Further More on Key Wrapping

Yasushi Osaki and Tetsu Iwata

Dept. of Computational Science and Engineering, Nagoya University, Japan
y_ozaki@echo.nuee.nagoya-u.ac.jp, iwata@cse.nagoya-u.ac.jp

Abstract. At SAC 2009, Gennaro and Halevi showed that a key wrapping scheme using a universal hash function and ECB mode (a HtECB scheme) is broken, and the security of a scheme based on a universal hash function and CBC mode (a HtCBC scheme) has been left as an open problem. In this paper, we first generalize classical notions of universal and uniform hash functions, and propose two new notions, where we consider the composition of the keyed hash function. We then prove that the HtECB scheme is a secure key wrapping scheme if a universal hash function satisfies uniformity and our notions. We further generalize the notions, and propose two more new notions for a keyed hash function. We present a partial answer to the open problem by showing that the HtCBC scheme using a universal hash function that also satisfies uniformity and the new notions is a secure key wrapping scheme. We then point out that a monic polynomial hash function satisfies all the new notions. Therefore, by combining the monic polynomial hash function with ECB mode and CBC mode, we obtain secure and efficient key wrapping schemes.

Keywords: Key wrapping scheme, universal hash function, uniform hash function, ECB mode, CBC mode, security proof

1 Introduction

A (symmetric) key wrapping scheme is a symmetric key primitive that is used to encrypt specialized data, such as cryptographic keys, without using a nonce or IV, for distribution or storage. An authenticated key wrapping scheme, which we write AKW, additionally ensures integrity. These schemes are widely used in practice [1, 12, 20], and NIST is in the process of specifying an AKW scheme [19].

There are mainly two approaches in designing an AKW scheme. The first approach is the dedicated construction. At EUROCRYPT 2006, Rogaway and Shrimpton proposed a concept of deterministic authenticated encryption (DAE), which formalizes the strongest notion of an authenticated encryption scheme without using a nonce or IV [22]. The security notion for DAE schemes demands that they achieve confidentiality and integrity even if the input data (plaintext) is chosen by an adversary, while it allows the adversary to detect a repetition of the same plaintexts being encrypted (and nothing more). They showed that a secure DAE scheme can be used in AKW applications. Then they proposed SIV mode as a concrete construction of the DAE scheme, which can be seen as the dedicated construction of the AKW scheme. Subsequent works in the dedicated designs of DAE schemes (and hence AKW schemes) include HBS mode [13], BTM mode [14], and DCM mode [7]. Other examples are in [1].

Another approach is the construction by the generic composition, where one combines an encryption mode and a hash function to construct an AKW scheme. This approach allows re-use of the existing primitives, thus the implementation and the deployment are potentially easier than the dedicated constructions. At SAC 2009, Gennaro and Halevi pointed out that the security requirement of DAE rules out many practical and "seemingly secure" implementations [9]. For this, they proposed a security notion which states that the AKW scheme is secure if it achieves confidentiality and integrity for random plaintexts. The notion is strictly weaker than the security requirement for the DAE scheme. However, the notion is sufficiently strong for most AKW applications, where the plaintext is intended to be a cryptographic key, and thus a random plaintext, instead of that chosen by the adversary. Then they studied a wide variety of AKW

Table 1. Security of Hash-then-Encrypt constructions [9]. In the table, "secure" means that the combination achieves provable security, "somewhat" means the security bound is worse than the birthday bound, and "broken" means that the combination allows an attack.

Encryption\Hash	XOR	Linear	2nd-preimage resistant	universal hash
CTR	broken	broken	secure	secure
ECB	broken	somewhat	secure	broken
CBC	broken	somewhat	secure	open problem
masked ECB/CBC	somewhat	somewhat	secure	secure
XEX	secure	secure	secure	secure

schemes by combining elementary encryption modes with hash functions. Specifically, Gennaro and Halevi extensively studied constructions of a so-called Hash-then-Encrypt approach, where one first derives the hash value of the random plaintext, and then encrypts the hash value together with the plaintext to obtain the ciphertext. Their results are summarized in Table 1. They examined combinations of XOR, Linear, 2nd-preimage resistant, and universal hash functions, and CTR mode [17], ECB mode [17], CBC mode [17], masked ECB/CBC mode [9], and XEX mode [21] (See [9] for definitions of XOR, Linear, and 2nd-preimage resistant hash functions).

Our Contributions. In this paper, we closely look at the security of the Hash-then-Encrypt approach using ECB and CBC modes, which are called Hash-then-ECB (HtECB) and Hash-then-CBC (HtCBC) schemes. In particular, we focus on the case where a universal hash function is used as the underlying hash function. These schemes are attractive in that they allow use of the most common encryption modes, ECB and CBC modes, which are likely deployed already in existing systems, and there are a large number of efficient constructions of a universal hash function including a polynomial hash function [16, 18, 4], MMH [10], Square Hash [8], NMH [10, 23], and NH [5].

From Table 1, we see that combining a universal hash function with ECB mode is broken, which means that there exists a universal hash function such that, when used with ECB mode, the resulting HtECB scheme allows an efficient attack. Indeed, Gennaro and Halevi showed that such a universal hash function exists, which implies that sole requirement of being universal is not sufficient for the security. However, we remark that this does not imply all the universal hash functions cannot be used with ECB mode, and in particular, this does not exclude a possibility that there exists a subset of universal hash functions that can securely be used with ECB mode. The first contribution of this paper is to identify such a subset, a sufficient condition for the universal hash function so that the resulting HtECB scheme becomes a secure AKW scheme. We propose two new notions for a keyed hash function which we call universal$_C$ and uniform$_C$ hash functions. The universal$_C$ hash function is roughly a universal hash function such that the collision probability stays small even when some part of the input is a hash value, i.e., even when we consider the composition of the hash function. The uniform$_C$ hash function is a uniform hash function such that the output value is close to uniform even when some part of the input is a hash value. Both notions are natural extensions of the classical notions of the universal and uniform hash functions [6]. We then show that if the underlying hash function is a universal, uniform, universal$_C$ and uniform$_C$ hash function, then the HtECB scheme is a secure AKW scheme.

Next, we look at a HtCBC scheme that uses a universal hash function. The security of this scheme has been left as an open problem, and conjectured to be secure in [9]. As a second contribution of this paper, we present a partial answer to the open problem. We propose two more new notions for a keyed hash function which we call universal$_{CC}$ and uniform$_{CC}$ hash

functions. They are the natural extensions of universal$_C$ and uniform$_C$ hash functions, where we allow some part of the input being "hash value xor constant," instead of merely a hash value. We then show that if the underlying hash function is a universal, uniform, universal$_{CC}$ and uniform$_{CC}$ hash function, then the HtCBC scheme is secure, giving a partial answer to the open problem.

We then discuss that our new notions are strictly stronger than the classical notions of universal and uniform hash functions. This may imply that constructing a hash function that meets new notions is hard, or computationally costly. For this, we point out that a simple and efficient construction exists, by showing that a monic polynomial hash function satisfies all the notions considered in this paper. Therefore, by combining the monic polynomial hash function with ECB mode and CBC mode, we obtain secure and efficient AKW schemes. We also remark that the CBC MAC satisfies all the notions.

Remarks. We note that the focus of this paper is AKW applications, and HtECB and HtCBC schemes that we prove secure may not maintain their security once the input data, which is supposed to be random, is chosen by the adversary (this is at least the case for the confidentiality of the HtECB scheme). We also note that dedicated DAE schemes in [22, 13, 14, 7] can be used as secure AKW schemes, and hence, whenever possible, one should prefer these constructions. Practical advantages of HtECB and HtCBC schemes include that they allow re-use of components that are already implemented in existing systems; ECB and CBC modes are the most commonly used encryption modes, and the monic polynomial hash function can be implemented efficiently from the polynomial hash function in a black-box manner. This feature helps reducing the load of the implementation and the deployment if the underlying primitives are already available. Finally, we note that none of the results of this paper contradicts the results in [9]. We also note that asymmetric key-wrapping schemes are studied in [11].

2 Preliminaries

2.1 Notation

For two bit strings X and Y of the same length, $X \oplus Y$ is their xor. For an integer $n \geq 1$, $\{0,1\}^n$ is the set of all bit strings of n bits. The set $\{0,1\}^n$ is also treated as the finite field GF(2^n) of 2^n elements (relative to some irreducible polynomial). We write $X \xleftarrow{\$} \mathcal{X}$ for sampling an element from the set $\mathcal{X}$ uniformly at random and assigning its value to the variable X. For two integers $l, n \geq 1$ and a bit string $X \in \{0,1\}^{nl}$, $(X[1], \ldots, X[l])$ is the partition of X into n-bit strings, i.e., $X[1], \ldots, X[l]$ are the unique bit strings such that $(X[1], \ldots, X[l]) = X$. We write $(X[1], \ldots, X[l]) \xleftarrow{n} X$ for the partitioning operation. We use the convention that when $X \in \{0,1\}^{n(l+1)}$, the partition of X is $(X[0], X[1], \ldots, X[l])$ and we write $(X[0], \ldots, X[l]) \xleftarrow{n} X$.

2.2 Blockciphers

A blockcipher is a family of permutations. Throughout this paper we fix the block size n. Typical values of n are 64 and 128. We write $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ for a blockcipher, where $\mathcal{K}$ is a non-empty set of keys, $K \in \mathcal{K}$ is a key, and $E_K(\cdot)$ is a permutation on $\{0,1\}^n$ specified by the key K. We write $E_K^{-1}(\cdot)$ for the inverse permutation of $E_K(\cdot)$. Let Perm(n) be a set of all permutations on $\{0,1\}^n$. This set can be regarded as a blockcipher by assigning a unique key to each permutation. We say that a permutation $P(\cdot)$ on $\{0,1\}^n$ is a random permutation if $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$, and we write $P^{-1}(\cdot)$ for the inverse permutation of $P(\cdot)$.

An adversary is an oracle machine. We consider two security notions for a blockcipher. The goal of a PRP-adversary (PseudoRandom Permutation-adversary) A is to distinguish the

blockcipher $E_K(\cdot)$ from a random permutation $P(\cdot)$. The success probability of A is measured by

$$\mathbf{Adv}_E^{\mathrm{prp}}(A) = \Pr[A^{E_K(\cdot)} \Rightarrow 1] - \Pr[A^{P(\cdot)} \Rightarrow 1],$$

where the first probability is over the random choices of $K \xleftarrow{\$} \mathcal{K}$ and A's coin (if any), and the last is over $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$ and A's coin (if any) [15, 2]. We also consider a stronger adversary, an SPRP-adversary (Strong PRP-adversary), whose goal is to distinguish $(E_K(\cdot), E_K^{-1}(\cdot))$ from $(P(\cdot), P^{-1}(\cdot))$. The success probability of A is measured by

$$\mathbf{Adv}_E^{\mathrm{sprp}}(A) = \Pr[A^{E_K(\cdot),E_K^{-1}(\cdot)} \Rightarrow 1] - \Pr[A^{P(\cdot),P^{-1}(\cdot)} \Rightarrow 1],$$

where the first probability is over the random choices of $K \xleftarrow{\$} \mathcal{K}$ and A's coin (if any), and the last is over $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$ and A's coin (if any) [15, 2].

The resource of A is measured by the number of oracle calls and the time complexity. The time complexity of A, which we write $\mathrm{Time}(A)$, is the sum of the actual running time (relative to some fixed RAM model of computation) and its description size (relative to some standard encoding of algorithms). The details of the big-O notation in a time complexity reference depend on the RAM model and the choice of encoding.

2.3 Encryption Modes

The blockcipher is used to encrypt a plaintext of fixed length, n bits. A longer plaintext can be encrypted by using an encryption mode. In this paper, we consider two encryption modes known as ECB mode and CBC mode [17].

Throughout this paper we fix an integer $l \geq 1$. This parameter corresponds to the length of the plaintext to encrypt in n-bit blocks. Typical values of l that we intend in this paper include $l = 1$, 2, 3, 4 and 8. Let $D \in \{0,1\}^{nl}$ be the plaintext to encrypt.

