HBS: A Single-Key Mode of Operation for Deterministic Authenticated Encryption

Tetsu Iwata[1] and Kan Yasuda[2]

[1] Dept. of Computational Science and Engineering, Nagoya University, Japan
iwata@cse.nagoya-u.ac.jp
[2] NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
yasuda.kan@lab.ntt.co.jp

Abstract. We propose the HBS (Hash Block Stealing) mode of operation. This is *the first* single-key mode that provably achieves the goal of providing deterministic authenticated encryption. The authentication part of HBS utilizes a newly-developed, vector-input polynomial hash function. The encryption part uses a blockcipher-based, counter-like mode. These two parts are combined in such a way as the numbers of finite-field multiplications and blockcipher calls are minimized. Specifically, for a header of h blocks and a message of m blocks, the HBS algorithm requires just $h + m + 2$ multiplications in the finite field and $m + 2$ calls to the blockcipher. Although the HBS algorithm is fairly simple, its security proof is rather complicated.

Keywords: universal hash function, counter mode, SIV, security proof.

1 Introduction

The goals for blockcipher modes of operation are twofold. One is to establish *authenticity*, or data integrity. The other is to preserve *privacy*, or data confidentiality. These two goals are realized by the mechanism of *authenticated encryption*, or AE for short. An AE mode establishes authenticity and preserves privacy concurrently by producing a ciphertext into which both the tag and the encrypted message are embedded.

A crucial aspect of an AE mode is that its security is based on the use of either a randomized salt or a state-dependent value, which has been formalized as *nonce-based* AE [13–15]. In fact, many of the modern AE modes, including CCM [19], GCM [9] and OCB [13], are all nonce-based.

The nonce, however, needs to be handled with great care, because the misuse of nonce (i.e., repeating the same value) would generally lead to the complete collapse of systems based on these AE modes. This is due to the fact that the security designs have no concern in what happens when the nonce assumption becomes no longer true.

The problem of nonce misuse has been settled by the introduction of *deterministic authenticated encryption* [16], or DAE for short. A DAE mode provides a deterministic, stateless algorithm that produces a ciphertext from the pair of a header and a message. A DAE mode can be used also as a conventional nonce-based AE mode by embedding a nonce value into the header. The virtue of a DAE mode is that it still ensures a certain level of security (i.e., all that an adversary can do is to detect a repetition of the same header-message pair) even when the DAE mode is not combined with a nonce element. Hence DAE is more robust than nonce-based AE.

The work [16] also proposes a concrete DAE mode of operation called SIV. The SIV mode is a blockcipher-based scheme, utilizing a vector-input version of the CMAC algorithm [5, 11] for its authentication part and the CTR (counter) mode for its encryption part. The SIV mode requires two independent keys for the underlying blockcipher, one being for the CMAC algorithm and another for the CTR mode.

The purpose of the current paper is to improve usability and performance over SIV. For this, we present a new DAE mode of operation called HBS (which stands for Hash Block Stealing), which has the following features.

1. The HBS mode requires just one key. It uses the same key for authentication and for encryption.
2. For authentication, the HBS mode adopts a vector-input polynomial-based universal hash function (rather than the blockcipher-based CMAC).
3. For encryption, the HBS mode uses a CTR-like scheme using a blockcipher, which operates differently from an ordinary CTR mode.
4. The numbers of finite-field multiplications (in hashing) and blockcipher calls are minimized.
5. We provide the HBS mode with concrete proofs of security.

In general, reducing keying material without compromising security reduces the cost of sharing, saving, and updating the secret key. Also, changing from a blockcipher-based MAC to polynomial-based universal hashing increases efficiency on many platforms [3, 17], as in changing from CCM [19], which uses a CBC-MAC, to GCM [9], which employs polynomial-based hashing.

It turns out that meeting the above objectives is demanding. We introduce the notion of a vector-input ϵ-almost XOR universal hash function and develop a new polynomial-based hashing that meets our requirements. We also devise a somewhat odd way of incrementation in a CTR mode, which is necessary for the security of the scheme and is non-trivially binded with the new polynomial hash function. As a result, the security proofs become rather involved.

2 Preliminaries

Notation. If x is a finite string, then $|x|$ denotes its length in bits. If x and y are two equal-length strings, then $x \oplus y$ denotes the XOR of x and y. If x and y are finite strings, then $x \| y$ denotes their concatenation. Given finite strings $x_0, x_1, \ldots, x_{m-1}$, we use the notation $x_0 \| x_1 \| \cdots \| x_{m-1}$ and the vector notation $(x_0, x_1, \ldots, x_{m-1})$ interchangeably. For a positive integer n, $\{0,1\}^n$ is the set of all n-bit strings, and $(\{0,1\}^n)^+$ is the set of all strings whose lengths are positive multiples of n bits. Whenever we write $X = (X[0], X[1], \ldots, X[m-1]) \in (\{0,1\}^n)^+$, we implicitly assume that $m \geq 1$ is an integer and $|X[i]| = n$ for $0 \leq i \leq m-1$. $\{0,1\}^*$ is the set of all finite strings (including the empty string), and for a positive integer ℓ, $(\{0,1\}^*)^\ell$ is the set of all vectors $(x_0, x_1, \ldots, x_{\ell-1})$, where $x_i \in \{0,1\}^*$ for $0 \leq i \leq \ell-1$. For positive integers n and j such that $n \leq 2^j - 1$, $\langle j \rangle_n$ is the big-endian n-bit binary representation of j. For a finite string x and a positive integer n such that $|x| \geq n$, $\mathrm{msb}(n, x)$ is the most significant n bits of x. For a positive integer n, 0^n is the n-times repetition of 0's.

We write $\boldsymbol{N}$ for the set of non-negative integers. Given a real number x, the symbol $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. If X is a finite set, then $\#X$ is the cardinality of X, and we let $x \overset{R}{\leftarrow} X$ denote the process of selecting an element from X uniformly at random and assigning it to x.

The Finite Field of 2^n Elements. We regard the set $\{0,1\}^n$ as the finite field of 2^n elements (relative to some irreducible polynomial). For $x, y \in \{0,1\}^n$, the symbol $x \cdot y$ denotes the product of x and y. We write $x^2 = x \cdot x$, $x^3 = x \cdot x \cdot x$, and so on. The $\oplus$ operation corresponds with the addition in the field.

Blockciphers and SPRP Adversaries. A blockcipher is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for any $K \in \mathcal{K}$, $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$ (i.e., a permutation family). The positive integer n is the block length, and an n-bit string is called a block. If $\mathcal{K} = \{0,1\}^k$, then k is the key length.

The SPRP notion for blockciphers was introduced in [6] and later made concrete in [1]. Let $\mathrm{Perm}(n)$ denote the set of all permutations on $\{0,1\}^n$. This set can be regarded as a blockcipher by assigning a unique string (a key) to each permutation. We say that P is a random permutation if $P \overset{R}{\leftarrow} \mathrm{Perm}(n)$. An adversary is a probabilistic algorithm (a program) with access to one or more oracles. An *SPRP-adversary* A has access to two oracles and returns a bit. The two oracles are either the encryption oracle $E_K(\cdot)$ and the decryption oracle $E_K^{-1}(\cdot)$, or a random permutation oracle $P(\cdot)$ and its inverse oracle $P^{-1}(\cdot)$. We define the advantage $\mathbf{Adv}_E^{\mathrm{sprp}}(A)$ as

$$\left| \Pr(K \overset{R}{\leftarrow} \mathcal{K} : A^{E_K(\cdot), E_K^{-1}(\cdot)} = 1) - \Pr(P \overset{R}{\leftarrow} \mathrm{Perm}(n) : A^{P(\cdot), P^{-1}(\cdot)} = 1) \right|.$$

For an adversary A, A's running time is denoted by *time*(A). The running time is its 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 running-time reference depend on the RAM model and the choice of encoding.

3 Specification of HBS

3.1 A Vector-Input Universal Hash Function F

In this section, we define a new vector-input universal hash function F. Let n be a fixed block length, e.g., $n = 128$. Before defining F, we first define a polynomial universal hash function $f : \{0,1\}^n \times (\{0,1\}^n)^+ \to \{0,1\}^n$. Let $L \in \{0,1\}^n$ be a key for f and $X = (X[0], X[1], \ldots, X[m-1]) \in (\{0,1\}^n)^+$ be an input. Define

$$f_L(X) = L^m \oplus L^{m-1} \cdot X[0] \oplus L^{m-2} \cdot X[1] \oplus \cdots \oplus X[m-1].$$

Note that we need $(m-1)$ multiplications to compute $f_L(X)$.

Now we define the vector-input hash function F. It internally uses a padding function $\mathrm{pad}(\cdot) : \{0,1\}^* \to (\{0,1\}^n)^+$, which takes $x \in \{0,1\}^*$ and outputs

$$\mathrm{pad}(x) = \begin{cases} x & \text{if } x \in (\{0,1\}^n)^+, \\ x \| 10^i & \text{otherwise}, \end{cases}$$

where i is the smallest non-negative integer such that the total length of $x\|10^i$ in bits becomes a positive multiple of n. F also takes a vector dimension ℓ as a parameter, and we write $F^{(\ell)}$ for F with a parameter ℓ. $F^{(\ell)}$ takes $L \in \{0,1\}^n$ as a key and $\mathcal{X} = (x_0, x_1, \ldots, x_{\ell-1}) \in (\{0,1\}^*)^\ell$ as its input. For $0 \le i \le \ell - 1$, let $X_i = \mathrm{pad}(x_i)$ and $z_i = f_L(X_i)$. Define $F^{(\ell)} : \{0,1\}^n \times (\{0,1\}^*)^\ell \to \{0,1\}^n$ as

$$\begin{aligned} F_L^{(\ell)}(\mathcal{X}) &= F_L^{(\ell)}(x_0, x_1, \ldots, x_{\ell-1}) \\ &= L \cdot z_0^\ell \cdot \langle c_0 \rangle_n \oplus L^2 \cdot z_1^\ell \cdot \langle c_1 \rangle_n \oplus \cdots \oplus L^\ell \cdot z_{\ell-1}^\ell \cdot \langle c_{\ell-1} \rangle_n, \end{aligned} \qquad (1)$$

where, for $0 \le i \le \ell - 1$, we set $z_i = f_L(\mathrm{pad}(x_i))$ and

$$c_i = \begin{cases} 1 & \text{if } x_i \notin (\{0,1\}^n)^+, \\ 2 & \text{otherwise}. \end{cases}$$

The degree of $F_L^{(\ell)}(\mathcal{X})$ as a polynomial in L is $\mu(\mathcal{X})$, which is defined as follows. For $\mathcal{X} = (x_0, \ldots, x_{\ell-1}) \in (\{0,1\}^*)^\ell$, define $\mu : (\{0,1\}^*)^\ell \to \boldsymbol{N}$ as

$$\mu(\mathcal{X}) = \max_{0 \le i \le \ell-1} \{i + 1 + \ell m_i\},$$

where $m_i = \lceil |x_i|/n \rceil$ for $0 \le i \le \ell - 1$. The value m_i is the length of x_i in blocks, where a partial block counts for one block. Note that each z_i is a polynomial in L of degree m_i.

