

A preliminary version of this paper appears in the proceedings of CRYPTO 2001. This is the full version.

On-Line Ciphers and the Hash-CBC Construction

M. BELLARE* A. BOLDYREVA† L. KNUDSEN‡ C. NAMPREMPRE§

June 13, 2001

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 i th 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 the HCBC1 construction, based on a given block cipher E and a family of AXU functions. HCBC1 is proven secure against chosen-plaintext attacks assuming that E is a PRP secure against chosen-plaintext attacks. We also look at some possible methods for achieving security against chosen-ciphertext attacks.

Keywords: Ciphers, pseudorandom permutations, universal hash functions, security proofs.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: <http://www-cse.ucsd.edu/users/mihir>. Supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering.

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: aboldre@cs.ucsd.edu. URL: <http://www-cse.ucsd.edu/users/aboldyre>. Supported in part by above-mentioned grants of first author.

‡Department of Informatics, PB 7800, N-5020 Bergen, Norway. E-Mail: lars@ramkilde.com. URL: <http://www.ramkilde.com>.

§Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: meaw@cs.ucsd.edu. URL: <http://www-cse.ucsd.edu/users/cnamprem>. Supported in part by above-mentioned grants of first author.

Contents

1	Introduction	3
1.1	On-line ciphers	3
1.2	A notion of security for on-line ciphers	3
1.3	Candidates for on-line ciphers	4
1.4	The Hash-CBC-1 on-line cipher and its security	4
1.5	Security against chosen ciphertext attacks	5
1.6	Usage and application of on-line ciphers	5
1.7	Related work	6
2	Definitions	6
3	On-line ciphers and their basic properties	7
4	Analysis of some candidate ciphers	11
4.1	CBC as an on-line cipher	11
4.2	ABC as an on-line cipher	12
5	Lemmas about AXU families	14
6	The Hash-CBC-1 cipher	16
6.1	Proof of Lemma 6.3	18
6.2	Proof of Lemma 6.4	19
7	A Chosen-Ciphertext Attack against HCBC1	24
8	The Hash-CBC-2 cipher	25
9	Usage of on-line ciphers	25
10	Acknowledgments	28
	References	28
A	Definitions for privacy and integrity of symmetric encryption schemes	29

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 $F: \{0, 1\}^k \times D \rightarrow D$ 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 some large value d , and is also divisible by n .) Matyas and Meyer refer to these as “general” ciphers [12].

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 $M = M[1] \dots M[l]$ to an instance $F(K, \cdot)$ of the cipher as a sequence of n -bit blocks, and similarly for the output ciphertext $F(K, M) = C[1] \dots C[l]$. Then, given the key K , for all i , it should be possible to compute output block $C[i]$ after having seen input blocks $M[1] \dots M[i]$. That is, $C[i]$ does not depend on blocks $i + 1, \dots, l$ of the plaintext.

An on-line cipher permits real-time, length-preserving encryption of a data stream without recourse to buffering, which can be attractive in some practical settings.

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 pseudorandom permutation (PRP), as defined by Luby and Rackoff [11]. 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 i th block of the ciphertext does not depend on blocks $i + 1, i + 2, \dots$ of the plaintext. This is necessary, since otherwise it would not be possible to output the i th 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 meet 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 i th output block of π depends only on the first i input blocks to π , and let $\text{OPerm}_{d,n}$ denote the set of all length-preserving permutations π on domain $D_{d,n}$. The rest is like for 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 $\text{OPerm}_{d,n}$. 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 the introduction of such a notion. However, finding appropriate balances between security and practical constraints is an impactful and active research endeavor where the goal is not necessarily to achieve some strong notion of security but to have the “best possible” security under given practical constraints, so that weaker notions of security are useful. Furthermore, we will see that in this case, even this weak primitive, if properly used, can provide strong security.

1.3 Candidates for on-line ciphers

To the best of our knowledge, the problem of designing on-line ciphers with security properties as strong as those required by our definition 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 chooses 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. Attacks demonstrating this are provided in Section 4.

We then consider the Accumulated Block Chaining (ABC) mode proposed by Knudsen in [9], which is a generalization of the Infinite Garble Extension mode proposed by Campbell [6]. 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 Section 4, we present attacks demonstrating that this is not a secure on-line cipher.