In ECB mode, we first parse the plaintext $D \in \{0,1\}^{nl}$ into n-bit blocks as $(D[1], \ldots, D[l]) \xleftarrow{n} D$. We then encrypt each block with the blockcipher, i.e., we let $C[j] \leftarrow E_K(D[j])$ for $1 \leq j \leq l$. The ciphertext is $C \leftarrow (C[1], \ldots, C[l])$. To decrypt the ciphertext $C \in \{0,1\}^{nl}$, we first parse it into n-bit blocks as $(C[1], \ldots, C[l]) \xleftarrow{n} C$. We then decrypt each block with the blockcipher, i.e., we let $D[j] \leftarrow E_K^{-1}(C[j])$ for $1 \leq j \leq l$. The plaintext is obtained by $D \leftarrow (D[1], \ldots, D[l])$.

In CBC mode, we first parse the plaintext $D \in \{0,1\}^{nl}$ into n-bit blocks as $(D[1], \ldots, D[l]) \xleftarrow{n} D$. Let $IV \in \{0,1\}^n$ be an initial value and let $C[0] \leftarrow IV$. Then, we compute $C[j] \leftarrow E_K(D[j] \oplus C[j-1])$ for $1 \leq j \leq l$. The ciphertext is $C \leftarrow (C[0], \ldots, C[l])$. To decrypt the ciphertext $C \in \{0,1\}^{n(l+1)}$, we first parse it into n-bit blocks as $(C[0], \ldots, C[l]) \xleftarrow{n} C$. Next, we let $D[j] \leftarrow E_K^{-1}(C[j]) \oplus C[j-1]$ for $1 \leq j \leq l$, and the plaintext is $D \leftarrow (D[1], \ldots, D[l])$.

2.4 Keyed Hash Functions

A keyed hash function is a family of functions. We write $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ for a keyed hash function, where $\mathcal{L}$ is a non-empty set of keys, $L \in \mathcal{L}$ is a key, and $H_L(\cdot) : \{0,1\}^{nl} \to \{0,1\}^n$ is a function specified by the key L.

We present two classical notions of a universal hash function and a uniform hash function [6].

Definition 1. *Let $X, X' \in \{0,1\}^{nl}$ be arbitrary bit strings such that $X \neq X'$. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an* ϵ_1-universal *hash function if*

$$\Pr[H_L(X) = H_L(X')] \leq \epsilon_1,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

Algorithm HtECB$[H, E]$-$\mathcal{E}_{L,K}(D)$	**Algorithm** HtECB$[H, E]$-$\mathcal{D}_{L,K}(C)$
1. $D[0] \leftarrow H_L(D)$	1. $(C[0], \ldots, C[l]) \xleftarrow{n} C$
2. $(D[1], \ldots, D[l]) \xleftarrow{n} D$	2. **for** $j \leftarrow 0$ **to** l **do**
3. **for** $j \leftarrow 0$ **to** l **do**	3. $\quad D[j] \leftarrow E_K^{-1}(C[j])$
4. $\quad C[j] \leftarrow E_K(D[j])$	4. $D \leftarrow (D[1], \ldots, D[l])$
5. $C \leftarrow (C[0], \ldots, C[l])$	5. **if** $H_L(D) = D[0]$ **then return** D
6. **return** C	6. **else return** $\perp$

Fig. 1. The encryption and decryption algorithms of HtECB$[H, E]$.

Definition 2. *Let $X \in \{0,1\}^{nl}$ and $Y \in \{0,1\}^n$ be arbitrary bit strings. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an* ϵ_2-uniform *hash function if*

$$\Pr[H_L(X) = Y] \leq \epsilon_2,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

We say that H is a universal (resp. uniform) hash function if it is an ϵ_1-universal (resp. ϵ_2-uniform) hash function for sufficiently small ϵ_1 (resp. ϵ_2). We use similar notation for other notions of the hash function.

3 Authenticated Key Wrapping Schemes

3.1 Definitions and Constructions

We define an authenticated key wrapping scheme, which we write AKW. The AKW scheme consists of an encryption algorithm AKW-$\mathcal{E}$ and a decryption algorithm AKW-$\mathcal{D}$. The encryption algorithm AKW-$\mathcal{E}$ takes a plaintext $D \in \{0,1\}^{nl}$ and a wrapping key $W \in \mathcal{W}$, where $\mathcal{W}$ is a non-empty set of keys, and outputs a ciphertext $C \in \{0,1\}^{n(l+1)}$. We write $C \leftarrow \text{AKW-}\mathcal{E}_W(D)$. The decryption algorithm AKW-$\mathcal{D}$ takes a ciphertext $C \in \{0,1\}^{n(l+1)}$ and the wrapping key $W \in \mathcal{W}$ and outputs either the plaintext $D \in \{0,1\}^{nl}$, or a special symbol $\perp$ which indicates that the ciphertext is invalid. We write $D \leftarrow \text{AKW-}\mathcal{D}_W(C)$ or $\perp \leftarrow \text{AKW-}\mathcal{D}_W(C)$. For consistency, for all $D \in \{0,1\}^{nl}$ and $W \in \mathcal{W}$, if $C \leftarrow \text{AKW-}\mathcal{E}_W(D)$, we require $D \leftarrow \text{AKW-}\mathcal{D}_W(C)$.

Gennaro and Halevi considered AKW constructions of a so-called Hash-then-Encrypt approach [9]. We describe two concrete constructions which are called Hash-then-ECB (HtECB) and Hash-then-CBC (HtCBC) schemes.

HtECB [9]: Hash-then-ECB scheme is specified by a (keyed) hash function H and a blockcipher E, and we write HtECB$[H, E]$ for the HtECB scheme that uses H and E. It consists of the encryption algorithm HtECB$[H, E]$-$\mathcal{E}$ and the decryption algorithm HtECB$[H, E]$-$\mathcal{D}$ which are defined in Fig. 1, and the encryption algorithm is illustrated in Fig. 3.

HtCBC [9]: Hash-then-CBC scheme is also specified by a (keyed) hash function H and a blockcipher E, and we write HtCBC$[H, E]$ for the HtCBC scheme that uses H and E. The encryption algorithm HtCBC$[H, E]$-$\mathcal{E}$ and the decryption algorithm HtCBC$[H, E]$-$\mathcal{D}$ are defined in Fig. 2, and the encryption algorithm is illustrated in Fig. 4.

3.2 Security Definitions for the AKW

We follow the security definitions proposed by Gennaro and Halevi [9]. Since the AKW scheme is used to encrypt and authenticate random plaintext, it is necessary to satisfy two security definitions. The first one is called the security against the Random-Plaintext Attack (RPA) and the other is called the INTegrity of ciphertext (INT).

Algorithm HtCBC$[H, E]$-$\mathcal{E}_{L,K}(D)$	**Algorithm** HtCBC$[H, E]$-$\mathcal{D}_{L,K}(C)$
1. $D[0] \leftarrow H_L(D)$	1. $(C[0], \ldots, C[l]) \xleftarrow{n} C$
2. $(D[1], \ldots, D[l]) \xleftarrow{n} D$	2. $D[0] \leftarrow E_K^{-1}(C[0])$
3. $C[0] \leftarrow E_K(D[0])$	3. **for** $j \leftarrow 1$ **to** l **do**
4. **for** $j \leftarrow 1$ **to** l **do**	4. $\quad I[j] \leftarrow E_K^{-1}(C[j])$
5. $\quad I[j] \leftarrow D[j] \oplus C[j-1]$	5. $\quad D[j] \leftarrow I[j] \oplus C[j-1]$
6. $\quad C[j] \leftarrow E_K(I[j])$	6. $D \leftarrow (D[1], \ldots, D[l])$
7. $C \leftarrow (C[0], \ldots, C[l])$	7. **if** $H_L(D) = D[0]$ **then return** D
8. **return** C	8. **else return** $\perp$

Fig. 2. The encryption and decryption algorithms of HtCBC$[H, E]$.

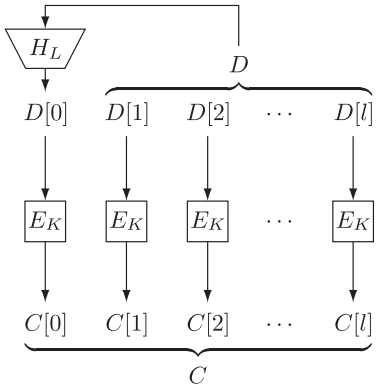

Fig. 3. The encryption algorithm of HtECB$[H, E]$.

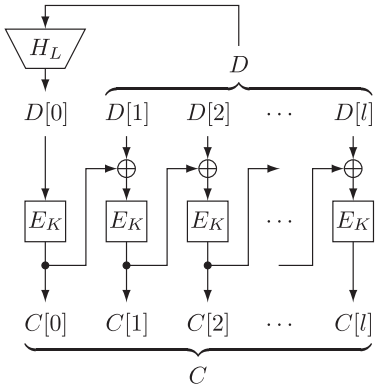

Fig. 4. The encryption algorithm of HtCBC$[H, E]$.

RPA-security [9]: The goal of an RPA-adversary A is to distinguish the left-encryption oracle \$AKW-$\mathcal{E}_W^0(\cdot)$ from the right-encryption oracle \$AKW-$\mathcal{E}_W^1(\cdot)$. The adversary can only invoke the oracle, and when invoked, the \$AKW-$\mathcal{E}_W^0(\cdot)$ oracle chooses two plaintexts $D^0, D^1 \xleftarrow{\$} \{0,1\}^{nl}$, and A receives (D^0, D^1, C), where $C \leftarrow$ AKW-$\mathcal{E}_W(D^0)$. Likewise, in the invocation of the \$AKW-$\mathcal{E}_W^1(\cdot)$ oracle, it chooses two plaintexts $D^0, D^1 \xleftarrow{\$} \{0,1\}^{nl}$, and A receives (D^0, D^1, C), where $C \leftarrow$ AKW-$\mathcal{E}_W(D^1)$. The success probability of A is measured by

$$\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{AKW}}(A) = \Pr[A^{\$\text{AKW-}\mathcal{E}_W^0(\cdot)} \Rightarrow 1] - \Pr[A^{\$\text{AKW-}\mathcal{E}_W^1(\cdot)} \Rightarrow 1],$$

where the probabilities are over the random choices of $W \xleftarrow{\$} \mathcal{W}$, coins used by the oracles, and A's coin (if any).

INT-security [9]: The goal of an INT-adversary A is to output a forgery. As in the game of RPA, the INT-adversary A can only invoke the oracle, and when invoked, the random-encryption oracle \$AKW-$\mathcal{E}_W(\cdot)$ chooses a plaintext $D \xleftarrow{\$} \{0,1\}^{nl}$, and A receives (D, C), where $C \leftarrow$ AKW-$\mathcal{E}_W(D)$. The adversary A subsequently outputs a challenge ciphertext C^*, and we say A forges if C^* is different from the ciphertexts returned to the adversary from the oracle and C^* is not invalid. That is, A forges if $C^* \notin \{C_1, \ldots, C_q\}$ and $\perp \neq$ AKW-$\mathcal{D}_W(C^*)$, where $C_1, \ldots, C_q$ denote the ciphertexts returned to the adversary. The success probability of A is measured by

$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{AKW}}(A) = \Pr[A^{\$\text{AKW-}\mathcal{E}_W(\cdot)} \text{ forges}],$$

where the probability is taken over the random choices of $W \xleftarrow{\$} \mathcal{W}$, coins used by the oracle, and A's coin (if any).

Intuitively, we say that the AKW scheme is secure if $\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{AKW}}(A)$ and $\mathbf{Adv}^{\mathrm{int}}_{\mathrm{AKW}}(A)$ are sufficiently small for any adversary A with reasonable resource, which is measured by the number of oracle invocations and the time complexity.