We remark that the multiplication by the constant $\langle 2 \rangle_n$ can be implemented efficiently. For example, for $n = 128$ we can choose $\mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$, which is one of the irreducible polynomials having the minimum number of non-zero coefficients. Then, for $x \in \{0,1\}^{128}$, we can compute the product $x \cdot \langle 2 \rangle_{128}$ as $x \cdot \langle 2 \rangle_{128} = x \cdot 0^{126}10 = x \ll 1$ if $\mathrm{msb}(1, x) = 0$, or as $x \cdot \langle 2 \rangle_{128} = x \cdot 0^{126}10 = (x \ll 1) \oplus 0^{120}10000111$ otherwise, where $x \ll 1$ is the left shift of x by one bit (The first bit of x disappears and a zero comes into the last bit). See [13] for details.

3.2 Vector-Input ϵ-Almost XOR Universal Hash Function

We next define the notion of a vector-input ϵ-almost XOR universal (VI-ϵ-AXU for short) hash function. This plays an important role in our security analysis.

```
Algorithm HBS.Enc_K(H, M)
100  $L \leftarrow E_K(0^n)$
101  $S \leftarrow F_L^{(2)}(H, M)$
102  $C \leftarrow \mathrm{CTR.Enc}_K(S, M)$
103  $T \leftarrow E_K(S)$
104  return $(T, C)$
```

```
Algorithm CTR.Enc_K(S, M)
300  for $i \leftarrow 0$ to $\lceil |M|/n \rceil - 1$ do
301      $R_i \leftarrow E_K(S \oplus \langle i+1 \rangle_n)$
302  $R \leftarrow (R_0, R_1, \ldots, R_{\lceil |M|/n \rceil - 1})$
303  $C \leftarrow M \oplus \mathrm{msb}(|M|, R)$
304  return $C$
```

```
Algorithm HBS.Dec_K(H, (T, C))
200  $L \leftarrow E_K(0^n)$
201  $S \leftarrow E_K^{-1}(T)$
202  $M \leftarrow \mathrm{CTR.Dec}_K(S, C)$
203  $S' \leftarrow F_L^{(2)}(H, M)$
204  if $S \neq S'$ then return $\perp$
205  else return $M$
```

```
Algorithm CTR.Dec_K(S, C)
400  for $i \leftarrow 0$ to $\lceil |C|/n \rceil - 1$ do
401      $R_i \leftarrow E_K(S \oplus \langle i+1 \rangle_n)$
402  $R \leftarrow (R_0, R_1, \ldots, R_{\lceil |C|/n \rceil - 1})$
403  $M \leftarrow C \oplus \mathrm{msb}(|C|, R)$
404  return $M$
```

Fig. 1. Definition of the encryption algorithm HBS.Enc (left top) and the decryption algorithm HBS.Dec (left bottom). CTR.Enc (right top) is used in HBS.Enc, and CTR.Dec (right bottom) is used in HBS.Dec.

Definition 1 (VI-ϵ-AXU hash function). *Let ℓ be an integer and $F^{(\ell)} : \{0,1\}^n \times (\{0,1\}^*)^\ell \to \{0,1\}^n$ be a vector-input function keyed by $L \in \{0,1\}^n$. $F^{(\ell)}$ is said to be VI-ϵ-AXU if, for any two distinct inputs $\mathcal{X}, \hat{\mathcal{X}} \in (\{0,1\}^*)^\ell$ and for any $Y \in \{0,1\}^n$,*

$$\Pr(L \xleftarrow{R} \{0,1\}^n : F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y) \leq \epsilon.$$

We show that the polynomial hash function $F^{(\ell)}$ defined in Sect. 3.1 is VI-ϵ-AXU for a small ϵ. A proof is given in Appendix A.

Theorem 1. *Let ℓ be an integer and $F^{(\ell)} : \{0,1\}^n \times (\{0,1\}^*)^\ell \to \{0,1\}^n$ be the polynomial hash function defined in Sect. 3.1, keyed by $L \in \{0,1\}^n$. Let $\mathcal{X} = (x_0, \ldots, x_{\ell-1}), \hat{\mathcal{X}} = (\hat{x}_0, \ldots, \hat{x}_{\ell-1}) \in (\{0,1\}^*)^\ell$ be any two distinct inputs to $F^{(\ell)}$. Then for any $Y \in \{0,1\}^n$,*

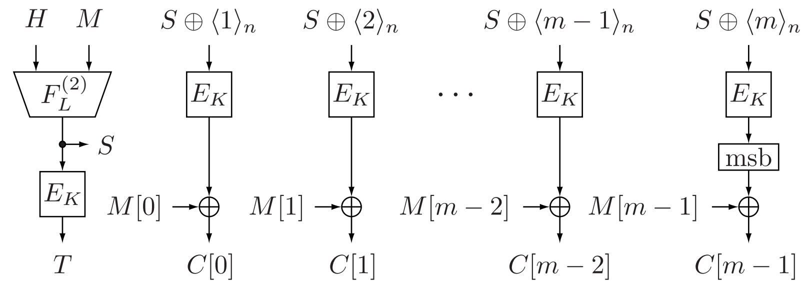
$$\Pr(L \xleftarrow{R} \{0,1\}^n : F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y) \leq \frac{\max\{\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})\}}{2^n}.$$

3.3 HBS: Hash Block Stealing

We present our HBS, Hash Block Stealing. It takes a blockcipher E as a parameter and uses $F^{(\ell)}$ defined in Sect. 3.1 with $\ell = 2$.

Fix a blockcipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. HBS consists of two algorithms, the encryption algorithm (HBS.Enc) and the decryption algorithm (HBS.Dec). These algorithms are defined in Fig. 1. See Fig. 2 for a picture illustrating HBS.Enc (See also Fig. 3 for $F_L^{(2)}(H, M)$).

The encryption algorithm HBS.Enc takes a key $K \in \{0,1\}^k$, a header $H \in \{0,1\}^*$, and a plaintext $M \in \{0,1\}^*$ to return a ciphertext $(T, C) \in \{0,1\}^n \times \{0,1\}^*$, where $|C| = |M|$. We write $(T, C) \leftarrow \mathrm{HBS.Enc}_K(H, M)$. The decryption



H
M
$F_L^{(2)}$
S
E_K
T
$S \oplus \langle 1 \rangle_n$
$S \oplus \langle 2 \rangle_n$
$S \oplus \langle m-1 \rangle_n$
$S \oplus \langle m \rangle_n$
E_K
msb
$M[0]$
$M[1]$
$M[m-2]$
$M[m-1]$
$C[0]$
$C[1]$
$C[m-2]$
$C[m-1]$

Fig. 2. Illustration of the encryption algorithm $(T, C) \leftarrow \text{HBS.Enc}_K(H, M)$. In the figure, $M = (M[0], M[1], \ldots, M[m-1])$, where $|M[0]| = \cdots = |M[m-2]| = n$ and $|M[m-1]| \leq n$. $L = E_K(0^n)$, and the output of "msb" is the most significant $|M[m-1]|$ bits of $E_K(S \oplus \langle m \rangle_n)$.

algorithm takes K, H, and (T, C) as its inputs to return the corresponding plaintext M or a special symbol $\perp$. The symbol $\perp$ indicates that the given inputs are invalid. We write $M \leftarrow \text{HBS.Dec}_K(H, (T, C))$. We have $\text{HBS.Dec}_K(H, (T, C)) = \perp$ when the decryption process fails.

HBS.Enc and HBS.Dec call subroutines CTR.Enc and CTR.Dec, respectively, which are the encryption and the decryption of the CTR mode using S as its initial counter value. By specification we restrict the message length to $2^{n/2} - 1$ blocks, as the security of the HBS mode becomes vacuous beyond this point. This restriction allows us to write $\langle i \rangle_n = 0^{n/2} \| \langle i \rangle_{n/2}$ for an integer $0 \leq i \leq 2^{n/2} - 1$ (We assume that the block length n is an even integer), which implies that the incrementation of the counter in CTR.Enc and CTR.Dec can be done by adding 1 modulo $2^{n/2}$ rather than modulo 2^n.

$F_L^{(2)}(H, M)$ can be implemented using $h+m$ multiplications and two squaring operations, as in Fig. 3. See also Fig. 4 for pseudocode. In the figures, we let

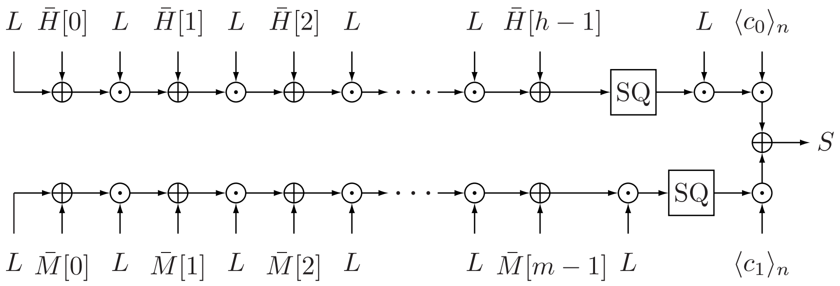


Fig. 3. Illustration of $F_L^{(2)}(H, M)$. In the figure, $\text{pad}(H) = (\bar{H}[0], \ldots, \bar{H}[h-1])$ and $\text{pad}(M) = (\bar{M}[0], \ldots, \bar{M}[m-1])$. $L = E_K(0^n)$, "SQ" outputs the square of its input, "$\odot$" is the multiplication, and $c_0, c_1 \in \{1, 2\}$ are constants.

```
Function $F_L^{(2)}(H, M)$
100    $z_H \leftarrow L \oplus \bar{H}[0]$
101    for $i = 1$ to $h - 1$ do
102        $z_H \leftarrow L \cdot z_H \oplus \bar{H}[i]$
103    $z_M \leftarrow L \oplus \bar{M}[0]$
104    for $i = 1$ to $m - 1$ do
105        $z_M \leftarrow L \cdot z_M \oplus \bar{M}[i]$
106    return $L \cdot z_H^2 \cdot \langle c_0 \rangle_n \oplus (L \cdot z_M)^2 \cdot \langle c_1 \rangle_n$
```

Fig. 4. Implementation example of $F_L^{(2)}(H, M)$, where $\mathrm{pad}(H) = (\bar{H}[0], \ldots, \bar{H}[h-1])$ and $\mathrm{pad}(M) = (\bar{M}[0], \ldots, \bar{M}[m-1])$. z_H and z_M are n-bit variables, and $c_0, c_1 \in \{1, 2\}$.