1.4 The Hash-CBC-1 on-line cipher and its security

We seek a construction of a secure on-line cipher based on a given block cipher $E: \{0,1\}^{ek} \times \{0,1\}^n \rightarrow \{0,1\}^n$. We provide a construction called HCBC1 that uses a family $H: \{0,1\}^{hk} \times \{0,1\}^n \rightarrow \{0,1\}^n$ of Almost-XOR-Universal (AXU) hash functions [10]. The key $eK \| hK$ for an instance $\text{HCBC1}(eK \| hK, \cdot)$ of the cipher 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 .) A picture is in Figure 3, and a full description of the construction is in Section 6. It is easy to see that this cipher is on-line.

We stress that the hash functions map n bits to n bits, meaning work on inputs of the block length, as does the given block cipher. Numerous designs of fast AXU families are known, so that our construction is quite efficient. For an overview of the state-of-the-art of AXU families refer to [14].

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. The proof involves finding and exploiting a way of looking at an on-line cipher as a 2^n -ary tree of permutations on n bits, and then going through a hybrid argument involving a sequence of different games that “move” from $\text{OPerm}_{d,n}$ to HCBC1.

1.5 Security against chosen ciphertext attacks

HCBC1 meets the notion of being an on-line PRP under chosen-plaintext attack. It is not secure against chosen-ciphertext attack. We consider the question of finding an on-line PRP secure against chosen-ciphertext attack. (This is the analogue, for on-line-PRPs, of the notion of a PRP secure against chosen-ciphertext attacks. The latter was called a strong PRP in [13] and a super-PRP in [11]. The adversary has an oracle not just for the challenge permutation, but also for its inverse.) We consider a natural extension of HCBC1 which we call HCBC2. It involves two hashings, and one block cipher application, per message block. We show that there are examples of AXU families H for which HCBC2 is insecure against chosen-ciphertext attack. This attack illustrates the power of CCA attacks and highlights the types of issues they raise. We know of no attack if the family H is PRF, but neither do we have a proof of security under this assumption.

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 or legacy code.

However, an on-line cipher is more generally useful, via the “encode-then-encipher” paradigm discussed in [5]. 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 [5] remain true even if the cipher is an on-line-PRP rather than a PRP. Specifically, the requirement on the message space must be strengthened to require not just that entropy be present, but that it be in the first blocks of the message; and similarly, that redundancy not just be present, but be at the end of the data. Again, one might already have data of such structure, in which case the encryption will be length preserving yet provide semantic security and integrity, or one can prepend a random number and append a constant to the message, getting the same properties but at the cost of data expansion.

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 Hash-CBC constructions is that of building a general cipher from a block cipher. Naor and Reingold [13] 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 [13, Section 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 [13] in using hash functions in combination with block ciphers.

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 [11, 13]. We are, however, interested in allowing inputs of varying and very large size, not merely twice the block size.

2 Definitions

We recall basic definitions of families of functions and ciphers following [3].

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 . If $x, y \in \{0, 1\}^*$ are strings, then we denote by $\text{LCP}_n(x, y)$ the *longest common n -prefix* of x, y . This is the longest string s such that $|s|$ is a multiple of n , and s is a prefix of both x and y . A map $f: D \rightarrow R$ is a *permutation* if $D = R$ and f is a bijection (i.e. one-to-one and onto). A map $f: D \rightarrow R$ is *length-preserving* if $|f(x)| = |x|$ for all $x \in D$. If $n \geq 1, d \geq 1$ are integers, then $D_{d,n}$ denotes the set of all strings whose length is a positive multiple of n bits and at most dn bits. If $P \in D_{d,n}$, then $P[i]$ denotes its i th block, meaning $P = P[1] \dots P[l]$ where $l = |P|/n$ and $|P[i]| = n$ for all $i = 1, \dots, l$. 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 i th block of $f(M)$. (Or ε if $|f(M)| < ni$.)