4 New Notions for Keyed Hash Functions

In this section, we present our new notions for a keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$. In the following two new notions, some of the input blocks can be the hash value, i.e., we consider the "composition" of the hash function.

Definition 3. *Let $X_1, \ldots, X_l \in \{0,1\}^{nl}, Z[1], \ldots, Z[l] \in \{0,1\}^n$ and $X' \in \{0,1\}^{nl}$ be arbitrary bit strings such that $X' \neq (Z[1], \ldots, Z[l])$. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an ϵ_3-universal$_C$ (ϵ_3-universal with composition) hash function if, for each of the 2^l possible choices of $X \in \{Z[1], H_L(X_1)\} \times \cdots \times \{Z[l], H_L(X_l)\}$, it holds that*

$$\Pr[H_L(X) = H_L(X')] \leq \epsilon_3,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

For example, if $l = 2$, we require

$$\begin{cases} \Pr[H_L(Z[1], Z[2]) = H_L(X')] \leq \epsilon_3, \\ \Pr[H_L(H_L(X_1), Z[2]) = H_L(X')] \leq \epsilon_3, \\ \Pr[H_L(Z[1], H_L(X_2)) = H_L(X')] \leq \epsilon_3, \text{and} \\ \Pr[H_L(H_L(X_1), H_L(X_2)) = H_L(X')] \leq \epsilon_3. \end{cases}$$

We remark that Definition 1 corresponds to above definition with $X = (Z[1], \ldots, Z[l])$ and thus Definition 1 is included in the above definition.

The next definition corresponds to the uniformity of the above definition.

Definition 4. *Let $X_1, \ldots, X_l \in \{0,1\}^{nl}, Z[1], \ldots, Z[l] \in \{0,1\}^n$ and $Y \in \{0,1\}^n$ be arbitrary bit strings. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an ϵ_4-uniform$_C$ (ϵ_4-uniform with composition) hash function if, for each of the 2^l possible choices of $X \in \{Z[1], H_L(X_1)\} \times \cdots \times \{Z[l], H_L(X_l)\}$, it holds that*

$$\Pr[H_L(X) = Y] \leq \epsilon_4,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

We further generalize the above two notions. In the following two notions, some of the input blocks can be "the hash value xor constant."

Definition 5. *Let $X_1, \ldots, X_l \in \{0,1\}^{nl}, Z[1], \ldots, Z[l] \in \{0,1\}^n, V[1], \ldots, V[l] \in \{0,1\}^n$ and $X' \in \{0,1\}^{nl}$ be arbitrary bit strings such that $X' \neq (Z[1], \ldots, Z[l])$. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an ϵ_5-universal$_{CC}$ (ϵ_5-universal with composition and constant xor) hash function if, for each of the 2^l possible choices of $X \in \{Z[1], H_L(X_1) \oplus V[1]\} \times \cdots \times \{Z[l], H_L(X_l) \oplus V[l]\}$, it holds that*

$$\Pr[H_L(X) = H_L(X')] \leq \epsilon_5,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

Definition 6. *Let $X_1, \ldots, X_l \in \{0,1\}^{nl}, Z[1], \ldots, Z[l] \in \{0,1\}^n, V[1], \ldots, V[l] \in \{0,1\}^n$ and $Y \in \{0,1\}^n$ be arbitrary bit strings. A keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ is an ϵ_6-uniform$_{CC}$ (ϵ_6-uniform with composition and constant xor) hash function if, for each of the 2^l possible choices of $X \in \{Z[1], H_L(X_1) \oplus V[1]\} \times \cdots \times \{Z[l], H_L(X_l) \oplus V[l]\}$, it holds that*

$$\Pr[H_L(X) = Y] \leq \epsilon_6,$$

where the probability is taken over the random choice of $L \xleftarrow{\$} \mathcal{L}$.

In the following sections, we show that, with these definitions, the security of HtECB and HtCBC schemes can be proved.

5 Analysis of Hash-then-ECB Scheme

As in Table 1, at SAC 2009, Gennaro and Halevi showed that a combination of a universal hash function and ECB mode is broken, by showing a concrete instance of a universal hash function that leads to a successful forgery. This implies that, in general, the combination does not satisfy INT-security.

In the following theorem, we show that the HtECB scheme is secure if the universal hash function satisfies additional requirements of being a uniform, universal$_C$ and uniform$_C$ hash function, under the assumption that the blockcipher is secure in sense of a PRP (for RPA) and an SPRP (for INT).

Theorem 1. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be an ϵ_1-universal, ϵ_2-uniform, ϵ_3-universal$_C$ and ϵ_4-uniform$_C$ hash function, and $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then for any A that invokes the oracle at most q times, there exist adversaries A' and A'' such that*

$$\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{HtECB}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(A') + \frac{2q^2l^2}{2^n} + 2q^2\epsilon_1 + 4q^2l\epsilon_2,$$
$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtECB}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{sprp}}_{E}(A'') + \frac{q^2}{2}\epsilon_1 + q^2l\epsilon_2 + q(l+1)\epsilon_2 + \max\{\epsilon_3, \epsilon_4\},$$

where A' makes at most $q(l+1)$ queries and A'' makes at most $(q+1)(l+1)$ queries. Furthermore, $\mathrm{Time}(A') = \mathrm{Time}(A) + O(nlq)$ and $\mathrm{Time}(A'') = \mathrm{Time}(A) + O(nlq)$.

We note that ϵ_1 and ϵ_2 can respectively be replaced by ϵ_3 and ϵ_4, since an ϵ_3-universal$_C$ and ϵ_4-uniform$_C$ hash function is always an ϵ_3-universal and ϵ_4-uniform hash function. We separate them as it makes clear the properties needed to achieve the security notion, i.e., we see that universality and uniformity are enough for RPA-security, while we require universal$_C$ and uniform$_C$ notions for INT-security. It also makes possible to obtain better security bounds for a keyed hash function such that $\epsilon_1 < \epsilon_3$ and $\epsilon_2 < \epsilon_4$.

The following lemma is the information theoretic counterpart of the above theorem, where a random permutation $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$ is used as the blockcipher. A proof of Theorem 1, given Lemma 1, is standard, e.g., see [2].

Lemma 1. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be an ϵ_1-universal, ϵ_2-uniform, ϵ_3-universal$_C$ and ϵ_4-uniform$_C$ hash function. Then for any A that invokes the oracle at most q times,*

$$\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{HtECB}[H,\mathrm{Perm}(n)]}(A) \leq \frac{2q^2l^2}{2^n} + 2q^2\epsilon_1 + 4q^2l\epsilon_2, \tag{1}$$
$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtECB}[H,\mathrm{Perm}(n)]}(A) \leq \frac{q^2}{2}\epsilon_1 + q^2l\epsilon_2 + q(l+1)\epsilon_2 + \max\{\epsilon_3, \epsilon_4\}. \tag{2}$$

A proof of (1) is simple, and we first present a sketch of it. In the game of RPA, we may without loss of generality assume that A invokes the oracle exactly q times. Let (D_i^0, D_i^1, C_i) be the tuple that A receives in the i-th invocation of the oracle, $(D_i^0[1], \ldots, D_i^0[l]) \xleftarrow{n} D_i^0$ and $(D_i^1[1], \ldots, D_i^1[l]) \xleftarrow{n} D_i^1$ be the partitions, and $D_i^0[0] \leftarrow H_L(D_i^0)$ and $D_i^1[0] \leftarrow H_L(D_i^1)$ be the output values of the hash function. Define $\mathcal{D}_{\mathrm{hash}}^0 = \{D_i^0[0] : 1 \le i \le q\}$, $\mathcal{D}_{\mathrm{hash}}^1 = \{D_i^1[0] : 1 \le i \le q\}$, $\mathcal{D}_{\mathrm{data}}^0 = \{D_i^0[j] : 1 \le i \le q, 1 \le j \le l\}$ and $\mathcal{D}_{\mathrm{data}}^1 = \{D_i^1[j] : 1 \le i \le q, 1 \le j \le l\}$. Suppose that $\mathcal{D}_{\mathrm{hash}}^0 \cup \mathcal{D}_{\mathrm{hash}}^1 \cup \mathcal{D}_{\mathrm{data}}^0 \cup \mathcal{D}_{\mathrm{data}}^1$ does not contain any collisions, i.e., all the elements in $\mathcal{D}_{\mathrm{hash}}^0 \cup \mathcal{D}_{\mathrm{hash}}^1 \cup \mathcal{D}_{\mathrm{data}}^0 \cup \mathcal{D}_{\mathrm{data}}^1$ are distinct. If $C_1, \ldots, C_q$ are generated by the \$HtECB$[H, \mathrm{Perm}(n)]$-$\mathcal{E}_{L,P}^0(\cdot)$ oracle, then $C_1, \ldots, C_q$ consist of the output values of the random permutation at distinct input points, and even if $C_1, \ldots, C_q$ are generated by the \$HtECB$[H, \mathrm{Perm}(n)]$-$\mathcal{E}_{L,P}^1(\cdot)$ oracle, we see that $C_1, \ldots, C_q$ follow exactly the same probability distribution, and hence there is no way that A can distinguish between the two oracles. This implies that A's success probability is bounded by the probability that there are some collision in $\mathcal{D}_{\mathrm{hash}}^0 \cup \mathcal{D}_{\mathrm{hash}}^1 \cup \mathcal{D}_{\mathrm{data}}^0 \cup \mathcal{D}_{\mathrm{data}}^1$, which can be proved to be at most $\frac{2q^2l^2}{2^n} + 2q^2\epsilon_1 + 4q^2l\epsilon_2$ from the randomness of L, $D_1^0, \ldots, D_q^0$, and $D_1^1, \ldots, D_q^1$.

A proof of (2) is more involved, and we show a full proof in Appendix A. In what follows, we present a sketch of it. We may without loss of generality assume that A invokes the oracle exactly q times. Let (D_i, C_i) be the pair that A receives in the i-th invocation of the oracle, $(D_i[1], \ldots, D_i[l]) \xleftarrow{n} D_i$ be the partition, and $D_i[0] \leftarrow H_L(D_i)$ be the output value of the hash function. Let $(C^*[0], \ldots, C^*[l]) \xleftarrow{n} C^*$ be the challenge ciphertext, and $D^*[j] \leftarrow P^{-1}(C^*[j])$ for $0 \le j \le l$ be the corresponding plaintext blocks.

Define $\mathcal{D}_{\mathrm{hash}} = \{D_i[0] : 1 \le i \le q\}$, $\mathcal{D}_{\mathrm{data}} = \{D_i[j] : 1 \le i \le q, 1 \le j \le l\}$, $\mathcal{C}_{\mathrm{hash}} = \{C_i[0] : 1 \le i \le q\}$, $\mathcal{C}_{\mathrm{data}} = \{C_i[j] : 1 \le i \le q, 1 \le j \le l\}$ and $\mathcal{C}_{\mathrm{rest}} = \{0,1\}^n \setminus (\mathcal{C}_{\mathrm{hash}} \cup \mathcal{C}_{\mathrm{data}})$. Note that $\mathcal{C}_{\mathrm{rest}}$ is the set of n-bit strings that are never returned to the adversary, and some of $C^*[0], \ldots, C^*[l]$ may be in this set. We let $\mathcal{D}_{\mathrm{rest}}$ be the corresponding plaintext blocks, i.e., we let $\mathcal{D}_{\mathrm{rest}} = \{D^*[j] : 0 \le j \le l, C^*[j] \in \mathcal{C}_{\mathrm{rest}}\}$.

We first define a bad event and, if this occurs, we give up the analysis and let the adversary win the game of INT. Intuitively, the bad event occurs if one of the elements in $\mathcal{D}_{\mathrm{hash}}$ is used multiple times, i.e., if the uniqueness of $D_1[0], \ldots, D_q[0]$ is lost. The bad event can be broken into the following three cases. Case 1: There is some collision between the elements in $\mathcal{D}_{\mathrm{hash}}$, Case 2: $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{data}} \neq \emptyset$, and Case 3: $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{rest}} \neq \emptyset$. It can be shown that the probabilities are at most $\frac{q^2}{2}\epsilon_1$ for Case 1, $q^2l\epsilon_2$ for Case 2, and $q(l+1)\epsilon_2$ for Case 3, which appear as the first three terms in (2). We note that the analysis of Case 3 requires a careful treatment in that we have to make sure that C^* is independent of $L \xleftarrow{\$} \mathcal{L}$. See Appendix A for details.