$\lceil |H|/n \rceil = h$ and $\lceil |M|/n \rceil = m$, so that $\mathrm{pad}(H) = (\bar{H}[0], \ldots, \bar{H}[h-1])$ and $\mathrm{pad}(M) = (\bar{M}[0], \ldots, \bar{M}[m-1])$. Also, let $c_0 = 1$ if $H \notin (\{0,1\}^n)^+$ and $c_0 = 2$ otherwise, and $c_1 = 1$ if $M \notin (\{0,1\}^n)^+$ and $c_1 = 2$ otherwise. Then $F_L^{(2)}(H, M)$ can be written as

$$\begin{aligned} & L \cdot \bigl(f_L(\bar{H}[0], \ldots, \bar{H}[h-1])\bigr)^2 \cdot \langle c_0 \rangle_n \oplus L^2 \cdot \bigl(f_L(\bar{M}[0], \ldots, \bar{M}[m-1])\bigr)^2 \cdot \langle c_1 \rangle_n \\ &\quad = L \cdot \bigl(L^h \oplus L^{h-1} \cdot \bar{H}[0] \oplus L^{h-2} \cdot \bar{H}[1] \oplus \cdots \oplus \bar{H}[h-1]\bigr)^2 \cdot \langle c_0 \rangle_n \\ &\qquad \oplus \bigl(L^{m+1} \oplus L^m \cdot \bar{M}[0] \oplus L^{m-1} \cdot \bar{M}[1] \oplus \cdots \oplus L \cdot \bar{M}[m-1]\bigr)^2 \cdot \langle c_1 \rangle_n . \end{aligned}$$

Recall that the multiplication by $\langle 2 \rangle_n$ can be implemented using a left shift and a conditional XOR, which is a small overhead compared to the multiplication by L.

Lastly, we make a remark about handling a message with no header. For such a message, simply ignore the computation of z_H and define the value $(L \cdot z_M)^2 \cdot \langle c_1 \rangle_n$ as the output of the hash function $F_L^{(2)}(\cdot, M)$. Note that this should be differentiated from the case $H = \varepsilon$ (the null string), where $\mathrm{pad}(H) = 10^{n-1}$.

4 Security Analysis of HBS

HBS is a mode of operation for deterministic authenticated encryption. Before presenting our security results, we define the security of the mode. Our security definition follows the one given by Rogaway and Shrimpton in [16].

4.1 Security Definition

Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. For notational simplicity, we write $\mathcal{E}_K(\cdot,\cdot)$ for $\mathrm{HBS.Enc}_K(\cdot,\cdot)$, and $\mathcal{D}_K(\cdot,\cdot)$ for $\mathrm{HBS.Dec}_K(\cdot,\cdot)$. Let $\mathrm{HBS}[E] = (\mathrm{HBS.Enc}_K, \mathrm{HBS.Dec}_K) = (\mathcal{E}_K, \mathcal{D}_K)$ be defined as in Sect. 3.3.

An adversary A is given access to either a pair of $\mathcal{E}_K(\cdot,\cdot)$ and $\mathcal{D}_K(\cdot,\cdot)$ oracles or a pair of $\mathcal{R}(\cdot,\cdot)$ and $\perp(\cdot,\cdot)$ oracles. On query $(H, M) \in (\{0,1\}^*)^2$, $\mathcal{E}_K(\cdot,\cdot)$ returns $(C, T) \leftarrow \mathcal{E}_K(H, M)$, and the random-bits oracle $\mathcal{R}(\cdot,\cdot)$ returns a random string of $n+|M|$ bits. On query $(H, (T, C)) \in \{0,1\}^* \times (\{0,1\}^n \times \{0,1\}^*)$, $\mathcal{D}_K(\cdot,\cdot)$

returns $\perp$ or $M \leftarrow \mathcal{D}_K(H,(T,C))$, and the $\perp(\cdot,\cdot)$ oracle returns $\perp$ on every input. Queries can be made adaptively, and A's goal is to distinguish between these pairs.

Definition 2. *The DAE-advantage* $\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[E]}(A)$ *of an adversary* A *in breaking* $\mathrm{HBS}[E]=(\mathcal{E}_K,\mathcal{D}_K)$ *is defined as*

$$\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[E]}(A)=\left|\Pr(A^{\mathcal{E}_K(\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot)}=1)-\Pr(A^{\mathcal{R}(\cdot,\cdot),\perp(\cdot,\cdot)}=1)\right|.$$

In what follows, the first oracle refers to $\mathcal{E}_K(\cdot,\cdot)$ or $\mathcal{R}(\cdot,\cdot)$, and the second oracle refers to $\mathcal{D}_K(\cdot,\cdot)$ or $\perp(\cdot,\cdot)$. We make the following assumptions about A.

- A does not make a query (H,M) to its first oracle if the second oracle has returned M in response to some previous query $(H,(T,C))$.
- A does not make a query $(H,(T,C))$ to its second oracle if the first oracle has returned (T,C) in response to some previous query (H,M).
- A does not repeat a query.

The last assumption is without loss of generality, and the first two assumptions are to prevent trivial wins.

We also consider the security of HBS in terms of privacy. An adversary B is given access to either $\mathcal{E}_K(\cdot,\cdot)$ or $\mathcal{R}(\cdot,\cdot)$, and B's goal is to distinguish between these oracles.

Definition 3. *The PRIV-advantage* $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[E]}(B)$ *of an adversary* B *in breaking the privacy of* $\mathrm{HBS}[E]=(\mathcal{E}_K,\mathcal{D}_K)$ *is defined as*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[E]}(B)=\left|\Pr(B^{\mathcal{E}_K(\cdot,\cdot)}=1)-\Pr(B^{\mathcal{R}(\cdot,\cdot)}=1)\right|.$$

We deal with $\mathrm{HBS}[\mathrm{Perm}(n)]=(\mathcal{E}_P,\mathcal{D}_P)$, where a random permutation $P\overset{R}{\leftarrow}\mathrm{Perm}(n)$ is used as the underlying blockcipher. That is, $\mathrm{HBS}[\mathrm{Perm}(n)]$ is defined by replacing E_K and E_K^{-1} in Fig. 1 by P and P^{-1}, respectively.

4.2 Security Theorem

Consider the DAE-advantage of an adversary A attacking $\mathrm{HBS}[\mathrm{Perm}(n)]$. We assume A makes a total of at most q queries. For $0\le i\le q-1$, A makes a query of the form (H_i,M_i) to the first oracle, or $(H_i,(T_i,C_i))$ to the second oracle. We assume that $\lceil |H_i|/n\rceil\le h_{\max}$, $\lceil |M_i|/n\rceil\le m_{\max}$, and $\lceil |C_i|/n\rceil\le m_{\max}$ for some $h_{\max},m_{\max}\ge 1$. That is, $h_{\max}$ is the maximum length of H_i, and $m_{\max}$ is the maximum length of M_i and C_i in blocks. The following theorem is our main result on the security of our HBS mode. It shows that HBS provides standard birthday-bound security.

Theorem 2. *Let* A *be an adversary described as above. Then*

$$\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(A)\le\frac{19q^2(1+h_{\max}+2m_{\max})^2}{2^n}.$$

Given Theorem 2, it is standard to pass to its complexity-theoretic bound, by replacing $P \overset{R}{\leftarrow} \mathrm{Perm}(n)$ with E_K (and hence P^{-1} with E_K^{-1}).

Lemma 1. *Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and A be an adversary attacking* HBS$[E]$*. Then there exists A' attacking E such that*

$$\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[E]}(A) \le \mathbf{Adv}^{\mathrm{sprp}}_{E}(A') + \frac{19q^2(1+h_{\max}+2m_{\max})^2}{2^n},$$

where A' makes at most $1 + q(1 + m_{\max})$ queries, and time(A') = time$(A) + O(nq(h_{\max} + m_{\max}))$.

The proof of Lemma 1 is standard and omitted (For example, see [1]). Now we turn to Theorem 2. The proof is based on the following lemma.

Lemma 2. *Let A be an adversary as in Theorem 2. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(A) \le 2\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B) + \frac{11q^2(1+h_{\max}+2m_{\max})^2}{2^n} + \frac{q}{2^n},$$

where B makes at most q queries, and it is subject to the same restriction as A on the length of its queries.

We give a proof sketch of Lemma 2, leaving a complete proof in Appendix B. We see that $\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(A)$ is at most $p_0 + p_1$, where we define p_0 and p_1 as

$$\begin{cases} p_0 = \left|\Pr\left(A^{\mathcal{E}_P(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\right) - \Pr\left(A^{\mathcal{R}(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\right)\right|, \\ p_1 = \left|\Pr\left(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)} = 1\right) - \Pr\left(A^{\mathcal{E}_P(\cdot,\cdot),\perp(\cdot,\cdot)} = 1\right)\right|. \end{cases}$$

We can view A in p_0 as attacking the privacy of HBS[Perm(n)], as the second oracle always returns $\perp$. So there exists B such that $p_0 \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B)$.

To derive the upper bound on p_1, we observe that the oracles are identical unless the $\mathcal{D}_P(\cdot,\cdot)$ oracle returns something other than $\perp$. We thus have

$$p_1 \le \Pr(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)} \text{ forges}),$$

where $A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)}$ forges if A makes a query $((H,M),T)$ to its second oracle such that $\mathcal{D}_P((H,M),T)) \ne \perp$. To derive the upper bound on the forgery probability, we introduce an oracle $\mathcal{V}_P(\cdot,\cdot)$ that takes $((H,M),T)$ as its input and returns T if $P(F^{(2)}_L(H,M)) = T$ or returns $\perp$ otherwise. We can show that

$$\Pr(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)} \text{ forges}) \le \Pr(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}) + \frac{11q^2(1+h_{\max}+2m_{\max})^2}{2^n}$$

and $\Pr(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B) + q/2^n$. The proof for the first claim is rather complicated but the second one follows from the standard argument that distinguishing is easier than forging, and Lemma 2 follows.