FUNCTION FAMILIES AND CIPHERS. A *family of functions* is a map $F: \text{Keys}(F) \times \text{Dom}(F) \rightarrow \text{Ran}(F)$ where $\text{Keys}(F)$ is the *key space* of F ; $\text{Dom}(F)$ is the *domain* of F ; and $\text{Ran}(F)$ is the *range* of F . If $\text{Keys}(F) = \{0, 1\}^k$, then we refer to k as the *key-length*. The two-input function F takes a key $K \in \text{Keys}(F)$ and an input $x \in \text{Dom}(F)$ to return a point $F(K, x) \in \text{Ran}(F)$. For each key $K \in \text{Keys}(F)$, we define the map $F_K: \text{Dom}(F) \rightarrow \text{Ran}(F)$ by $F(K, x)$ for all $x \in \text{Dom}(F)$. Thus, F specifies a collection of maps from $\text{Dom}(F)$ to $\text{Ran}(F)$, 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 $K \stackrel{R}{\leftarrow} \text{Keys}(F)$. We write $f \stackrel{R}{\leftarrow} F$ for the operation $K \stackrel{R}{\leftarrow} \text{Keys}(F); f \leftarrow F(K, \cdot)$. That is, $f \stackrel{R}{\leftarrow} F$ 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 $\text{Rand}_{n,n}$ be the family of all functions mapping $\{0, 1\}^n$ to $\{0, 1\}^n$ so that $f \stackrel{R}{\leftarrow} \text{Rand}_{n,n}$ denotes the operation of selecting at random a function from $\{0, 1\}^n$ to $\{0, 1\}^n$. Similarly, let Perm_n be the family of all permutations mapping $\{0, 1\}^n$ to $\{0, 1\}^n$ so that $\pi \stackrel{R}{\leftarrow} \text{Perm}_n$ denotes the operation of selecting at random a permutation on $\{0, 1\}^n$. We say that F is a *cipher* if $\text{Dom}(F) = \text{Ran}(F)$ and each instance $F(K, \cdot)$ of F is a length-preserving permutation. A *block cipher* is a cipher whose domain and range equal $\{0, 1\}^n$ 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 $K \in \text{Keys}(F)$ and $x \in \text{Dom}(F)$.

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 [7, 11]. A *distinguisher* is an algorithm that has access to one or more oracles and outputs a bit. Let $F: \text{Keys}(F) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a family of functions with domain and range $\{0, 1\}^n$. Let A_1 be a distinguisher with one oracle and A_2 a distinguisher with two oracles. Let

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

If $F: \text{Keys}(F) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a cipher, then we also let

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

These capture the *advantage* of the distinguisher in question in the task of distinguishing a random instance of F from a random permutation on D . In the first case, the distinguisher gets to query the challenge instance. In the second, it also gets to query the inverse of the challenge instance. For any integers $t, q_e, q_d, \mu_e, \mu_d$, we now let

$$\begin{aligned} \mathbf{Adv}_F^{\text{prp-cpa}}(t, q_e, \mu_e) &= \max_{A_1} \left\{ \mathbf{Adv}_F^{\text{prp-cpa}}(A_1) \right\} \\ \mathbf{Adv}_F^{\text{prp-cca}}(t, q_e, \mu_e, q_d, \mu_d) &= \max_{A_2} \left\{ \mathbf{Adv}_F^{\text{prp-cca}}(A_2) \right\} . \end{aligned}$$

The maximum is over all distinguishers having time-complexity t , making to the g oracle at most q_e queries totaling at most μ_e bits, and, in the second case, also making to the g^{-1} oracle at most q_d queries totaling at most μ_d bits. We say that a PRP F is *secure against chosen-plaintext attacks* if the function $\mathbf{Adv}_F^{\text{prp-cpa}}(t, q_e)$ grows “slowly.” Similarly, we say that a PRP F is *secure against chosen-ciphertext attacks* if the function $\mathbf{Adv}_F^{\text{prp-cca}}(t, q_e, q_d)$ grows “slowly.” Time complexity includes the time to reply to oracle calls by computation of $F(K, \cdot)$ or $F(K, \cdot)^{-1}$.

3 On-line ciphers and their basic properties

We say that a function $f: D_{d,n} \rightarrow D_{d,n}$ is *n-on-line* if the i -th block of the output is determined completely by the first i blocks of the input. A more formal definition follows. We refer the reader to Section 2 for the definition of $f^{(i)}$.