Now it remains to show that the probability of A winning the game of INT is at most $\max\{\epsilon_3, \epsilon_4\}$. We present intuitive discussions by showing situations such that our new notions of the keyed hash function prevent the adversary from winning the game, and the classical notions are not sufficient for proving its security.

For simplicity consider the case $l = 3$ and where the adversary A invokes the oracle only once. In this case, A obtains (D_1, C_1) by invoking the oracle, where $D_1 = (D_1[1], D_1[2], D_1[3])$ and $C_1 = (C_1[0], C_1[1], C_1[2], C_1[3])$. At this point, A learns that $C_1[0] = P(H_L(D_1))$, $C_1[1] = P(D_1[1])$, $C_1[2] = P(D_1[2])$ and $C_1[3] = P(D_1[3])$ hold. Then, A outputs a challenge ciphertext $C^* = (C^*[0], C^*[1], C^*[2], C^*[3])$. Recall that A wins the game of INT if $D^*[0] = H_L(D^*[1], D^*[2], D^*[3])$ holds, where $D^*[j] = P^{-1}(C^*[j])$ for $0 \le j \le 3$.

We see that $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$, $C^*[j] \in \mathcal{C}_{\mathrm{data}}$ or $C^*[j] \in \mathcal{C}_{\mathrm{rest}}$ must hold for all $0 \le j \le 3$. For $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$, A knows that the corresponding $D^*[j]$ is a hash value, and knows X such that $D^*[j] = H_L(X)$, while $D^*[j]$ itself is not known to A. For $C^*[j] \in \mathcal{C}_{\mathrm{data}}$, A knows the value of the corresponding $D^*[j]$. For $C^*[j] \in \mathcal{C}_{\mathrm{rest}}$, we fix the randomness for $D^*[j] \leftarrow P^{-1}(C^*[j])$, and treat as if $D^*[j]$ is a fixed constant even though A does not know its value. By doing so, we can treat $\mathcal{C}_{\mathrm{data}}$ and $\mathcal{C}_{\mathrm{rest}}$ identically. We consider two cases depending on the value of $C^*[0]$. Case

A: $C^*[0] \in \mathcal{C}_{\text{hash}}$ and Case B: $C^*[0] \in \mathcal{C}_{\text{data}} \cup \mathcal{C}_{\text{rest}}$. Note that, in our example, $\mathcal{C}_{\text{hash}} = \{C_1[0]\}$, $\mathcal{C}_{\text{data}} = \{C_1[1], C_1[2], C_1[3]\}$ and $\mathcal{C}_{\text{rest}}$ is the remaining n-bit strings.

Case A includes, for instance, $C^* = (C_1[0], C_1[1], C_1[2], C_1[0])$. A wins the game with this C^* if $H_L(D_1) = H_L(D_1[1], D_1[2], H_L(D_1))$. However, we see that this situation is not covered by Definition 1, and it is not enough to conclude that the probability is low. In fact, we show in Sect. 7.1 that there exists a universal hash function such that this sort of probability is high. On the other hand, this is the situation where our new notion of universal$_C$ plays a role. We see that the probability is at most ϵ_3 by setting $X = (D_1[1], D_1[2], H_L(D_1))$ and $X' = D_1$ in Definition 3, and thus A's success probability is at most ϵ_3. Furthermore, for any combinations of $C^*[1], C^*[2], C^*[3] \in \mathcal{C}_{\text{hash}} \cup \mathcal{C}_{\text{data}} \cup \mathcal{C}_{\text{rest}}$, we see that the event $H_L(D_1) = H_L(D^*[1], D^*[2], D^*[3])$ is covered by one of the 2^3 cases considered in Definition 3, since $D^*[j]$ is either "a hash value" or "a fixed constant." Therefore, A's success probability is at most ϵ_3 in Case A.

Case B includes, for instance, $C^* = (C_1[3], C_1[1], C_1[2], C_1[0])$, and A wins the game if $D_1[3] = H_L(D_1[1], D_1[2], H_L(D_1))$. We see that Definition 2 is not enough to conclude that the probability is low, and, as we show in Sect. 7.1, there exists a uniform hash function such that the probability is high. On the other hand, this is covered by our new notion of uniform$_C$, and the probability is at most ϵ_4 by setting $X = (D_1[1], D_1[2], H_L(D_1))$ and $Y = D_1[3]$ in Definition 4. Furthermore, any combinations of $C^*[1], C^*[2], C^*[3] \in \mathcal{C}_{\text{hash}} \cup \mathcal{C}_{\text{data}} \cup \mathcal{C}_{\text{rest}}$ is one of the 2^3 cases in Definition 4, and therefore, A's success probability is at most ϵ_4 in Case B.

6 Analysis of Hash-then-CBC Scheme

As in Table 1, the security of a combination of CBC mode and a universal hash function has been left as an open problem. We present a partial answer to the open problem by showing that the HtCBC scheme is secure if H is a universal, uniform, universal$_{CC}$ and uniform$_{CC}$ hash function.

Theorem 2. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be an ϵ_1-universal, ϵ_2-uniform, ϵ_5-universal$_{CC}$ and ϵ_6-uniform$_{CC}$ hash function, and $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then for any A that invokes the oracle at most q times, there exist adversaries A' and A'' such that*

$$\mathbf{Adv}^{\text{rpa}}_{\text{HtCBC}[H,E]}(A) \le \mathbf{Adv}^{\text{prp}}_{E}(A') + 2q^2\epsilon_1 + \frac{14q^2(l+1)^2}{2^n},$$
$$\mathbf{Adv}^{\text{int}}_{\text{HtCBC}[H,E]}(A) \le \mathbf{Adv}^{\text{sprp}}_{E}(A'') + \frac{q^2}{2}\epsilon_1 + q^2 l\epsilon_2 + q(l+1)\epsilon_2 + \max\{\epsilon_5, \epsilon_6\},$$

where A' makes at most $q(l+1)$ queries and A'' makes at most $(q+1)(l+1)$ queries. Furthermore, $\text{Time}(A') = \text{Time}(A) + O(nlq)$ *and* $\text{Time}(A'') = \text{Time}(A) + O(nlq)$.

The above theorem can be proved from the following lemma, where we use a random permutation $P(\cdot) \xleftarrow{\$} \text{Perm}(n)$ as the blockcipher, by following the standard argument, e.g., see [2].

Lemma 2. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be an ϵ_1-universal, ϵ_2-uniform, ϵ_5-universal$_{CC}$ and ϵ_6-uniform$_{CC}$ hash function. Then for any A that invokes the oracle at most q times,*

$$\mathbf{Adv}^{\text{rpa}}_{\text{HtCBC}[H,\text{Perm}(n)]}(A) \le 2q^2\epsilon_1 + \frac{14q^2(l+1)^2}{2^n}, \tag{3}$$
$$\mathbf{Adv}^{\text{int}}_{\text{HtCBC}[H,\text{Perm}(n)]}(A) \le \frac{q^2}{2}\epsilon_1 + q^2 l\epsilon_2 + q(l+1)\epsilon_2 + \max\{\epsilon_5, \epsilon_6\}. \tag{4}$$

A proof of (3) is similar to that of (1), and we present a brief sketch of it. We use the same notation of (D_i^0, D_i^1, C_i), $(D_i^0[1], \ldots, D_i^0[l]) \xleftarrow{n} D_i^0$, $(D_i^1[1], \ldots, D_i^1[l]) \xleftarrow{n} D_i^1$, $D_i^0[0] \leftarrow H_L(D_i^0)$ and $D_i^1[0] \leftarrow H_L(D_i^1)$. Let $I_i^0[j] \leftarrow D_i^0[j] \oplus C_i[j-1]$ and $I_i^1[j] \leftarrow D_i^1[j] \oplus C_i[j-1]$ be the input values of the blockcipher. Define $\mathcal{D}_{\text{hash}}^0 = \{D_i^0[0] : 1 \le i \le q\}$, $\mathcal{D}_{\text{hash}}^1 = \{D_i^1[0] : 1 \le i \le q\}$, $\mathcal{D}_{\text{data}}^0 = \{I_i^0[j] : 1 \le i \le q, 1 \le j \le l\}$ and $\mathcal{D}_{\text{data}}^1 = \{I_i^1[j] : 1 \le i \le q, 1 \le j \le l\}$. Then A's success probability is bounded by the probability that there are some collision in $\mathcal{D}_{\text{hash}}^0 \cup \mathcal{D}_{\text{hash}}^1 \cup \mathcal{D}_{\text{data}}^0 \cup \mathcal{D}_{\text{data}}^1$, and this can be proved to be at most $2q^2\epsilon_1 + \frac{14q^2(l+1)^2}{2^n}$.

A full proof of (4) is presented in Appendix B, and a sketch is shown below. As in the analysis of (2), we first exclude a bad event by giving up the analysis and letting the adversary win the game. The bad event is essentially the same as the one defined in the analysis of (2), and this corresponds to the first three terms in (4).

We next discuss that the probability of A winning the game of INT is at most $\max\{\epsilon_5, \epsilon_6\}$. We again consider the case $l = 3$ and where the adversary A invokes the oracle only once, and illustrate situations such that our new notions prevent the adversary from winning the game. Now A obtains (D_1, C_1) by invoking the oracle, where $D_1 = (D_1[1], D_1[2], D_1[3])$ and $C_1 = (C_1[0], C_1[1], C_1[2], C_1[3])$. Let $(I_1[1], I_1[2], I_1[3]) = (D_1[1] \oplus C_1[0], D_1[2] \oplus C_1[1], D_1[3] \oplus C_1[2])$ be the input values of the blockcipher, which are known to A. Then A outputs a challenge ciphertext C^*. We consider $C^* = (C_1[0], C_1[1], C_1[2], C_1[0])$ and $(C_1[3], C_1[1], C_1[2], C_1[0])$ that we used as the examples of the analysis of (2).

For $C^* = (C_1[0], C_1[1], C_1[2], C_1[0])$, A wins the game if $H_L(D_1) = H_L(I_1[1] \oplus C_1[0], I_1[2] \oplus C_1[1], H_L(D_1) \oplus C_1[2])$, which is not covered by Definition 1 nor Definition 3, while this is the situation where our notion of universal$_{CC}$ covers. We see that the probability is at most ϵ_5 by setting $X = (I_1[1] \oplus C_1[0], I_1[2] \oplus C_1[1], H_L(D_1) \oplus C_1[2])$ and $X' = D_1$ in Definition 5, and thus A's success probability is at most ϵ_5. By a similar argument, the probability of $D^*[0] = H_L(D^*[1], D^*[2], D^*[3])$ is at most ϵ_5 if $P^{-1}(C^*[0])$ is "a hash value."

For $C^* = (C_1[3], C_1[1], C_1[2], C_1[0])$, A wins the game if $I_1[3] = H_L(I_1[1] \oplus C_1[3], I_1[2] \oplus C_1[1], H_L(D_1) \oplus C_1[2])$, which is not covered by Definition 2 nor Definition 4, but it is one of the cases covered by Definition 6. We see that the probability is at most ϵ_6 by setting $X = (I_1[1] \oplus C_1[3], I_1[2] \oplus C_1[1], H_L(D_1) \oplus C_1[2])$ and $Y = I_1[3]$. By a similar argument, the probability of $D^*[0] = H_L(D^*[1], D^*[2], D^*[3])$ is at most ϵ_6 if $P^{-1}(C^*[0])$ is "a fixed constant."