Now to prove Theorem 2, it suffices to bound $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B)$. Recall that B is an adversary attacking the privacy of HBS[Perm(n)], where B makes a query of the form (H_i, M_i) for $0 \le i \le q-1$. We assume that H_i is at most $h_{\max}$ blocks, and M_i is at most $m_{\max}$ blocks. We have the following result.

Lemma 3. *Let B be an adversary as described above. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B) \le \frac{3q^2(1+h_{\max}+2m_{\max})^2}{2^n} + \frac{(1+q+qm_{\max})^2}{2^{n+1}}.$$

Let us explain the intuitive reasoning behind the bound, leaving a complete proof in Appendix C. Suppose B makes q queries $(H_0, M_0), \ldots, (H_{q-1}, M_{q-1})$, where M_i is m_i blocks. Let $S_i = F_L^{(2)}(H_i, M_i)$, which corresponds to the initial counter value, and we consider the set $\mathbb{S}_i = \{S_i, S_i \oplus \langle 1\rangle_n, S_i \oplus \langle 2\rangle_n, \ldots, S_i \oplus \langle m_i\rangle_n\}$. Each element in $\mathbb{S}_i$ is the input value to P at the i-th query. Notice that we never have a collision within $\mathbb{S}_i$, but we may have $0^n \in \mathbb{S}_i$, or $\mathbb{S}_i \cap \mathbb{S}_j \neq \emptyset$. Both events may be useful for B to mount an distinguishing attack. Now for the first event, we know that $F_L^{(2)}(H_i, M_i)$ is a non-zero polynomial in L, and hence, for $0 \le s \le m_i$, the probability that $F_L^{(2)}(H_i, M_i) = \langle s\rangle_n$ is small. For the second event to occur, we need

$$F_L^{(2)}(H_i, M_i) \oplus \langle s\rangle_n = F_L^{(2)}(H_j, M_j) \oplus \langle t\rangle_n,$$

where $0 \le s \le m_i$ and $0 \le t \le m_j$. Theorem 1 ensures that the equality holds with only a small probability. It turns out that during the attack, with a high probability, we have $0^n \notin \mathbb{S}_i$ for $0 \le i \le q-1$, and $\mathbb{S}_i \cap \mathbb{S}_j = \emptyset$ for $0 \le i < j \le q-1$. Under this assumption, what B learns is the output values of P for distinct input values, and hence B cannot distinguish it from a truly random string.

Given Lemma 2 and Lemma 3, we find the proof of Theorem 2 straightforward, as we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dae}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(A) &\le 2\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS}[\mathrm{Perm}(n)]}(B) + \frac{11q^2(1+h_{\max}+2m_{\max})^2}{2^n} + \frac{q}{2^n} \\
&\le \frac{6q^2(1+h_{\max}+2m_{\max})^2}{2^n} + \frac{(1+q+qm_{\max})^2}{2^n} \\
&\qquad + \frac{11q^2(1+h_{\max}+2m_{\max})^2}{2^n} + \frac{q}{2^n} \\
&\le \frac{19q^2(1+h_{\max}+2m_{\max})^2}{2^n},
\end{aligned}$$

where the first inequality follows from Lemma 2, the next one from Lemma 3, and the last one from easy simplification.

5 Rationale of f_L and $F_L^{(\ell)}$ and Comparison with SIV

Rationale of f_L and $F_L^{(\ell)}$. We adopt $f_L(X[0], \ldots, X[m-1]) = L^m \oplus L^{m-1} \cdot X[0] \oplus L^{m-2} \cdot X[1] \oplus \cdots \oplus X[m-1]$ as the polynomial hash in HBS, rather than the usual $g_L(X[0], \ldots, X[m-1]) = L^{m-1} \cdot X[0] \oplus L^{m-2} \cdot X[1] \oplus \cdots \oplus X[m-1]$ as in [18, 9]. The choice of f_L enables us to handle variable-length inputs without increasing the number of multiplications. We note that the degree of our polynomial f_L is higher than that of g_L. However, the difference is only one, and the decrease in the security bound due to the increase in the degree should be negligible in practice.

$F_L^{(\ell)}$ is designed to handle an ℓ-dimensional vector efficiently. For an input $(x_0, \dots, x_{\ell-1}) \in (\{0,1\}^*)^\ell$, we need just $\lceil |x_0|/n \rceil + \cdots + \lceil |x_0|/n \rceil + \ell$ multiplications. We may need additional ℓ shifts and XOR's depending on the length of each component, but the cost for these operations is fairly low. Moreover, $F_L^{(2)}$ allows us to reuse z_H or z_M in Fig. 4 if H or M stays fixed, and the same is true for $F_L^{(\ell)}$ with $\ell \geq 3$.

In contrast, the polynomial hash in GCM was designed to accept only two-dimensional vectors (H, M), and the lengths of H and M were encoded into a block as $\langle |H| \rangle_{n/2} \| \langle |M| \rangle_{n/2}$. One could handle, say, three-dimensional vectors by encoding the lengths into a block, but such an approach would severely limit the maximum length of each component, which is solved in our hash function $F_L^{(\ell)}$.

A drawback of $F_L^{(\ell)}$, as compared to the vectorized CMAC [16], is that the (maximum) dimension ℓ needs to be fixed in advance. In order to lift this restriction, we can start with the hash function $F_L^{(2)}$ for two-dimensional vectors and then "increase" the dimension by utilizing the values $\sqrt{L}$, $\sqrt[4]{L}$, $\sqrt[8]{L}$, $\dots$, upon receiving 3-, 4-, 5-, $\dots$, dimensional vectors. Although we can efficiently compute the square root $\sqrt{L}$ of an element $L \in GF(2^n)$, we admit that this solution results in rather complicated algorithms.

As with the polynomial hash in GCM [9], $L = 0^n$ is a weak key for $F_L^{(\ell)}$ [4]. We may let $L \leftarrow \mathrm{msb}(n-1, E_K(0^n)) \| 1$ to avoid the problem at the cost of slight decrease in the security bound. But since $L = 0^n$ only occurs with a probability $1/2^n$, we do not adopt this approach. The birthday attack on GCM described in [4] can be made also on our HBS mode. Therefore, as with GCM, one needs to update the secret key well before processing $2^{n/2}$ blocks.

Comparison with SIV. HBS basically follows the design of SIV [16], but there are important differences. HBS works with a single key, and SIV [16] uses two separate keys, one for encryption and the other for MAC. SIV works as follows. First, let $IV \leftarrow \mathrm{CMAC}^*_{K_1}(H, M)$ and $C \leftarrow \mathrm{CTR.Enc}_{K_2}(IV, M)$. Then the output is (IV, C), where CMAC* is constructed from CMAC [5, 11] to handle a vector input (H, M).

In Table 1, we make a brief comparison between SIV and HBS. HBS reduces the keying material and replaces $\lceil |M|/n \rceil + \lceil |H|/n \rceil$ blockcipher calls by $\lceil |M|/n \rceil + \lceil |H|/n \rceil + 2$ multiplications at the cost of a stronger assumption about the blockcipher. The SPRP assumption is needed as we "steal" the result of hash computation (hash block) and use it as the initial counter value. However, we argue that this does not seem to make a substantial difference in practice, because many of the modern blockciphers, such as AES, seem to satisfy the strong property of SPRP. Rather, we admit that it may be disadvantageous of HBS to require the inverse cipher. This is especially true for blockciphers of asymmetric encryption/decryption design, such as AES.

Table 1. Comparison between SIV and HBS.

	SIV	HBS
Keying material	two blockcipher keys	single blockcipher key
Number of blockcipher calls	$2\lceil \lvert M\rvert/n\rceil + \lceil \lvert H\rvert/n\rceil + 2$	$\lceil \lvert M\rvert/n\rceil + 2$
Number of multiplications	0	$\lceil \lvert M\rvert/n\rceil + \lceil \lvert H\rvert/n\rceil + 2$
Assumption about blockcipher	PRP	SPRP
Security bound	$O(\sigma^2/2^n)$	$O(q^2(h_{\max} + m_{\max})^2/2^n)$

We note that, out of $2\lceil |M|/n\rceil + \lceil |H|/n\rceil + 2$ blockcipher calls in SIV, two calls can be done during idle time (without H or M). Similarly, out of $\lceil |M|/n\rceil + 2$ blockcipher calls in HBS, one blockcipher call (for $L = E_K(0^n)$) does not need H or M. We also note that there is a subtle difference in the security bounds. In SIV, the bound is $O(\sigma^2/2^n)$, while in HBS, the bound is $O(q^2(h_{\max} + m_{\max})^2/2^n)$, where σ is the total block length of H and M. It remains open if our analysis of HBS can be improved to give an $O(\sigma^2/2^n)$ security bound.

6 Further Discussion: Beyond the Birthday Bound

HBS delivers fine performance and provides security up to the birthday bound. It might be beneficial to come up with a DAE construction whose security is *beyond* the birthday bound. Such a construction would most likely be less efficient than HBS but might be desirable for some cases, as explained below.

Recall that DAE demands that the message space be of high entropy. An important example is the *key wrap* [10], since a key space obviously has high entropy. Although a key length is usually very short and a query complexity exceeding the birthday bound is hard to imagine in such a scenario, the highest security possible is desired for key-wrap applications. Therefore, highly secure constructions stand as suitable candidates for such systems, even if their performance is relatively modest.

We could give a beyond-the-birthday-bound construction as follows. First construct a $2n$-to-$2n$-bit blockcipher $E' : \{0,1\}^{2n} \to \{0,1\}^{2n}$, which has a certain key space, from a blockcipher $E_K : \{0,1\}^n \to \{0,1\}^n$ with $K \in \{0,1\}^k$ via the sum construction [7] and the Feistel network of six rounds [12]. Then construct the HBS mode with the block size of $2n$ bits, using E' as its underlying blockcipher. This construction ensures security beyond the $2^{n/2}$ bound but has the following three problems.

1. It is inefficient and impractical. One call to E' requires twelve calls to E.
2. It requires more than one key. The key space of E' is large.
3. It produces a long ciphertext. The tag size is $2n$-bit rather than n-bit.

The last point might be contrasted with the double-pipe hash [8], which has $2n$-bit intermediate values but outputs n-bit hash values. It remains open to provide a beyond-the-birthday-bound construction which resolves these problems.

Acknowledgments

The authors would like to express their thanks to the anonymous reviewers of FSE 2009 for many insightful comments.