Definition 3.1 Let $n, d \geq 1$ be integers, and let $f: D_{d,n} \rightarrow D_{d,n}$ be a length-preserving function. We say that f is *n-on-line* if there exists a function $X: D_{d,n} \rightarrow \{0, 1\}^n$ such that for every $M \in D_{d,n}$ and every $i \in \{1, \dots, |M|/n\}$ it is the case that

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

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

An equivalent characterization can be made in terms of the longest common prefix function defined in Section 2, as follows.

Proposition 3.2 Let $n, d \geq 1$ be integers, and let $f: D_{d,n} \rightarrow D_{d,n}$ be a length-preserving function. Then f is *n-on-line* if and only if the following is true:

$$|\text{LCP}_n(f(x), f(y))| \geq |\text{LCP}(x, y)| . \tag{2}$$

for every $x, y \in D_{d,n}$.

Definition 3.3 Let f be an *n-on-line* function. Let $i \geq 1$. Fix $M[1], \dots, M[i-1] \in \{0, 1\}^n$. Define the function $\Pi_{M[1] \dots M[i-1]}^f: \{0, 1\}^n \rightarrow \{0, 1\}^n$ by

$$\Pi_{M[1] \dots M[i-1]}^f(x) = f^{(i)}(M[1] \dots M[i-1]x)$$

for all $x \in \{0, 1\}^n$. ■

Proposition 3.4 If f is an n -on-line permutation, $i \geq 1$ and $M[1], \dots, M[i-1] \in \{0, 1\}^n$, then the map $\Pi_{M[1]\dots M[i-1]}^f$ is a permutation on $\{0, 1\}^n$.

Proof of Proposition 3.4: Let $M' = M[1] \dots M[i-1]$, and let x, y be distinct n -bit strings. We claim that

$$\Pi_{M'}^f(x) \neq \Pi_{M'}^f(y). \quad (3)$$

Since $\Pi_{M'}^f$ maps $\{0, 1\}^n$ to $\{0, 1\}^n$, this implies that it is a permutation. We now proceed to establish Equation (3).

Since f is n -on-line, we may fix a function $X: D_{d,n} \rightarrow \{0, 1\}^n$ meeting the conditions of Definition 3.1. Let $M_x = M' \| x$ and $M_y = M' \| y$. Applying f to M_x , we get

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

Similarly,

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

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 $j = 1, \dots, i-1$, the j -th output block of $f(M_x)$ and the j -th output block of $f(M_y)$ are equal. So it must be that the i -th blocks of the outputs are unequal, meaning $X(M_x) \neq X(M_y)$. Finally, we observe that

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

so Equation (3) is established. ■

PSEUDORANDOMNESS OF ON-LINE CIPHERS. Let $\text{OPerm}_{d,n}$ denote the family of all n -on-line, length-preserving permutations on $D_{d,n}$. A “secure” on-line cipher is one that closely approximates $\text{OPerm}_{d,n}$; 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 $F: \text{Keys}(F) \times D_{d,n} \rightarrow D_{d,n}$ be a family of functions with domain and range $D_{d,n}$. Let A_1 be a distinguisher with one oracle and A_2 a distinguisher with two oracles. Let

$$\mathbf{Adv}_F^{\text{oprpr-cpa}}(A_1) = \Pr \left[g \stackrel{R}{\leftarrow} F : A_1^g = 1 \right] - \Pr \left[g \stackrel{R}{\leftarrow} \text{OPerm}_{d,n} : A_1^g = 1 \right].$$

If $F: \text{Keys}(F) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a cipher, then we also let

$$\mathbf{Adv}_F^{\text{oprpr-cca}}(A_2) = \Pr \left[g \stackrel{R}{\leftarrow} F : A_2^{g, g^{-1}} = 1 \right] - \Pr \left[g \stackrel{R}{\leftarrow} \text{OPerm}_{d,n} : A_2^{g, g^{-1}} = 1 \right].$$

