J. Cryptol. (2012) 25: 640–679
DOI: 10.1007/s00145-011-9106-1

Journal of
CRYPTOLOGY

On-line Ciphers and the Hash-CBC Constructions

M. Bellare

Dept. of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
mihir@cs.ucsd.edu
url: http://www-cse.ucsd.edu/users/mihir

A. Boldyreva*

Department of Computer Science, Georgia Institute of Technology, 266 Ferst Dr., Atlanta, GA 30332-0765, USA
sasha@gatech.edu
url: http://www-static.cc.gatech.edu/~aboldyre/index.html

L. Knudsen

Department of Mathematics, Technical University of Denmark, Building 303, 2800 Lyngby, Denmark
Lars.R.Knudsen@mat.dtu.dk
url: http://www2.mat.dtu.dk/people/Lars.R.Knudsen

C. Namprempre

Electrical and Computer Engineering, Faculty of Engineering, Thammasat University, Klong Luang, Patumtani, 12121 Thailand
nchanath@engr.tu.ac.th
url: http://chanathip.ece.engr.tu.ac.th

Communicated by Kenneth Paterson

Received 29 May 2007
Online publication 7 September 2011

Abstract. We initiate a study of on-line ciphers. These are ciphers that can take input plaintexts of large and varying lengths and will output the ith block of the ciphertext after having processed only the first i blocks of the plaintext. Such ciphers permit length-preserving encryption of a data stream with only a single pass through the data. We provide security definitions for this primitive and study its basic properties. We then provide attacks on some possible candidates, including CBC with fixed IV. We then provide two constructions, HCBC1 and HCBC2, based on a given block cipher E and a family of computationally AXU functions. HCBC1 is proven secure against chosen-plaintext attacks assuming that E is a PRP secure against chosen-plaintext attacks, while HCBC2 is proven secure against chosen-ciphertext attacks assuming that E is a PRP secure against chosen-ciphertext attacks.

Key words. On-line ciphers, Pseudorandomness

* A. Boldyreva work done while at University of California, San Diego.

1. Introduction

We begin by saying what we mean by on-line ciphers. We then describe a notion of security for them, and discuss constructions and analyses. Finally, we discuss usage, applications, and related work.

1.1. *On-line Ciphers*

A *cipher* over domain D is a function $F\colon \{0,1\}^k \times D \to D$ such that for each key K the map $F(K,\cdot)$ is a length-preserving permutation on D, and possession of K enables one to both compute and invert $F(K,\cdot)$. The most popular examples are block ciphers, where $D = \{0,1\}^n$ for some n called the block length; these are fundamental tools in cryptographic protocol design. However, one might want to encipher data of large size, in which case one needs a cipher whose domain D is appropriately large. (A common choice, which we make, is to set the domain to $D_{d,n}$, the set of all strings having a length that is at most nd for some large number d.) Matyas and Meyer refer to these as "general" ciphers [21].

In this paper, we are interested in general ciphers that are computable in an on-line manner. Specifically, cipher F is said to be *on-line* if the following is true. View the input plaintext $M = M[1]\cdots M[l]$ to an instance $F(K,\cdot)$ of the cipher as a sequence of n-bit blocks, and similarly for the output ciphertext $F(K,M) = C[1]\cdots C[l]$. Then, given the key K, for all i, it should be possible to compute output block $C[i]$ after having seen input blocks $M[1]\cdots M[i]$. That is, $C[i]$ does not depend on blocks $i+1,\ldots,l$ of the plaintext.

On-line ciphers permit real-time, length-preserving encryption of a data stream without recourse to buffering, which can be attractive in some practical settings. On-line ciphers also have other applications that we discuss later in Sects. 1.6, 1.7 and 8.

The intent of this paper is to find efficient, proven secure constructions of on-line ciphers and to further explore the applications. Let us now present the relevant security notions and our results.

1.2. *A Notion of Security for On-line Ciphers*

A commonly accepted notion of security to target for a cipher is that it be a pseudo-random permutation (PRP), as defined by Luby and Rackoff [19]. Namely, for a cipher F to be a PRP, it should be computationally infeasible, given an oracle g, to have non-negligible advantage in distinguishing between the case where g is a random instance of F and the case where g is a randomly-chosen, length-preserving permutation on the domain of the cipher. However, if a cipher is on-line, then the ith block of the ciphertext does not depend on blocks $i+1, i+2, \ldots$ of the plaintext. This is necessary, since otherwise it would not be possible to output the ith ciphertext block having seen only the first i plaintext blocks. Unfortunately, this condition impacts security, since a cipher with this property certainly cannot be a PRP. An easy distinguishing test is to ask the given oracle g the two-block queries AB and AC, getting back outputs WX and YZ, respectively, and if $W = Y$ then bet that g is an instance of the cipher. This test has a very high advantage since the condition being tested fails with high probability for a random length-preserving permutation.

For an on-line cipher, then, we must give up on the requirement that it meets the security property of being a PRP. Instead, we define and target an appropriate alternative notion of security. This is quite natural; we simply ask that the cipher behave "as randomly as possible" subject to the constraint of being on-line. We say that a length-preserving permutation π is *on-line* if for all i the ith output block of π depends only on the first i input blocks to π, and let $\mathsf{OPerm}_{d,n}$ denote the set of all length-preserving permutations π on domain $D_{d,n}$. The rest of the definition follows that of a PRP, with members of this new set playing the role of the "ideal" objects to which cipher instances are compared: it should be computationally infeasible, given an oracle g, to have non-negligible advantage in distinguishing between the case where g is a random instance of F and the case where g is a random member of $\mathsf{OPerm}_{d,n}$. A cipher secure in this sense is called an on-line-PRP.

The fact that an on-line-PRP meets a notion of security that is relatively weak compared to a PRP might at first lead one to question its value. The point, however, is that it requires two (or more) passes through the data to compute a PRP, which is prohibitive or even impossible in some settings. An on-line-PRP offers a good security to cost trade-off, and in fact the best possible security subject to the constraint of being on-line and length-preserving.

1.3. *Candidates for On-line Ciphers*

To the best of our knowledge, the problem of designing on-line ciphers satisfying our definition of security has not been explicitly addressed before. When one comes to consider this problem, however, it is natural to test first some existing candidate ciphers or natural constructions from the literature. We consider some of them and present attacks that are helpful to gather intuition about the kinds of security properties we are seeking.

It is natural to begin with standard modes of operation of a block cipher, such as CBC. However, CBC is an encryption scheme, not a cipher; each invocation uses a new random initial vector as a starting point and makes this part of the ciphertext. In particular, it is not length-preserving. The natural way to modify it to be a cipher is to fix the initial vector. There are a couple of choices: make it a known public value, or, hopefully better for security, make it a key that will be part of the secret key of the cipher. The resulting ciphers are certainly on-line, but they do not meet the notion of security we have defined. In other words, the CBC cipher with fixed IV, whether public or private, can be easily distinguished from a random on-line permutation. The attack works by creating and exploiting input-block collisions. See Sect. 4.

We then consider the Accumulated Block Chaining (ABC) mode proposed by Knudsen in [16], which is a generalization of the Infinite Garble Extension mode proposed by Campbell [11]. It was designed to have "infinite error propagation," a property that intuitively seems necessary for a secure on-line cipher but which, as we will see, is not sufficient. In Sect. 4, we present attacks demonstrating that this is not a secure on-line cipher.

1.4. *The HCBC1 On-line Cipher and Its Security*

We provide a construction of a secure on-line cipher based on a given block cipher $E\colon \{0,1\}^{ek} \times \{0,1\}^n \to \{0,1\}^n$ and an auxiliary family $H\colon \{0,1\}^{hk} \times \{0,1\}^n \to \{0,1\}^n$.

The key $eK \| hK$ for an instance of the cipher $\mathsf{HCBC1}(eK \| hK, \cdot)$ consists of a key eK for the block cipher and a key hK specifying a member $H(hK, \cdot)$ of the family H. The construction is just like CBC, except that a ciphertext block is first hashed via $H(hK, \cdot)$ before being XORed with the next plaintext block. (The initial vector is fixed to 0^n.) We stress that the hash functions map $2n$ bits to n bits, meaning work on inputs of the block length, as does the given block cipher. A picture is in Fig. 5, and a full description of the construction is in Sect. 6. It is easy to see that this cipher is on-line.

We prove that HCBC1 meets the notion of security for an on-line cipher that we discussed above, assuming that the underlying block cipher E is a PRP and that H is computationally AXU (Almost XOR Universal). The family H can be instantiated either via an AXU family [17,24,26] or a block cipher, with the latter again assumed to be a PRP. With the latter, we obtain a purely block-cipher-based instantiation of HCBC1 that uses two block-cipher operations per block and has a key consisting of two block-cipher keys.

1.5. *The HCBC2 On-line Cipher and Its Security*

We provide a second, related construction which is proven to meet a stronger notion of security, namely it is a PRP under chosen-ciphertext attack. This is the analog, for on-line-PRPs, of the notion of a PRP secure against chosen-ciphertext attacks. (The latter was called a strong PRP in [23] and a super-PRP in [19].) The adversary has an oracle not just for the challenge permutation, but also for its inverse. The assumption we make in the security result is that the underlying block cipher is a PRP under chosen-ciphertext attack. Additionally, we need a computationally AXU family of $2n$ bits to n bits. Instantiating the latter with a CBC MAC, we obtain a purely block-cipher-based instantiation of HCBC2 that uses three block-cipher operations per block and has a key consisting of two block-cipher keys. We note that this construction and result were not included in the preliminary version of this paper appearing in the Crypto 2001 conference [4].

1.6. *Usage and Application of On-line Ciphers*

There are settings in which the input plaintext is being streamed to a device that has limited memory for buffering and wants to produce output at the same rate at which it is getting input. The on-line property becomes desirable in these settings.

The most direct usage of an on-line cipher will be in settings where, additionally, there is a constraint requiring the length of the ciphertext to equal the length of the plaintext. (Otherwise, one can use a standard mode of encryption like CBC, since it has the on-line property. But it is length expanding in the sense that the length of the ciphertext exceeds that of the plaintext, due to the changing initial vector.) This type of constraint occurs when one is dealing with fixed packet formats, legacy code, or disk-sector encryption.

However, an on-line cipher is more generally useful, via the "encode-then-encipher" paradigm discussed in [8]. This paradigm was presented for ciphers that are PRPs, and says that enciphering provides semantic security if the message space has enough entropy, and provides integrity if the message space contains enough redundancy. Entropy and redundancy might be present in the data, as often happens when enciphering structured data like packets, which have fixed formats and often contain counters. Or, entropy

and redundancy can be explicitly added, for example by inserting a random value and a constant string in the message. (This will of course increase the size of the plaintext, so is only possible when data expansion is permitted.)

Claims similar to those made in [8] remain true even if the cipher is an on-line-PRP rather than a PRP, but more restrictions on the message space are required. Specifically, we require not just that entropy be present in a message but that the same high entropy block be present both at the beginning and at the end of the message. While it is less likely that data already have such a structure, one can prepend and append a random number to the message, getting the same properties but at the cost of small data expansion. In Sect. 8 we also discuss two other possibilities of message encodings for use with on-line ciphers.

In summary, an on-line cipher is a versatile tool that, when appropriately used, is able to provide some security under severe practical constraints, and yet provide security as high as that provided by standard primitives when practical constraints are less severe, motivating its isolation and study as a primitive in its own right.

1.7. *Related Work*

The problem addressed by our HCBC constructions is that of building a general cipher from a block cipher. Naor and Reingold [23] consider this problem for the case where the general cipher is to be a PRP or strong PRP, while we want the general cipher to be an on-line-PRP or strong-on-line-PRP. The constructions of [23, Sect. 7] are not on-line; indeed, they cannot be, since they achieve the stronger security notion of a PRP. Our construction, however, follows that of [23] in using hash functions in combination with block ciphers.

Similarly, the CMC [14] and EME [15] enciphering modes are PRPs and not on-line. These constructs all incur latency that grows with the length of the message: the first bit of the output ciphertext is not produced until the entire message has been processed. In contrast, on-line ciphers like HCBC1 and HCBC2 have no latency: an output block is produced as soon as the corresponding input block is processed.

A problem that has received a lot of attention is to take a PRP and produce another having twice the input block length of the original [19,23]. We are, however, interested in allowing inputs of varying and very large size, not merely twice the block size.

Following the appearance of the preliminary version of our work [4], there has been further research on on-line ciphers and encryption schemes [2,3,10,12]. Our HCBC2 cipher was used by [1] to achieve efficiently searchable symmetric encryption. Our work has been simplified and generalized by Rogaway and Zhang [25] who show how to build on-line ciphers from tweakable block ciphers [18].

1.8. *Versions of This Paper*

A preliminary version of this paper appeared in Crypto 2001 [4]. The current version includes a new construction, namely HCBC2, which did not appear in the preliminary version. Section 8 has also been updated to include an on-line cipher-based authenticated encryption scheme for variable input-length data. (The scheme in the preliminary version was only secure for fixed-length data.) The proofs in the current version of our paper, as opposed to the ones from (the full version of) our Crypto 2001 paper, use code-based game playing [9]. This simplifies the proofs and also bypasses various conditional

probability claims made in the prior analyses. (Some of these claims were pointed out by out to be false by Nandi [22], who also provides his own proofs of security for HCBC1 and HCBC2.)

2. Definitions

Notation A *string* is a member of $\{0,1\}^*$. If x is a string, then $|x|$ denotes its length. The empty string is denoted ε. For integer $n, d \geq 1$, let $D_n = \{0,1\}^n$ and let $D_{d,n}$ be the set of all strings in D_n^* whose length is at most dn bits. We adopt the convention that $D_n^0 = \{\varepsilon\}$. If $P \in D_n^*$, then we let $\|P\|_n$ be the number of blocks in P, namely the value i such that $P \in D_n^i$. If $P \in D_n^*$, then $P[i]$ denotes its ith block, meaning $P = P[1] \cdots P[\|P\|_n]$ where $P[i] \in D_n$ for all $i = 1, \ldots, \|P\|_n$. We let $P[1..i]$ denote $P[1] \cdots P[i]$ for any $0 \leq i \leq \|P\|_n$. If $x, y \in D_n^*$ are strings, then we define the *longest common n-prefix* of x, y, denoted by $\mathsf{LCP}_n(x, y)$, as the largest integer $i \geq 0$ such that there exists a string $z \in D_n^i$ which is a prefix of both x and y. For $i \geq 1$ and $M_1, \ldots, M_i \in D_n^*$, we also define

$\mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
  $(s, p) \leftarrow (i, 0)$
  For $j = 1, \ldots, i-1$ do
    $l \leftarrow \mathsf{LCP}_n(M_i, M_j)$
    If $l > p$ then $p \leftarrow l$; $s \leftarrow j$
  Return (s, p).

That is, $\mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$ returns (s, p) such that the following is true. If $\mathsf{LCP}_n(M_i, M_j) = 0$ for all $1 \leq j \leq i-1$ then $s = i$ and $p = 0$. Else $p = \mathsf{LCP}_n(M_i, M_s)$ and $p > \mathsf{LCP}_n(M_i, M_j)$ for all $1 \leq j < s$. A map $f\colon D \to R$ is a *permutation* if $D = R$ and f is a bijection (i.e. one-to-one and onto). A map $f\colon D \to R$ is *length-preserving* if $|f(x)| = |x|$ for all $x \in D$. We will typically consider functions whose inputs and outputs are in $D_{d,n}$, so that both are viewed as sequences of blocks where each block is n bits long. We let $f^{(i)}$ denote the function which on input M returns the ith block of $f(M)$, or ε if $|f(M)| < ni$. If i, j are integers, then we let $[i..j]$ denote $\{i, \ldots, j\}$. For any $i, j \geq 1$, we let

$$L(i, j) = \big\{(i', j') : 1 \leq i' \leq i \text{ and } 1 \leq j' \text{ and } (i' < i \text{ or } j' < j)\big\}.$$

This set contains all pairs (i', j') that come before (i, j) in the lexicographic order.

Function Families and Ciphers A *family of functions* is a map $F\colon \mathrm{Keys}(F) \times \mathrm{Dom}(F) \to \mathrm{Rng}(F)$ where $\mathrm{Keys}(F)$ is the *key space* of F; $\mathrm{Dom}(F)$ is the *domain* of F; and $\mathrm{Rng}(F)$ is the *range* of F. If $\mathrm{Keys}(F) = \{0,1\}^k$, then we refer to k as the key-length. The two-input function F takes a key $K \in \mathrm{Keys}(F)$ and an input $x \in \mathrm{Dom}(F)$ to return a point $F(K, x) \in \mathrm{Rng}(F)$. For each key $K \in \mathrm{Keys}(F)$, we define the map $F_K\colon \mathrm{Dom}(F) \to \mathrm{Rng}(F)$ by $F_K(\cdot) = F(K, \cdot)$ for all $x \in \mathrm{Dom}(F)$. Thus, F specifies a collection of maps from $\mathrm{Dom}(F)$ to $\mathrm{Rng}(F)$, each map being associated with a key. (That is why F is called a family of functions.) We refer to $F(K, \cdot)$ as an *instance* of F. The operation of choosing a key at random from the key space is denoted

$K \xleftarrow{\$} \mathrm{Keys}(F)$. We write $f \xleftarrow{\$} F$ for the operation $K \xleftarrow{\$} \mathrm{Keys}(F)$; $f \leftarrow F(K, \cdot)$. That is, $f \xleftarrow{\$} F$ denotes the operation of selecting at random a function from the family F. When f is so selected it is called a *random instance* of F. Let Rand_n be the family of all functions mapping D_n to D_n so that $f \xleftarrow{\$} \mathsf{Rand}_n$ denotes the operation of selecting at random a function from D_n to D_n. Similarly, let Perm_n be the family of all permutations mapping D_n to D_n so that $\pi \xleftarrow{\$} \mathsf{Perm}_n$ denotes the operation of selecting at random a permutation on D_n. We say that F is a *cipher* if $\mathrm{Dom}(F) = \mathrm{Rng}(F)$ and each instance $F(K, \cdot)$ of F is a length-preserving permutation. A *block cipher* is a cipher whose domain and range equal D_n for some integer n called the *block size*. (For example, the AES has block size 128.) If F is a cipher, then F^{-1} is the *inverse cipher*, defined by $F^{-1}(K, x) = F_K(\cdot)^{-1}(x)$ for all $K \in \mathrm{Keys}(F)$ and $x \in \mathrm{Dom}(F)$.