References

1. Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3) (2000) 362–399
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay, S., ed.: EUROCRYPT 2006. Volume 4004 of LNCS., Springer (2006) 409–426
3. Gladman, B.: AES and combined encryption/authentication modes. Available at `http://www.gladman.me.uk/` (2006)
4. Handschuh, H., Preneel, B.: Key-recovery attacks on universal hash function based MAC algorithms. In Wagner, D., ed.: CRYPTO 2008. Volume 5157 of LNCS., Springer (2008) 144–161
5. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In Johansson, T., ed.: FSE 2003. Volume 2887 of LNCS., Springer (2003) 129–153
6. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. **17**(2) (1988) 373–386
7. Lucks, S.: The sum of PRPs is a secure PRF. In Preneel, B., ed.: EUROCRYPT 2000. Volume 1807 of LNCS., Springer (2000) 470–484
8. Lucks, S.: A failure-friendly design principle for hash functions. In Roy, B.K., ed.: ASIACRYPT 2005. Volume 3788 of LNCS., Springer (2005) 474–494
9. McGrew, D.A., Viega, J.: The security and performance of the Galois/counter mode (GCM) of operation. In Canteaut, A., Viswanathan, K., eds.: INDOCRYPT 2004. Volume 3348 of LNCS., Springer (2004) 343–355
10. NIST: AES key wrap specification (2001)
11. NIST: Recommendation for block cipher modes of operation: the CMAC mode for authentication (2005)
12. Patarin, J.: Security of random Feistel schemes with 5 or more rounds. In Franklin, M.K., ed.: CRYPTO 2004. Volume 3152 of LNCS., Springer (2004) 106–122
13. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS, ACM Press (2001) 196–205
14. Rogaway, P.: Authenticated-encryption with associated-data. In Atluri, V., ed.: ACM CCS, ACM Press (2002) 98–107
15. Rogaway, P.: Nonce-based symmetric encryption. In Roy, B.K., Meier, W., eds.: FSE 2004. Volume 3017 of LNCS., Springer (2004) 348–359
16. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In Vaudenay, S., ed.: EUROCRYPT 2006. Volume 4004 of LNCS., Springer (2006) 373–390
17. Satoh, A.: High-speed hardware architectures for authenticated encryption mode GCM. In Friedman, E.G., Theodoridis, S., eds.: IEEE ISCAS 2006. IEEE Press (2006) 4831–4844
18. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. **22**(3) (1981) 265–279
19. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). Submission to NIST, Available at `http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html` (2002)

A Proof of Theorem 1

We have 2^n possible values of $L \in \{0,1\}^n$. We show that

$$\#\{L \mid F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y\} \le \max\{\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})\}.$$

It suffices to show that $F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y$ is a non-trivial equation in L of degree at most $\max\{\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})\}$. Let $x_i = (x_i[0], \ldots, x_i[m_i - 1])$, $X_i = \mathrm{pad}(x_i) = (X_i[0], \ldots, X_i[m_i - 1])$, $z_i = f_L(X_i)$. Also, let $c_i = 1$ if $x_i \notin (\{0,1\}^n)^+$, and $c_i = 2$ if $x_i \in (\{0,1\}^n)^+$. We use $\hat{x}_i$, $\hat{X}_i$, $\hat{z}_i$ and $\hat{c}_i$ in the same way.

Observe that, from (1), $F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}})$ can be written as

$$F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = \bigoplus_{0 \le i \le \ell-1} L^{i+1} \cdot (z_i^\ell \cdot \langle c_i \rangle_n \oplus \hat{z}_i^\ell \cdot \langle \hat{c}_i \rangle_n). \tag{2}$$

We show (2) is a non-zero, non-constant polynomial in the following three cases.

Case 1: $|X_i| \ne |\hat{X}_i|$ for some $0 \le i \le \ell - 1$,
Case 2: $|X_i| = |\hat{X}_i|$ for all $0 \le i \le \ell-1$, but $X_i[j] \ne \hat{X}_i[j]$ for some $0 \le i \le \ell-1$ and $0 \le j \le m_i - 1$, and
Case 3: $|X_i| = |\hat{X}_i|$ for all $0 \le i \le \ell - 1$, and $X_i[j] = \hat{X}_i[j]$ for all $0 \le i \le \ell - 1$ and $0 \le j \le m_i - 1$.

For the first case, if $m_i > \hat{m}_i$, then $(z_i^\ell \cdot \langle c_i \rangle_n \oplus \hat{z}_i^\ell \cdot \langle \hat{c}_i \rangle_n)$ is a polynomial in L of degree ℓm_i. To see this, we note that the polynomial is equivalent to

$$\begin{aligned}&\big(L^{m_i} \oplus L^{m_i-1} \cdot X_i[0] \oplus L^{m_i-2} \cdot X_i[1] \oplus \cdots \oplus X_i[m_i - 1]\big)^\ell \cdot \langle c_i \rangle_n \\ \oplus &\big(L^{\hat{m}_i} \oplus L^{\hat{m}_i-1} \cdot \hat{X}_i[0] \oplus L^{\hat{m}_i-2} \cdot \hat{X}_i[1] \oplus \cdots \oplus \hat{X}_i[\hat{m}_i - 1]\big)^\ell \cdot \langle \hat{c}_i \rangle_n\end{aligned} \tag{3}$$

and therefore the coefficient of $L^{\ell m_i}$ is $\langle c_i \rangle_n \ne 0^n$. This implies $F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y$ is a non-trivial equation of degree at most $\max\{\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})\}$. Similarly, if $m_i < \hat{m}_i$, then $(z_i^\ell \cdot \langle c_i \rangle_n \oplus \hat{z}_i^\ell \cdot \langle \hat{c}_i \rangle_n)$ is a polynomial in L of degree $\ell \hat{m}_i$, and therefore $F_L^{(\ell)}(\mathcal{X}) \oplus F_L^{(\ell)}(\hat{\mathcal{X}}) = Y$ is a non-trivial equation.

For the second case, consider i and j such that $0 \le i \le \ell - 1$, $0 \le j \le m_i - 1$, and $X_i[j] \ne \hat{X}_i[j]$. Now $(z_i^\ell \cdot \langle c_i \rangle_n \oplus \hat{z}_i^\ell \cdot \langle \hat{c}_i \rangle_n)$ can be written as (3), and since we have $m_i = \hat{m}_i$, a simplification shows that the coefficient of $L^{\ell m_i}$ is $\langle c_i \rangle_n \oplus \langle \hat{c}_i \rangle_n$, and the coefficient of $L^{\ell(m_i - j - 1)}$ is $(X_i[j])^\ell \cdot \langle c_i \rangle_n \oplus (\hat{X}_i[j])^\ell \cdot \langle \hat{c}_i \rangle_n$. If $\langle c_i \rangle_n \ne \langle \hat{c}_i \rangle_n$, then the coefficient of $L^{\ell m_i}$ is non-zero. Otherwise the coefficient of $L^{\ell(m_i-j-1)}$ is non-zero since

$$(X_i[j])^\ell \cdot \langle c_i \rangle_n \oplus (\hat{X}_i[j])^\ell \cdot \langle \hat{c}_i \rangle_n = (X_i[j] \oplus \hat{X}_i[j])^\ell \cdot \langle c_i \rangle_n,$$

and the right hand side is zero if and only if $X_i[j] = \hat{X}_i[j]$.

Finally, we consider the third case. In this case, we claim that we must have $x_i \in (\{0,1\}^n)^+$ and $\hat{x}_i \notin (\{0,1\}^n)^+$ (or $x_i \notin (\{0,1\}^n)^+$ and $\hat{x}_i \notin (\{0,1\}^n)^+$) for some $0 \le i \le \ell-1$. Indeed, if $x_i, \hat{x}_i \in (\{0,1\}^n)^+$ holds for all $0 \le i \le \ell-1$, then we have $x_i = \hat{x}_i$ for all $0 \leq i \leq \ell-1$, since $X_i = \mathrm{pad}(x_i) = x_i$ and $\hat{X}_i = \mathrm{pad}(\hat{x}_i) = \hat{x}_i$. This contradicts the condition $\mathcal{X} \neq \hat{\mathcal{X}}$. Similarly, if $x_i, \hat{x}_i \notin (\{0,1\}^n)^+$ holds for all $0 \leq i \leq \ell - 1$, then again it contradicts the fact $\mathcal{X} \neq \hat{\mathcal{X}}$. Now without loss of generality, we may assume that $x_i \in (\{0,1\}^n)^+$ and $\hat{x}_i \notin (\{0,1\}^n)^+$ for some $0 \leq i \leq \ell - 1$, which implies $\langle c_i \rangle_n \neq \langle \hat{c}_i \rangle_n$. Then $(z_i^\ell \cdot \langle c_i \rangle_n \oplus \hat{z}_i^\ell \cdot \langle \hat{c}_i \rangle_n)$ can be written as (3), and since we have $m_i = \hat{m}_i$, the coefficient of $L^{\ell m_i}$ is $\langle c_i \rangle_n \oplus \langle \hat{c}_i \rangle_n \neq 0^n$. □

B Proof of Lemma 2

Let p_0 and p_1 be as defined in Sect. 4.2. We have already shown that $p_0 \leq \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{HBS[Perm}(n)]}(B)$ and $p_1 \leq \Pr(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)}$ forges). In the rest of this section we evaluate the last probability.

We introduce two oracles $\mathcal{O}_P(\cdot,\cdot)$ and $\mathcal{V}_P(\cdot,\cdot)$. The $\mathcal{O}_P(\cdot,\cdot)$ oracle takes $(T, s) \in \{0,1\}^n \times \boldsymbol{N}$ $(s < 2^n)$ as its input and returns $P(P^{-1}(T) \oplus \langle s \rangle_n)$. The $\mathcal{V}_P(\cdot,\cdot)$ oracle takes $((H, M), T)$ as its input and returns T if $P\big(F_L^{(2)}(H, M)\big) = T$ or returns $\perp$ otherwise. We observe that the $\mathcal{D}_P(\cdot,\cdot)$ oracle can be perfectly simulated by using these two oracles $\mathcal{O}_P(\cdot,\cdot)$ and $\mathcal{V}_P(\cdot,\cdot)$. Therefore, there exists an adversary A_1 such that

$$\Pr\big(A^{\mathcal{E}_P(\cdot,\cdot),\mathcal{D}_P(\cdot,\cdot)} \text{ forges}\big) \leq \Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big).$$

We use the following system of notation. Consider all queries to the $\mathcal{E}_P(\cdot,\cdot)$ oracle. Also consider those queries to the $\mathcal{V}_P(\cdot,\cdot)$ oracle which make the oracle return T (i.e., forgery). Enumerate these queries, in the order of being made, as (H_1, M_1), (H_2, M_2), $\ldots$ and set $S_i = F_L^{(2)}(H_i, M_i)$. Define sets $\mathbb{S}_i = \{S_i, S_i \oplus \langle 1 \rangle_n, S_i \oplus \langle 2 \rangle_n, \ldots, S_i \oplus \langle m_i \rangle_n\}$. Define vectors $\mathbb{Y}_i$ as follows. In the case of $\mathcal{E}_P(\cdot,\cdot)$-query, define $\mathbb{Y}_i = \big(P(S_i), P(S_i \oplus \langle 1 \rangle_n), \ldots, P(S_i \oplus \langle m_i \rangle_n)\big)$ so that $\mathbb{Y}_i[j] = P(S_i \oplus \langle j \rangle_n)$. In the case of $\mathcal{V}_P(\cdot,\cdot)$-query, define $\mathbb{Y}_i = \big(P(S_i)\big)$ (A 1-dimensional vector). With abuse of notation we identify $\mathbb{Y}_i$ with the set $\{\mathbb{Y}_i[0], \ldots, \mathbb{Y}_i[m_i]\}$.