These capture the *advantage* of the distinguisher 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 distinguisher gets to query the challenge instance. In the second, it also gets to query the inverse of the challenge instance. For any integers $t, q_e, \mu_e, q_d, \mu_d$, we now let

$$\begin{aligned} \mathbf{Adv}_F^{\text{oprpr-cpa}}(t, q_e, \mu_e) &= \max_{A_1} \left\{ \mathbf{Adv}_F^{\text{oprpr-cpa}}(A_1) \right\} \\ \mathbf{Adv}_F^{\text{oprpr-cca}}(t, q_e, \mu_e, q_d, \mu_d) &= \max_{A_2} \left\{ \mathbf{Adv}_F^{\text{oprpr-cca}}(A_2) \right\}. \end{aligned}$$

The maximum is over all distinguishers having time-complexity t , making to the oracle g at most q_e queries totaling at most μ_e bits, and, in the second case, also making to the g^{-1} oracle at most q_d queries totaling at most μ_d bits. We say that an online PRP (OPRP) F is secure against chosen plaintext attacks if the function $\mathbf{Adv}_F^{\text{oprp-cpa}}(t, q_e, \mu_e)$ grows “slowly.” Similarly, we say that an OPRP F is secure against chosen ciphertext attacks if the function $\mathbf{Adv}_F^{\text{oprp-cca}}(t, q_e, \mu_e, q_d, \mu_d)$ grows “slowly.” Time complexity includes the time to reply to oracle calls by computation of $F(K, \cdot)$ or $F(K, \cdot)^{-1}$.

TREE-BASED CHARACTERIZATION. We present a tree-based characterization of n -on-line ciphers that is useful to gain intuition and to analyze constructs. Let $N = 2^n$. An N -ary tree of functions is an N -ary tree T each node of which is labeled by a function mapping $\{0, 1\}^n$ to $\{0, 1\}^n$. We label each edge in the tree in a natural way via a string in $\{0, 1\}^n$. 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 $0 \leq i \leq d$, 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} .

$$\begin{array}{l|l}
 T(M[1] \dots M[l]) & T^{-1}(C[1] \dots C[l]) \\
 x \leftarrow \varepsilon & x \leftarrow \varepsilon \\
 \text{For } i = 1, \dots, l \text{ do} & \text{For } i = 1, \dots, l \text{ do} \\
 \quad C[i] \leftarrow T_x(M[i]) & \quad M[i] \leftarrow T_x^{-1}(C[i]) \\
 \quad x \leftarrow x \| C[i] & \quad x \leftarrow x \| C[i] \\
 \text{EndFor} & \text{EndFor} \\
 \text{Return } C[1] \dots C[l] & \text{Return } M[1] \dots M[l]
 \end{array}$$

Here, $1 \leq l \leq d$. Let $G : \text{Keys}(G) \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a function family. (We are most interested in the case where G is Perm_n or $\text{Rand}_{n,n}$.) We let $\text{Tree}(n, G, d)$ 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 3.5 There is a bijection between $\text{Tree}(n, \text{Perm}_n, d)$ and $\text{OPerm}_{d,n}$.

Proof of Proposition 3.5: We specify a map $Z: \text{OPerm}_{n,d} \rightarrow \text{Tree}(n, \text{Perm}_n, d)$ and then argue that it is a bijection. Given $f \in \text{OPerm}_{n,d}$, the map Z returns the tree $T_f = Z(f) \in \text{Tree}(n, \text{Perm}_n, d)$ defined as follows: for any $l = 0, \dots, d - 1$ and any $M[1] \dots M[l] \in D_{d,n}$, $C = f(M)$, node $C[1] \dots C[l]$ of tree T_f is labeled by the permutation $\Pi_{M[1] \dots M[l]}^f$. Equivalently, T_f can be defined as the function which, for any $l = 1, \dots, d$ and any input $M[1] \dots M[l] \in D_{d,n}$, works as follows:

$$\begin{array}{l}
 T_f(M[1] \dots M[l]) \\
 \quad \text{For } i = 1, \dots, l \text{ do} \\
 \quad \quad C[i] \leftarrow \Pi_{M[1] \dots M[i-1]}^f(M[i]) \\
 \quad \text{EndFor} \\
 \quad \text{Return } C[1] \dots C[l]
 \end{array}$$

