, the values of the corresponding hashing keys $(\tau_s, \beta_{1,s})$ and $(\tau_r, \beta_{1,r})$ are equal. Consequently, using Theorem 1(2), the probability that the elements of a special pair are equal is $m_k/2^n$. Hence, the total probability of equality of special pairs is

$$\sum_{k=1}^{p} \binom{l_k}{2} m_k \times \frac{m_k}{2^n} \leq \frac{\sum_{k=1}^{p}(l_k m_k)^2}{2^n} \leq \frac{(\sum_{k=1}^{p} l_k m_k)^2}{2^n} \leq \frac{\sigma_n^2}{2^n}.$$

The number of non-special pairs in $\mathcal{D}$ is less than σ_n^2. For any non-special pair, the probability that the two elements are equal is either zero or $1/2^n$. Hence, the total probability of equality of non-special pairs is at most $\sigma_n^2/2^n$. Consequently, the total probability of collision in $\mathcal{D}$ is at most $2\sigma_n^2/2^n$. A similar analysis shows that the probability of collision in $\mathcal{R}$ is also at most $2\sigma_n^2/2^n$ and hence the total probability of collision in either $\mathcal{D}$ or $\mathcal{R}$ is at most $4\sigma_n^2/2^n$. Since this is also the probability that bad is set to true, we obtain the desired bound.

Collision Analysis for HEHp**:** The structures of $\mathcal{D}$ and $\mathcal{R}$ in this case are the following. Recall that in this case, the hashing key τ does not depend on the tweak T. It is chosen randomly and is same for all queries. The hashing key $\beta_{1,s}$, however, is still derived from the tweak and the number of blocks.

Elements in $\mathcal{D}$: T_s, $\gamma_s \oplus \mathsf{bin}_n(m_s)$;
$PP_{s,1} = U_s \oplus P_{s,1} \oplus x\beta_{1,s}, \ldots, PP_{s,m_s-1} = U_s \oplus P_{s,m_s-1} \oplus x^{m_s-1}\beta_{1,s}$,
$PP_{s,m_s} = U_s \oplus \beta_{1,s}$.

Here $U_s = \bigoplus_{i=1}^{m_s} P_{s,i}\tau^{m_s-i}$.

Elements in $\mathcal{R}$: γ_s, $\beta_{1,s}$;
$CC_{s,1} = V_s \oplus C_{s,1} \oplus x\beta_{2,s}, \ldots, CC_{s,m_s-1} = V_s \oplus C_{s,m_s-1} \oplus x^{m_s-1}\beta_{2,s}$,
$CC_{s,m_s} = V_s \oplus \beta_{2,s}$.

Here, $V_s = \bigoplus_{i=1}^{m_s} C_{s,i}\tau^{m_s-i}$ and $\beta_{2,s} = x\beta_{1,s}$. Now a collision analysis along the lines performed for HEH provide us the same bound.

Collision Analysis for HEHfp**:** In this case, the number of blocks in each query is fixed, i.e., m does not depend on s. Also, $\beta_{1,s}$ is obtained directly by encrypting T_s. As in the case of HEHp, the hashing key τ is the same for all queries.

Elements in $\mathcal{D}$: T_s,
$PP_{s,1} = U_s \oplus P_{s,1} \oplus x\beta_{1,s}, \ldots, PP_{s,m-1} = U_s \oplus P_{s,m-1} \oplus x^{m-1}\beta_{1,s}$,
$PP_{s,m} = U_s \oplus \beta_{1,s}$.

Here $U_s = \bigoplus_{i=1}^{m} P_{s,i}\tau^{m-i}$.

Elements in $\mathcal{R}$: $\beta_{1,s}$;
$CC_{s,1} = V_s \oplus C_{s,1} \oplus x\beta_{2,s}, \ldots, CC_{s,m-1} = V_s \oplus C_{s,m-1} \oplus x^{m-1}\beta_{2,s}$,
$CC_{s,m} = V_s \oplus \beta_{2,s}$.

Here $V_s = \bigoplus_{i=1}^{m} C_{s,i}\tau_s^{m-i}$ and $\beta_{2,s} = x\beta_{1,s}$. Again a collision analysis similar to that of HEH shows the desired bound. $\square$