7 Discussions on Hash Functions

7.1 Relations between the Notions

In this subsection, we show that the new notions are strictly stronger the classical notions. Let $U, U' \in \{\text{universal}, \text{uniform}, \text{universal}_C, \text{uniform}_C, \text{universal}_{CC}, \text{uniform}_{CC}\}$. We write $U \to U'$ if an ϵ-U hash function, for sufficiently small ϵ, is always an ϵ'-U' hash function, for sufficiently small ϵ'. Otherwise we write $U \not\to U'$.

We first consider the universality. From the definitions, we obviously have universal$_{CC} \to$ universal$_C$, universal$_{CC} \to$ universal, and universal$_C \to$ universal.

Now we show universal $\not\to$ universal$_C$. Consider a keyed hash function $H : \{0,1\}^n \times \{0,1\}^{3n} \to \{0,1\}^n$ such that $H_L(X[1], X[2], X[3]) = L^2 \cdot X[1] \oplus L \cdot X[2] \oplus X[3]$, where the multiplication is over GF(2^n). We see that H is a $\frac{2}{2^n}$-universal hash function, but $\Pr[H_L(X) = H_L(X')] = 1$, where $X = (X[1], X[2], H_L(X[1], X[2], X[3]))$ and $X' = (0^n, 0^n, X[3])$, and hence it is not a universal$_C$ hash function. This also implies universal $\not\to$ universal$_{CC}$, and hence the new notions, universal$_C$ and universal$_{CC}$, are strictly stronger than the universality.

As for the uniformity, from the definitions, we obviously have uniform$_{CC} \to$ uniform$_C$, uniform$_{CC} \to$ uniform and uniform$_C \to$ uniform.

We show uniform $\not\to$ uniform$_C$. Consider a keyed hash function $H : \{0,1\}^n \times \{0,1\}^{3n} \to \{0,1\}^n$ such that $H_L(X[1], X[2], X[3]) = L^3 \oplus L^2 \cdot X[1] \oplus L \cdot X[2] \oplus X[3]$, where the multiplication is over GF(2^n). We see that H is a $\frac{3}{2^n}$-uniform hash function, but $\Pr[H_L(X) = Y] = 1$, where $X = (X[1], X[2], H_L(X[1], X[2], X[3]))$ and $Y = X[3]$, and hence it is not a uniform$_C$ hash function. This also implies uniform $\not\to$ uniform$_{CC}$.

7.2 Construction of a Hash Function

The discussions in the previous section indicate that new notions that we propose are strictly stronger than a universal and uniform hash function. However, we show that a monic polynomial hash function suffices to obtain an ϵ_1-universal, ϵ_2-uniform, ϵ_3-universal$_C$, ϵ_4-uniform$_C$, ϵ_5-universal$_{CC}$ and ϵ_6-uniform$_{CC}$ hash function for sufficiently small $\epsilon_1, \ldots, \epsilon_6$.

Let $\mathcal{L} = \{0,1\}^n$ and define a keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ by

$$H_L(X) = L^{l+1} \oplus L^l \cdot X[1] \oplus \cdots \oplus L \cdot X[l], \tag{5}$$

where $X = (X[1], \ldots, X[l]) \in \{0,1\}^{nl}$ and the multiplication is over GF(2^n).

Observe that the construction is close to the polynomial hash function, the basic construction of a universal hash function used, e.g., in [16, 18, 4], except that we always have a leading term L^{l+1}. We note that it can be implemented easily from the polynomial hash function. That is, from a polynomial hash function $G_L(X) = L^{l+1} \cdot X[1] \oplus L^l \cdot X[2] \oplus \cdots \oplus L \cdot X[l+1]$, the keyed hash function defined in (5) can be obtained by $G_L(1_n, X[2], \ldots, X[l+1])$, where 1_n is the n-bit representation of integer 1. Furthermore, we note that the keyed hash function is efficient. It can be implemented with l multiplications by using Horner's rule.

The following lemma shows that the keyed hash function defined in (5) satisfies all the notions with sufficiently small $\epsilon_1, \ldots, \epsilon_6$.

Lemma 3. *The keyed hash function $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$, where $\mathcal{L} = \{0,1\}^n$, defined in (5) is a $\frac{l}{2^n}$-universal, $\frac{l+1}{2^n}$-uniform, $\frac{2l+1}{2^n}$-universal$_C$, $\frac{2l+1}{2^n}$-uniform$_C$, $\frac{2l+1}{2^n}$-universal$_{CC}$, and $\frac{2l+1}{2^n}$-uniform$_{CC}$ hash function.*

Proof. Let $X_1, \ldots, X_l \in \{0,1\}^{nl}, Z[1], \ldots, Z[l] \in \{0,1\}^n, V[1], \ldots, V[l] \in \{0,1\}^n$, $X' \in \{0,1\}^{nl}$ and $Y \in \{0,1\}^n$ be arbitrary bit strings such that $X' \neq (Z[1], \ldots, Z[l])$.

It is easy to show that H is a $\frac{l}{2^n}$-universal and $\frac{l+1}{2^n}$-uniform hash function. We first show that H is a $\frac{2l+1}{2^n}$-universal$_C$ hash function. Let $\mathcal{Z} = \{Z[1], \ldots, Z[l]\}$ and $\mathcal{H} = \{H_L(X_1), \ldots, H_L(X_l)\}$. Recall that $X = (X[1], \ldots, X[l])$, and for each $1 \le j \le l$, we have $X[j] \in \{Z[j], H_L(X_j)\}$. We say $X[j]$ is chosen from $\mathcal{Z}$ if $X[j] = Z[j]$, otherwise we say it is chosen from $\mathcal{H}$. If all blocks of X are chosen from $\mathcal{Z}$, i.e., if $X = (Z[1], \ldots, Z[l])$, we have $\Pr[H_L(X) = H_L(X')] \le \frac{l}{2^n}$ since $X' \neq (Z[1], \ldots, Z[l])$ and H is a $\frac{l}{2^n}$-universal hash function. Next, let $1 \le s \le l$ be an integer and consider the case where $l - s$ blocks of X are chosen from $\mathcal{Z}$ and s blocks are chosen from $\mathcal{H}$. Without loss of generality, let $i_1, \ldots, i_s$ be the s indexes such that $1 \le i_1 < \cdots < i_s \le l$ and $X[i_1], \ldots, X[i_s]$ are the s blocks that are chosen from $\mathcal{H}$. Then $H_L(X)$ can be written as

$$H_L(X) = L^{l+1} \oplus L^l \cdot Z[1] \oplus \cdots \oplus L^{l+1-i_1} \cdot H_L(X_{i_1}) \oplus \cdots \oplus L \cdot Z[l], \tag{6}$$

by substituting $X[i_1] = H_L(X_{i_1})$. Now we see that the right hand side of (6) has a term L^{2l+2-i_1} since $H_L(X_{i_1})$ itself has a term L^{l+1}. We also see that the term L^{2l+2-i_1} is not canceled when we simplify the equation $H_L(X) = H_L(X')$, because the highest degree term in the right hand side is L^{l+1}, and we have $2l + 2 - i_1 \ge l + 2$. Therefore, $H_L(X) = H_L(X')$ is a non-trivial equation in L of degree $2l + 2 - i_1 \le 2l + 1$, which implies that $\Pr[H_L(X) = H_L(X')] \le \frac{2l+1}{2^n}$.

We next show that H is a $\frac{2l+1}{2^n}$-uniform$_C$ hash function. If $X = (Z[1], \ldots, Z[l])$, i.e., if all blocks are chosen from $\mathcal{Z}$, then we have $\Pr[H_L(X) = Y] \leq \frac{l+1}{2^n}$ since H is a $\frac{l+1}{2^n}$-uniform hash function. If $l - s$ blocks of X are chosen from $\mathcal{Z}$ and s blocks are chosen from $\mathcal{H}$ for some $1 \leq s \leq l$, we see that $H_L(X) = Y$ is a non-trivial equation in L of degree at most $2l + 1$ by following a similar argument as above, which implies that $\Pr[H_L(X) = Y] \leq \frac{2l+1}{2^n}$.

In order to show that H is a $\frac{2l+1}{2^n}$-universal$_{CC}$ and $\frac{2l+1}{2^n}$-uniform$_{CC}$ hash function, we consider $\mathcal{Z} = \{Z[1], \ldots, Z[l]\}$ and $\mathcal{H}' = \{H_L(X_1) \oplus V[1], \ldots, H_L(X_l) \oplus V[l]\}$. We see that, changing $\mathcal{H}$ to $\mathcal{H}'$ does not affect the above arguments because the highest degree term in $H_L(X)$ is not affected by the constants $V[1], \ldots, V[l]$, and we have both $\Pr[H_L(X) = H_L(X')] \leq \frac{2l+1}{2^n}$ and $\Pr[H_L(X) = Y] \leq \frac{2l+1}{2^n}$ for any $X \in \{Z[1], H_L(X_1) \oplus V[1]\} \times \cdots \times \{Z[l], H_L(X_l) \oplus V[l]\}$. □

We remark that a PRF (PseudoRandom Function) $F_L : \{0,1\}^{nl} \to \{0,1\}^n$ is another example of an ϵ_1-universal, ϵ_2-uniform, ϵ_3-universal$_C$, ϵ_4-uniform$_C$, ϵ_5-universal$_{CC}$ and ϵ_6-uniform$_{CC}$ hash function for sufficiently small $\epsilon_1, \ldots, \epsilon_6$. For example, the CBC MAC based on a PRP is a PRF [2], and hence it satisfies the six notions.

7.3 Applications

By applying Lemma 3 to Theorem 1, we obtain the following corollary.

Corollary 1. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be the keyed hash function defined in (5), and $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then for any A that invokes the oracle at most q times, there exist adversaries A' and A'' such that*

$$\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{HtECB}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(A') + \frac{8q^2(l+1)^2}{2^n},$$
$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtECB}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{sprp}}_{E}(A'') + \frac{3q^2(l+1)^2}{2^n},$$

where A' makes at most $q(l+1)$ queries and A'' makes at most $(q+1)(l+1)$ queries. Furthermore, $\mathrm{Time}(A') = \mathrm{Time}(A) + O(nlq)$ *and* $\mathrm{Time}(A'') = \mathrm{Time}(A) + O(nlq)$.

By applying Lemma 3 to Theorem 2, we obtain the following result.

Corollary 2. *Let $H : \mathcal{L} \times \{0,1\}^{nl} \to \{0,1\}^n$ be the keyed hash function defined in (5), and $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then for any A that invokes the oracle at most q times, there exist adversaries A' and A'' such that*

$$\mathbf{Adv}^{\mathrm{rpa}}_{\mathrm{HtCBC}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(A') + \frac{16q^2(l+1)^2}{2^n},$$
$$\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtCBC}[H,E]}(A) \leq \mathbf{Adv}^{\mathrm{sprp}}_{E}(A'') + \frac{3q^2(l+1)^2}{2^n},$$

where A' makes at most $q(l+1)$ queries and A'' makes at most $(q+1)(l+1)$ queries. Furthermore, $\mathrm{Time}(A') = \mathrm{Time}(A) + O(nlq)$ *and* $\mathrm{Time}(A'') = \mathrm{Time}(A) + O(nlq)$.

8 Conclusions

In this paper, we proposed a total of four new notions of a keyed hash function. Based on the new notions, we showed that HtECB and HtCBC schemes are secure AKW schemes, and the result on the HtCBC scheme partially solves the open problem posed by Gennaro and Halevi [9].

We also showed that there exists an efficient construction of a keyed hash function that satisfies all the six notions considered in this paper.

There are several interesting open problems. First, we still do not know the security of the HtCBC scheme that is based only on the universality assumption. Next, it remains to identify the missing relations between the notions of a keyed hash function. In particular, we do not know if $\text{universal}_C \rightarrow \text{universal}_{CC}$ and $\text{uniform}_C \rightarrow \text{uniform}_{CC}$ hold. Also, it would be interesting to examine if existing universal hash functions, e.g., MMH, Square Hash, NMH, and NH, satisfy the notions proposed in this paper, or to see if these hash functions can be used in a black-box manner to obtain a construction that satisfies the new notions.