Pseudorandomness of Ciphers A "secure" cipher is one that approximates a family of random permutations; the "better" the approximation, the more secure the cipher. This is formalized following [13,19]. Let $F\colon \mathrm{Keys}(F) \times D_n \to D_n$ be a family of functions with domain and range D_n. Let A_1 be an adversary (algorithm) that has access to one oracle and outputs a bit. Let

$$\mathbf{Adv}^{\text{prp-cpa}}_F(A_1) = \Pr\left[\, g \xleftarrow{\$} F \;:\; A_1^g = 1 \,\right] - \Pr\left[\, g \xleftarrow{\$} \mathsf{Perm}_n \;:\; A_1^g = 1 \,\right].$$

Let A_2 be an adversary that has access to two oracles and outputs a bit. If $F\colon \mathrm{Keys}(F) \times D_n \to D_n$ is a cipher, then we also let

$$\mathbf{Adv}^{\text{prp-cca}}_F(A_2) = \Pr\left[\, g \xleftarrow{\$} F \;:\; A_2^{g, g^{-1}} = 1 \,\right] - \Pr\left[\, g \xleftarrow{\$} \mathsf{Perm}_n \;:\; A_2^{g, g^{-1}} = 1 \,\right].$$

These capture the *advantage* of the adversary in question in the task of distinguishing a random instance of F from a random permutation on D. In the first case, the adversary gets to query the challenge instance. In the second, it also gets to query the inverse of the challenge instance. We note that in measuring time complexity of an adversary, the time to reply to oracle calls by computation of $F(K, \cdot)$ and $F(K, \cdot)^{-1}$ is included.

Games Our proofs will use code-based game playing [9]. We recall some background here. A game—look at Fig. 6 for an example—has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A as follows. First, **Initialize** executes and its outputs are the inputs to A. Then, the latter executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted G^A, is called the output of the game, and we let "$\mathrm{G}^A \Rightarrow y$" denote the event that this game output takes value y. The boolean flag bad is assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true. For example, games $\mathrm{G}_1, \mathrm{G}_2$ of Fig. 7 are identical until bad. The following is the Fundamental Lemma of game playing of [9].

Lemma 2.1 [9]. *Let* $\mathrm{G}_i, \mathrm{G}_j$ *be identical until* bad *games, and A an adversary. Then*

$$\Pr\left[\, \mathrm{G}_i^A \Rightarrow 1 \,\right] - \Pr\left[\, \mathrm{G}_j^A \Rightarrow 1 \,\right] \le \Pr\left[\, \mathrm{G}_j \textit{ sets } \mathsf{bad} \,\right].$$

Lemma 2.1 can also be derived as a corollary of [20, Theorem 1]. We will also use the following:

Lemma 2.2 [9]. *If* $\mathrm{G}_i, \mathrm{G}_j$ *are identical until* bad *then for any* A

$$\Pr\left[\mathrm{G}_i^A \textit{ sets } \mathsf{bad}\right] = \Pr\left[\mathrm{G}_j^A \textit{ sets } \mathsf{bad}\right].$$

We often have a figure describe multiple games by indicating next to each procedure the games to which the procedure belongs. See for example Fig. 2. This compacts the representation of a game sequence where successive games tend to have a lot of code in common.

3. On-line Ciphers and Their Basic Properties

We say that a function $f\colon D_{d,n} \to D_{d,n}$ is n-on-line if the ith block of the output is determined completely by the first i blocks of the input. A more formal definition follows. We remind the reader that $f^{(i)}(M)$ returns the ith block of $f(M)$, or ε if $|f(M)| < ni$.

Definition 3.1. Let $n, d \geq 1$ be integers, and let $f\colon D_{d,n} \to D_{d,n}$ be a length-preserving function. We say that f is *n-on-line* if there exists a function $X\colon D_{d,n} \to D_n$ such that for every $M \in D_{d,n}$ and every $i \in \{1, \ldots, \|M\|_n\}$ it is the case that

$$f^{(i)}(M) \;=\; X\big(M[1]\cdots M[i]\big). \tag{1}$$

A cipher F having domain and range a subset of $D_{d,n}$ is said to be n-on-line if for every $K \in \mathrm{Keys}(F)$ the function $F(K, \cdot)$ is n-on-line.

Definition 3.2. Let f be an n-on-line function. Let $i \geq 1$. Fix $M[1], \ldots, M[i-1] \in D_n$. Define the function $\Pi^f_{M[1]\cdots M[i-1]}\colon D_n \to D_n$ by

$$\Pi^f_{M[1]\cdots M[i-1]}(x) = f^{(i)}\big(M[1]\cdots M[i-1]x\big)$$

for all $x \in D_n$.

Proposition 3.3. *If* f *is an* n*-on-line permutation,* $i \geq 1$ *and* $M[1], \ldots, M[i-1] \in D_n$*, then the map* $\Pi^f_{M[1]\cdots M[i-1]}$ *is a permutation on* D_n*.*

Proof of Proposition 3.3. Let $M' = M[1]\cdots M[i-1]$, and let x, y be distinct n-bit strings. We claim that

$$\Pi^f_{M'}(x) \neq \Pi^f_{M'}(y). \tag{2}$$

Since $\Pi^f_{M'}$ maps D_n to D_n, this implies that it is a permutation. We now proceed to establish (2).

Since f is n-on-line, we may fix a function $X\colon D_{d,n} \to D_n$ meeting the conditions of Definition 3.1. Let $M_x = M'\|x$ and $M_y = M'\|y$. Applying f to M_x, we get

$$\begin{aligned} f(M_x) &= f^{(1)}(M_x)\big\|f^{(2)}(M_x)\big\|\cdots\big\|f^{(i-1)}(M_x)\big\|f^{(i)}(M_x) \\ &= X\big(M[1]\big)\big\|X\big(M[1]M[2]\big)\big\|\cdots\big\|X\big(M[1]\cdots M[i-1]\big)\big\|X(M_x). \end{aligned}$$

Similarly,

$$f(M_y) = X\big(M[1]\big)\big\|X\big(M[1]M[2]\big)\big\|\cdots\big\|X\big(M[1]\cdots M[i-1]\big)\big\|X(M_y).$$

By assumption $x \neq y$, which implies $M_x \neq M_y$. But f is a permutation, so it must be that $f(M_x) \neq f(M_y)$. However, from the above we see that, for every $j = 1, \ldots, i-1$, the jth output block of $f(M_x)$ and the jth output block of $f(M_y)$ are equal. So it must be that the ith blocks of the outputs are unequal, meaning $X(M_x) \neq X(M_y)$. Finally, we observe that

$$\begin{aligned} X(M_x) &= f^{(i)}(M'x) = \Pi^f_{M'}(x), \\ X(M_y) &= f^{(i)}(M'y) = \Pi^f_{M'}(y), \end{aligned}$$

so (2) is established. □

Pseudorandomness of On-line Ciphers Let $\mathsf{OPerm}_{d,n}$ denote the family of all n-on-line, length-preserving permutations on $D_{d,n}$. A "secure" on-line cipher (namely, an on-line-PRP or OPRP) is one that closely approximates $\mathsf{OPerm}_{d,n}$; the "better" the approximation, the more "secure" the on-line cipher. This formalization is analogous to the previously presented formalization of the pseudorandomness of ciphers. Let $F\colon \mathrm{Keys}(F) \times D_{d,n} \to D_{d,n}$ be a family of functions with domain and range $D_{d,n}$. Let A_1 be an adversary with one oracle and A_2 an adversary with two oracles. Let

$$\mathbf{Adv}^{\text{oprp-cpa}}_F(A_1) = \Pr\big[\, g \xleftarrow{\$} F \;:\; A_1^g = 1 \,\big] - \Pr\big[\, g \xleftarrow{\$} \mathsf{OPerm}_{d,n} \;:\; A_1^g = 1 \,\big].$$

If F is a cipher, then we also let

$$\mathbf{Adv}^{\text{oprp-cca}}_F(A_2) = \Pr\big[g \xleftarrow{\$} F \;:\; A_2^{g,g^{-1}} = 1\big] - \Pr\big[g \xleftarrow{\$} \mathsf{OPerm}_{d,n} \;:\; A_2^{g,g^{-1}} = 1\big].$$

These capture the *advantage* of the adversary in question in the task of distinguishing a random instance of F from a random, length-preserving, n-on-line permutation on $D_{d,n}$. In the first case, the adversary gets to query the challenge instance. In the second, it also gets to query the inverse of the challenge instance.

Tree-Based Characterization We present a tree-based characterization of n-on-line ciphers that is useful to gain intuition and to analyze constructs. A 2^n-ary tree of functions is a 2^n-ary tree T each node of which is labeled by a function mapping D_n to D_n. We label each edge in the tree in a natural way via a string in D_n. Then, each node in the tree is described by a sequence of edge labels defining the path from the root to the node in question. The function labeling node x in the tree, where x is a string of length ni

for some $0 \le i \le d$, is then denoted T_x. A tree defines a function T from $D_{d,n}$ to $D_{d,n}$ as described below. If the nodes in the tree are labeled with permutations, then the tree also defines an inverse function T^{-1}.

$T(M[1]\cdots M[l])$	$T^{-1}(C[1]\cdots C[l])$
$x \leftarrow \varepsilon$	$x \leftarrow \varepsilon$
For $i = 1,\ldots,l$ do	For $i = 1,\ldots,l$ do
$\quad C[i] \leftarrow T_x(M[i])$	$\quad M[i] \leftarrow T_x^{-1}(C[i])$
$\quad x \leftarrow x\|C[i]$	$\quad x \leftarrow x\|C[i]$
Return $C[1]\cdots C[l]$	Return $M[1]\cdots M[l]$.

Here, $1 \le l \le d$. Let $G : \mathrm{Keys}(G) \times D_n \to D_n$ be a function family. (We are most interested in the case where G is Perm_n or Rand_n.) We let $\mathsf{Tree}(n, G, d)$ denote the set of all 2^n-ary trees of functions in which each function is an instance of G and the depth of the tree is d. This set is viewed as equipped with a distribution under which each node of the tree is assigned a random instance of G, and the assignments to the different nodes are independent. We claim that a tree-based construction defined above is a valid characterization of on-line ciphers, as stated in the following proposition.

Proposition 3.4. *There is a bijection between* $\mathsf{Tree}(n, \mathsf{Perm}_n, d)$ *and* $\mathsf{OPerm}_{d,n}$.

Proof of Proposition 3.4. We specify a map Z: $\mathsf{OPerm}_{n,d} \to \mathsf{Tree}(n, \mathsf{Perm}_n, d)$ and then argue that it is a bijection. Given $f \in \mathsf{OPerm}_{n,d}$, the map Z returns the tree $T_f = Z(f) \in \mathsf{Tree}(n, \mathsf{Perm}_n, d)$ defined as follows: for any $l = 0,\ldots,d-1$ and any $M[1]\cdots M[l] \in D_{d,n}$, $C = f(M)$, node $C[1]\cdots C[l]$ of tree T_f is labeled by the permutation $\Pi^f_{M[1]\cdots M[l]}$. Equivalently, T_f can be defined as the function which, for any $l = 1,\ldots,d$ and any input $M[1]\cdots M[l] \in D_{d,n}$, works as follows:

$T_f(M[1]\cdots M[l])$
  For $i = 1,\ldots,l$ do
    $C[i] \leftarrow \Pi^f_{M[1]\cdots M[i-1]}(M[i])$
  Return $C[1]\cdots C[l]$.

Proposition 3.3 implies that the functions labeling the nodes of the tree are indeed in Perm_n, therefore $T_f \in \mathsf{Tree}(n, \mathsf{Perm}_n, d)$. Now we want to show that Z is a bijection. We need to show that it is injective (i.e. one-to-one) and surjective (i.e. onto). We prove these in turn.

To show that Z is injective, let f and g be n-on-line permutations such that $Z(f) = Z(g)$. We show that $f = g$. As per the above and the assumption that $T_f = T_g$, the function labeling a node $C[1]\cdots C[l]$ in T_f is $\Pi^f_{M[1]\cdots M[l]}$ in T_f and $\Pi^g_{M[1]\cdots M[l]}$ in T_g. The assumption $T_f = T_g$ also implies that $\Pi^f_{M[1]\cdots M[l]} = \Pi^g_{M[1]\ldots M[l]}$ for all $l = 0,\ldots,d-1$ and all $M[1]\cdots M[l] \in D_{d,n}$. By Definition 3.2 we have

$$f^{(l+1)}\big(M[1]\cdots M[l]x\big) = g^{(l+1)}\big(M[1]\cdots M[l]x\big)$$

for all $l = 0,\ldots,d-1$ and all $M[1]\cdots M[l] \in D_{d,n}$. Therefore, $f = g$.

Next we show that Z is surjective. Let $T \in \mathsf{Tree}(n, \mathsf{Perm}_n, d)$. We need to show that there exists an $f \in \mathsf{OPerm}_{n,d}$ such that $T_f = T$. We let f be the function defined by T.

From the definition, it is clear that it is n-on-line, and since the inverse function T^{-1} is also defined, is a permutation. □

Inversion It turns out that the inverse of an on-line permutation is itself on-line:

Proposition 3.5. *Let $f\colon D_{d,n} \to D_{d,n}$ be an n-on-line permutation, and let $g = f^{-1}$. Then g is an n-on-line permutation.*

Proof of Proposition 3.5. Since f is by assumption a length-preserving permutation, so is g. So as per Definition 3.1, it suffices to show that there exists a function Y so that, for every $C \in D_{d,n}$ and for every $i \le \|C\|_n$, we have

$$g^{(i)}(C) = Y\big(C[1]\cdots C[i]\big). \tag{3}$$

We define $Y : D_{d,n} \to D_n$ as follows for any $i = 1, \ldots, d$ and any input $C[1]\cdots C[i] \in D_{d,n}$:

$Y(C[1]\cdots C[i])$
  $M[0] \leftarrow \varepsilon$
  For $j = 1, \ldots, i$ do
    $M[j] \leftarrow (\Pi^f_{M[0]\cdots M[j-1]})^{-1}(C[j])$
  Return $M[i]$.

Here we used Proposition 3.3 and the assumption that f is n-on-line to guarantee that the inverse of $\Pi^f_{M[0]\cdots M[j-1]}$ is well-defined. Now, suppose $C \in D_{d,n}$ and $1 \le i \le \|C\|_n$. We prove that (3) holds. Letting $M = g(C)$ we have

$$\begin{aligned}
Y(C) &= Y\big(f\big(g(C)\big)\big)\\
&= Y\big(f(M)\big)\\
&= Y\big(f^{(1)}(M)\,\|\,f^{(2)}(M)\,\|\cdots\|\,f^{(i)}(M)\,\|\big)\\
&= Y\big(\Pi^f_{\varepsilon}\big(M[1]\big)\,\|\,\Pi^f_{M[1]}\big(M[2]\big)\,\|\cdots\|\,\Pi^f_{M[1]\cdots M[i-1]}\big(M[i]\big)\,\|\big)\\
&= M[i]\\
&= g^{(i)}(C),
\end{aligned}$$

as desired. The first line above is true because $g = f^{-1}$. The second line is true because $M = g(C)$. The third line is by definition of $f^{(j)}$ for $j = 1, \ldots, i$. The fourth line is by Proposition 3.3. The fifth line follows by applying the definition of Y. The sixth line is because $M = g(C)$. □

We note that the proof does not tell us anything about the computational complexity of function f^{-1}, meaning it could be the case that f is efficiently computable, but the f^{-1} given by Proposition 3.5 is not. However, whenever we design a cipher F, we will make sure that both $F(K,\cdot)$ and $F^{-1}(K,\cdot)$ are efficiently computable given K, and will explicitly specify F^{-1} in order to make this clear.

proc Initialize
$i \leftarrow 0$

proc Encipher(M)
$i \leftarrow i + 1;\ M_i \leftarrow M;\ l_i \leftarrow \|M_i\|_n$
$(s, p) \leftarrow \mathsf{LCP}^*_n(M_i, M_1, \ldots, M_{i-1})$
For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
For $j = p + 1, \ldots, l_i$ do
  $\Pi_{M_i[1..j-1]}(M_i[j]) \xleftarrow{\$} \overline{\mathsf{Rng}}(\Pi_{M_i[1\ldots j-1]})$
  $C_i[j] \leftarrow \Pi_{M_i[1\ldots j-1]}(M_i[j])$
Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C)
$i \leftarrow i + 1;\ C_i \leftarrow C;\ l_i \leftarrow \|C_i\|_n$
$(s, p) \leftarrow \mathsf{LCP}^*_n(C_i, C_1, \ldots, C_{i-1})$
For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
For $j = p + 1, \ldots, l_i$ do
  $\Pi^{-1}_{M_i[1\ldots j-1]}(C_i[j]) \xleftarrow{\$} \overline{\mathsf{Dom}}(\Pi_{M_i[1\ldots j-1]})$
  $M_i[j] \leftarrow \Pi^{-1}_{M_i[1\ldots j-1]}(C_i[j])$
Return $M_i \leftarrow M_i[1 \ldots l_i]$

proc Finalize(d)
Return d

proc Initialize
$K \xleftarrow{\$} \mathrm{Keys}(F)$

proc Encipher(M)
Return $F_K(M)$

proc Decipher(C)
Return $F_K^{-1}(C)$

proc Finalize(d)
Return d

Fig. 1. Games for defining security of on-line ciphers. *On the left* are the procedures defining games $\mathrm{OPRPCPA}_{\mathsf{Perm}}$ and $\mathrm{OPRPCCA}_{\mathsf{Perm}}$, where the **Decipher** procedure is included only in the latter. *On the right* are the procedures defining games $\mathrm{OPRPCPA}_F$ and $\mathrm{OPRPCCA}_F$, where the **Decipher** procedure is included only in the latter.