Proposition 3.4 implies that the functions labeling the nodes of the tree are indeed in Perm_n , so $T_f \in \text{Tree}(n, \text{Perm}_n, d)$. 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 $C[1] \dots C[l]$ in T_f is $\Pi_{M[1] \dots M[l]}^f$ in T_f and $\Pi_{M[1] \dots M[l]}^g$ in T_g . The assumption $T_f = T_g$ also implies that $\Pi_{M[1] \dots M[l]}^f = \Pi_{M[1] \dots M[l]}^g$ for all $l = 0, \dots, d-1$ and all $M[1] \dots M[l] \in D_{d,n}$. By Definition 3.3 we have

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

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

Next we show that Z is surjective. Let $T \in \text{Tree}(n, \text{Perm}_n, d)$. We need to show that there exists an $f \in \text{OPerm}_{n,d}$ such that $T_f = T$. We simply let f be the function defined by T . It is clear from the definition that it is n -on-line, and since the inverse function T^{-1} is also defined, is a permutation. \blacksquare

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

Proposition 3.6 Let $f: D_{d,n} \rightarrow D_{d,n}$ be an n -on-line permutation, and let $g = f^{-1}$. Then g is an n -on-line permutation.

Proof of Proposition 3.6: 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 $i \leq |C|/n$, we have

$$g^{(i)}(C) = Y(C[1] \dots C[i]). \quad (4)$$

We define $Y: D_{d,n} \rightarrow \{0, 1\}^n$ as follows for any $i = 1, \dots, d$ and any input $C[1] \dots C[i] \in D_{d,n}$:

```

Y(C[1] ... C[i])
  M[0] ← ε
  For j = 1, ..., i do
    M[j] ← (ΠM[0]...M[j-1]f)-1(C[j])
  EndFor
  Return M[i].

```

Here we used Proposition 3.4 and the assumption that f is n -on-line to guarantee that the inverse of $\Pi_{M[0] \dots M[j-1]}^f$ is well-defined. Now, suppose $C \in D_{d,n}$ and $1 \leq i \leq |C|/n$. We prove that Equation (4) holds. Letting $M = g(C)$ we have

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

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 $j = 1, \dots, i$. The fourth line is by Proposition 3.4. The fifth line follows by applying the definition of Y . The sixth line is because $M = g(C)$. \blacksquare

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 3.6 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.

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 [9], which is a generalization of the Infinite Garble Extension mode proposed by Campbell [6]. In this section, we let $E: \{0, 1\}^{ek} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ 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 *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 $C[0] \in \{0, 1\}^n$, we define

$$\begin{array}{l} \text{OCBC}(eK \| C[0], M) \\ \text{Parse } M \text{ as } M[1] \dots M[l] \text{ with } l \geq 1 \\ \text{For } i = 1, \dots, l \text{ do} \\ \quad C[i] \leftarrow E(eK, M[i] \oplus C[i-1]) \\ \text{Return } C[1] \dots C[l] \end{array} \quad \left| \quad \begin{array}{l} \text{OCBC}^{-1}(eK \| C[0], C) \\ \text{Parse } C \text{ as } C[1] \dots C[l] \text{ with } l \geq 1 \\ \text{For } i = 1, \dots, l \text{ do} \\ \quad M[i] \leftarrow E^{-1}(eK, C[i] \oplus C[i-1]) \\ \text{Return } M[1] \dots M[l] \end{array} \right.$$

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 Figure 1 gets an oracle g where g is either an instance of OCBC or an instance of $\text{OPerm}_{d,n}$. 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 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

$$\mathbf{Adv}_{\text{OCBC}}^{\text{odrp-cpa}}(A) \geq 1 - 2^{-n}. \quad (5)$$

We now justify Equation (5). We claim that

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

from which Equation (5) from Section 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

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

Distinguisher A^g

Let $M[2], \dots, M[l]$ be any n -bit strings
 Let $M_1 = 0^n M[2] \dots M[l]$ and let $M_2 = 1^n M[2] \dots M[l]$
 Let $C_1[1] \dots C_1[l] \leftarrow g(M_1)$ and let $C_2[1] \dots C_2[l] \leftarrow g(M_2)$
 Let $M_3[2] = M[2] \oplus C_1[1] \oplus C_2[1]$ and let $M_3 = 1^n M_3[2] M[3] \dots M[l]$
 Let $C_3[1] \dots C_3[l] \leftarrow g(M_3)$
 If $C_3[2] = C_1[2]$ then return 1 else return 0

Figure 1: Attack on the CBC based on-line cipher.