Consider all queries to the $\mathcal{O}_P(\cdot,\cdot)$ oracle. Let (T_1, s_1), (T_2, s_2), $\ldots$ be these queries and $Y_1, Y_2, \ldots$ the values returned by the oracle so that we have $Y_i = \mathcal{O}_P(T_i, s_i)$. We classify the queries to the $\mathcal{O}_P(\cdot,\cdot)$ oracle into three categories: *root* queries, *chain* queries, and *extension* queries.

Root. We say that (T_i, s_i) is a *root query* if $T_i \neq T_j$, $T_i \neq Y_j$ for all $j < i$ and $T_i \notin \mathbb{Y}_k$ for each previous k-th query to the $\mathcal{E}_P(\cdot,\cdot)$ / $\mathcal{V}_P(\cdot,\cdot)$ oracle.

Chain. We say that (T_i, s_i) is a *chain query* if there exists a $j < i$ such that the j-th query (T_j, s_j) was a root query and either $T_i = T_j$ or $T_i = Y_j$. Recursively, we call (T_i, s_i) a chain query also if there exists a $j < i$ such that the j-th query (T_j, s_j) was a chain query and $T_i = Y_j$.

Extension. We say that (T_i, s_i) is an *extension query* if there exists some previous k-th query to the $\mathcal{E}_P(\cdot,\cdot)$ / $\mathcal{V}_P(\cdot,\cdot)$ oracle such that $T_i \in \mathbb{Y}_k$. Recursively, we call (T_i, s_i) an extension query also if there exists a $j < i$ such that the j-the query (T_j, s_j) was an extension query and $T_i = Y_j$.

Note that some of the chain/extension queries to the $\mathcal{O}_P(\cdot,\cdot)$ oracle, even if the query itself is new, may be *trivial* in the sense that the adversary A_1 already knows the to-be-returned value. For example, if $P(T \oplus \langle s_1 \rangle_n) = Y_1$ and $P(T \oplus \langle s_2 \rangle_n) = Y_2$, then we know that $Y_2 = \mathcal{O}_P(Y_1, \langle s_1 \rangle_n \oplus \langle s_2 \rangle_n)$ (The second argument is treated as an integer). Whenever possible, we implicitly exclude these trivial queries from our consideration.

Now we are ready to introduce a modified oracle $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ for A_1, where $\tilde{P} \overset{R}{\leftarrow} \mathrm{Perm}(n)$ is a random permutation independent of P. The oracle is defined in the style of lazy sampling for $\tilde{P}$. The oracle keeps a record of domain points $\tilde{S}_1, \tilde{S}_2, \ldots$ and that of range points $\tilde{Y}_1, \tilde{Y}_2, \ldots$ in a linked way. For this, the oracle needs to observe the vectors $\mathbb{Y}_i$ output by the $\mathcal{E}_P(\cdot,\cdot)$ / $\mathcal{V}_P(\cdot,\cdot)$ oracle. Let (T, s) be the current query made to the $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ oracle.

1. If (T, s) is a trivial query, then the oracle simply returns the expected value.
2. If (T, s) is a root query, the oracle adds the point T to the set of range points $\{\tilde{Y}_1, \tilde{Y}_2, \ldots\}$. Then the oracle picks $\tilde{S} \overset{R}{\leftarrow} \{0,1\}^n \setminus \big(\{0^n\} \cup \bigcup_i \{\tilde{S}_i\}\big)$, where i runs over already-defined domain points. The oracle updates the record $\{\tilde{S}_1, \tilde{S}_2, \ldots\}$ by adding the new domain point $\tilde{S}$ (The oracle establishes a link $\tilde{P}(\tilde{S}) = T$). If $s = 0$, then the oracle returns T. If $s \geq 1$, then the oracle checks if $\tilde{S} \oplus \langle s \rangle_n \in \{\tilde{Y}_1, \tilde{Y}_2, \ldots\}$. If so, then the oracle returns $\tilde{P}(\tilde{S} \oplus \langle s \rangle_n)$. If not, then the oracle adds the point $\tilde{S} \oplus \langle s \rangle_n$ to the set of domain points, picks $\tilde{Y} \overset{R}{\leftarrow} \{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_j \{\tilde{Y}_j\}\big)$, adds $\tilde{Y}$ to the set of range points and returns $\tilde{Y}$.
3. If (T, s) is a chain query, then it means that $\tilde{S} = \tilde{P}^{-1}(T)$ is in the set $\{\tilde{S}_1, \tilde{S}_2, \ldots\}$ but $\tilde{S} \oplus \langle s \rangle_n$ is not. So the oracle adds the point to the set $\{\tilde{S}_1, \tilde{S}_2, \ldots\}$, picks $\tilde{Y} \overset{R}{\leftarrow} \{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_j \{\tilde{Y}_j\}\big)$ and adds $\tilde{Y}$ to the set of range points. This establishes a link $\tilde{P}(\tilde{S} \oplus \langle s \rangle_n) = \tilde{Y}$, and the value $\tilde{Y}$ is returned.
4. Finally, if (T, s) is an extension query, then it means that the oracle has located a vector $\mathbb{Y}_i$ and an integer t such that $T = \mathcal{O}_{P,\tilde{P}}(\mathbb{Y}_i[0], t)$. The task is to return a value corresponding to $\mathcal{O}_{P,\tilde{P}}(\mathbb{Y}_i[0], \langle t \rangle_n \oplus \langle s \rangle_n)$. For this, the oracle picks $\tilde{Y} \overset{R}{\leftarrow} \{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_j \{\tilde{Y}_j\}\big)$, adds $\tilde{Y}$ to the set of range points and returns $\tilde{Y}$. Note that no domain point is selected; no domain point is linked to the range point $\tilde{Y}$. However, the range point $\tilde{Y}$ is linked to the vector $\mathbb{Y}_i$ through the "distance" $\langle t \rangle_n \oplus \langle s \rangle_n$.

We want to evaluate the quantity

$$p_2 = \Big|\Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big) - \Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big)\Big|. \quad (4)$$

It turns out that the evaluation of p_2 requires a fair amount of work. So we defer the treatment of p_2 until we finish evaluating the second forgery probability in (4). Observe that there exists an adversary A_2 such that

$$\Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big) \leq \Pr\big(A_2^{\mathcal{E}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big),$$

because the adversary A_2 can perfectly simulate the $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ oracle by observing the vectors $\mathbb{Y}_i$ and performing lazy sampling for $\tilde{P}$.

We apply the standard argument that distinguishing is easier than forging. Namely, there exists an adversary B such that

$$\Pr(A_2^{\mathcal{E}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}) \leq \mathbf{Adv}^{\text{priv}}_{\text{HBS[Perm}(n)]}(B) + \frac{q}{2^n}.$$

The adversary B simulates the two oracles for A_2 by using its own oracle and outputs 1 as soon as A_2 forges. If B's oracle is $\mathcal{E}_P(\cdot,\cdot)$, the simulation is perfect and hence $\Pr\big(B^{\mathcal{E}_P(\cdot,\cdot)} = 1\big) = \Pr\big(A_2^{\mathcal{E}_P(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ forges}\big)$. If B's oracle is $\mathcal{R}(\cdot,\cdot)$, then each time A_2 makes a query to the second oracle, the probability of forgery is $1/2^n$ (prior to the execution of the game) and so $\Pr\big(B^{\mathcal{R}(\cdot,\cdot)} = 1\big) \leq q/2^n$.

Now we come back to the evaluation of p_2. Consider those queries to the $\mathcal{V}_P(\cdot,\cdot)$ oracle for which the oracle returns $\perp$. Enumerate these queries, in the order of being made, as $((H_1,M_1),T_1), ((H_2,M_2),T_2),\ldots$. Set $U_i = F_L^{(2)}(H_i,M_i)$ and $Z_i = P(U_i)$.

For $S \in \{0,1\}^n$, define $\mathcal{N}(S) = \bigcup_{s_1,\ldots,s_q}\{S \oplus \langle s_1\rangle_n \oplus \cdots \oplus \langle s_q\rangle_n\}$, where each s_i runs over $0 \leq s_i \leq m_{\max}$. This set is not so large, as we have $\mathcal{N}(S) \subset \{S, S\oplus\langle 1\rangle_n, \ldots, S\oplus\langle 2m_{\max}\rangle_n\}$. Put $\mathbb{S}_i^* = \mathcal{N}(S_i)$, $\tilde{\mathbb{S}}_i^* = \mathcal{N}(\tilde{S}_i)$ and $\mathbb{U}_i^* = \mathcal{N}(U_i)$.

We consider the following bad events in the latter game in (4). (i) $\mathbb{S}_i^* \ni 0^n$ or $\mathbb{U}_i^* \ni 0^n$ for some i. (ii) $\mathbb{S}_i^* \cap \mathbb{S}_j^* \neq \emptyset$, $\mathbb{U}_i^* \cap \mathbb{U}_j^* \neq \emptyset$ or $\mathbb{S}_i^* \cap \mathbb{U}_j^* \neq \emptyset$ for some i and j such that $(H_i,M_i) \neq (H_j,M_j)$. (iii) $\mathbb{S}_i^* \cap \tilde{\mathbb{S}}_j^* \neq \emptyset$ or $\mathbb{U}_i^* \cap \tilde{\mathbb{S}}_j^* \neq \emptyset$ for some i and j. (iv) $\tilde{\mathbb{S}}_i^* \ni 0^n$ for some i. (v) $\mathbb{Y}_i \ni \tilde{Y}_j$ for some i and j. (vi) $Z_i = \tilde{Y}_j$ for some i and j. (vii) $L = \tilde{Y}_i$ for some i.