Game-Based Formulations For our proofs, it is helpful to re-cast the advantages via the games shown in Fig. 1. Games $\mathrm{OPRPCPA}_{\mathsf{Perm}}$ and $\mathrm{OPRPCPA}_F$ provide the adversary with an **Encipher** oracle while games $\mathrm{OPRPCCA}_{\mathsf{Perm}}$ and $\mathrm{OPRPCCA}_F$ additionally give it a **Decipher** oracle, the latter in the case where F is a cipher. Games $\mathrm{OPRPCPA}_{\mathsf{Perm}}$ and $\mathrm{OPRPCCA}_{\mathsf{Perm}}$ lazily pick a permutation $\Pi_{M[1..j]}$ for the tree node with label $M[1..j]$. We have

$$\mathbf{Adv}^{\text{oprp-cpa}}_F(A) = \Pr\big[\,\mathrm{OPRPCPA}^A_F \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{OPRPCPA}^A_{\mathsf{Perm}} \Rightarrow 1\,\big],$$
$$\mathbf{Adv}^{\text{oprp-cca}}_F(A) = \Pr\big[\,\mathrm{OPRPCCA}^A_F \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{OPRPCCA}^A_{\mathsf{Perm}} \Rightarrow 1\,\big].$$

In the games, the domain $\mathsf{Dom}(\Pi_{M[1..j]})$ and range $\mathsf{Rng}(\Pi_{M[1..j]})$ of $\Pi_{M[1..j]}$ start out empty, and an assignment $\Pi_{M[1..j]}(x) \leftarrow y$ adds x to the first set and y to the second. We are denoting $D_n \setminus \mathsf{Dom}(\Pi_{M[1..j]})$ by $\overline{\mathsf{Dom}}(\Pi_{M[1..j]})$ and $D_n \setminus \mathsf{Rng}(\Pi_{M[1..j]})$ by $\overline{\mathsf{Rng}}(\Pi_{M[1..j]})$.

Switching Lemma It will be useful in our proofs to consider games which reply to queries with random block values unless constrained otherwise by the prefix condition. That is, consider games $\text{OPRFCPA}_{\text{Perm}}$ and $\text{OPRFCCA}_{\text{Perm}}$ of Fig. 1. In analogy with the PRP/PRF Switching Lemma of [9], we have the following lemma.

Lemma 3.6 (OPRP/OPRF Switching Lemma). *Let A be an adversary making oracle queries totalling at most μ blocks. Then,*

$$\left|\Pr\left[\,\text{OPRPCPA}^{A}_{\text{Perm}} \Rightarrow 1\,\right] - \Pr\left[\,\text{OPRFCPA}^{A}_{\text{Rand}} \Rightarrow 1\,\right]\right| \le \frac{\mu(\mu-1)}{2^{n+1}} \quad \textit{and} \tag{4}$$

$$\left|\Pr\left[\,\text{OPRPCCA}^{A}_{\text{Perm}} \Rightarrow 1\,\right] - \Pr\left[\,\text{OPRFCCA}^{A}_{\text{Rand}} \Rightarrow 1\,\right]\right| \le \frac{\mu(\mu-1)}{2^{n+1}}. \tag{5}$$

Note in the second case we mean the total number of blocks across queries to both oracles together is at most μ.

Proof of Lemma 3.6. We have

$$\begin{aligned}
&\left|\Pr\left[\,\text{OPRPCPA}^{A}_{\text{Perm}} \Rightarrow 1\,\right] - \Pr\left[\,\text{OPRFCPA}^{A}_{\text{Rand}} \Rightarrow 1\,\right]\right| \\
&\quad = \left|\Pr\left[\,\text{G}^{A}_{\text{cpa}} \Rightarrow 1\,\right] - \Pr\left[\,\text{OPRFCPA}^{A}_{\text{Rand}} \Rightarrow 1\,\right]\right| \\
&\quad \le \Pr\left[\,\text{OPRFCPA}^{A}_{\text{Rand}} \text{ sets } \mathsf{bad}\,\right],
\end{aligned}$$

the last line by Lemmas 2.1 and 2.2. However, the last probability is certainly at most $\mu(\mu-1)/2^{n+1}$. The analysis for the CCA case is similar and is omitted. □

We conjecture that the bounds in (4) and (5) can be improved to $q(q-1)/2^{n+1}$ where q is an upper bound on the number of oracle queries made by A. (In the CCA case, we mean the total number of queries across both oracles.) That is, the bound does not depend on the number of blocks but only on the number of queries. We do not attempt to prove this since it does not (much) affect our results. We leave settling this as an open question.

4. Analysis of Some Candidate Ciphers

We consider several candidates for on-line ciphers. First, we consider one based on the basic CBC mode. Then, we consider the Accumulated Block Chaining (ABC) proposed by Knudsen in [16], which is a generalization of the Infinite Garble Extension mode proposed by Campbell [11]. In this section, we let $E\colon \{0,1\}^{ek} \times D_n \to D_n$ be a given block cipher with key size ek and block size n.

4.1. *CBC as an On-line Cipher*

In CBC encryption based on E, one usually uses a new, random IV for every message. This does not yield a cipher, let alone an on-line one. To get an on-line cipher, we fix the IV. We can, however, make it secret; this can only increase security. In more detail, the

proc Initialize Games $\mathrm{OPRFCPA}_{\mathsf{Rand}}$, $\mathrm{OPRFCCA}_{\mathsf{Rand}}$, $\boxed{\mathrm{G}_{\mathsf{cpa}}}$, $\boxed{\mathrm{G}_{\mathsf{cca}}}$

$i \leftarrow 0$

proc Encipher(M) Games $\mathrm{OPRFCPA}_{\mathsf{Rand}}$, $\mathrm{OPRFCCA}_{\mathsf{Rand}}$, $\boxed{\mathrm{G}_{\mathsf{cpa}}}$, $\boxed{\mathrm{G}_{\mathsf{cca}}}$

$i \leftarrow i + 1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$
$(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
For $j = p + 1, \ldots, l_i$ do
  $y \xleftarrow{\$} D_n$
  If $y \in \mathsf{Rng}(\Pi_{M_i[1..j-1]})$ then $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{;\ y \xleftarrow{\$} \overline{\mathsf{Rng}}(\Pi_{M_i[1..j-1]})}$
  $\Pi_{M_i[1..j-1]}(M_i[j]) \leftarrow y$; $C_i[j] \leftarrow \Pi_{M_i[1..j-1]}(M_i[j])$
Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Games $\mathrm{OPRFCCA}_{\mathsf{Rand}}$, $\boxed{\mathrm{G}_{\mathsf{cca}}}$

$i \leftarrow i + 1$; $C_i \leftarrow C$; $l_i \leftarrow \|C_i\|_n$
$(s, p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
For $j = p + 1, \ldots, l_i$ do
  $x \xleftarrow{\$} D_n$
  If $x \in \mathsf{Dom}(\Pi_{M_i[1..j-1]})$ then $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{;\ x \xleftarrow{\$} \overline{\mathsf{Dom}}(\Pi_{M_i[1..j-1]})}$
  $\Pi^{-1}_{M_i[1..j-1]}(C_i[j]) \leftarrow x$; $M_i[j] \leftarrow \Pi^{-1}_{M_i[1..j-1]}(C_i[j])$
Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Games $\mathrm{OPRFCPA}_{\mathsf{Rand}}$, $\mathrm{OPRFCCA}_{\mathsf{Rand}}$, $\boxed{\mathrm{G}_{\mathsf{cpa}}}$, $\boxed{\mathrm{G}_{\mathsf{cca}}}$

Return d

Fig. 2. Games for proof of Lemma 3.6. Games $\mathrm{G}_{\mathsf{cpa}}$ and $\mathrm{G}_{\mathsf{cca}}$ include the boxed statements while the other games do not.

CBC cipher associated to E, denoted OCBC, has key space $\{0,1\}^{ek+n}$. For $M, C \in D_{d,n}$, $eK \in \{0,1\}^{ek}$ and $C[0] \in D_n$, we define

$\mathsf{OCBC}(eK \| C[0], M)$	$\mathsf{OCBC}^{-1}(eK \| C[0], C)$
For $i = 1, \ldots, l$ do	For $i = 1, \ldots, l$ do
$\quad C[i] \leftarrow E(eK, M[i] \oplus C[i-1])$	$\quad M[i] \leftarrow E^{-1}(eK, C[i]) \oplus C[i-1]$
Return $C[1..l]$	Return $M[1..l]$.

Here, $C[0]$ is the IV. The key is the pair $eK \| C[0]$, consisting of a key eK for the block cipher, and the IV. It is easy to check that the above cipher is on-line. For clarity, we have also shown the inverse cipher. We now present the attack. The adversary A shown in Fig. 3 gets an oracle g where g is either an instance of OCBC or an instance of $\mathsf{OPerm}_{d,n}$. The idea of the attack is to gather some input-output pairs for the cipher. Then we use these values to construct a new sequence of input blocks so that one of the

Adversary A^g
Let $M[2]$ be any n-bit string
Let $M_1 = 0^n M[2]$ and let $M_2 = 1^n M[2]$
Let $C_1[1]C_1[2] \leftarrow g(M_1)$ and let $C_2[1]C_2[2] \leftarrow g(M_2)$
Let $M_3[2] = M[2] \oplus C_1[1] \oplus C_2[1]$ and let $M_3 = 1^n M_3[2]$
Let $C_3[1]C_3[2] \leftarrow g(M_3)$
If $C_3[2] = C_1[2]$ then return 1 else return 0

Fig. 3. Attack on the CBC-based on-line cipher.

input blocks to E collides with one of the previous input blocks to E. This enables us to predict an output block of the cipher. If our prediction is correct, then we know that the oracle is an instance of OCBC with overwhelming probability. Specifically, we claim that

$$\mathbf{Adv}^{\text{oprp-cpa}}_{\mathsf{OCBC}}(A) \geq 1 - 2^{-n}. \tag{6}$$

We now justify (6). We claim that

$$\Pr\left[g \xleftarrow{\$} \mathsf{OCBC} \ : \ A^g = 1 \right] = 1 \quad \text{and} \quad \Pr\left[g \xleftarrow{\$} \mathsf{OPerm}_{n,d} \ : \ A^g = 1 \right] \leq 2 \cdot 2^{-n},$$

from which (6) from Sect. 4 follows. We justify these two claims as follows. First, suppose g is an instance of OCBC. Since the first block of M_3 is $M_2[1]$, we have

$$\begin{aligned} C_3[2] &= E\big(eK,\ C_2[1] \oplus M_3[2]\big) \\ &= E\big(eK,\ C_2[1] \oplus M[2] \oplus C_1[1] \oplus C_2[1]\big) \\ &= E\big(eK,\ M[2] \oplus C_1[1]\big) \\ &= C_1[2]. \end{aligned}$$

This means that adversary A will always return 1 when g is an instance of OCBC, and the first equation is true. Now, consider the case where g is a random instance of $\mathsf{OPerm}_{n,d}$. Here, there are two possible ways in which $C_3[2] = C_1[2]$ holds. First, $M_3[2]$ can happen to be the same as $M[2]$. This happens with the probability at most 2^{-n} when g is a random instance of $\mathsf{OPerm}_{n,d}$. Second, if $M_3[2] \neq M[2]$, then it can happen that $C_3[2] = C_1[2]$ with the probability at most 2^{-n} when g is a random instance of $\mathsf{OPerm}_{n,d}$. Therefore, the adversary A^g outputs 1 with the probability at most $2 \cdot 2^{-n}$, and this justifies the second equation.

Since A made only three oracle queries, this shows that the CBC mode with a fixed IV is not a secure on-line cipher.

4.2. *ABC as an On-line Cipher*

Knudsen in [16] proposes the Accumulated Block Chaining (ABC) mode of operation for block ciphers. This is an on-line cipher that is a natural starting point in the problem of finding a secure on-line cipher because it has the property of "infinite error propagation." We formalize and analyze ABC with regard to meeting our security requirements.

Description The mode is parameterized by initial values $P[0], C[0] \in D_n$ and also by a public function $h\colon D_n \to D_n$. (Instantiations for h suggested in [16] include the identity function, the constant function always returning 0^n, and the function which rotates its input by one bit.) We are interested in the security of the mode across various settings and choices of these parameters. (In particular, we want to consider the case where the initial values are public and also the case where they are secret, and see how the choice of h impacts security in either case.) Accordingly, it is convenient to first introduce auxiliary functions EABC and DABC. For $M, C \in D_{d,n}$ and $eK \in \{0,1\}^k$, we define

$\mathrm{EABC}(eK, P[0], C[0], M)$	$\mathrm{DABC}(eK, P[0], C[0], C)$
For $i = 1, \ldots, l$ do	For $i = 1, \ldots, l$ do
$\quad P[i] \leftarrow M[i] \oplus h(P[i-1])$	$\quad P[i] \leftarrow E^{-1}(eK, C[i] \oplus P[i-1])$
$\quad C[i] \leftarrow E(eK, P[i] \oplus C[i-1])$	$\qquad\qquad \oplus C[i-1]$
$\qquad\qquad \oplus P[i-1]$	$\quad M[i] \leftarrow P[i] \oplus h(P[i-1])$
Return $C[1..l]$	Return $M[1..l]$.

We now define two versions of the ABC cipher. The first uses public initial values, while the second uses secret initial values. The *ABC cipher with public initial values* associated to E, denoted PABC, has key space $\{0,1\}^k$ and domain and range $D_{d,n}$. We fix values $P[0], C[0] \in D_n$ which are known to all parties including the adversary. We then define the cipher and the inverse cipher by

$\mathsf{PABC}(eK, M)$	$\mathsf{PABC}^{-1}(eK, C)$
Return $\mathrm{EABC}(eK, P[0], C[0], M)$	Return $\mathrm{DABC}(eK, P[0], C[0], C)$.

The *ABC cipher with secret initial values* associated to E, denoted SABC, has key space $\{0,1\}^{k+2n}$ and domain and range $D_{d,n}$. The key is $eK \| P[0] \| C[0]$. We then define the cipher and the inverse cipher by

$\mathsf{SABC}(eK \| P[0] \| C[0], M)$	$\mathsf{SABC}^{-1}(eK \| P[0] \| C[0], C)$
Return $\mathrm{EABC}(eK, P[0], C[0], M)$	Return $\mathrm{DABC}(eK, P[0], C[0], C)$.

It is easy to check that both the above ciphers are n-on-line.

Security of PABC We show that the ABC cipher with public initial values is not a secure OPRP for *all* choices of the function h. The attack is shown in Fig. 4. The adversary A gets an oracle g where g is either an instance of PABC or an instance of $\mathsf{OPerm}_{d,n}$. The adversary can mount this attack because the function h as well as the value $P[0]$ are public. We claim that

$$\mathbf{Adv}^{\text{oprp-cpa}}_{\mathsf{PABC}}(A) \geq 1 - 2 \cdot 2^{-n}. \tag{7}$$

Since A made only three oracle queries, this means that PABC is not a secure on-line cipher.

We now analyze the attack against PABC, meaning we justify (7). We claim that

$$\Pr\left[\, g \xleftarrow{\$} \mathsf{PABC} \;:\; A^g = 1 \,\right] = 1 \quad \text{and} \quad \Pr\left[\, g \xleftarrow{\$} \mathsf{OPerm}_{n,d} \;:\; A^g = 1 \,\right] \leq 3 \cdot 2^{-n},$$

Adversary A^g
Let $M[2]$ be any n-bit string
Let $M_1 = 0^n M[2]$ and let $M_2 = 1^n M[2]$
Let $C_1[1] \cdots C_1[l] \leftarrow g(M_1)$ and let $C_2[1]C_2[2] \leftarrow g(M_2)$
Let $M_3[2] = M[2] \oplus C_1[1] \oplus C_2[1] \oplus h(0^n \oplus h(P[0])) \oplus h(1^n \oplus h(P[0]))$
Let $M_3 = 1^n M_3[2]$
Let $C_3[1]C_3[2] \leftarrow g(M_3)$
If $C_3[2] = C_1[2] \oplus 1^n$, then return 1 else return 0

Fig. 4. Attack on the ABC-based on-line cipher.

from which (7) follows. We justify these two claims below. First, suppose g is an instance of PABC, namely $g(\cdot) = \mathsf{PABC}(eK, \cdot)$. Since the first block of M_3 is 1^n, we have

$$\begin{aligned}
C_3[2] &= E\big(eK,\ P_3[2] \oplus C_3[1]\big) \oplus P_3[1] \\
&= E\big(eK,\ M_3[2] \oplus h\big(P_3[1]\big) \oplus C_3[1]\big) \oplus P_3[1] \\
&= E\big(eK,\ M_3[2] \oplus h\big(P_2[1]\big) \oplus C_2[1]\big) \oplus P_2[1] \\
&= E\big(eK,\ M_3[2] \oplus h\big(1^n \oplus h\big(P[0]\big)\big) \oplus C_2[1]\big) \oplus P_2[1] \\
&= E\big(eK,\ M[2] \oplus C_1[1] \oplus h\big(0^n \oplus h\big(P[0]\big)\big)\big) \oplus P_2[1] \\
&= E\big(eK,\ M[2] \oplus C_1[1] \oplus h\big(P_1[1]\big)\big) \oplus P_2[1] \\
&= E\big(eK,\ P_1[2] \oplus C_1[1]\big) \oplus P_2[1] \\
&= \big(C_1[2] \oplus P_1[1]\big) \oplus P_2[1] \\
&= C_1[2] \oplus \big(0^n \oplus h\big(P[0]\big)\big) \oplus \big(1^n \oplus h\big(P[0]\big)\big) \\
&= C_1[2] \oplus 1^n.
\end{aligned}$$

This means that adversary A will always return 1 when g is an instance of PABC. Now, consider the case where g is an instance of $\mathsf{OPerm}_{n,d}$. We claim that this event has low probability, namely $2 \cdot 2^{-n}$. The reason is similar to that in Sect. 4.1. In particular, there are two possible ways in which $C_3[2] = C_1[2] \oplus 1^n$ holds. First, it may be the case that $M_3[2] = M[2]$. This means that the attack is invalid since $M_3 = M_2$. Second, if $M_3[2] \neq M[2]$, then it may be the case that $C_3[2] = C_1[2] \oplus 1^n$. Each of these events happens with probability at most 2^{-n} when g is a random instance of $\mathsf{OPerm}_{n,d}$. Upper-bounding the aggregate probability, we obtain $2 \cdot 2^{-n}$ and (7) follows.