Acknowledgments

The authors would like to thank Joan Daemen for pointing out a reference and the anonymous reviewers for insightful comments. A part of this work was supported by MEXT KAKENHI, Grant-in-Aid for Young Scientists (A), 22680001.

References

1. ANSI: Symmetric Key Cryptography for the Financial Services Industry — Wrapping of Keys and Associated Data. X9.102-2008 (2008)
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. 61 (3), 362–399, 2000
3. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
4. Bernstein, D.J.: The Poly1305-AES Message-Authentication Code. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 32–49. Springer, Heidelberg (2005)
5. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and Secure Message Authentication. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 216–233. Springer, Heidelberg (1999)
6. Carter, J.L., Wegman, M.N.: Universal Classes of Hash Functions. J. Comput. Syst. Sci. 18 (2), 143–154, 1979
7. Chakraborty, D., Mancillas-López, C.: Double Ciphertext Mode: A Proposal for Secure Backup. IACR Cryptology ePrint Archive: Report 2010/369, 2010
8. Etzel, M., Patel, S., Ramzan, Z.: Square Hash: Fast Message Authentication via Optimized Universal Hash Functions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 234–251. Springer, Heidelberg (1999)
9. Gennaro, R., Halevi, S.: More on Key Wrapping. In: Jacobson, J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 53–70. Springer, Heidelberg (2009)
10. Halevi, S., Krawczyk, H.: MMH: Software Message Authentication in the Gbit/second Rates. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 172–189. Springer, Heidelberg (1997)
11. Halevi, S., Krawczyk, H.: One-Pass HMQV and Asymmetric Key-Wrapping. In: Gennaro, R. (ed.) PKC 2011. to appear in LNCS. Springer, Heidelberg (2011)
12. Hoyer, P., Pei, M., Machani, S.: Portable Symmetric Key Container (PSKC). IETF RFC 6030 (2010)
13. Iwata, T., Yasuda, K.: HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 394–415. Springer, Heidelberg (2009)
14. Iwata, T., Yasuda, K.: BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In: Jacobson, J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 313–330. Springer, Heidelberg (2009)
15. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17 (2), 373–386, 1988
16. McGrew, D., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)
17. NIST: Recommendation for Block Cipher Modes of Operation, Methods and Techniques. NIST Special Publication 800-38A 2001 Edition, 2001
18. NIST: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007
19. NIST: `http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html`

20. OASIS: Key Management Interoperability Protocol Specification Version 1.0. http://www.oasis-open.org/ (2010)
21. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
22. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
23. Wegman, M.N., Carter, J.L.: New Hash Functions and Their Use in Authentication and Set Equality. J. Comput. Syst. Sci. 22 (3), 265–279, 1981

A Proof of Lemma 1, (2)

We present a proof of Lemma 1, (2) based on the code based game playing proof [3]. Without loss of generality, we assume that A is deterministic and invokes the $\$\mathrm{HtECB}[H, \mathrm{Perm}(n)]\text{-}\mathcal{E}_{L,P}(\cdot)$ oracle exactly q times. Let (D_i, C_i) be the plaintext-ciphertext pair that A receives in the i-th invocation of the oracle, and let $(D_i[1], \ldots, D_i[l]) \xleftarrow{n} D_i$ and $(C_i[0], \ldots, C_i[l]) \xleftarrow{n} C_i$ be their partitions. By invoking the oracle q times, A receives $(D_1, C_1), \ldots, (D_q, C_q)$, and A subsequently outputs a challenge ciphertext C^*. Without loss of generality, we assume that $C^* \notin \{C_1, \ldots, C_q\}$. We consider four games, Game $G_0, \ldots, G_3$, to complete our proof.

The first game, Game G_0, precisely simulates the $\$\mathrm{HtECB}[H, \mathrm{Perm}(n)]\text{-}\mathcal{E}_{L,P}(\cdot)$ oracle. In Game G_0 we first choose $L \xleftarrow{\$} \mathcal{L}$ and $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$ in the initialization phase, and when A invokes the oracle, the game chooses $D_i \xleftarrow{\$} \{0,1\}^{nl}$ and returns (D_i, C_i), where $C_i \leftarrow \mathrm{HtECB}[H, \mathrm{Perm}(n)]\text{-}\mathcal{E}_{L,P}(D_i)$. When A outputs the challenge ciphertext C^* after the q invocations of the oracle, the game returns 1 iff $\perp \neq \mathrm{HtECB}[H, \mathrm{Perm}(n)]\text{-}\mathcal{D}_{L,P}(C^*)$ holds. We write $G_0^A \Rightarrow 1$ for the event that Game G_0 interacting with A returns 1 (and we use a similar notation for the remaining games), and we obviously have $\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtECB}[H,\mathrm{Perm}(n)]}(A) = \Pr[G_0^A \Rightarrow 1]$.

We next define Game G_1. First, instead of choosing the random permutation $P(\cdot)$ at the initialization phase, Game G_1 uses the lazy sampling [3]. That is, the array $P(X)$ is initialized to "undefined" for all $X \in \{0,1\}^n$ and we maintain two sets, $\mathcal{D}_P$ and $\mathcal{R}_P$, which keep the record of domain and range points, respectively, that have already used. When we need a value of $P(X)$, the value is $Y \in \mathcal{R}_P$ such that $Y = P(X)$ if $X \in \mathcal{D}_P$. Otherwise we let $Y \xleftarrow{\$} \overline{\mathcal{R}_P}$ and set $P(X) \leftarrow Y$, where $\overline{\mathcal{R}_P} = \{0,1\}^n \setminus \mathcal{R}_P$. We then let $\mathcal{D}_P \leftarrow \mathcal{D}_P \cup \{X\}$ and $\mathcal{R}_P \leftarrow \mathcal{R}_P \cup \{Y\}$. Similarly, when we need a value of $P^{-1}(Y)$, the value is $X \in \mathcal{D}_P$ such that $Y = P(X)$ if $Y \in \mathcal{R}_P$. Otherwise we let $X \xleftarrow{\$} \overline{\mathcal{D}_P}$ and set $P(X) \leftarrow Y$, which is equivalent to set $P^{-1}(Y) \leftarrow X$, where $\overline{\mathcal{D}_P} = \{0,1\}^n \setminus \mathcal{D}_P$. We then let $\mathcal{D}_P \leftarrow \mathcal{D}_P \cup \{X\}$ and $\mathcal{R}_P \leftarrow \mathcal{R}_P \cup \{Y\}$. Using the lazy sampling does not change A's success probability. Next, we see that since A can only invoke the oracle, the adaptivity does not help in increasing or decreasing A's success probability. Therefore, instead of asking A to invoke the oracle q times, we assume that our adversary A can invoke the oracle only once, and when invoked, we return the q plaintext-ciphertext pairs, $(D_1, C_1), \ldots, (D_q, C_q)$, simultaneously. This does not change A's success probability. Third, since $D_1, \ldots, D_q$ are chosen randomly from $\{0,1\}^{nl}$, $D_i = D_{i'}$ may hold for some $1 \leq i' < i \leq q$. We see that, unlike in the game of RPA, the collision in $\{D_1, \ldots, D_q\}$ does not increase A's success probability, and hence when we choose D_i, we let $D_i \xleftarrow{\$} \{0,1\}^{nl} \setminus \{D_1, \ldots, D_{i-1}\}$ rather than $D_i \xleftarrow{\$} \{0,1\}^{nl}$. This only increases A's success probability. Finally, we maintain a bad flag, which is initialized to false. This flag gets set if some "bad event" occurs. Let $D_i[0] \leftarrow H_L(D_i)$ for $1 \leq i \leq q$. Define $\mathcal{D}_{\mathrm{hash}} = \{D_i[0] : 1 \leq i \leq q\}$ and $\mathcal{D}_{\mathrm{data}} = \{D_i[j] : 1 \leq i \leq q, 1 \leq j \leq l\}$. The bad flag gets set if there is some collision within the elements of $\mathcal{D}_{\mathrm{hash}}$, or some element of $\mathcal{D}_{\mathrm{hash}}$ and some element of $\mathcal{D}_{\mathrm{data}}$ collide, i.e., if $D_i[0] = D_{i'}[0]$ holds for some $1 \leq i' < i \leq q$, or $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{data}} \neq \emptyset$. Furthermore, when the bad flag gets set, we let the adversary win the game, i.e., we let $G_1^A \Rightarrow 1$. Since this only increases the probability that the game returns 1, overall, we have $\Pr[G_0^A \Rightarrow 1] \leq \Pr[G_1^A \Rightarrow 1]$.

Next, we define Game G_2 in Fig. 5. After the initialization and when A invokes the oracle, Procedure (invoke) is curried out to generate the q plaintext-ciphertext pairs. When A outputs a challenge ciphertext $C^* = (C^*[0], \ldots, C^*[l])$, Procedure (C^*) is curried out to test if the adversary wins the game, i.e., if $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ holds where $D^*[j] \leftarrow P^{-1}(C^*[j])$ for $0 \le i \le l$. The procedure is followed by the finalization to return the outcome of the game, which is maintained by a win flag. We note that the lazy sampling is implicit, i.e., $P(X)$ is initialized to "undefined" for all $X \in \{0,1\}^n$, and two sets $\mathcal{D}_P$ and $\mathcal{R}_P$ are initialized to empty, and they are updated when "$P(X) \leftarrow Y$" is carried out for some X and Y. The initialization of the flags is also implicit. Game G_2 is obtained from Game G_1 as follows. In Game G_1, A does not see $D_1[0], \ldots, D_q[0]$, and we delay the computation of these hash values until line 12, which is after A outputs the challenge ciphertext. Accordingly, we generate $C_1[0], \ldots, C_q[0]$ without specifying the corresponding input values, but assuming that the input values are all distinct. Besides, as we have delayed the computation of the hash values, we delay the evaluation of the bad flag until line 13. Game G_2 is designed to set the bad flag iff the flag gets set in Game G_1. Furthermore, we see that whenever Game G_1 returns 1, Game G_2 returns 1. Therefore, we have $\Pr[G_1^A \Rightarrow 1] \le \Pr[G_2^A \Rightarrow 1]$.

The terminal game, Game G_3, is defined in Fig. 5. There is no initialization in Game G_3 (except for the implicit initialization of the lazy sampling and the flags), and the sampling of $L \xleftarrow{\$} \mathcal{L}$ is delayed until line 14. Procedure (invoke) and finalization are identical to those in Game G_2. Now we define three sets, $\mathcal{C}_{\text{hash}}$, $\mathcal{C}_{\text{data}}$ and $\mathcal{C}_{\text{rest}}$. Let $\mathcal{C}_{\text{hash}} = \{C_i[0] : 1 \le i \le q\}$, $\mathcal{C}_{\text{data}} = \{C_i[j] : 1 \le i \le q, 1 \le j \le l\}$ and $\mathcal{C}_{\text{rest}} = \{0,1\}^n \setminus (\mathcal{C}_{\text{hash}} \cup \mathcal{C}_{\text{data}})$. For the challenge ciphertext $C^* = (C^*[0], \ldots, C^*[l])$, some blocks may be completely new, i.e., $C^*[j] \in \mathcal{C}_{\text{rest}}$ may hold for some $0 \le j \le l$. In Game G_2, these blocks are decrypted in line 17, but in Game G_3, these blocks are decrypted in line 13, which is before the sampling of $L \xleftarrow{\$} \mathcal{L}$. Adding these blocks to $\mathcal{R}_P$ and the corresponding input blocks to $\mathcal{D}_P$ only increases the probability that the bad flag gets set. Furthermore, we see Game G_3 always returns 1 whenever Game G_2 does, which implies that $\Pr[G_2^A \Rightarrow 1] \le \Pr[G_3^A \Rightarrow 1]$.