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 $\text{OPerm}_{n,d}$. 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 $\text{OPerm}_{n,d}$. 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 $\text{OPerm}_{n,d}$. Therefore, the distinguisher A^g outputs 1 with the probability at most $2 \cdot 2^{-n}$, and this justifies the second equation.

Since A made only 3 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 [9] 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 $P[0], C[0] \in \{0, 1\}^n$ and also by a public function $h: \{0, 1\}^n \rightarrow \{0, 1\}^n$. (Instantiations for h suggested in [9] 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

EABC($eK, P[0], C[0], M$) Parse M as $M[1] \dots M[l]$ with $l \geq 1$ For $i = 1, \dots, l$ do $P[i] \leftarrow M[i] \oplus h(P[i-1])$ $C[i] \leftarrow E(eK, P[i] \oplus C[i-1])$ $\oplus P[i-1]$ EndFor Return $C[1] \dots C[l]$	DABC($eK, P[0], C[0], C$) Parse C as $C[1] \dots C[l]$ with $l \geq 1$ For $i = 1, \dots, l$ do $P[i] \leftarrow E^{-1}(eK, C[i] \oplus P[i-1])$ $\oplus C[i-1]$ $M[i] \leftarrow P[i] \oplus h(P[i-1])$ EndFor Return $M[1] \dots M[l]$
--	---

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,

Distinguisher A^g

Let $M[2], \dots, M[l]$ be any n -bit strings
 Let $M_1 = 0^n M[2] \dots M[l]$ and let $M_2 = 1^n M[2] \dots M[l]$
 Let $C_1[1] \dots C_1[l] \leftarrow g(M_1)$ and let $C_2[1] \dots C_2[l] \leftarrow g(M_2)$
 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 $M_3 = 1^n M_3[2] M[3] \dots M[l]$
 Let $C_3[1] \dots C_3[l] \leftarrow g(M_3)$
 If $C_3[2] = C_1[2] \oplus 1^n$, then return 1 else return 0

Figure 2: Attack on the ABC based on-line cipher.

has key space $\{0, 1\}^k$ and domain and range $D_{d,n}$. We fix values $P[0], C[0] \in \{0, 1\}^n$ which are known to all parties including the adversary. We then define the cipher and the inverse cipher as follows:

$$\begin{array}{l} \text{PABC}(eK, M) \\ \text{Return EABC}(eK, P[0], C[0], M) \end{array} \quad \left| \quad \begin{array}{l} \text{PABC}^{-1}(eK, C) \\ \text{Return DABC}(eK, P[0], C[0], C) \end{array} \right.$$

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 as follows:

$$\begin{array}{l} \text{SABC}(eK \| P[0] \| C[0], M) \\ \text{Return EABC}(eK, P[0], C[0], M) \end{array} \quad \left| \quad \begin{array}{l} \text{SABC}^{-1}(eK \| P[0] \| C[0], C) \\ \text{Return DABC}(eK, P[0], C[0], C) \end{array} \right.$$

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 Figure 2. The adversary A gets an oracle g where g is either an instance of PABC or an instance of $\text{OPerm}_{d,n}$. The adversary can mount this attack because the function h as well as the value $P[0]$ are public. We claim that

$$\mathbf{Adv}_{\text{PABC}}^{\text{oprp-cpa}}(A) \geq 1 - 2 \cdot 2^{-n}. \quad (6)$$

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 Equation (6). We claim that