Security of SABC We show that the ABC cipher with secret initial values is not a secure OPRP for a class of functions h that includes the ones suggested in [16]. Specifically, let us say that a function $h\colon D_n \to D_n$ is *linear* if $h(x \oplus y) = h(x) \oplus h(y)$ for all $x, y \in D_n$. (Notice that the identity function, the constant function always returning 0^n, and the function which rotates its input by one bit are all linear.) For any linear hash function h, we simply note that the above attack applies. This is because the fourth line of the adversary's code can be replaced by

$$\text{Let } M_3[2] = M[2] \oplus C_1[1] \oplus C_2[1] \oplus h\big(0^n\big) \oplus h\big(1^n\big).$$

The adversary can compute $M_3[2]$ because h is public. The fact that h is linear means that the value $M_3[2]$ is the same as before, so the attack has the same success probability.

Avoiding the Attacks There are several ways one might try to avoid such attacks while keeping intact the basic structure of the ABC mode. One could use secret initial values and a more complex public function h that in particular is non-linear. Another suggestion is to allow the function h to depend on a secret key. A concrete suggestion in this regard is to choose some family of functions $H\colon \{0,1\}^{2n} \times D_n \to D_n$ that is pairwise independent. Then, the key for the ABC cipher is $eK \| hK$ where $eK \in \{0,1\}^k$ and $hK \in \{0,1\}^{2n}$, and the construction replaces $h(\cdot)$ by $H(hK, \cdot)$. We do not attempt to analyze these, since we propose a somewhat simpler construct that we prove to be secure.

5. (Computational) AXU Families

Our constructions utilize a block cipher and an auxiliary family H that meets a computational relaxation of the notion of AXU (Almost XOR Universal) of Krawczyk [17]. To detail this, let us begin by recalling the measure of [17]:

Definition 5.1. Let $m, n, hk \geq 1$ be integers, and let $H\colon \{0,1\}^{hk} \times \{0,1\}^m \to D_n$ be a family of functions. Let

$$\mathbf{Adv}^{\mathrm{axu}}_H = \max_{x_1, x_2, y} \left\{ \Pr[K \xleftarrow{\$} \{0,1\}^{hk} : H(K, x_1) \oplus H(K, x_2) = y] \right\}$$

where the maximum is over all *distinct* $x_1, x_2 \in \{0,1\}^m$ and all $y \in D_n$.

The "advantage function"-based notation we are introducing is novel: previous works used instead the term "ϵ-AXU" family to refer to a family H that, in our notation, has $\mathbf{Adv}^{\mathrm{axu}}_H \leq \epsilon$. We find the advantage function-based notation more convenient, and more consistent with the rest of our security definitions. We now define an adversary-based measure that will underly a computational relaxation of the above:

Definition 5.2. Let $m, n, hk \geq 1$ be integers, and let $H\colon \{0,1\}^{hk} \times \{0,1\}^m \to D_n$ be a family of functions. Let X be an adversary that takes no inputs and outputs a set $S \subseteq \{0,1\}^m \times D_n$. Consider the experiment in which we first run X to get S and then pick K at random from $\{0,1\}^{hk}$. Let $\mathbf{Adv}^{\mathrm{axu}}_H(X)$ denote the probability that there exists $(x_1, y_1), (x_2, y_2) \in S$ such that $H(K, x_1) \oplus y_1 = H(K, x_2) \oplus y_2$ and $x_1 \neq x_2$, where the probability is over the coins of X and the choice of K.

We (informally) say that H is cAXU if $\mathbf{Adv}^{\mathrm{axu}}_H(X)$ is "small" for all X of "practical" resources. (Resources means running time and size of the output set S.) The following shows that any AXU family is cAXU:

Proposition 5.3. *Let $m, n, hk \geq 1$ be integers, and let $H\colon \{0,1\}^{hk} \times \{0,1\}^m \to D_n$ be a family of functions. Let X be an adversary that takes no inputs and returns a set*

$S \subseteq \{0,1\}^m \times D_n$ of size at most s. Then,

$$\mathbf{Adv}^{\mathrm{axu}}_H(X) \le \frac{s(s-1)}{2} \cdot \mathbf{Adv}^{\mathrm{axu}}_H .$$

Proof of Proposition 5.3. Fix a sequence of coins that maximize the success probability of X, and let S be the set it outputs with these coins. For each $(x_1, y_1), (x_2, y_2) \in S$ with $x_1 \neq x_2$, we have

$$\Pr\left[K \xleftarrow{\$} \{0,1\}^{hk} \ : \ H(K, x_1) \oplus y_1 = H(K, x_2) \oplus y_2 \right] \le \mathbf{Adv}^{\mathrm{axu}}_H .$$

The lemma follows from the union bound. □

However, there are cAXU families that are not necessarily AXU. For example, any PRF is cAXU. To detail this, we let

$$\mathbf{Adv}^{\mathrm{prf}}_H(B) = \Pr\left[g \xleftarrow{\$} H \ : \ B^g = 1 \right] - \Pr\left[g \xleftarrow{\$} \mathsf{Rand}_{m,n} \ : \ B^g = 1 \right]$$

where $\mathsf{Rand}_{m,n}$ is the family of all functions mapping $\{0,1\}^m$ to D_n.

Proposition 5.4. *Let $m, n, hk \ge 1$ be integers, and let $H\colon \{0,1\}^{hk} \times \{0,1\}^m \to D_n$ be a family of functions. Let X be an adversary of running time at most t that takes no inputs and outputs a set of size at most s. Then, there is an adversary B such that*

$$\mathbf{Adv}^{\mathrm{axu}}_H(X) \le \mathbf{Adv}^{\mathrm{prf}}_H(B) + \frac{s(s-1)}{2^{n+1}} .$$

Furthermore, B makes at most s oracle queries and has running time at most $t + O((m+n)s \log s)$.

The proof is trivial and is omitted. Now, from the above we have multiple ways to obtain cAXU families suitable for our constructs. First, by Proposition 5.3, any AXU family suffices. Refer to [17,24,26] for constructions and performance comparisons for such families. These constructs are all unconditionally secure. On the other hand, by Proposition 5.4, any PRF suffices. In particular, if $m = n$, a block cipher suffices, and if $m = 2n$, a 2-fold CBC MAC suffices [6]. These constructs are conditionally secure, the condition being that the block cipher is a PRP.

6. The HCBC1 Cipher

In this section, we provide a construction of an on-line cipher that we call HCBC1. We prove it is secure against chosen-plaintext attacks. This construction is similar to the CBC mode of encryption. The difference is that each output block passes through a keyed hash function before getting exclusive-or-ed with the next input block and there is no IV. The key of the hash function is kept secret.

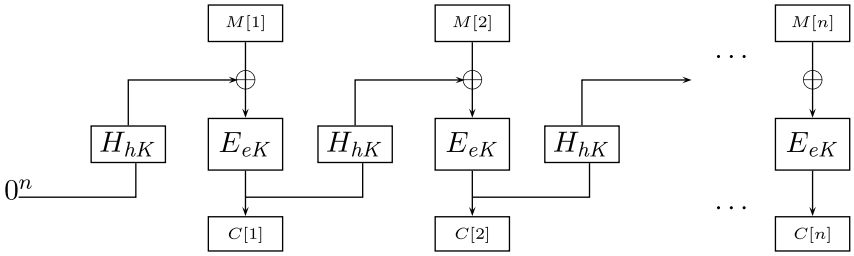

Fig. 5. The HCBC1 cipher.

Construction 6.1. Let $n, d \geq 1$ be integers, and let $E\colon \{0,1\}^{ek} \times D_n \to D_n$ be a block cipher. Let $H\colon \{0,1\}^{hk} \times D_n \to D_n$ be a family of hash functions. We associate to them a cipher $\mathsf{HCBC1}\colon \{0,1\}^{ek+hk} \times D_{d,n} \to D_{d,n}$. A key for it is a pair $eK\|hK$ where eK is a key for E and hK is a key for H. The cipher and its inverse are defined as follows for $M, C \in D_{d,n}$. Figure 5 illustrates the cipher.

$\mathsf{HCBC1}(eK\|hK, M)$
$C[0] \leftarrow 0^n$; $l \leftarrow \|M\|_n$
For $j = 1, \ldots, l$ do
  $P[j] \leftarrow H(hK, C[j-1]) \oplus M[j]$
  $C[j] \leftarrow E(eK, P[j])$
Return $C[1..l]$

$\mathsf{HCBC1}^{-1}(eK\|hK, C)$
$C[0] \leftarrow 0^n$; $l \leftarrow \|C\|_n$
For $j = 1, \ldots, l$ do
  $P[j] \leftarrow E^{-1}(eK, C[j])$
  $M[j] \leftarrow H(hK, C[j-1]) \oplus P[j]$
Return $M[1..l]$.

The following theorem implies that if E is a PRP secure against chosen-plaintext attacks and H is a cAXU family of hash functions, then HCBC1 is an OPRP secure against chosen-plaintext attacks.

Theorem 6.2. *Let $E\colon \{0,1\}^{ek} \times D_n \to D_n$ be a block cipher, and let $H\colon \{0,1\}^{hk} \times D_n \to D_n$ be a family of hash functions. Let* HCBC1 *be the n-on-line cipher associated to them as per Construction 6.1. Then, for any adversary A against* HCBC1 *running in time t and making oracle queries totalling at most μ blocks, there is an adversary B against E and an adversary X against H such that*

$$\mathbf{Adv}^{\text{oprp-cpa}}_{\mathsf{HCBC1}}(A) \leq \mathbf{Adv}^{\text{prp-cpa}}_{E}(B) + \frac{3\mu(\mu-1)}{2^{n+1}} + \mathbf{Adv}^{\text{axu}}_{H}(X).$$

Furthermore, B runs in time $t + O(n\mu + hk)$ and makes at most μ oracle queries, while X runs in time $t + O(n\mu)$ and outputs a set of size at most μ.

By Proposition 5.4, we can simply let $H = E$ to obtain a purely block-cipher-based instantiation of HCBC1 that has cost two block-cipher computations per block and uses two block-cipher keys. Proposition 5.3 says that other (possibly more efficient) instantiations are possible by setting H to an AXU family.

proc Initialize Game G_0

000 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$
001 $eK \xleftarrow{\$} \{0,1\}^{ek}$; $\pi \leftarrow E(eK, \cdot)$

proc Initialize Game G_1

100 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$
101 $\pi \xleftarrow{\$} \mathsf{Perm}_n$

proc Encipher(M) Games G_0, G_1

010 $i \leftarrow i+1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$
011 $(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
012 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
013 For $j = p+1, \ldots, l_i$ do
014 $\quad P_i[j] \leftarrow H(hK, C_i[j-1]) \oplus M_i[j]$
015 $\quad C_i[j] \leftarrow \pi(P_i[j])$
016 Return $C_i \leftarrow C_i[1..l_i]$

proc Finalize(d) Games G_0, G_1

020 Return d

Fig. 6. Games G_0, G_1 for proof of Theorem 6.2. The variable i counts the number of queries. Game G_0 is to imitate $\mathrm{OPRPCPA}_{\mathsf{HCBC1}}$ (hence the keys are chosen as shown on lines 000 and 001, and **Encipher** is as prescribed by Construction 6.1). In contrast, Game G_1 uses a random permutation (hence line 101). Line 011 finds the sth query that has the longest common prefix with the current query. The length of the common prefix thus found is named p. The outputs for all input blocks up to and including p are the same as previous corresponding outputs for the sth query (hence line 012). The outputs for the rest of the blocks are computed anew (hence lines 013–015).

Proof of Theorem 6.2. The proof is based on the sequence G_0–G_6 of games in Figs. 6, 7, and 8. Let us begin with some intuition.

Say A has queried $M_1, \ldots, M_{i-1}$ ($i \geq 1$) and received back $C_1, \ldots, C_{i-1}$. Say it now queries M_i. Let $(s, p) = \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$. Then, blocks $1, \ldots, p$ of C_i are determined via $C_i[1..p] = C_s[1..p]$. What about $C_i[j]$ for $j \geq p+1$? We have $C_i[j] = E(eK, P_i[j])$ where $P_i[j] = H(hK, C_i[j-1]) \oplus M_i[j]$. Intuitively, the cxu property tells us that $P_i[j] \neq P_{i'}[j']$ for all $(i', j') \in L(i, j)$—refer to Sect. 2 for the definition of the latter—where $P_{i'}[j'] = H(hK, C_{i'}[j'-1]) \oplus M_{i'}[j']$. Since E is a PRP, this means that $E(eK, P_i[j])$ looks random.

There are a couple of difficulties. One is that the cxu property as per Definition 5.2 refers to a setting where adversary X chooses its set S before hK is chosen. Yet, above, A is getting information about hK via responses to its queries. We address this by moving to a game in which blocks in replies to queries are random unless prefix-constrained, and in particular independent of hK. The second difficulty is that the cxu property is only violated by the collision $H(hK, C_i[j-1]) \oplus M_i[j] = H(hK, C_{i'}[j'-1]) \oplus M_{i'}[j']$ if $C_i[j-1] \neq C_{i'}[j'-1]$. We will use the randomness of the blocks to show that the last condition is usually true.

Let us now provide the full proof. The **Initialize** procedure of Game G_0 picks a key hK for the hash function and a key eK to define the instance π of the block cipher E. The **Encipher** procedure implements the HCBC1 function of Construction 6.1. Game G_1 is identical to G_0 except that π is a random permutation.

proc Initialize Game $\boxed{\mathrm{G}_2}$, G_3

200 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

proc Encipher(M) Game $\boxed{\mathrm{G}_2}$, G_3

210 $i \leftarrow i+1$; $M_i \leftarrow M$
211 $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$
212 $(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
213 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
214 For $j = p+1, \ldots, l_i$ do
215 $\quad P_i[j] \leftarrow H(hK, C_i[j-1]) \oplus M_i[j]$
216 $\quad$ If $P_i[j] \notin \mathsf{Dom}(\pi)$ then
217 $\quad\quad y \xleftarrow{\$} D_n$
218 $\quad\quad$ If $y \in \mathsf{Rng}(\pi)$ then
219 $\quad\quad\quad$ bad $\leftarrow$ true; $\boxed{y \xleftarrow{\$} \overline{\mathsf{Rng}}(\pi)}$
220 $\quad\quad \pi(P_i[j]) \leftarrow y$
221 $\quad C_i[j] \leftarrow \pi(P_i[j])$
222 Return $C_i \leftarrow C_i[1..l_i]$

proc Finalize(d) Game $\boxed{\mathrm{G}_2}$, G_3

230 Return d

proc Initialize Games $\boxed{\mathrm{G}_4}$, G_5

400 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

proc Encipher(M) Games $\boxed{\mathrm{G}_4}$, G_5

410 $i \leftarrow i+1$; $M_i \leftarrow M$
411 $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$
412 $(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
413 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
414 For $j = p+1, \ldots, l_i$ do
415 $\quad P_i[j] \leftarrow H(hK, C_i[j-1]) \oplus M_i[j]$
416 $\quad C_i[j] \xleftarrow{\$} D_n$
417 $\quad B(i,j) \leftarrow \{\, (i', j') \in L(i,j) \,:\, P_i[j] = P_{i'}[j'] \,\}$
418 $\quad$ If $B(i,j) \neq \emptyset$ then
419 $\quad\quad$ bad $\leftarrow$ true
420 $\quad\quad \boxed{(i', j') \xleftarrow{\$} B(i,j);\ C_i[j] \leftarrow C_{i'}[j']}$
421 Return $C_i \leftarrow C_i[1..l_i]$

proc Finalize(d) Games $\boxed{\mathrm{G}_4}$, G_5

430 Return d

Fig. 7. Games $\mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$, and G_5 for proof of Theorem 6.2. Game G_2 and G_4 include the boxed code while G_3 and G_5 do not. Game G_2 is like G_1 except that it samples π lazily in lines 216–220. Game G_3 is like G_2 except that in the former π is a random function rather than a permutation (hence the omission of the boxed code). Game G_4 is a rewrite of G_3 so that the outputs of π that need to be computed anew are optimistically picked at random in line 416 then corrected later in line 420 if necessary. Line 417 identifies the points that need correction. Game G_5 omits the corrections.