Now we have $\mathbf{Adv}^{\text{int}}_{\text{HtECB}[H,\text{Perm}(n)]}(A) \le \Pr[G_3^A \Rightarrow 1]$, which can be bounded as

$$\Pr[G_3^A \Rightarrow 1] \le \Pr[G_3^A \text{ sets bad}] \tag{7}$$
$$+ \Pr[G_3^A \text{ sets win in line 21}]. \tag{8}$$

It remains to evaluate (7) and (8).

First, we show (7) $\le \frac{q^2}{2}\epsilon_1 + q^2 l \epsilon_2 + q(l+1)\epsilon_2$. Recall that $\mathcal{D}_{\text{hash}} = \{D_i[0] : 1 \le i \le q\}$ and $\mathcal{D}_{\text{data}} = \{D_i[j] : 1 \le i \le q, 1 \le j \le l\}$. Let $\mathcal{D}_{\text{rest}}$ be the set of $D^*[j]$'s that are chosen in line 13. We derive the claimed bound by fixing the randomness used to choose these $D^*[j]$'s. In other words, we treat these $D^*[j]$'s as constants, and consider only $L \xleftarrow{\$} \mathcal{L}$ as the source of randomness. There are three cases to consider. Case 1: $D_i[0] = D_{i'}[0]$ holds for some $1 \le i' < i \le q$, Case 2: $\mathcal{D}_{\text{hash}} \cap \mathcal{D}_{\text{data}} \ne \emptyset$, and Case 3: $\mathcal{D}_{\text{hash}} \cap \mathcal{D}_{\text{rest}} \ne \emptyset$.

Case 1: For any fixed $1 \le i' < i \le q$, we have $\Pr[D_i[0] = D_{i'}[0]] = \Pr[H_L(D_i) = H_L(D_{i'})] \le \epsilon_1$ since H is an ϵ_1-universal hash function. Therefore, $\Pr[G_3^A \text{ sets bad in Case 1}] \le \frac{q^2}{2}\epsilon_1$ as there are at most $\frac{q^2}{2}$ choices for (i, i').

Case 2: For any fixed $1 \le i, i' \le q$ and $1 \le j' \le l$, we have $\Pr[D_i[0] = D_{i'}[j']] = \Pr[H_L(D_i) = D_{i'}[j']] \le \epsilon_2$ since H is an ϵ_2-uniform hash function. Therefore, $\Pr[G_3^A \text{ sets bad in Case 2}] \le q^2 l \epsilon_2$ as there are at most $q^2 l$ choices for (i, i', j').

Game G_2
Initialize

1. $L \xleftarrow{\$} \mathcal{L}$

Procedure (invoke)

2. **for** $i \leftarrow 1$ **to** q **do**
3. $\quad D_i \xleftarrow{\$} \{0,1\}^{nl} \setminus \{D_1, \ldots, D_{i-1}\}$, $(D_i[1], \ldots, D_i[l]) \xleftarrow{n} D_i$
4. $\quad$ **for** $j \leftarrow 1$ **to** l **do**
5. $\quad\quad$ **if** $D_i[j] \in \mathcal{D}_P$ **then** $C_i[j] \leftarrow P(D_i[j])$
6. $\quad\quad$ **else** $C_i[j] \xleftarrow{\$} \overline{\mathcal{R}_P}$, $P(D_i[j]) \leftarrow C_i[j]$
7. **for** $i \leftarrow 1$ **to** q **do**
8. $\quad C_i[0] \xleftarrow{\$} \overline{\mathcal{R}_P \cup \{C_1[0], \ldots, C_{i-1}[0]\}}$, $C_i \leftarrow (C_i[0], \ldots, C_i[l])$
9. **return** $((D_1, C_1), \ldots, (D_q, C_q))$

Procedure (C^*)

10. $(C^*[0], \ldots, C^*[l]) \xleftarrow{n} C^*$
11. **for** $i \leftarrow 1$ **to** q **do**
12. $\quad D_i[0] \leftarrow H_L(D_i[1], \ldots, D_i[l])$
13. $\quad$ **if** $D_i[0] \in \mathcal{D}_P$ **then** bad $\leftarrow$ true, win $\leftarrow$ true, **go Finalize**
14. $\quad P(D_i[0]) \leftarrow C_i[0]$
15. **for** $j \leftarrow 0$ **to** l **do**
16. $\quad$ **if** $C^*[j] \in \mathcal{R}_P$ **then** $D^*[j] \leftarrow P^{-1}(C^*[j])$
17. $\quad$ **else** $D^*[j] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(D^*[j]) \leftarrow C^*[j]$
18. **if** $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ **then** win $\leftarrow$ true, **go Finalize**

Finalize

19. **return** win

Game G_3
Procedure (invoke)
// same as Game G_2
Procedure (C^*)

10. $(C^*[0], \ldots, C^*[l]) \xleftarrow{n} C^*$
11. **for** $j \leftarrow 1$ **to** l **do**
12. $\quad$ **if** $C^*[j] \notin \mathcal{R}_P \cup \{C_1[0], \ldots, C_q[0]\}$
13. $\quad\quad$ **then** $D^*[j] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(D^*[j]) \leftarrow C^*[j]$
14. $L \xleftarrow{\$} \mathcal{L}$
15. **for** $i \leftarrow 1$ **to** q **do**
16. $\quad D_i[0] \leftarrow H_L(D_i[1], \ldots, D_i[l])$
17. $\quad$ **if** $D_i[0] \in \mathcal{D}_P$ **then** bad $\leftarrow$ true, win $\leftarrow$ true, **go Finalize**
18. $\quad P(D_i[0]) \leftarrow C_i[0]$
19. **for** $j \leftarrow 0$ **to** l **do**
20. $\quad D^*[j] \leftarrow P^{-1}(C^*[j])$
21. **if** $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ **then** win $\leftarrow$ true, **go Finalize**

Finalize

22. **return** win

Fig. 5. Game G_2 (top) and G_3 (bottom). Procedure (invoke) in Game G_3 is identical to that in Game G_2.

Case 3: For any fixed $1 \le i \le q$ and j such that $D^*[j] \in \mathcal{D}_{\mathrm{rest}}$, we have $\Pr[D_i[0] = D^*[j]] = \Pr[H_L(D_i) = D^*[j]] \le \epsilon_2$ since H is an ϵ_2-uniform hash function. Therefore, we obtain that $\Pr[G_3^A \text{ sets bad in Case 3}] \le q(l+1)\epsilon_2$ as there are at most q choices for i, and the cardinality of $\mathcal{D}_{\mathrm{rest}}$ is at most $(l+1)$, which implies that we have at most $(l+1)$ choices for j.

We next prove $(8) \le \max\{\epsilon_3, \epsilon_4\}$ to complete our proof. Recall that $\mathcal{C}_{\mathrm{hash}} = \{C_i[0] : 1 \le i \le q\}$, $\mathcal{C}_{\mathrm{data}} = \{C_i[j] : 1 \le i \le q, 1 \le j \le l\}$ and $\mathcal{C}_{\mathrm{rest}} = \{0,1\}^n \setminus (\mathcal{C}_{\mathrm{hash}} \cup \mathcal{C}_{\mathrm{data}})$. For each $0 \le j \le l$, we have $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$ or $C^*[j] \in \mathcal{C}_{\mathrm{data}} \cup \mathcal{C}_{\mathrm{rest}}$. As in the analysis of (7), we derive the claimed bound based only on $L \xleftarrow{\$} \mathcal{L}$ by fixing the remaining randomness, and in particular $D^*[j]$'s chosen in line 13 are fixed. These $D^*[j]$'s correspond to $P^{-1}(C^*[j])$ such that $C^*[j] \in \mathcal{C}_{\mathrm{rest}}$. We consider two cases depending on how $C^*[0]$ is chosen by the adversary. Case A: $C^*[0] \in \mathcal{C}_{\mathrm{hash}}$ and Case B: $C^*[0] \in \mathcal{C}_{\mathrm{data}} \cup \mathcal{C}_{\mathrm{rest}}$.

Case A: We have to bound $\Pr[D^*[0] = H_L(D^*[1], \ldots, D^*[l])]$, where $D^*[0] = H_L(D_i)$ for some $1 \le i \le q$. Now for each $1 \le j \le l$, $D^*[j] = H_L(D_{i'})$ for some $1 \le i' \le q$ if $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$, $D^*[j] = D_{i'}[j']$ for some $1 \le i' \le q$ and $1 \le j' \le l$ if $C^*[j] \in \mathcal{C}_{\mathrm{data}}$, and $D^*[j]$ itself is a constant if $C^*[j] \in \mathcal{C}_{\mathrm{rest}}$. In particular, we have $D^*[0] = H_L(X')$, and $D^* = (D^*[1], \ldots, D^*[l]) \in \{Z[1], H_L(X_1)\} \times \cdots \times \{Z[l], H_L(X_l)\}$ for some constants $X_1, \ldots, X_l, Z[1], \ldots, Z[l]$ and X'. This implies that the event $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ must be one of the 2^l cases covered in Definition 3, and we therefore have $\Pr[H_L(D_i) = H_L(D^*[1], \ldots, D^*[l])] \le \epsilon_3$, which proves $(8) \le \epsilon_3$ in this case.

Case B: By a similar argument to Case A, we have $D^*[0] = Y$, and $D^* = (D^*[1], \ldots, D^*[l]) \in \{Z[1], H_L(X_1)\} \times \cdots \times \{Z[l], H_L(X_l)\}$ for some constants $X_1, \ldots, X_l$, $Z[1], \ldots, Z[l]$ and Y, and hence $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ is one of the 2^l events covered in Definition 4. Therefore, we have $(8) \le \epsilon_4$ in this case.

This completes the proof of Lemma 1, (2). □

B Proof of Lemma 2, (4)

A proof is similar to that of Lemma 1, (2). Without loss of generality, we assume that A is deterministic, invokes the $\$\mathrm{HtCBC}[H, \mathrm{Perm}(n)]$-$\mathcal{E}_{L,P}(\cdot)$ oracle exactly q times, and the challenge ciphertext C^* always satisfy $C^* \notin \{C_1, \ldots, C_q\}$. We consider four games, Game $G_0, \ldots, G_3$.

The first game, Game G_0, precisely simulates the $\$\mathrm{HtCBC}[H, \mathrm{Perm}(n)]$-$\mathcal{E}_{L,P}(\cdot)$ oracle, and the game returns 1 iff $\perp \not\leftarrow \mathrm{HtCBC}[H, \mathrm{Perm}(n)]$-$\mathcal{D}_{L,P}(C^*)$. We have $\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtCBC}[H,\mathrm{Perm}(n)]}(A) = \Pr[G_0^A \Rightarrow 1]$.

In the next Game G_1, we use the lazy sampling of $P(\cdot)$, and we change the adversary to invoke the oracle only once by returning the q plaintext-ciphertext pairs, $(D_1, C_1), \ldots, (D_q, C_q)$, simultaneously. We also make sure that we do not choose the same D_i twice by letting $D_i \xleftarrow{\$} \{0,1\}^{nl} \setminus \{D_1, \ldots, D_{i-1}\}$. Finally, we maintain a bad flag, which is set as follows. Let $(D_i[1], \ldots, D_i[l]) \xleftarrow{n} D_i$ and $(C_i[0], \ldots, C_i[l]) \xleftarrow{n} C_i$ be the partitions of D_i and C_i. Let $D_i[0] \leftarrow H_L(D_i)$ for $1 \le i \le q$ and define $\mathcal{D}_{\mathrm{hash}} = \{D_i[0] : 1 \le i \le q\}$. Besides, let $I_i[j] \leftarrow D_i[j] \oplus C_i[j-1]$ for $1 \le i \le q$ and $1 \le j \le l$, and define $\mathcal{D}_{\mathrm{data}} = \{I_i[j] : 1 \le i \le q, 1 \le j \le l\}$. The bad flag gets set if there is some collision within the elements of $\mathcal{D}_{\mathrm{hash}}$, or some element of $\mathcal{D}_{\mathrm{hash}}$ and some element of $\mathcal{D}_{\mathrm{data}}$ collide, i.e., if $D_i[0] = D_{i'}[0]$ holds for some $1 \le i' < i \le q$, or $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{data}} \neq \emptyset$. When the bad flag gets set, the adversary wins the game and we let $G_1^A \Rightarrow 1$. Since Game G_1 returns 1 if Game G_0 returns 1, we have $\Pr[G_0^A \Rightarrow 1] \le \Pr[G_1^A \Rightarrow 1]$.