We argue that the two games in (4) (referred to as the *first* and the *second* games) proceed exactly the same as long as none of (i)–(vii) bad events occurs. This means that we have

$$p_2 \leq \Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ causes one of (i)–(vii)}\big).$$

To see this, it is helpful to consider the permutation P as lazily sampled. We check this one by one.

1. Consider a query (say the i-th query) either to the $\mathcal{E}_P(\cdot,\cdot)$ oracle or to the $\mathcal{V}_P(\cdot,\cdot)$ oracle (returning 1). Thanks to (i), (ii) and (iii), it is guaranteed that the set $\mathbb{S}_i$ consists of entirely fresh domain points for P (unless a query with (H_i,M_i) has been made to the $\mathcal{V}_P(\cdot,\cdot)$ oracle returning $\perp$). This results in lazy sampling of points for $\mathbb{Y}_i$. In the first game, these points are sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_j \mathbb{Y}_j \cup \bigcup_j\{\tilde{Y}_j\} \cup \bigcup_j\{Z_j\}\big)$, where j runs over already-defined points. In the second game, the points are sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_j \mathbb{Y}_j \cup \bigcup_j\{Z_j\}\big)$. The two distributions remain the same due to (v).
2. Consider a root query (T,s) to the $\mathcal{O}_P(\cdot,\cdot)$ or $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ oracle. Note that (vii) eliminates the possibility $T = L$. In the first game, a domain point S is sampled for $P^{-1}(T)$ from $\{0,1\}^n \setminus \big(\{0^n\} \cup \bigcup_i \mathbb{S}_i \cup \bigcup_i\{\tilde{S}_i\} \cup \bigcup_i\{U_i\}\big)$. In the second game, a domain point S is sampled (for $\tilde{P}^{-1}(T)$) from $\{0,1\}^n\setminus\big(\{0^n\}\cup$

$\bigcup_i \{\tilde{S}_i\}\big)$. These are the same distribution due to (iii). In both the games, the shifted domain point $S \oplus \langle s \rangle_n$ is not fresh under the same condition that there exists some i such that $S \oplus \langle s \rangle_n = \tilde{S}_i$. This is due to (iii) and (iv). If it is fresh, then in the first game a range point $\tilde{Y}$ is sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\} \cup \bigcup_i \{Z_i\}\big)$, whereas in the second game a range point $\tilde{Y}$ is sampled from $\{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\}\big)$. These are the same owing to (vi) and (vii).
3. Consider a chain query to the $\mathcal{O}_P(\cdot,\cdot)$ or $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ oracle. Thanks to (i), (ii) and (iii), the query results in a fresh domain point $P^{-1}(T) \oplus \langle s \rangle_n$ in the first game and $\tilde{P}^{-1}(T) \oplus \langle s \rangle_n$ in the second. In the first game, the corresponding range point is sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\} \cup \bigcup_i \{Z_i\}\big)$. In the second game, the range point is sampled from $\{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\}\big)$. These two yield the same distribution because of (vi) and (vii).
4. Consider an extension query (T, s) to the $\mathcal{O}_P(\cdot,\cdot)$ or $\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot)$ oracle. Thanks to (i), (ii) and (iii), the query results in a fresh domain point $P^{-1}(T) \oplus \langle s \rangle_n$. In the first game, the corresponding range point is sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\} \cup \bigcup_i \{Z_i\}\big)$. In the second game, the range point is sampled from $\{0,1\}^n \setminus \big(\bigcup_i \mathbb{Y}_i \cup \bigcup_i \{\tilde{Y}_i\}\big)$. These two sampling processes yield the same distribution due to (vi) and (vii).
5. Consider a query (say the i-th query) to the $\mathcal{V}_P(\cdot,\cdot)$ oracle returning $\bot$. If a query with (H_i, M_i) has been made to one of the oracles, then no sampling is performed. Otherwise, owing to (i), (ii) and (iii), the point U_i is a fresh domain point. This results in lazy sampling of Z_i. In the first game, Z_i is sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_j \mathbb{Y}_j \cup \bigcup_j \{\tilde{Y}_j\} \cup \bigcup_j \{Z_j\}\big)$. In the second game, it is sampled from $\{0,1\}^n \setminus \big(\{L\} \cup \bigcup_j \mathbb{Y}_j \cup \bigcup_j \{Z_j\}\big)$. These two distributions remain the same because of (vi).

Now we replace P with a random function $R \overset{R}{\leftarrow} \mathrm{Func}(n)$ (The set of all functions on $\{0,1\}^n$). This is possible because the inverse cipher P^{-1} never appears in the definitions of the three oracles. By the PRP/PRF switching lemma [2], we get

$$\Big|\Pr\big(A_1^{\mathcal{E}_P(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P(\cdot,\cdot)} \text{ causes one of (i)–(vii)}\big) \\ - \Pr\big(A_1^{\mathcal{E}_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}_R(\cdot,\cdot)} \text{ causes one of (i)–(vii)}\big)\Big| \le \frac{(1+q+qm_{\max})^2}{2^{n+1}}.$$

We introduce a modified oracle $\mathcal{E}^{\bullet}_R(\cdot,\cdot)$. This oracle behaves just like the $\mathcal{E}_R(\cdot,\cdot)$ oracle, computing the values for $\mathbb{S}_i$ using $L = R(0^n)$, except that at the end of query process, even if a bad event has occurred, the oracle returns a random string, which defines the value of $\mathbb{Y}_i$ (unless a query with (H_i, M_i) has been made to one of the oracles, in which case those already-defined values are used). The $\mathcal{V}^{\bullet}_R(\cdot,\cdot)$ oracle is defined similarly. That is, unless a query with (H_i, M_i) has been already made, the oracle computes the value S_i / U_i using $L = R(0^n)$ and, regardless of a bad event, picks a random point in $\{0,1\}^n$. If this point happens to be the same as T, then the oracle returns T. Otherwise, the random point is set to Z_i, and the oracle returns $\perp$. The modified oracles are identical to the original ones until a bad event occurs, so we have

$$\begin{aligned}\Pr\big(A_1^{\mathcal{E}_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}_R(\cdot,\cdot)} &\text{ causes one of (i)–(vii)}\big)\\ &= \Pr\big(A_1^{\mathcal{E}^\bullet_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}^\bullet_R(\cdot,\cdot)} \text{ causes one of (i)–(vii)}\big).\end{aligned}$$

Now observe that the values returned by the three oracles are independent of the value $L = R(0^n)$. So we are ready to evaluate the probabilities of (i)–(vii). Without loss of generality we assume that the bad events are disjoint. This can be done by defining a bad event to occur prior to any other bad event. So, for example, when we say "the event (i) occurs," we implicitly mean that no other bad event has occurred.

An event of type (i) occurs upon a query (H_i, M_i), made to the $\mathcal{E}^\bullet_R(\cdot,\cdot)$ oracle or to $\mathcal{V}^\bullet_R(\cdot,\cdot)$, which satisfies an equation $F_L^{(2)}(H_i, M_i) = \langle s\rangle_n$ for some $0 \le s \le 2m_{\max}$. Here, we have $1 \le i \le q$, and the degree of the equation is at most $2(1 + h_{\max} + m_{\max})$. Therefore, we have

$$\Pr\big(A_1^{\mathcal{E}^\bullet_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}^\bullet_R(\cdot,\cdot)} \text{ causes (i)}\big) \le \frac{2q(1+2m_{\max})(1+h_{\max}+m_{\max})}{2^n}.$$

Event (ii) corresponds to queries (H_i, M_i) and (H_j, M_j) $(1 \le i < j \le q)$, made to the $\mathcal{E}^\bullet_R(\cdot,\cdot)$ oracle or to $\mathcal{V}^\bullet_R(\cdot,\cdot)$, which satisfy an equation $F_L^{(2)}(H_i, M_i) \oplus F_L^{(2)}(H_j, M_j) = \langle s\rangle_n$ with $0 \le s \le 2m_{\max}$. We have

$$\Pr\big(A_1^{\mathcal{E}^\bullet_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}^\bullet_R(\cdot,\cdot)} \text{ causes (ii)}\big) \le \frac{q^2(1+2m_{\max})(1+h_{\max}+m_{\max})}{2^n}.$$

Event (iii) corresponds to a query (H_i, M_i) and a root query (T_j, s_j) (either one may be made before the other) such that $F_L^{(2)}(H_i, M_i) = \tilde{S}_j \oplus \langle s\rangle_n$ with $0 \le s \le 2m_{\max}$. Note that A_1 makes at most q-many root queries (rather than $q(1 + m_{\max})$-many). So we have

$$\Pr\big(A_1^{\mathcal{E}^\bullet_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}^\bullet_R(\cdot,\cdot)} \text{ causes (iii)}\big) \le \frac{2q^2(1+2m_{\max})(1+h_{\max}+m_{\max})}{2^n}.$$

An event of type (iv) occurs upon a root query (T_i, s_i) satisfying $\tilde{S}_j = \langle s\rangle_n$ for some $0 \le s \le 2m_{\max}$. The sampling is performed from the space $\{0,1\}^n \setminus \{0^n\}$ for at most q-many distinct points, so we have

$$\Pr\big(A_1^{\mathcal{E}^\bullet_R(\cdot,\cdot),\mathcal{O}_{R,\tilde{P}}(\cdot,\cdot),\mathcal{V}^\bullet_R(\cdot,\cdot)} \text{ causes (iv)}\big) \le \frac{1+2m_{\max}-1+1}{2^n} = \frac{1+2m_{\max}}{2^n}.$$

An event of type (v) occurs upon a new query (H_i, M_i) to the $\mathcal{E}^\bullet_R(\cdot,\cdot)$ oracle or to the $\mathcal{V}^\bullet_R(\cdot,\cdot)$ oracle returning T. Such an query causes random sampling of the vector $\mathbb{Y}_i$. The values $\tilde{Y}_1, \tilde{Y}_2, \ldots$ consist of root points selected by A_1 plus additional "semi-random" points. The number of random sampling for the vectors $\mathbb{Y}_i$ is at most $q(1+m_{\max})$, and the size of $\{\tilde{Y}_1, \tilde{Y}_2, \ldots\}$ is at most $q(1+m_{\max})$. So we obtain

$$\Pr\bigl(A_1^{\mathcal{E}_P^\bullet(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P^\bullet(\cdot,\cdot)} \text{ causes (v)}\bigr) \le \frac{q^2(1+m_{\max})^2}{2^n}.$$