We note that

$$\begin{aligned}\Pr\big[\,\mathrm{OPRPCPA}^A_{\mathsf{HCBC1}} \Rightarrow 1\,\big] &= \Pr\big[\,\mathrm{G}_0^A \Rightarrow 1\,\big] \\ &= \Pr\big[\,\mathrm{G}_1^A \Rightarrow 1\,\big] + \big(\Pr\big[\,\mathrm{G}_0^A \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{G}_1^A \Rightarrow 1\,\big]\big). \quad (8)\end{aligned}$$

It is easy to define an adversary B so that

$$\Pr\big[\,\mathrm{G}_0^A \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{G}_1^A \Rightarrow 1\,\big] \le \mathbf{Adv}^{\text{prp-cpa}}_E(B). \quad (9)$$

Namely, B^π initializes i to 0 and picks hK at random from $\{0,1\}^{hk}$. It then runs A, answering its oracle queries with the code of the **Encipher** procedure of G_0 but with the role of π played by its own oracle. B returns what A returns. The running time of B is that of A plus the time to compute LCP_n^* for each oracle invocation and the time to pick hK at random. Implementing LCP_n^* with a tree data structure, rather than directly as described in Sect. 2, allows B to answer the ith **Encipher** query in time $O(nl_i)$. Thus, B's total running time is that of A plus $O(n\mu + hk)$.

Game G_2 is the same as G_1 except that it samples π lazily. The convention is that $\mathsf{Rng}(\pi)$ and $\mathsf{Dom}(\pi)$ are initially empty. $\overline{\mathsf{Rng}(\pi)}$ always denotes $D_n \setminus \mathsf{Rng}(\pi)$, and an assignment $\pi(x) \leftarrow y$ adds y to $\mathsf{Rng}(\pi)$ and x to $\mathsf{Dom}(\pi)$. Game G_3 is identical to G_2 except that in the former π is a (lazily sampled) random function rather than

proc Initialize Game G_6

600 $i \leftarrow 0$; $C_0[0] \leftarrow 0^n$

proc Encipher(M) Game G_6

610 $i \leftarrow i+1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $a(i,0) \leftarrow 0$
611 $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
612 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$; $a(i,j) \leftarrow a(s,j)$
613 For $j = p+1, \ldots, l_i$ do $C_i[j] \xleftarrow{\$} D_n$; $a(i,j) \leftarrow i$
614 Return $C_i \leftarrow C_i[1..l_i]$

proc Finalize(d) Game G_6

620 $hK \xleftarrow{\$} \{0,1\}^{hk}$
621 For $i = 1, \ldots, q$ do
622 $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
623 For $j = p+1, \ldots, l_i$ do
624 $P_i[j] \leftarrow H(hK, C_i[j-1]) \oplus M_i[j]$
625 $B(i,j) \leftarrow \{\, (i',j') \in L(i,j) \,:\, P_i[j] = P_{i'}[j'] \,\}$
626 If $B(i,j) \neq \emptyset$ then
627 $(i',j') \xleftarrow{\$} B(i,j)$
628 If $C_i[j-1] \neq C_{i'}[j'-1]$ then $\mathsf{bad} \leftarrow \mathsf{true}$
629 If $C_i[j-1] = C_{i'}[j'-1]$ then $\mathsf{bad} \leftarrow \mathsf{true}$
630 Return d

Fig. 8. Game G_6 for proof of Theorem 6.2. Here, q denotes the number of oracle queries made by A, with the wlog assumption that it always makes exactly q queries. An *ancestor* function a is defined in lines 610, 612, and 613 for later case analysis. Everything that can be delayed to **Finalize** is (hence lines 620–629). Finally, lines 627–629 together are equivalent to line 419 in Fig. 7.

permutation. Now, we have

$$\begin{aligned}\Pr\left[\mathrm{G}_1^A \Rightarrow 1\right] &= \Pr\left[\mathrm{G}_2^A \Rightarrow 1\right] \\ &= \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right] + \left(\Pr\left[\mathrm{G}_2^A \Rightarrow 1\right] - \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right]\right) \\ &\leq \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right] + \Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right], \end{aligned} \tag{10}$$

the last by Lemma 2.1. However,

$$\Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right] \leq \frac{\mu(\mu-1)}{2^{n+1}}. \tag{11}$$

Game G_4 writes the **Encipher** procedure of G_3 in a different but equivalent way: it optimistically picks $C_i[j]$ at random and then, if $\pi(P_i[j])$ was defined, appropriately corrects at line 420. Note that π is implicit, not appearing in the code, and choosing (i',j') at random from $B(i,j)$ is only for notational convenience; any member of this

set would do. Now,

$$\begin{aligned}\Pr\big[\,\mathrm{G}_3^A \Rightarrow 1\,\big] &= \Pr\big[\,\mathrm{G}_4^A \Rightarrow 1\,\big] \\ &= \Pr\big[\,\mathrm{G}_5^A \Rightarrow 1\,\big] + \big(\Pr\big[\,\mathrm{G}_4^A \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{G}_5^A \Rightarrow 1\,\big]\big) \\ &\le \Pr\big[\,\mathrm{G}_5^A \Rightarrow 1\,\big] + \Pr\big[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\big], \end{aligned} \tag{12}$$

the last by Lemma 2.1. Next, we see that

$$\begin{aligned}&\Pr\big[\,\mathrm{G}_5^A \Rightarrow 1\,\big] \\ &= \Pr\big[\,\mathrm{OPRFCPA}_{\mathsf{Rand}}^A \Rightarrow 1\,\big] \\ &= \Pr\big[\,\mathrm{OPRPCPA}_{\mathsf{Perm}}^A \Rightarrow 1\,\big] \\ &\quad + \big(\Pr\big[\,\mathrm{OPRFCPA}_{\mathsf{Rand}}^A \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{OPRPCPA}_{\mathsf{Perm}}^A \Rightarrow 1\,\big]\big) \\ &\le \Pr\big[\,\mathrm{OPRPCPA}_{\mathsf{Perm}}^A \Rightarrow 1\,\big] + \frac{\mu(\mu-1)}{2^{n+1}}\end{aligned}$$

where the last line holds due to Lemma 3.6. Putting (8), (9), (10), (11), and (12) together, we have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{HCBC1}}^{\text{oprp-cpa}}(A) &= \Pr\big[\,\mathrm{OPRPCPA}_{\mathsf{HCBC1}}^A \Rightarrow 1\,\big] - \Pr\big[\,\mathrm{OPRPCPA}_{\mathsf{Perm}}^A \Rightarrow 1\,\big] \\ &\le \mathbf{Adv}_E^{\text{prp-cpa}}(B) + \frac{\mu(\mu-1)}{2^n} + \Pr\big[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\big]. \end{aligned} \tag{13}$$

It remains to upper bound the probability that the execution of G_5 with A sets bad. The **Encipher** procedure of G_5 sets bad but does not use it, so G_6 delays its setting to **Finalize**, also breaking line 419 into the equivalent lines 628 and 629. It additionally introduces the *ancestor* function a, defining it at lines 610, 612, and 613, but this does not affect the setting of bad, so

$$\Pr\big[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\big] = \Pr\big[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}\,\big]. \tag{14}$$

Consider the cxu-adversary X of Fig. 9. Then,

$$\Pr[\mathrm{G}_6 \text{ sets } \mathsf{bad} \text{ at line } 628] \le \mathbf{Adv}_H^{\text{axu}}(X), \tag{15}$$

and X runs in time that of A plus $O(n\mu)$ and outputs a set of size at most μ. We now claim that

$$\Pr[\mathrm{G}_6 \text{ sets } \mathsf{bad} \text{ at line } 629] \le \frac{\mu(\mu-1)}{2^{n+1}}. \tag{16}$$

To justify these important claims, we consider the following cases.

Case 1: $j \ge p+2$. In this case, $j-1 \ge p+1$, so $C_i[j-1]$ was chosen at random at 613 after $C_{i'}[j'-1]$ was determined. So the probability of their being equal is at most 2^{-n}.

Case 2: $j = p+1$ and $j' = p+1$ and $\mathsf{LCP}_n(M_i, M_{i'}) \ge p$. In this case, we claim that the condition $P_i[j] = P_{i'}[j']$ implies that $C_i[j-1] \ne C_{i'}[j'-1]$, meaning bad cannot be set at line 629. We now justify this. Since $(i', j') \in L(i, j)$ and $j = j'$, it must be

Adversary X
  $i \leftarrow 0;\ S \leftarrow \emptyset$
  Run A
  On query **Encipher**(M)
    $i \leftarrow i+1;\ M_i \leftarrow M;\ l_i \leftarrow \|M_i\|_n;\ C_i[0] \leftarrow 0^n$
    $(s, p) \leftarrow \mathsf{LCP}^*_n(M_i, M_1, \ldots, M_{i-1})$
    For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
    For $j = p+1, \ldots, l_i$ do $S \leftarrow S \cup \{(C_i[j-1], M_i[j])\};\ C_i[j] \xleftarrow{\$} D_n$
    Return $C_i \leftarrow C_i[1..l_i]$ to A
  Until A halts
  Return S

Fig. 9. Adversary X for the proof of Theorem 6.2. It outputs the collision(s) identified in the case that bad is set in line 628 of Fig. 8 to break the cAXU property.

that $i' < i$. But $(s, p) = \mathsf{LCP}^*_n(M_i, M_1, \ldots, M_{i-1})$, so $\mathsf{LCP}_n(M_i, M_{i'}) \geq p$ implies that in fact $\mathsf{LCP}_n(M_i, M_{i'}) = p$. So $M_i[p+1] \neq M_{i'}[p+1]$. Now note that

$$P_i[j] = P_i[p+1] = H\big(hK, C_i[p]\big) \oplus M_i[p+1],$$
$$P_{i'}[j'] = P_{i'}[p+1] = H\big(hK, C_{i'}[p]\big) \oplus M_{i'}[p+1].$$

So $C_i[p] = C_{i'}[p]$ would imply $P_i[j] \neq P_{i'}[j']$.

A proof by induction can be used to verify that the ancestor function a has the following properties. First, $C_k[m] = C_{a(k,m)}[m]$, for any $k \in [1..q]$ and $m \in [0..l_k]$. Second, $C_{a(k,m)}[m]$ was chosen at random by Game G_6 (at line 613), for any $k \in [1..q]$ and $m \in [1..l_k]$. Third, if $a(k, m) < k$ then $\mathsf{LCP}_n(M_k, M_{a(k,m)}) \geq m$, for any $k \in [1..q]$ and $m \in [1..l_k]$. We are now ready to tackle the last case.

Case 3: $j = p+1$ and ($j' \neq p+1$ or $\mathsf{LCP}_n(M_i, M_{i'}) \leq p-1$). By the first property of the ancestor function a noted above, we have $C_i[j-1] = C_{a(i,j-1)}[j-1]$ and $C_{i'}[j'-1] = C_{a(i',j'-1)}[j'-1]$. So we want to upper bound the probability that $C_{a(i,j-1)}[j-1] = C_{a(i',j'-1)}[j'-1]$. By the second property of a noted above, if $j \neq 1$ then $C_{a(i,j-1)}[j-1]$ was chosen at random, and if $j' \neq 1$, then $C_{a(i',j'-1)}[j'-1]$ was chosen at random. So as long as $(j, j') \neq (1, 1)$ and $(a(i, j-1), j) \neq (a(i', j'-1), j')$, the probability that $C_{a(i,j-1)}[j-1] = C_{a(i',j'-1)}[j'-1]$ is 2^{-n}. Let us now check that the conditions above are met. We know that $j = p+1$, so if $p \geq 1$ then $j \neq 1$, so $(j, j') \neq (1, 1)$. If $p = 0$ then the condition $\mathsf{LCP}_n(M_i, M_{i'}) \leq p-1$ is not met (because $\mathsf{LCP}_n(M_i, M_{i'})$ is always non-negative) so it must be that $j' \neq p+1 = 1$, meaning we again have $(j, j') \neq (1, 1)$. Now consider whether $(a(i, j-1), j) = (a(i', j'-1), j')$. If $j' \neq p+1 = j$, then certainly this condition is not true. So suppose $j' = p+1 = j$. In that case, we are given that $\mathsf{LCP}_n(M_i, M_{i'}) \leq p-1$. Suppose toward a contradiction that $a(i, j-1) = a(i', j'-1)$ and call this common value α. Since $(i', j') \in L(i, j)$ and $j = j'$, it must be that $i' < i$. So $1 \leq \alpha \leq i' < i$. Then, by the third property of the ancestor function a noted above, it must be that $\mathsf{LCP}_n(M_i, M_\alpha) \geq p$. If $\alpha = i'$, this contradicts $\mathsf{LCP}_n(M_i, M_{i'}) \leq p-1$, so assume $\alpha < i'$. But then, again by the third property, we have $\mathsf{LCP}_n(M_{i'}, M_\alpha) \geq p$, which, together with $\mathsf{LCP}_n(M_i, M_\alpha) \geq p$, implies $\mathsf{LCP}_n(M_i, M_{i'}) \geq p$, contradicting $\mathsf{LCP}_n(M_i, M_{i'}) \leq p-1$.

Adversary $A^{g,g^{-1}}$
Let $C_1 = 1^n$ and let $C_2 = 0^n 1^n$
Let $M_1[1] \leftarrow g^{-1}(C_1)$ and let $M_2[1]M_2[2] \leftarrow g^{-1}(C_2)$
If $M_1[1] = M_2[2]$, then return 1 else return 0

Fig. 10. Chosen-ciphertext attack on HCBC1.

Equations (14), (15), and (16) imply that

$$\Pr\left[\, \mathrm{G}_5^A \text{ sets bad} \,\right] \le \mathbf{Adv}_H^{\mathrm{axu}}(X) + \frac{\mu(\mu-1)}{2^{n+1}}. \tag{17}$$

Combining (13) and (17) completes the proof. □

A Chosen-Ciphertext Attack Against HCBC1 We just showed that HCBC1 is secure against chosen-plaintext attack. It is, however, not secure against chosen-ciphertext attacks, as we now observe. Figure 10 shows the attack. The adversary A is given oracle access to g and g^{-1} where g is either an instance of HCBC1 or an instance of $\mathsf{OPerm}_{n,d}$. We claim that

$$\mathbf{Adv}_{\mathsf{HCBC1}}^{\text{oprp-cca}}(A) \ge 1 - 2^{-n}. \tag{18}$$

Since A made only 2 oracle queries, this shows that, as an on-line cipher, HCBC1 is not secure against chosen-ciphertext attack. We claim that

$$\Pr\left[\, g \xleftarrow{\$} \mathsf{HCBC1} \;:\; A^g = 1 \,\right] = 1 \quad \text{and} \quad \Pr\left[\, g \xleftarrow{\$} \mathsf{OPerm}_{n,d} \;:\; A^g = 1 \,\right] \le 2^{-n},$$

from which (18) follows. We justify these two equations as follows. First, suppose g is an instance of HCBC1. Since the first block of C_1 is 1^n, we have

$$M_2[2] = E^{-1}\big(eK,\ 1^n\big) \oplus H\big(hK, 0^n\big) = M_1[1].$$

This means that adversary A will always return 1 when g is an instance of OCBC, and the first equation is true. Now, consider the case where g is a random instance of $\mathsf{OPerm}_{n,d}$. Here, $M_1[1] = M_2[2]$ holds with probability at most 2^{-n}. Therefore, the adversary A^g outputs 1 with the probability at most 2^{-n}, and this justifies the second equation.

7. The HCBC2 Cipher

We construct an on-line cipher that is secure against chosen-ciphertext attacks.

Construction 7.1. Let $E\colon \{0,1\}^{ek} \times D_n \to D_n$ be a block cipher. Let $H\colon \{0,1\}^{hk} \times D_n^2 \to D_n$ be a family of functions. We associate to them a cipher HCBC2: $\{0,1\}^{ek+hk} \times D_{d,n} \to D_{d,n}$. A key for it is a pair $eK \| hK$ where eK is a key for E and hK is a key for H. The cipher and its inverse are defined as follows for $M, C \in D_{d,n}$. Fig. 11 illustrates the cipher.

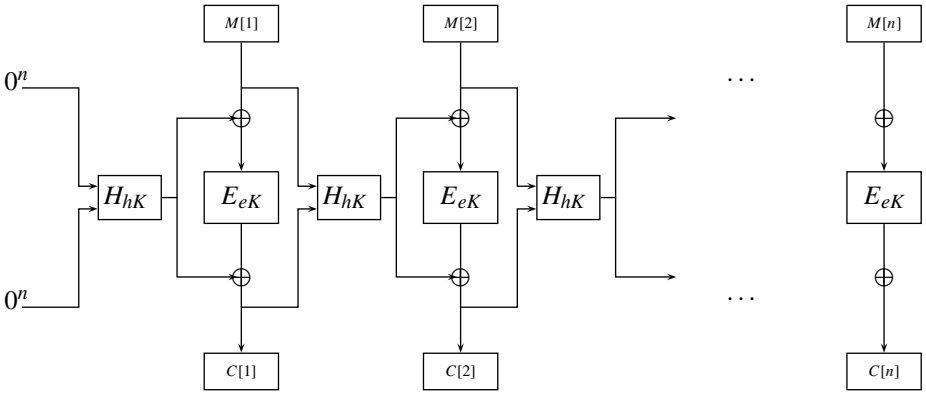

Fig. 11. The HCBC2 cipher.

HCBC2$(eK\|hK, M)$
$C[0] \leftarrow 0^n$; $M[0] \leftarrow 0^n$; $l \leftarrow \|M\|_n$
For $j = 1, \ldots, l$ do
  $h[j] \leftarrow H(hK, M[j-1]\|C[j-1])$
  $P[j] \leftarrow h[j] \oplus M[j]$
  $Q[j] \leftarrow E(eK, P[j])$
  $C[j] \leftarrow h[j] \oplus Q[j]$
Return $C[1..l]$

HCBC2$^{-1}(eK\|hK, C)$
$C[0] \leftarrow 0^n$; $M[0] \leftarrow 0^n$; $l \leftarrow \|C\|_n$
For $j = 1, \ldots, l$ do
  $h[j] \leftarrow H(hK, M[j-1]\|C[j-1])$
  $Q[j] \leftarrow h[j] \oplus C[j]$
  $P[j] \leftarrow E^{-1}(eK, Q[j])$
  $M[j] \leftarrow h[j] \oplus P[j]$
Return $M[1..l]$.