Next, we define Game G_2 in Fig. 6. Game G_2 is obtained from Game G_1 as follows. We delay the computation of the hash values until line 13, and generate $C_1[0], \ldots, C_q[0]$ without specifying the corresponding input values by assuming that the input values are all distinct.

Game G_2 sets the bad flag iff the flag gets set in Game G_1, and Game G_2 returns 1 if Game G_1 returns 1. Therefore, we have $\Pr[G_1^A \Rightarrow 1] \leq \Pr[G_2^A \Rightarrow 1]$.

The terminal game, Game G_3, is also defined in Fig. 6. In Game G_3, the sampling of $L \xleftarrow{\$} \mathcal{L}$ is moved to line 17, and the new blocks in the challenge ciphertext $C^* = (C^*[0], \ldots, C^*[l])$, i.e., the blocks $C^*[j] \notin \{C_1[0], \ldots, C_1[l], \ldots, C_q[0], \ldots, C_q[l]\}$, are decrypted in lines 13 or 16. By adding these blocks to $\mathcal{R}_P$ and the corresponding input blocks to $\mathcal{D}_P$ increases the probability that the bad flag gets set. Furthermore, Game G_3 always returns 1 whenever Game G_2 does, and we therefore have $\Pr[G_2^A \Rightarrow 1] \leq \Pr[G_3^A \Rightarrow 1]$.

Now we have $\mathbf{Adv}^{\mathrm{int}}_{\mathrm{HtCBC}[H,\mathrm{Perm}(n)]}(A) \leq \Pr[G_3^A \Rightarrow 1]$, which can be bounded as

$$\Pr[G_3^A \Rightarrow 1] \leq \Pr[G_3^A \text{ sets bad}] \tag{9}$$
$$+ \Pr[G_3^A \text{ sets win in line 26}]. \tag{10}$$

We evaluate (9) and (10).

First, we show (9) $\leq \frac{q^2}{2}\epsilon_1 + q^2 l \epsilon_2 + q(l+1)\epsilon_2$ by a similar argument to (7). Let $\mathcal{D}_{\mathrm{rest}}$ be the set of $D^*[0]$ and $I^*[j]$'s that are chosen in lines 13 or 16. There are three cases to consider. Case 1: $D_i[0] = D_{i'}[0]$ holds for some $1 \leq i' < i \leq q$, Case 2: $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{data}} \neq \emptyset$, and Case 3: $\mathcal{D}_{\mathrm{hash}} \cap \mathcal{D}_{\mathrm{rest}} \neq \emptyset$.

Case 1: For any fixed $1 \leq i' < i \leq q$, we have $\Pr[D_i[0] = D_{i'}[0]] = \Pr[H_L(D_i) = H_L(D_{i'})] \leq \epsilon_1$ since H is an ϵ_1-universal hash function, and we have $\Pr[G_3^A \text{ sets bad in Case 1}] \leq \frac{q^2}{2}\epsilon_1$ as there are at most $\frac{q^2}{2}$ choices for (i, i').

Case 2: For any fixed $1 \leq i, i' \leq q$ and $1 \leq j' \leq l$, we have $\Pr[D_i[0] = I_{i'}[j']] = \Pr[H_L(D_i) = I_{i'}[j']] \leq \epsilon_2$ since H is an ϵ_2-uniform hash function, and we have $\Pr[G_3^A \text{ sets bad in Case 2}] \leq q^2 l \epsilon_2$ as there are at most $q^2 l$ choices for (i, i', j').

Case 3: For any fixed $1 \leq i \leq q$, if $D^*[0] \in \mathcal{D}_{\mathrm{rest}}$, then $\Pr[D_i[0] = D^*[0]] = \Pr[H_L(D_i) = D^*[0]] \leq \epsilon_2$ since H is an ϵ_2-uniform hash function. With the same reasoning, for any fixed $1 \leq i \leq q$ and j such that $I^*[j] \in \mathcal{D}_{\mathrm{rest}}$, we have $\Pr[D_i[0] = I^*[j]] = \Pr[H_L(D_i) = I^*[j]] \leq \epsilon_2$. Therefore, we have $\Pr[G_3^A \text{ sets bad in Case 3}] \leq q(l+1)\epsilon_2$ as the cardinality of $\mathcal{D}_{\mathrm{rest}}$ is at most $(l+1)$.

We next show (10) $\leq \max\{\epsilon_5, \epsilon_6\}$. Let $\mathcal{C}_{\mathrm{hash}} = \{C_i[0] : 1 \leq i \leq q\}$, $\mathcal{C}_{\mathrm{data}} = \{C_i[j] : 1 \leq i \leq q, 1 \leq j \leq l\}$ and $\mathcal{C}_{\mathrm{rest}} = \{0,1\}^n \setminus (\mathcal{C}_{\mathrm{hash}} \cup \mathcal{C}_{\mathrm{data}})$. For each $0 \leq j \leq l$, we have $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$ or $C^*[j] \in \mathcal{C}_{\mathrm{data}} \cup \mathcal{C}_{\mathrm{rest}}$. There are two cases to consider. Case A: $C^*[0] \in \mathcal{C}_{\mathrm{hash}}$ and Case B: $C^*[0] \in \mathcal{C}_{\mathrm{data}} \cup \mathcal{C}_{\mathrm{rest}}$.

Case A: We have to bound $\Pr[D^*[0] = H_L(D^*[1], \ldots, D^*[l])]$, where $D^*[0] = H_L(D_i)$ for some $1 \leq i \leq q$. Now $D^*[j] = I^*[j] \oplus C^*[j-1]$ for $1 \leq j \leq l$. If $C^*[j] \in \mathcal{C}_{\mathrm{hash}}$, then $I^*[j]$ is a hash value, i.e., $D^*[j] = H_L(D_{i'}) \oplus C^*[j-1]$ for some $1 \leq i' \leq q$. If $C^*[j] \in \mathcal{C}_{\mathrm{data}} \cup \mathcal{C}_{\mathrm{rest}}$, then $I^*[j]$ is a constant and thus $D^*[j]$ is also a constant. Therefore, we have $D^* = (D^*[1], \ldots, D^*[l]) \in \{Z[1], H_L(X_1) \oplus V[1]\} \times \cdots \times \{Z[l], H_L(X_l) \oplus V[l]\}$ for some constants $X_1, \ldots, X_l$, $Z[1], \ldots, Z[l]$ and $V[1], \ldots, V[l]$. This implies that the event $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ must be one of the 2^l events covered in Definition 5, and hence we have (10) $\leq \epsilon_5$ in this case.

Case B: By a similar argument to Case A, we have $D^*[0] = Y$, and $D^* = (D^*[1], \ldots, D^*[l]) \in \{Z[1], H_L(X_1) \oplus V[1]\} \times \cdots \times \{Z[l], H_L(X_l) \oplus V[l]\}$ for some constants $X_1, \ldots, X_l$, $Z[1], \ldots, Z[l]$, $V[1], \ldots, V[l]$ and Y, and hence $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ is one of the 2^l events covered in Definition 6. Therefore, we have (10) $\leq \epsilon_6$ in this case.

This completes the proof of Lemma 2, (4). □

Game G_2
Initialize

1. $L \xleftarrow{\$} \mathcal{L}$

Procedure (invoke)

2. **for** $i \leftarrow 1$ **to** q **do**
3. $\quad D_i \xleftarrow{\$} \{0,1\}^{nl} \setminus \{D_1, \ldots, D_{i-1}\}$, $(D_i[1], \ldots, D_i[l]) \xleftarrow{n} D_i$
4. $\quad C_i[0] \xleftarrow{\$} \overline{\mathcal{R}_P \cup \{C_1[0], \ldots, C_{i-1}[0]\}}$
5. $\quad$ **for** $j \leftarrow 1$ **to** l **do**
6. $\quad\quad I_i[j] \leftarrow D_i[j] \oplus C_i[j-1]$
7. $\quad\quad$ **if** $I_i[j] \in \mathcal{D}_P$ **then** $C_i[j] \leftarrow P(I_i[j])$
8. $\quad\quad$ **else** $C_i[j] \xleftarrow{\$} \overline{\mathcal{R}_P \cup \{C_1[0], \ldots, C_i[0]\}}$, $P(I_i[j]) \leftarrow C_i[j]$
9. $\quad C_i \leftarrow (C_i[0], \ldots, C_i[l])$
10. **return** $((D_1, C_1), \ldots, (D_q, C_q))$

Procedure (C^*)

11. $(C^*[0], \ldots, C^*[l]) \xleftarrow{n} C^*$
12. **for** $i \leftarrow 1$ **to** q **do**
13. $\quad D_i[0] \leftarrow H_L(D_i[1], \ldots, D_i[l])$
14. $\quad$ **if** $D_i[0] \in \mathcal{D}_P$ **then** bad $\leftarrow$ true, win $\leftarrow$ true, **go Finalize**
15. $\quad P(D_i[0]) \leftarrow C_i[0]$
16. **if** $C^*[0] \in \mathcal{R}_P$ **then** $D^*[0] \leftarrow P^{-1}(C^*[0])$
17. **else** $D^*[0] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(D^*[0]) \leftarrow C^*[0]$
18. **for** $j \leftarrow 1$ **to** l **do**
19. $\quad$ **if** $C^*[j] \in \mathcal{R}_P$ **then** $I^*[j] \leftarrow P^{-1}(C^*[j])$
20. $\quad$ **else** $I^*[j] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(I^*[j]) \leftarrow C^*[j]$
21. $\quad D^*[j] \leftarrow I^*[j] \oplus C^*[j-1]$
22. **if** $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ **then** win $\leftarrow$ true, **go Finalize**

Finalize

23. **return** win

Game G_3
Procedure (invoke)
$\quad$ // same as Game G_2
Procedure (C^*)

11. $(C^*[0], \ldots, C^*[l]) \xleftarrow{n} C^*$
12. **if** $C^*[0] \notin \mathcal{R}_P \cup \{C_1[0], \ldots, C_q[0]\}$
13. $\quad$ **then** $D^*[0] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(D^*[0]) \leftarrow C^*[0]$
14. **for** $j \leftarrow 1$ **to** l **do**
15. $\quad$ **if** $C^*[j] \notin \mathcal{R}_P \cup \{C_1[0], \ldots, C_q[0]\}$
16. $\quad\quad$ **then** $I^*[j] \xleftarrow{\$} \overline{\mathcal{D}_P}$, $P(I^*[j]) \leftarrow C^*[j]$
17. $L \xleftarrow{\$} \mathcal{L}$
18. **for** $i \leftarrow 1$ **to** q **do**
19. $\quad D_i[0] \leftarrow H_L(D_i[1], \ldots, D_i[l])$
20. $\quad$ **if** $D_i[0] \in \mathcal{D}_P$ **then** bad $\leftarrow$ true, win $\leftarrow$ true, **go Finalize**
21. $\quad P(D_i[0]) \leftarrow C_i[0]$
22. $D^*[0] \leftarrow P^{-1}(C^*[0])$
23. **for** $j \leftarrow 1$ **to** l **do**
24. $\quad I^*[j] \leftarrow P^{-1}(C^*[j])$
25. $\quad D^*[j] \leftarrow I^*[j] \oplus C^*[j-1]$
26. **if** $D^*[0] = H_L(D^*[1], \ldots, D^*[l])$ **then** win $\leftarrow$ true

Finalize

27. **return** win

Fig. 6. Game G_2 (top) and G_3 (bottom). Procedure (invoke) in Game G_3 is identical to that in Game G_2.