For event (vi), there are three cases. The first case is upon the sampling of Z_i. This case can be treated similarly to event (v), and the probability is at most $q \cdot q(1+m_{\max})/2^n = q^2(1+m_{\max})/2^n$. The second case is upon a root query (T, s) such that $T = Z_i$ for some i. The values $Z_1, Z_2, \ldots$ are simply random points, the number of which is at most q. There are at most q-many root queries, so the probability is at most $q^2/2^n$. The third case is upon the sampling of a range point $\tilde{Y}_j$. Note that such a sampling is performed for at most $q(1+m_{\max})$ times. So the probability is at most $q^2(1+m_{\max})/2^n$. Overall, the probability for event (vi) is at most

$$\frac{q^2(1+m_{\max})}{2^n} + \frac{q^2}{2^n} + \frac{q^2(1+m_{\max})}{2^n} \le \frac{2q^2(1+2m_{\max})}{2^n}.$$

For event (vii), observe that the number of points $\tilde{Y}_1, \tilde{Y}_2, \ldots$ is at most $q(1+m_{\max})$. So we simply have

$$\Pr\bigl(A_1^{\mathcal{E}_P^\bullet(\cdot,\cdot),\mathcal{O}_{P,\tilde{P}}(\cdot,\cdot),\mathcal{V}_P^\bullet(\cdot,\cdot)} \text{ causes (vii)}\bigr) \le \frac{q(1+m_{\max})}{2^n}.$$

Finally we sum up the terms. This yields

$$\begin{aligned} p_2 \le (1+2+1+2+1+1+2+1) & \frac{q^2(1+h_{\max}+2m_{\max})^2}{2^n} \\ & = \frac{11q^2(1+h_{\max}+2m_{\max})^2}{2^n}. \end{aligned}$$

□

C Proof of Lemma 3

We consider the encryption algorithm $\mathcal{E}_R(\cdot,\cdot)$ of "HBS[Func(n)]," where Func(n) is the set of all functions over $\{0,1\}^n$, and a random function $R \overset{R}{\leftarrow} \text{Func}(n)$ is used as the underlying blockcipher (Obviously the decryption algorithm cannot be defined). We see that

$$\mathbf{Adv}^{\text{priv}}_{\text{HBS[Perm}(n)]}(B) \le \mathbf{Adv}^{\text{priv}}_{\text{"HBS[Func}(n)]\text{"}}(B) + \frac{(1+q+qm_{\max})^2}{2^{n+1}} \qquad (5)$$

from the PRP/PRF switching lemma [2]. We then use the following lemma to prove Lemma 3.

Lemma 4. *Let $h_0, \ldots, h_{q-1}$, $m_0, \ldots, m_{q-1}$ be integers such that $h_i \le h_{\max}$ and $m_i \le m_{\max}$ for $0 \le i \le q-1$. Also, let $H_0, \ldots, H_{q-1}$, $M_0, \ldots, M_{q-1}$, $T_0, \ldots, T_{q-1}$, and $C_0, \ldots, C_{q-1}$ be bit strings that satisfy the following conditions for $0 \le i \le q-1$. $H_i \in (\{0,1\}^n)^{h_i}$, $M_i \in (\{0,1\}^n)^{m_i}$, $T_i \in \{0,1\}^n$, and $C_i \in (\{0,1\}^n)^{m_i}$. Furthermore, assume $(H_i, M_i) \ne (H_j, M_j)$ holds for $0 \le i < j \le q-1$. Then we have*

$$\frac{p_3}{p_4} \ge 1 - \frac{3q^2(1 + h_{\max} + 2m_{\max})^2}{2^n}, \tag{6}$$

where we define p_3 and p_4 as $p_3 \overset{\text{def}}{=} \Pr(\mathcal{E}_R(H_i, M_i) = (T_i, C_i)$ for $0 \le i \le q-1)$ and $p_4 \overset{\text{def}}{=} \Pr(\mathcal{R}(H_i, M_i) = (T_i, C_i)$ for $0 \le i \le q-1)$.

Proof. The proof is based on a counting argument. We count the number of functions $R \in \text{Func}(n)$ that satisfy $\mathcal{E}_R(H_i, M_i) = (T_i, C_i)$ for $0 \le i \le q-1$. We first count the number of $L = R(0^n)$ and then count the rest. Let $S_i = F_L^{(2)}(H_i, M_i)$. S_i corresponds to the initial counter value. Consider the set $\mathbb{S}_i = \{S_i, S_i \oplus \langle 1 \rangle_n, S_i \oplus \langle 2 \rangle_n, \ldots, S_i \oplus \langle m_i \rangle_n\}$. Now we claim that the number of $L \in \{0,1\}^n$ such that

$$\{0^n\} \cap \mathbb{S}_i = \emptyset \text{ for } 0 \le i \le q-1 \text{ and } \mathbb{S}_i \cap \mathbb{S}_j = \emptyset \text{ for } 0 \le i < j \le q-1 \tag{7}$$

is at least $2^n - 2q(1+m_{\max})(1+h_{\max}+m_{\max}) - q^2(1+2m_{\max})(1+h_{\max}+m_{\max})$.

Fix $0 \le i \le q-1$ and $0 \le s \le m_i$. Consider the equation $F_L^{(2)}(H_i, M_i) = \langle s \rangle_n$. As we have seen in Sect. 3.1, $F_L^{(2)}(H_i, M_i)$ is a non-zero polynomial in L of degree $\mu(H_i, M_i) = \max\{1 + 2h_i, 2 + 2m_i\}$. We therefore have $\#\{L \mid F_L^{(2)}(H_i, M_i) = \langle s \rangle_n\} \le \max\{1 + 2h_i, 2 + 2m_i\} \le 2(1 + h_{\max} + m_{\max})$, which implies

$$\#\{L \mid \{0^n\} \cap \mathbb{S}_i \ne \emptyset \text{ for some } 0 \le i \le q-1\} \le 2q(1 + m_{\max})(1 + h_{\max} + m_{\max}).$$

Next, fix $0 \le i < j \le q-1$ and consider the equation $F_L^{(2)}(H_i, M_i) \oplus \langle s \rangle_n = F_L^{(2)}(H_j, M_j) \oplus \langle t \rangle_n$, where $0 \le s \le m_i$ and $0 \le t \le m_j$. Theorem 1 implies that this equation has at most $\max\{\mu(H_i, M_i), \mu(H_j, M_j)\} = \max\{1 + 2h_i, 2 + 2m_i, 1 + 2h_j, 2 + 2m_j\}$ solutions. Observe that the equation is equivalent to $F_L^{(2)}(H_i, M_i) \oplus F_L^{(2)}(H_j, M_j) = \langle s \rangle_n \oplus \langle t \rangle_n$, and the right hand side takes at most $(1 + m_i + m_j)$ values (rather than $(1 + m_i)(1 + m_j)$ values). We thus have

$$\begin{aligned}\#\{L \mid \mathbb{S}_i \cap \mathbb{S}_j \ne \emptyset\} &\le (1 + m_i + m_j) \max\{1 + 2h_i, 2 + 2m_i, 1 + 2h_j, 2 + 2m_j\} \\ &\le 2(1 + 2m_{\max})(1 + h_{\max} + m_{\max}),\end{aligned}$$

which implies $\#\{L \mid \mathbb{S}_i \cap \mathbb{S}_j \ne \emptyset$ for some $0 \le i < j \le q-1\}$ is at most

$$q^2(1 + 2m_{\max})(1 + h_{\max} + m_{\max}).$$

Once we fix any L that satisfies (7), the inputs to R, $\{0^n\} \cup \mathbb{S}_0 \cup \cdots \cup \mathbb{S}_{q-1}$, are all distinct. Therefore, the left hand side of (6) is at least

$$\frac{2^n - 2q(1 + m_{\max})(1 + h_{\max} + m_{\max}) - q^2(1 + 2m_{\max})(1 + h_{\max} + m_{\max})}{2^n},$$

which is at least $1 - (3q^2(1 + h_{\max} + 2m_{\max})^2)/2^n$. □

Proof (of Lemma 3). Without loss of generality, we assume that B makes exactly q oracle queries. Also, since B is computationally unbounded, we assume that B is deterministic. Now we can regard B as a function $f_B : (\{0,1\}^n)^{q(1+m_{\max})} \to \{0,1\}$. To see this, let $Y \in (\{0,1\}^n)^{q(1+m_{\max})}$ be an arbitrary bit string of $q(1+m_{\max})$ blocks. The first query, (H_0, M_0), is determined by B. If we return the first $n + |M_0|$ bits of Y, then the next query, (H_1, M_1), is determined. Similarly, if we return the next $n + |M_1|$ bits of Y, then the next query, (H_2, M_2), is determined. By continuing the procedure, the output of B, either 0 or 1, is determined. Therefore, the output of B and the value of q queries, $(H_0, M_0), \ldots, (H_{q-1}, M_{q-1})$, are all determined by fixing Y. Let $\mathbf{v}_{\text{one}} = \{Y \mid f_B(Y) = 1\}$, and $P_{\mathcal{R}} \overset{\text{def}}{=} \Pr(B^{\mathcal{R}(\cdot,\cdot)} = 1)$. Then we have

$$P_{\mathcal{R}} = \sum_{Y \in \mathbf{v}_{\text{one}}} p_4. \tag{8}$$

On the other hand, let $P_{\text{HBS}} \overset{\text{def}}{=} \Pr(A^{\mathcal{E}_R(\cdot,\cdot)} = 1)$ and observe

$$P_{\text{HBS}} = \sum_{Y \in \mathbf{v}_{\text{one}}} p_3 \geq \left(1 - \frac{3q^2(1 + h_{\max} + 2m_{\max})^2}{2^n}\right) \sum_{Y \in \mathbf{v}_{\text{one}}} p_4,$$

where the last inequality follows from Lemma 4. Then P_{HBS} is at least

$$\left(1 - \frac{3q^2(1 + h_{\max} + 2m_{\max})^2}{2^n}\right) P_{\mathcal{R}} \geq P_{\mathcal{R}} - \frac{3q^2(1 + h_{\max} + 2m_{\max})^2}{2^n}$$

from (8). Now, we have $P_{\text{HBS}} \geq P_{\mathcal{R}} - (3q^2(1 + h_{\max} + 2m_{\max})^2)/2^n$, and by applying the same argument to $1 - P_{\text{HBS}}$ and $1 - P_{\mathcal{R}}$, we have $1 - P_{\text{HBS}} \geq 1 - P_{\mathcal{R}} - (3q^2(1 + h_{\max} + 2m_{\max})^2)/2^n$. Finally, from (5), we obtain the claimed bound. $\square$