HCBC2 is slightly less efficient than HCBC1 because in the former the hash function takes a longer input. The following theorem implies that if E is a PRP secure against chosen-ciphertext attacks and H is an cAXU family of hash functions, then HCBC2 is an OPRP secure against chosen-ciphertext attacks.

Theorem 7.2. *Let $E\colon \{0,1\}^{ek} \times D_n \to D_n$ be a block cipher, and let $H\colon \{0,1\}^{hk} \times D_{2n} \to D_n$ be a family of hash functions. Let* HCBC2 *be the n-on-line cipher associated to them as per Construction 7.1. Then, for any adversary A against* HCBC2 *running in time t and making forward queries totalling at most μ_e blocks and backward queries totalling at most μ_d blocks, there is an adversary B against E and an adversary X against H such that*

$$\mathbf{Adv}^{\text{oprp-cca}}_{\text{HCBC2}}(A) \le \mathbf{Adv}^{\text{prp-cca}}_{E}(B) + \frac{3\mu(\mu-1)}{2^{n+1}} + \mathbf{Adv}^{\text{axu}}_{H}(X),$$

where $\mu = \mu_e + \mu_d$. Furthermore, B runs in time $t + O(n\mu + hk)$ and makes at most μ_e forward queries and at most μ_d backward queries, while X runs in time $t + O(n\mu)$ and outputs a set of size at most 2μ.

By Proposition 5.4, we can simply let $H = E$ to obtain a purely block-cipher-based instantiation of HCBC2 that has cost three block-cipher computations per block and uses two block-cipher keys. Proposition 5.3 says that other (possibly more efficient) instantiations are possible by setting H to an AXU family.

proc Initialize Game G_0

000 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

001 $eK \xleftarrow{\$} \{0,1\}^{ek}$; $\pi \leftarrow E(eK, \cdot)$

proc Initialize Game G_1

100 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

101 $\pi \xleftarrow{\$} \mathsf{Perm}_n$

proc Encipher(M) Games G_0, G_1

010 $i \leftarrow i+1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$

011 $(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$

012 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$

013 For $j = p+1, \ldots, l_i$ do

014 $\quad h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$

015 $\quad P_i[j] \leftarrow h_i[j] \oplus M_i[j]$; $Q_i[j] \leftarrow \pi(P_i[j])$; $C_i[j] \leftarrow h_i[j] \oplus Q_i[j]$

016 Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Games G_0, G_1

020 $i \leftarrow i+1$; $C_i \leftarrow C$; $l_i \leftarrow \|C_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$

021 $(s, p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$

022 For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$

023 For $j = p+1, \ldots, l_i$ do

024 $\quad h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$

025 $\quad Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$; $P_i[j] \leftarrow \pi^{-1}(Q_i[j])$; $M_i[j] \leftarrow h_i[j] \oplus P_i[j]$

026 Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Games G_0, G_1

020 Return d

Fig. 12. Games G_0, G_1 for proof of Theorem 7.2. The variable i counts the number of queries. Game G_0 is to imitate $\mathrm{OPRPCCA}_{\mathsf{HCBC2}}$ (hence the keys are chosen as shown on lines 000 and 001, and **Encipher** and **Decipher** are as prescribed by Construction 7.1). In contrast, Game G_1 uses a random permutation (hence line 101). The values s and p are computed in the same way as in the proof of Theorem 6.2. The outputs for all inputs blocks up to and including p are the same as previous outputs for the sth query (hence lines 012 and 022). The outputs for the rest of the blocks are computed anew (hence lines 013–015 and 023–025).

Proof of Theorem 7.2. The proof is based on the sequence G_0–G_7 of games in Figs. 12, 13, 14, and 15. The games first move us to a point where blocks in replies to (both forward and backward) oracle queries are random unless prefix-constrained, and in particular independent of hK. This is at the cost of the probability of setting bad. The latter is bounded using the axu property and the randomness of the blocks.

Let us now provide the full proof. It is similar to the proof of Theorem 6.2 so we will omit some explanations and details.

We note that

$$\begin{aligned}\Pr\left[\,\mathrm{OPRPCCA}^A_{\mathsf{HCBC2}} \Rightarrow 1\,\right] &= \Pr\left[\,\mathrm{G}_0^A \Rightarrow 1\,\right] \\ &= \Pr\left[\,\mathrm{G}_1^A \Rightarrow 1\,\right] + \left(\Pr\left[\,\mathrm{G}_0^A \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_1^A \Rightarrow 1\,\right]\right). \quad (19)\end{aligned}$$

It is easy to define an adversary B so that

$$\Pr\left[\,\mathrm{G}_0^A \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_1^A \Rightarrow 1\,\right] \le \mathbf{Adv}^{\mathrm{prp\text{-}cca}}_E(B). \quad (20)$$

Namely, $B^{\pi,\pi^{-1}}$ initializes i to 0 and picks hK at random from $\{0,1\}^{hk}$. It then runs A, answering its forward and backward oracle queries with the code of the **Encipher** and

proc Initialize Game $\boxed{\text{G}_2}$, G_3

200 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

proc Encipher(M) Game $\boxed{\text{G}_2}$, G_3

210 $i \leftarrow i+1$; $M_i \leftarrow M$
211 $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
212 $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
213 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
214 For $j = p+1, \ldots, l_i$ do
215 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
216 $P_i[j] \leftarrow h_i[j] \oplus M_i[j]$
217 If $P_i[j] \notin \mathsf{Dom}(\pi)$ then
218 $y \xleftarrow{\$} D_n$
229 If $y \in \mathsf{Rng}(\pi)$ then
220 bad $\leftarrow$ true; $\boxed{y \xleftarrow{\$} \overline{\mathsf{Rng}(\pi)}}$
221 $\pi(P_i[j]) \leftarrow y$
222 $Q_i[j] \leftarrow \pi(P_i[j])$
223 $C_i[j] \leftarrow h_i[j] \oplus Q_i[j]$
224 Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Games $\boxed{\text{G}_2}$, G_3

230 $i \leftarrow i+1$; $C_i \leftarrow C$
231 $l_i \leftarrow \|C_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
232 $(s,p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
233 For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
234 For $j = p+1, \ldots, l_i$ do
235 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
236 $Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$
237 If $Q_i[j] \notin \mathsf{Rng}(\pi)$ then
238 $x \xleftarrow{\$} D_n$
239 If $x \in \mathsf{Dom}(\pi)$ then
240 bad $\leftarrow$ true; $\boxed{x \xleftarrow{\$} \overline{\mathsf{Dom}(\pi)}}$
241 $\pi^{-1}(Q_i[j]) \leftarrow x$
242 $P_i[j] \leftarrow \pi^{-1}(Q_i[j])$
243 $M_i[j] \leftarrow h_i[j] \oplus P_i[j]$
244 Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Game $\boxed{\text{G}_2}$, G_3

250 Return d

proc Initialize Games $\boxed{\text{G}_4}$, G_5

400 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

proc Encipher(M) Games $\boxed{\text{G}_4}$, G_5

410 $i \leftarrow i+1$; $M_i \leftarrow M$
411 $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
412 $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
413 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
414 For $j = p+1, \ldots, l_i$ do
415 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
416 $P_i[j] \leftarrow h_i[j] \oplus M_i[j]$
417 $Q_i[j] \xleftarrow{\$} D_n$
418 $B(i,j) \leftarrow \{(i',j') \in L(i,j) P_i[j] = P_{i'}[j']\}$
419 If $B(i,j) \neq \emptyset$ then
420 bad $\leftarrow$ true
421 $\boxed{(i',j') \xleftarrow{\$} B(i,j);\ Q_i[j] \leftarrow Q_{i'}[j']}$
422 $C_i[j] \leftarrow h_i[j] \oplus Q_i[j]$
423 Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Games $\boxed{\text{G}_4}$, G_5

430 $i \leftarrow i+1$; $C_i \leftarrow C$
431 $l_i \leftarrow \|C_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
432 $(s,p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
433 For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
434 For $j = p+1, \ldots, l_i$ do
435 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
436 $Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$
437 $P_i[j] \xleftarrow{\$} D_n$
438 $B(i,j) \leftarrow \{(i',j') \in L(i,j) Q_i[j] = Q_{i'}[j']\}$
439 If $B(i,j) \neq \emptyset$ then
440 bad $\leftarrow$ true
441 $\boxed{(i',j') \xleftarrow{\$} B(i,j);\ P_i[j] \leftarrow P_{i'}[j']}$
442 $M_i[j] \leftarrow h_i[j] \oplus P_i[j]$
443 Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Games $\boxed{\text{G}_4}$, G_5

450 Return d

Fig. 13. Games G_2–G_5 for proof of Theorem 7.2. Games G_2 and G_4 include the boxed code while G_3 and G_5 do not. Game G_2 is like G_1 except that it samples π lazily in lines 217–221. Game G_3 is like G_2 except that in the former π is a random function rather than a permutation. Game G_4 is a rewrite of G_3 so that, for **Encipher** (resp. **Decipher**), the outputs (resp. inputs) of π that need to be computed anew are optimistically picked at random in line 417 (resp. 437) then corrected in line 421 (resp. 441) if necessary. Game G_5 omits the corrections.

proc Initialize Game G_6

600 $i \leftarrow 0$; $hK \xleftarrow{\$} \{0,1\}^{hk}$

proc Encipher(M) Game G_6

610 $i \leftarrow i+1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
611 $(s, p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
612 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
613 For $j = p+1, \ldots, l_i$ do
614 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
615 $P_i[j] \leftarrow h_i[j] \oplus M_i[j]$; $C_i[j] \xleftarrow{\$} D_n$; $Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$
616 $B(i,j) \leftarrow \{(i', j') \in L(i,j) P_i[j] = P_{i'}[j']\}$
617 If $B(i,j) \neq \emptyset$ then bad $\leftarrow$ true
618 Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Game G_6

630 $i \leftarrow i+1$; $C_i \leftarrow C$; $l_i \leftarrow \|C_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
631 $(s, p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
632 For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
633 For $j = p+1, \ldots, l_i$ do
634 $h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$
635 $Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$; $M_i[j] \xleftarrow{\$} D_n$; $P_i[j] \leftarrow h_i[j] \oplus M_i[j]$
636 $B(i,j) \leftarrow \{(i', j') \in L(i,j) Q_i[j] = Q_{i'}[j']\}$
637 If $B(i,j) \neq \emptyset$ then bad $\leftarrow$ true
638 Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Game G_6

640 Return d

Fig. 14. Game G_6 for proof of Theorem 7.2 is similar to G_5 except that, for **Encipher** (resp. **Decipher**), the C-values (resp. the M-values) are chosen at random in line 615 (resp. 635) rather than the Q-values (resp. the P-values).

Decipher procedures of G_0, respectively, but with the roles of π and π^{-1} played by its own oracles. B returns what A returns. The running time of B is that of A plus the time to compute LCP_n^* for each oracle invocation and the time to pick hK at random. Implementing LCP_n^* with a tree data structure, rather than directly as described in Sect. 2, allows B to answer the ith **Encipher** or **Decipher** query in time $O(nl_i)$. Thus, B's total running time is $O(n\mu + hk)$.

Game G_2 is the same as G_1 except that it samples π lazily. The convention is that $\mathsf{Rng}(\pi)$ and $\mathsf{Dom}(\pi)$ are initially empty. $\overline{\mathsf{Rng}(\pi)}$ always denotes $D_n \setminus \mathsf{Rng}(\pi)$, and $\overline{\mathsf{Dom}}(\pi)$ always denotes $D_n \setminus \mathsf{Dom}(\pi)$. An assignment $\pi(x) \leftarrow y$ adds y to $\mathsf{Rng}(\pi)$ and x to $\mathsf{Dom}(\pi)$. Now, we have

$$\begin{aligned}\Pr\left[\mathrm{G}_1^A \Rightarrow 1\right] &= \Pr\left[\mathrm{G}_2^A \Rightarrow 1\right] \\ &= \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right] + \left(\Pr\left[\mathrm{G}_2^A \Rightarrow 1\right] - \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right]\right)\end{aligned}$$

proc Initialize Game G_7

700 $i \leftarrow 0$; $C_0[0] \leftarrow 0^n$

proc Encipher(M) Game G_7

710 $i \leftarrow i+1$; $M_i \leftarrow M$; $l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$; $\mathsf{anc}(i,0) \leftarrow 0$; $\mathsf{dir}(i) \leftarrow \mathsf{frwd}$
711 $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
712 For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$; $\mathsf{anc}(i,j) \leftarrow \mathsf{anc}(s,j)$
713 For $j = p+1, \ldots, l_i$ do $C_i[j] \xleftarrow{\$} D_n$; $\mathsf{anc}(i,j) \leftarrow i$
714 Return $C_i \leftarrow C_i[1..l_i]$

proc Decipher(C) Game G_7

720 $i \leftarrow i+1$; $C_i \leftarrow C$; $l_i \leftarrow \|C_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$; $\mathsf{anc}(i,0) \leftarrow 0$; $\mathsf{dir}(i) \leftarrow \mathsf{bkwd}$
721 $(s,p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
722 For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$; $\mathsf{anc}(i,j) \leftarrow \mathsf{anc}(s,j)$
723 For $j = p+1, \ldots, l_i$ do $M_i[j] \xleftarrow{\$} D_n$; $\mathsf{anc}(i,j) \leftarrow i$
724 Return $M_i \leftarrow M_i[1..l_i]$

proc Finalize(d) Game G_7

730 $hK \xleftarrow{\$} \{0,1\}^{hk}$
731 For $i = 1, \ldots, q$ do
732 $\quad l_i \leftarrow \|M_i\|_n$; $C_i[0] \leftarrow 0^n$; $M_i[0] \leftarrow 0^n$
733 $\quad$ If $\mathsf{dir}(i) = \mathsf{frwd}$ then $(s,p) \leftarrow \mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$
734 $\quad$ Else $(s,p) \leftarrow \mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$
735 $\quad$ For $j = p+1, \ldots, l_i$ do
736 $\quad\quad h_i[j] \leftarrow H(hK, M_i[j-1] \| C_i[j-1])$; $P_i[j] \leftarrow h_i[j] \oplus M_i[j]$; $Q_i[j] \leftarrow h_i[j] \oplus C_i[j]$
737 $\quad\quad$ If $\mathsf{dir}(i) = \mathsf{frwd}$ then $B(i,j) \leftarrow (i',j') \in L(i,j) P_i[j] = P_{i'}[j']\}$
738 $\quad\quad$ Else $B(i,j) \leftarrow \{(i',j') \in L(i,j) Q_i[j] = Q_{i'}[j']\}$
739 $\quad\quad$ If $B(i,j) \neq \emptyset$ then
740 $\quad\quad\quad (i',j') \xleftarrow{\$} B(i,j)$
741 $\quad\quad\quad$ If $(M_i[j-1], C_i[j-1]) \neq (M_{i'}[j'-1], C_{i'}[j'-1])$ then $\mathsf{bad} \leftarrow \mathsf{true}$
742 $\quad\quad\quad$ If $(M_i[j-1], C_i[j-1]) = (M_{i'}[j'-1], C_{i'}[j'-1])$ then $\mathsf{bad} \leftarrow \mathsf{true}$
743 Return d

Fig. 15. Game G_7 for proof of Theorem 7.2. Here, q denotes the number of forward and backward oracle queries made by A, with the wlog assumption that it always makes exactly q queries. Everything that can be delayed to **Finalize** is (hence lines 730–742). Finally, lines 739–742 together are equivalent to line 637 in Fig. 14. The functions anc and dir are defined here for later case analysis.

$$\leq \Pr\left[\mathrm{G}_3^A \Rightarrow 1\right] + \Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right], \tag{21}$$

the last by Lemma 2.1. However,

$$\Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right] \leq \frac{\mu(\mu-1)}{2^{n+1}}. \tag{22}$$

Game G_4 is very similar to G_3, the only difference is that the code for sampling a random function is written differently, with tracking down when collisions between P-values in the **Encipher** procedure and Q-values in the **Decipher** procedure happen; without affecting any results. We note that $(i'j')$ in line 421 are picked at random only for simplicity. Game G_5 is identical to G_4 until they set bad, but Game G_5 does not use the collision check, i.e. sampling of random function may not be consistent for equal

points. Now,

$$\begin{aligned}\Pr\left[\,\mathrm{G}_3^A \Rightarrow 1\,\right] &= \Pr\left[\,\mathrm{G}_4^A \Rightarrow 1\,\right] \\ &= \Pr\left[\,\mathrm{G}_5^A \Rightarrow 1\,\right] + \left(\Pr\left[\,\mathrm{G}_4^A \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_5^A \Rightarrow 1\,\right]\right) \\ &\le \Pr\left[\,\mathrm{G}_5^A \Rightarrow 1\,\right] + \Pr\left[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\right], \end{aligned} \tag{23}$$

the last by Lemma 2.1.

To compare Games G_5 and G_6 note that the distribution of $(P_i[j], Q_i[j], C_i[j])$ is the same whether the quantities are chosen by lines 416, 417, and 422 or by line 615. Similarly, the distribution of $(P_i[j], Q_i[j], M_i[j])$ is the same whether the quantities are chosen by lines 436, 437, 442 or by line 635. So,

$$\Pr\left[\,\mathrm{G}_5^A \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{G}_6^A \Rightarrow 1\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\right] = \Pr\left[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}\,\right]. \tag{24}$$

Lines 614–617 and 634–637 determine the setting of bad but do not influence replies to oracle queries. Accordingly, Game G_7 moves them into **Finalize**. This game also defines the ancestor function anc and the function dir and splits the setting of bad into two parts at lines 741 and 742. The value of $\mathsf{anc}(i, j)$ is the smallest (earliest) query index $l \le i$ such that $M_i[j] = M_l[j]$. Function $\mathsf{dir}(i)$ returns frwd if M_i was created by a forward (i.e. **Encipher**) query and bkwd if it was created by a backward (i.e. **Decipher**) query. These changes affect neither the game output nor the probability of setting bad, hence

$$\Pr\left[\,\mathrm{G}_6^A \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{G}_7^A \Rightarrow 1\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}\,\right] = \Pr\left[\,\mathrm{G}_7^A \text{ sets } \mathsf{bad}\,\right]. \tag{25}$$

We comment that $\mathsf{LCP}_n^*(M_i, M_1, \ldots, M_{i-1})$ at line 733 of G_7 and $\mathsf{LCP}_n^*(C_i, C_1, \ldots, C_{i-1})$ at line 734 of G_7 may not be equal, which is why we considered separately the cases $\mathsf{dir}(i) = \mathsf{frwd}$ and $\mathsf{dir}(i) = \mathsf{bkwd}$ in defining (s, p) at lines 733 and 734. Next, we observe that

$$\begin{aligned}&\Pr\left[\,\mathrm{G}_7^A \Rightarrow 1\,\right] \\ &= \Pr\left[\,\mathrm{OPRFCCA}_{\mathsf{Rand}}^A \Rightarrow 1\,\right] \\ &= \Pr\left[\,\mathrm{OPRPCCA}_{\mathsf{Perm}}^A \Rightarrow 1\,\right] \\ &\quad + \left(\Pr\left[\,\mathrm{OPRFCCA}_{\mathsf{Rand}}^A \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{OPRPCCA}_{\mathsf{Perm}}^A \Rightarrow 1\,\right]\right) \\ &\le \Pr\left[\,\mathrm{OPRPCCA}_{\mathsf{Perm}}^A \Rightarrow 1\,\right] + \frac{\mu(\mu-1)}{2^{n+1}} \end{aligned} \tag{26}$$

where the last line holds due to Lemma 3.6. Putting (19), (20), (21), (22), (23), (24), (25), and (26) together, we have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{HCBC2}}^{\text{oprp-cca}}(A) &= \Pr\left[\,\mathrm{OPRPCCA}_{\mathsf{HCBC2}}^A \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{OPRPCCA}_{\mathsf{Perm}}^A \Rightarrow 1\,\right] \\ &\le \mathbf{Adv}_E^{\text{prp-cca}}(B) + \frac{\mu(\mu-1)}{2^n} + \Pr\left[\,\mathrm{G}_7^A \text{ sets } \mathsf{bad}\,\right]. \end{aligned} \tag{27}$$

Adversary X
  $i \leftarrow 0;\ S \leftarrow \emptyset$
  Run A
  On query **Encipher**(M)
    $i \leftarrow i+1;\ M_i \leftarrow M;\ l_i \leftarrow \|M_i\|_n;\ C_i[0] \leftarrow 0^n$
    $(s, p) \leftarrow \mathsf{LCP}^*_n(M_i, M_1, \ldots, M_{i-1})$
    For $j = 1, \ldots, p$ do $C_i[j] \leftarrow C_s[j]$
    For $j = p+1, \ldots, l_i$ do
      $C_i[j] \xleftarrow{\$} D_n;\ S \leftarrow S \cup \{(M_i[j-1] \| C_i[j-1], M_i[j])\} \cup \{(M_i[j-1] \| C_i[j-1], C_i[j])\}$
    Return $C_i \leftarrow C_i[1..l_i]$ to A
  On query **Decipher**(C)
    $i \leftarrow i+1;\ C_i \leftarrow C;\ l_i \leftarrow \|C_i\|_n;\ C_i[0] \leftarrow 0^n;\ M_i[0] \leftarrow 0^n$
    $(s, p) \leftarrow \mathsf{LCP}^*_n(C_i, C_1, \ldots, C_{i-1})$
    For $j = 1, \ldots, p$ do $M_i[j] \leftarrow M_s[j]$
    For $j = p+1, \ldots, l_i$ do
      $M_i[j] \xleftarrow{\$} D_n;\ S \leftarrow S \cup \{(M_i[j-1] \| C_i[j-1], M_i[j])\} \cup \{(M_i[j-1] \| C_i[j-1], C_i[j])\}$
    Return $M_i \leftarrow M_i[1..l_i]$ to A
  Until A halts
  Return S

Fig. 16. Adversary X for the proof of Theorem 7.2. It outputs the collision(s) identified in the case that bad is set in line 741 of Fig. 15 to break the cAXU property.

It remains to upper bound the probability that the execution of G_7 with A sets bad. Consider the cxu-adversary X of Fig. 16. Then,

$$\Pr[\mathrm{G}_7 \text{ sets bad at line } 741] \le \mathbf{Adv}^{\mathrm{axu}}_H(X), \tag{28}$$

and X runs in time that of A plus $O(n\mu)$ and outputs a set of size at most 2μ. We now claim that

$$\Pr[\mathrm{G}_7 \text{ sets bad at line } 742] \le \frac{\mu(\mu-1)}{2^{n+1}}. \tag{29}$$

To justify this important claim, we consider the following cases.

Case 1: $j \ge p+2$. In this case, $j-1 \ge p+1$. So if $\mathsf{dir}(i) = \mathsf{frwd}$ then $C_i[j-1]$ was chosen at random due to line 713, and if $\mathsf{dir}(i) = \mathsf{bkwd}$ then $M_i[j-1]$ was chosen at random due to line 723, in either case after $(M_{i'}[j'-1], C_{i'}[j'-1])$ was determined. So the probability that $(M_i[j-1], C_i[j-1])$ equals $(M_{i'}[j'-1], C_{i'}[j'-1])$ is at most 2^{-n}.

Case 2: $j = p+1$ and $\mathsf{dir}(i) = \mathsf{frwd}$ and $j' = p+1$ and $\mathsf{LCP}_n(M_i, M_{i'}) \ge p$. In this case, we claim that the condition $P_i[j] = P_{i'}[j']$ implies that $(M_i[j-1], C_i[j-1]) \ne (M_{i'}[j'-1], C_{i'}[j'-1])$, meaning bad cannot be set at line 742. We now justify this. Since $(i', j') \in L(i, j)$ and $j = j'$, it must be that $i' < i$. But $(s, p) = \mathsf{LCP}^*_n(M_i, M_1, \ldots, M_{i-1})$, so $\mathsf{LCP}_n(M_i, M_{i'}) \ge p$ implies that in fact $\mathsf{LCP}_n(M_i, M_{i'}) = p$. So $M_i[p+1] \ne M_{i'}[p+1]$. Now note that

$$P_i[j] = P_i[p+1] = H\big(hK, M_i[p] \| C_i[p]\big) \oplus M_i[p+1],$$
$$P_{i'}[j'] = P_{i'}[p+1] = H\big(hK, M_{i'}[p] \| C_{i'}[p]\big) \oplus M_{i'}[p+1].$$

So $(M_i[p], C_i[p]) = (M_{i'}[p], C_{i'}[p])$ would imply $P_i[j] \ne P_{i'}[j']$.

Case 3: $j = p + 1$ and $\mathsf{dir}(i) = \mathsf{bkwd}$ and $j' = p + 1$ and $\mathsf{LCP}_n(C_i, C_{i'}) \geq p$. In this case, we claim that the condition $Q_i[j] = Q_{i'}[j']$ implies that $(M_i[j-1], C_i[j-1]) \neq (M_{i'}[j'-1], C_{i'}[j'-1])$, meaning bad cannot be set at line 742. We now justify this. Since $(i', j') \in L(i, j)$ and $j = j'$, it must be that $i' < i$. But $(s, p) = \mathsf{LCP}^*_n(C_i, C_1, \ldots, C_{i-1})$, so $\mathsf{LCP}_n(C_i, C_{i'}) \geq p$ implies that in fact $\mathsf{LCP}_n(C_i, C_{i'}) = p$. So $C_i[p+1] \neq C_{i'}[p+1]$. Now note that

$$Q_i[j] = Q_i[p+1] = H\big(hK, M_i[p] \| C_i[p]\big) \oplus C_i[p+1],$$
$$Q_{i'}[j'] = Q_{i'}[p+1] = H\big(hK, M_{i'}[p] \| C_{i'}[p]\big) \oplus C_{i'}[p+1].$$

So $(M_i[p], C_i[p]) = (M_{i'}[p], C_{i'}[p])$ would imply $Q_i[j] \neq Q_{i'}[j']$.

A proof by induction can be used to verify that the ancestor function anc has the following properties. First, $(M_k[m], C_k[m]) = (M_{\mathsf{anc}(k,m)}[m], C_{\mathsf{anc}(k,m)}[m])$, for any $k \in [1..q]$ and $m \in [0..l_k]$. Second, at least one component of the pair $(M_{\mathsf{anc}(k,m)}[m], C_{\mathsf{anc}(k,m)}[m])$ was chosen at random by Game G_8 (at line 713 or 723), for any $k \in [1..q]$ and $m \in [1..l_k]$. Third, if $\mathsf{anc}(k, m) < k$ for any $k \in [1..q]$ and $m \in [1..l_k]$, then $\mathsf{LCP}_n(M_k, M_{\mathsf{anc}(k,m)}) \geq m$ and $\mathsf{LCP}_n(C_k, C_{\mathsf{anc}(k,m)}) \geq m$. We are now ready to tackle the last case.

Case 4: $j = p + 1$ and ($j' \neq p + 1$ or $\mathsf{LCP}_n(X_i, X_{i'}) \leq p - 1$) where $(X_i, X_{i'}) = (M_i, M_{i'})$ if $\mathsf{dir}(i) = \mathsf{frwd}$ and $(X_i, X_{i'}) = (C_i, C_{i'})$ if $\mathsf{dir}(i) = \mathsf{bkwd}$. By the first property of the ancestor function a noted above, we have $(M_i[j-1], C_i[j-1]) = (M_{\mathsf{anc}(i,j-1)}[j-1], C_{\mathsf{anc}(i,j-1)}[j-1])$ and $(M_{i'}[j'-1], C_{i'}[j'-1]) = (M_{\mathsf{anc}(i',j'-1)}[j'-1], C_{\mathsf{anc}(i',j'-1)}[j'-1])$. So we want to upper bound the probability that $(M_{\mathsf{anc}(i,j-1)}[j-1], C_{\mathsf{anc}(i,j-1)}[j-1]) = (M_{\mathsf{anc}(i',j'-1)}[j'-1], C_{\mathsf{anc}(i',j'-1)}[j'-1])$. By the second property of a noted above, if $j \neq 1$ then at least one component of $(M_{\mathsf{anc}(i,j-1)}[j-1], C_{\mathsf{anc}(i,j-1)}[j-1])$ was chosen at random, and if $j' \neq 1$, then at least one component of $(M_{\mathsf{anc}(i',j'-1)}[j'-1], C_{\mathsf{anc}(i',j'-1)}[j'-1])$ was chosen at random. So as long as $(j, j') \neq (1, 1)$ and $(\mathsf{anc}(i, j-1), j) \neq (\mathsf{anc}(i', j'-1), j')$, the probability that $(M_{\mathsf{anc}(i,j-1)}[j-1], C_{\mathsf{anc}(i,j-1)}[j-1]) = (M_{\mathsf{anc}(i',j'-1)}[j'-1], C_{\mathsf{anc}(i',j'-1)}[j'-1])$ is at most 2^{-n}. Let us now check that the conditions above are met. We know that $j = p + 1$, so if $p \geq 1$ then $j \neq 1$, so $(j, j') \neq (1, 1)$. If $p = 0$ then the condition $\mathsf{LCP}_n(X_i, X_{i'}) \leq p - 1$ is not met (because $\mathsf{LCP}_n(X_i, X_{i'})$ is always non-negative) so it must be that $j' \neq p + 1 = 1$, meaning we again have $(j, j') \neq (1, 1)$. Now consider whether $(\mathsf{anc}(i, j-1), j) = (\mathsf{anc}(i', j'-1), j')$. If $j' \neq p + 1 = j$, then certainly this condition is not true. So suppose $j' = p + 1 = j$. In that case, we are given that $\mathsf{LCP}_n(X_i, X_{i'}) \leq p - 1$. Suppose toward a contradiction that $\mathsf{anc}(i, j-1) = \mathsf{anc}(i', j'-1)$ and call this common value α. Since $(i', j') \in L(i, j)$ and $j = j'$, it must be that $i' < i$. So $1 \leq \alpha \leq i' < i$. Then, by the third property of the ancestor function a noted above, it must be that $\mathsf{LCP}_n(X_i, X_\alpha) \geq p$ where $X_\alpha = M_\alpha$ if $\mathsf{dir}(i) = \mathsf{frwd}$ and $X_\alpha = C_\alpha$ if $\mathsf{dir}(i) = \mathsf{bkwd}$. If $\alpha = i'$, this contradicts $\mathsf{LCP}_n(X_i, X_{i'}) \leq p - 1$, so assume $\alpha < i'$. But then, again by the third property, we have $\mathsf{LCP}_n(X_{i'}, X_\alpha) \geq p$, which, together with $\mathsf{LCP}_n(X_i, X_\alpha) \geq p$, implies $\mathsf{LCP}_n(X_i, X_{i'}) \geq p$, contradicting $\mathsf{LCP}_n(X_i, X_{i'}) \leq p - 1$.

proc Initialize Game IND-CPA$_{\mathcal{SE}}$
$K \xleftarrow{\$} \mathcal{K}$; $b \xleftarrow{\$} \{0,1\}$

proc LR(M_0, M_1) Game IND-CPA$_{\mathcal{SE}}$
$C \xleftarrow{\$} \mathcal{E}(K, M_b)$; Return C

proc Finalize(d) Game IND-CPA$_{\mathcal{SE}}$
If $d = b$ then return 1 else return 0

proc Initialize Game INT-CTXT$_{\mathcal{SE}}$
$K \xleftarrow{\$} \mathcal{K}$; $S \leftarrow \emptyset$

proc Enc(M) Game INT-CTXT$_{\mathcal{SE}}$
$C \xleftarrow{\$} \mathcal{E}(K, M)$; $S \leftarrow S \cup \{C\}$; Return C

proc Finalize(C) Game INT-CTXT$_{\mathcal{SE}}$
If $\mathcal{D}_K(C) = \perp$ then return 0
If $C \notin S$ then return 1 else return 0

Fig. 17. Games IND-CPA$_{\mathcal{SE}}$ and INT-CTXT$_{\mathcal{SE}}$. Above, $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Equations (28) and (29) imply that

$$\Pr\left[\, \mathrm{G}_7^A \text{ sets } \mathsf{bad} \,\right] \leq \mathbf{Adv}^{\mathrm{axu}}_H(X) + \frac{\mu(\mu-1)}{2^{n+1}}. \tag{30}$$

Combining (27) and (30) completes the proof. □

8. Usage of On-line Ciphers

On-line ciphers can be used to encrypt and authenticate data in such a way that strong privacy and authenticity properties result, if the plaintext space has appropriate characteristics. This follows via the encode-then-encipher paradigm of [8]. It is shown in [8] that it suffices to apply a cipher which is a pseudorandom permutation to a message which contains some randomness and redundancy. However, we cannot apply this paradigm in as much generality as [8] do since we deal with weaker ciphers: ones which are pseudorandom *on-line* permutations. Being more specific, however, allows us to solve the problem. We suggest and discuss the following three message-encoding possibilities.

(1) *Prepend* randomness and *append* redundancy.[1]
(2) *Prepend* randomness and message length and *append* redundancy.
(3) *Prepend* and *append the same* randomness.

Before we formally define and analyze the resulting schemes we recall the standard notions of privacy and integrity for symmetric encryption.

Definitions Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, defined as usual via its key-generation, encryption, and decryption algorithms [5]. We use the IND-CPA notion of privacy, measured via the "left-or-right" model of [5]. Consider game IND-CPA$_{\mathcal{SE}}$ of Fig. 17. An oracle query of the adversary A must be a pair M_0, M_1 of equal-length strings belonging to the message space MsgSp associated to the scheme. The *advantage* of A is defined via

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) = 2 \cdot \Pr\left[\, \text{IND-CPA}^A_{\mathcal{SE}} \Rightarrow 1 \,\right] - 1.$$

[1] The preliminary version of this paper proposed only this encoding construction, which, as we discuss below, is useful only if all messages have the same length.

We use a simplified version of the INT-CTXT notion of integrity of [7]. Consider game $\text{INT-CTXT}_{\mathcal{SE}}$ of Fig. 17. An oracle query of the adversary B must be an element of the message space MsgSp associated to the scheme. We define the *advantage* of B via

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(B) = \Pr\big[\,\text{INT-CTXT}^{B}_{\mathcal{SE}} \Rightarrow 1\,\big].$$

On-line-Cipher-Based Encryption Schemes We now formally define and discuss the encryption schemes which use an on-line cipher applied to encoded messages.

Construction 8.1. Let n, d be integers with $d \geq 3$, and let $F\colon \text{Keys}(F) \times D_{d,n} \to D_{d,n}$ be a cipher. We associate to them the following symmetric encryption scheme $\mathcal{SE}_1 = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

Algorithm $\mathcal{K}$	Algorithm $\mathcal{E}(K, M)$	Algorithm $\mathcal{D}(K, C)$
$K \xleftarrow{\$} \text{Keys}(F)$	$r \xleftarrow{\$} D_n$	$x \leftarrow F^{-1}(K, C)$
Return K	$x \leftarrow r \| M \| 0^n$	If $\lvert x\rvert < 3n$ then return $\perp$
	$C \leftarrow F(K, x)$	Parse x as $r \| M \| \tau$ with $\lvert r\rvert = \lvert\tau\rvert = n$
	Return C	If $\tau = 0^n$ then return M
		Else return $\perp$.

The security depends on the message space MsgSp of $\mathcal{SE}_1$. If MsgSp is D^l_n for some fixed $l \in [1..d-2]$, meaning all messages are restricted to be of the same fixed length, then $\mathcal{SE}_1$ is IND-CPA secure if F is an n-on-line cipher secure against chosen-plaintext attacks and INT-CTXT secure if F is an n-on-line cipher secure against chosen-ciphertext attacks. However, this scheme is not INT-CTXT secure if variable-length (but still in $D_{d,n}$) messages can be encrypted, meaning the message space is, say, $D_{d-2,n}$. To see this, consider the following adversary B attacking the INT-CTXT security of $\mathcal{SE}_1$. B queries to its encryption oracle the message $0^n \| 0^n$. Let $C[1] \| C[2] \| C[3] \| C[4]$ be the reply. Then B outputs $C[1] \| C[2] \| C[3]$. It is easy to see that $\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(B) = 1$ since $C[1] \| C[2] \| C[3]$ is a valid encryption of 0^n.

In order to defend against the above attack, in addition to prepending randomness to the message one might want to prepend message length. The following construction formalizes this obvious fix.

Construction 8.2. Let n be an integer, and let $d \in [4..2^n - 1]$. For $x \in D_{d,n}$, let $\langle x \rangle_n$ denote the n-bit binary encoding of $\|x\|_n$. Let $F\colon \text{Keys}(F) \times D_{d,n} \to D_{d,n}$ be a cipher. Define symmetric encryption scheme $\mathcal{SE}_2 = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ via

Algorithm $\mathcal{K}$	Algorithm $\mathcal{E}(K, M)$	Algorithm $\mathcal{D}(K, C)$
$K \xleftarrow{\$} \text{Keys}(F)$	$r \xleftarrow{\$} D_n$	$x \leftarrow F^{-1}(K, C)$
Return K	$x \leftarrow r \| \langle M \rangle_n \| M \| 0^n$	If $\lvert x\rvert < 4n$ then return $\perp$
	$C \leftarrow F(K, x)$	Parse x as $r \| m \| M \| \tau$
	Return C	with $\lvert r\rvert = \lvert m\rvert = \lvert\tau\rvert = n$
		If $\langle M \rangle_n = m$ and $\tau = 0^n$ then return M
		Else return $\perp$.

The message space associated to $\mathcal{SE}_2$ is $D_{d-3,n}$.

While it is possible to show that $\mathcal{SE}_2$ is IND-CPA secure assuming F is an n-on-line cipher secure against chosen-plaintext attacks and INT-CTXT secure assuming F is an n-on-line cipher secure against chosen-ciphertext attacks, encryption requires knowing the length of the message in advance, which does not suit well the "on-line" goal, since with the latter we may want to encrypt messages in a single pass, and we may not know the message length in advance. The following scheme overcomes the difficulties pertaining to the previous constructions.

Construction 8.3. Let n, d be integers with $d \geq 3$, and let $F\colon \mathrm{Keys}(F) \times D_{d,n} \to D_{d,n}$ be a cipher. We associate to them the following symmetric encryption scheme $\mathcal{SE}_3 = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

Algorithm $\mathcal{K}$	Algorithm $\mathcal{E}(K, M)$	Algorithm $\mathcal{D}(K, C)$
$K \xleftarrow{\$} \mathrm{Keys}(F)$	$r \xleftarrow{\$} D_n$	$x \leftarrow F^{-1}(K, C)$
Return K	$x \leftarrow r \| M \| r$	If $\lvert x\rvert < 3n$ then return $\perp$
	$C \leftarrow F(K, x)$	Parse x as $r \| M \| r'$
	Return C	with $\lvert r\rvert = \lvert r'\rvert = n$
		If $r = r'$ then return M
		Else return $\perp$.

The message space associated to $\mathcal{SE}_3$ is $D_{d-2,n}$.

We show that $\mathcal{SE}_3$ is IND-CPA secure, when F is an n-on-line cipher secure against chosen-plaintext attacks, and INT-CTXT secure, when F is an n-on-line cipher secure against chosen-ciphertext attacks. The following claims formalize this.

Claim 8.4. *Let $F\colon \mathrm{Keys}(F) \times D_{d,n} \to D_{d,n}$ be an n-on-line cipher, and let $\mathcal{SE}_3 = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the symmetric encryption scheme defined in Consteuction 8.3. Then, for any adversary A against $\mathcal{SE}_3$ running in time t and making at most q oracle queries totalling at most μ blocks, there is an adversary D against F running in time t and making at most q oracle queries totalling at most $\mu + 2q$ blocks such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}_3}(A) \leq 2 \cdot \mathbf{Adv}^{\text{oprp-cpa}}_{F}(D) + \frac{q(q-1)}{2^n}.$$

Above, the convention is that the length of a query M_0, M_1 made by A is $|M_0|$.

Claim 8.5. *Let $F\colon \mathrm{Keys}(F) \times D_{d,n} \to D_{d,n}$ be an n-on-line cipher, and let $\mathcal{SE}_3 = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the symmetric encryption scheme defined in Construction 8.3. Then, for any adversary B against $\mathcal{SE}_3$ running in time t making at most q_e queries totalling at most μ_e blocks and outputting a string of length at most μ_d blocks, there exists an adversary D against F running in time t making at most q_e forward queries totalling at most $\mu_e + 2q_e$ blocks and making one backward query totalling at most μ_d blocks such that*

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}_3}(B) \leq \mathbf{Adv}^{\text{oprp-cca}}_{F}(D) + \frac{\mu_e}{2^n}.$$

The proofs of the above claims follow the ideas of [8].

Proof of Claim 8.4. Adversary D has access to an oracle g which is either a random instance of the on-line cipher F or a random instance of $\mathsf{OPerm}_{n,d}$. First, D picks a random bit b. Then, for each encryption oracle query M_0, M_1 made by A, the adversary D chooses a random element $r \xleftarrow{\$} D_n$, queries $r\|M_b\|r$ to its own oracle g and forwards the reply to A. At some point A outputs its guess d. If $b = d$, then D returns 1. Otherwise, it returns 0. We have

$$\begin{aligned} \Pr\big[\, g \xleftarrow{\$} F \;:\; D^g = 1 \,\big] &= \Pr\big[\, \text{IND-CPA}^A_{\mathcal{SE}_3} \Rightarrow 1 \,\big], \\ \Pr\big[\, g \xleftarrow{\$} \mathsf{OPerm}_{n,d} \;:\; D^g = 1 \,\big] &\le \frac{1}{2} + \frac{q(q-1)}{2^{n+1}}. \end{aligned} \tag{31}$$

To justify (31), we observe that, if g is an ideal on-line cipher, then A cannot get any information about the bit b unless at least two of the random elements chosen by D happen to be the same. Subtracting the above equations, we get

$$\begin{aligned} \mathbf{Adv}^{\text{oprp-cpa}}_F(D) &\ge \Pr\big[\, \text{IND-CPA}^A_{\mathcal{SE}_3} \Rightarrow 1 \,\big] - \frac{1}{2} - \frac{q(q-1)}{2^{n+1}} \\ &= \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}_3}(A) - \frac{q(q-1)}{2^{n+1}} \end{aligned}$$

as claimed. □

Proof of Claim 8.5. An adversary D has access to oracles g, g^{-1} which are either a random instance of the on-line cipher F and its inverse or a random instance of $\mathsf{OPerm}_{n,d}$ and its inverse. For each encryption oracle query M made by B, the adversary D chooses a random element $r \xleftarrow{\$} D_n$, submits $r\|M\|r$ to its own oracle g and forwards the reply to B. When B finally outputs C, the adversary D checks whether C has been previously returned to B as an answer to an encryption oracle query. If so, D returns 0. If not, D queries C to its second oracle g^{-1}. Let X be the reply. If $|X| < 3n$ then D returns 0, otherwise D parses X as $r\|M\|r'$, where $|r| = |r'| = n$. If $r = r'$, then D returns 1, otherwise, it returns 0. We have

$$\Pr\big[\, g \xleftarrow{\$} F \;:\; D^{g,g^{-1}} = 1 \,\big] = \Pr\big[\, \text{INT-CTXT}^B_{\mathcal{SE}_3} \Rightarrow 1 \,\big], \tag{32}$$

$$\Pr\big[\, g \xleftarrow{\$} \mathsf{OPerm}_{n,d} \;:\; D^{g,g^{-1}} = 1 \,\big] \le \frac{\mu_e}{2^n}. \tag{33}$$

Subtracting, we have

$$\begin{aligned} \mathbf{Adv}^{\text{oprp-cpa}}_F(D) &\ge \Pr\big[\, \text{INT-CTXT}^B_{\mathcal{SE}_3} \Rightarrow 1 \,\big] - \frac{1}{2^n} \\ &= \mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(B) - \frac{1}{2^n} \end{aligned}$$

which yields the claim. We now justify the above equations. Equation (32) is clear. To justify (33), let $M_1, \ldots, M_{q_e}$ denote B's queries and $C_1, \ldots, C_{q_e}$ the responses. Let r_i be the randomness chosen in encrypting M_i, and let C denote the output of B. Let $x = g^{-1}(C)$ and parse x as $r\|M\|r'$. Let $l = \|C\|_n$ and $(s, p) = \mathsf{LCP}^*_n(C, C_1, \ldots, C_{q_e})$.

We consider two cases. If $p < l$, then the permutation applied to $C[l]$ when computing $g^{-1}(C)$ is at a tree node that has never been visited before, and so the result r' has probability 2^{-n} of equalling r. On the other hand, if $p = l$, then $C[l] = C_s[l]$. Now let $l_s = \|C_s\|_n$. If $C \neq C_s$ (otherwise D returns 0), it must be that $l_s \geq l + 1$ and thus $\|M_s\|_n \geq l - 1$. So $r' = M_s[l - 1]$. The latter was, however, chosen by B before r_s was chosen. □

We conjecture that the $\mu_e 2^{-n}$ term in the bound of Claim 8.5 can be reduced to 2^{-n} by a better analysis of our adversary D in the proof. We leave settling this as an open question.

Acknowledgements

We thank Anand Desai, Bogdan Warinschi, Phillip Rogaway and anonymous reviewers for their helpful comments.

M. Bellare was supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering, NSF grants CCR-0098123, CNS-0524765, and CNS-0627779, and a gift from Intel Corporation.

A. Boldyreva was supported in part by above-mentioned grants of first author.

C. Namprempre was supported in part by above-mentioned grants of first author and the Thailand Research Fund.

References

[1] G. Amanatidis, A. Boldyreva, A. O'Neill, New security models and provably-secure schemes for basic query support in outsourced databases, in *21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, ed. by S. Barker, G.-J. Ahn, California, USA, 8–11 July 2007. Lecture Notes in Computer Science (Springer, Berlin, 2007)

[2] G.V. Bard, A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL, in *SECRYPT 2006, Proceedings of the International Conference on Security and Cryptography*, ed. by M. Malek, E. Fernández-Medina, J. Hernando, Setúbal, Portugal, 7–10 August 2006 (INSTICC Press, Setubol, 2006), pp. 99–109

[3] G.V. Bard, Blockwise-adaptive chosen-plaintext attack and online modes of encryption, in *Cryptography and Coding*, vol. 4887, ed. by S. Galbraith (Springer, Berlin, 2007), pp. 129–151

[4] M. Bellare, A. Boldyreva, L.R. Knudsen, C. Namprempre, Online ciphers and the hash-CBC construction, in *Advances in Cryptology—CRYPTO 2001* ed. by J. Kilian, Santa Barbara, CA, USA, 19–23 August 2001. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 292–309

[5] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption, in *38th Annual Symposium on Foundations of Computer Science*, Miami Beach, Florida, 19–22 October 1997 (IEEE Computer Society, Los Alamitos, 1997), pp. 394–403

[6] M. Bellare, J. Kilian, P. Rogaway, The security of the cipher block chaining message authentication code, in *Advances in Cryptology—CRYPTO'94*, ed. by Y. Desmedt. Santa Barbara, CA, USA, 21–25 August 1994. Lecture Notes in Computer Science, vol. 839 (Springer, Berlin, 1994), pp. 341–358

[7] M. Bellare, C. Namprempre, Authenticated encryption: relations among notions and analysis of the generic composition paradigm, in *Advances in Cryptology—ASIACRYPT 2000*, ed. by T. Okamoto, Kyoto, Japan, 3–7 December 2000. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 531–545

[8] M. Bellare, P. Rogaway, Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography, in *Advances in Cryptology—ASIACRYPT 2000*, ed. by T. Okamoto, Kyoto, Japan, 3–7 December 2000. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 317–330

[9] M. Bellare, P. Rogaway, The security of triple encryption and a framework for code-based game-playing proofs, in *Advances in Cryptology—EUROCRYPT 2006*, ed. by S. Vaudenay, St. Petersburg, Russia, 28 May–1 June 2006. Lecture Notes in Computer Science, vol. 4004 (Springer, Berlin, 2006), pp. 409–426

[10] A. Boldyreva, N. Taesombut, Online encryption schemes: new security notions and constructions, in *Topics in Cryptology—CT-RSA 2004*, ed. by T. Okamoto, San Francisco, CA, USA, 23–27 February 2004. Lecture Notes in Computer Science, vol. 2964 (Springer, Berlin, 2004), pp. 1–14

[11] C. Campbell, Design and specification of cryptographic capabilities. National Bureau of Standards Special Publications 500-27, US Department of Commerce, February 1978

[12] P.-A. Fouque, A. Joux, G. Poupard, Blockwise adversarial model for on-line ciphers and symmetric encryption schemes, in *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, ed. by H. Handschuh, A. Hasan, Waterloo, Ontario, Canada, 9–10 August 2004. Lecture Notes in Computer Science, vol. 3357 (Springer, Berlin, 2004)

[13] O. Goldreich, S. Goldwasser, S. Micali, On the cryptographic applications of random functions, in *Advances in Cryptology—CRYPTO'84*, ed. by G.R. Blakley, D. Chaum, Santa Barbara, CA, USA, 19–23 August 1985. Lecture Notes in Computer Science, vol. 196 (Springer, Berlin, 1985), pp. 276–288

[14] S. Halevi, P. Rogaway, A tweakable enciphering mode, in *Advances in Cryptology—CRYPTO 2003*, ed. by D. Boneh, Santa Barbara, CA, USA, 17–21 August 2003. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 482–499

[15] S. Halevi, P. Rogaway, A parallelizable enciphering mode, in *Topics in Cryptology—CT-RSA 2004*, ed. by T. Okamoto, San Francisco, CA, USA, 23–27 February 2004. Lecture Notes in Computer Science, vol. 2964, (Springer, Berlin, 2004), pp. 292–304

[16] L. Knudsen, Block chaining modes of operation, in *Symmetric Key Block Cipher Modes of Operation Workshop*. http://csrc.nist.gov/encryption/modes/workshop1/ 29 October 2000

[17] H. Krawczyk, LFSR-based hashing and authenticating, in *Advances in Cryptology—CRYPTO'94*, ed. by Y. Desmedt, Santa Barbara, CA, USA, 21–25 August 1994. Lecture Notes in Computer Science, vol. 839 (Springer, Berlin, 1994), pp. 129–139

[18] M. Liskov, R.L. Rivest, D. Wagner, Tweakable block ciphers, in *Advances in Cryptology—CRYPTO 2002*, ed. by M. Yung, Santa Barbara, CA, USA, 18–22 August 2002. Lecture Notes in Computer Science, vol. 2442 (Springer, Berlin, 2002), pp. 31–46

[19] M. Luby, C. Rackoff, How to construct pseudo-random permutations from pseudo-random functions, in *Advances in Cryptology—CRYPTO'85*, ed. by H.C. Williams, Santa Barbara, CA, USA, 18–22 August 1985. Lecture Notes in Computer Science, vol. 218 (Springer, Berlin, 1985), p. 447

[20] U.M. Maurer, Indistinguishability of random systems, in *Advances in Cryptology—EUROCRYPT 2002*, ed. by L.R. Knudsen, Amsterdam, The Netherlands, 28 April–2 May 2002. Lecture Notes in Computer Science, vol. 2332 (Springer, Berlin, 2002), pp. 110–132

[21] C. Meyer, S. Matyas, *A New Dimension in Computer Data Security* (Wiley, New York, 1982)

[22] M. Nandi, A simple and unified method of proving indistinguishability, in *Progress in Cryptology—INDOCRYPT 2006*, ed. by R. Barua, T. Lange, Kolkata, India, 11–13 December 2006. Lecture Notes in Computer Science, vol. 4329 (Springer, Berlin, 2006), pp. 317–334

[23] M. Naor, O. Reingold, On the construction of pseudorandom permutations: Luby-Rackoff revisited. *J. Cryptol.* **12**(1), 29–66 (1999). doi:10.1007/PL00003817

[24] W. Nevelsteen, B. Preneel, Software performance of universal hash functions, in *Advances in Cryptology—EUROCRYPT'99*, ed. by J. Stern, Prague, Czech Republic, 2–6 May 2–6 1999. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 24–41

[25] P. Rogaway, H. Zhang, Online ciphers from tweakable blockciphers, in *Topics in Cryptology—CT-RSA 2011*, ed. by A. Kiayias, San Francisco, CA, USA, 14–18 February 2011. Lecture Notes in Computer Science, vol. 6558 (Springer, Berlin, 2011), pp. 237–249

[26] V. Shoup, On fast and provably secure message authentication based on universal hashing, in *Advances in Cryptology—CRYPTO'96*, ed. by N. Koblitz, Santa Barbara, CA, USA, 18–22 August 1996. Lecture Notes in Computer Science, vol. 1109 (Springer, Berlin, 1996), pp. 313–328