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

from which Equation (6) follows. We justify these two claims below. First, suppose g is an instance of PABC, namely $g(\cdot) = \text{PABC}(eK, \cdot)$. Since the first block of x_3 is 1^n , we have

$$\begin{aligned} C_3[2] &= E(eK, P_3[2] \oplus C_3[1]) \oplus P_3[1] \\ &= E(eK, x_3[2] \oplus h(P_3[1]) \oplus C_3[1]) \oplus P_3[1] \\ &= E(eK, x_3[2] \oplus h(P_2[1]) \oplus C_2[1]) \oplus P_2[1] \\ &= E(eK, x_3[2] \oplus h(1^n \oplus h(P[0])) \oplus C_2[1]) \oplus P_2[1] \\ &= E(eK, x[2] \oplus C_1[1] \oplus h(0^n \oplus h(P[0]))) \oplus P_2[1] \\ &= E(eK, x[2] \oplus C_1[1] \oplus h(P_1[1])) \oplus P_2[1] \\ &= E(eK, P_1[2] \oplus C_1[1]) \oplus P_2[1] \end{aligned}$$

$$\begin{aligned}
&= (C_1[2] \oplus P_1[1]) \oplus P_2[1] \\
&= C_1[2] \oplus (0^n \oplus h(P[0])) \oplus (1^n \oplus h(P[0])) \\
&= C_1[2] \oplus 1^n .
\end{aligned}$$

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 $\text{OPerm}_{n,d}$. We claim that this event has low probability, namely $2 \cdot 2^{-n}$. The reason is similar to that in Section 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 $x_3[2] = x[2]$. This means that the attack is invalid since $x_3 = x_2$. Second, if $x_3[2] \neq x[2]$, 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 $\text{OPerm}_{n,d}$. Upper-bounding the aggregate probability, we obtain $2 \cdot 2^{-n}$ and Equation (6) 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 [9]. Specifically, let us say that a function $h: \{0, 1\}^n \rightarrow \{0, 1\}^n$ is *linear* if $h(x \oplus y) = h(x) \oplus h(y)$ for all $x, y \in \{0, 1\}^n$. (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

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

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 $H: \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ 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 Lemmas about AXU families

Our constructions of on-line ciphers will use the families of AXU (Almost Xor Universal) functions as defined by Krawczyk [10]. We recall the definition, and then prove some lemmas that will be helpful in our analyses.

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

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

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

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 $\mathbf{Adv}_H^{\text{axu}} \leq \epsilon$. We find the “advantage function” based notation more convenient, and more consistent with the rest of our security definitions.

The definition is information-theoretic, talking of the maximum value of some probability. We will find it convenient to think in terms of an adversary attacking the scheme, and will use the following lemma. We stress that below there are no limits on the running time of the adversary. This lemma is standard, and follows easily from Definition 5.1, so we omit the proof.

Lemma 5.2 Let $n, hk \geq 1$ be integers, and let $H: \{0, 1\}^{hk} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a family of functions. Let A be any possibly probabilistic algorithm that takes no inputs and returns a triple (x_1, x_2, y) of n -bit strings. Then

$$\Pr \left[(x_1, x_2, y) \stackrel{R}{\leftarrow} A ; K \stackrel{R}{\leftarrow} \{0, 1\}^{hk} : H(K, x_1) \oplus H(K, x_2) = y \right] \leq \mathbf{Adv}_H^{\text{axu}}.$$

In the formulation of Lemma 5.2, it is important that the adversary is constrained to pick x_1, x_2, y before the K is chosen. In our upcoming analyses, we will, in contrast, be considering an adversary that obtains some partial information regarding $H(K, \cdot)$ in the course of its search for a certain kind of “collision,” and uses this to guide its search. Specifically, our adversary B can be viewed as having access to an oracle that knows a key K . The adversary functions in stages. In stage i , it produces a pair (x_i, y_i) of values which it submits to the oracle. The latter responds with a bit indicating whether or not there exists some $j \in \{1, \dots, i-1\}$ such that $H(K, x_j) \oplus H(K, x_i) = y_j \oplus y_i$. (The oracle is stateful because it has to remember the adversary queries from previous stages in order to be able to answer the current query.) We wish to argue that the partial information about $H(K, \cdot)$ that is obtained by the adversary via this process is not too large. Specifically, we argue that the probability that the adversary ever gets back a positive response from the oracle is $O(q^2) \cdot \mathbf{Adv}_H^{\text{axu}}$.

In the formal definition that follows, we first describe an algorithm that serves as a stateful oracle discussed above. Then, we describe an experiment in which the adversary B with oracle access to the algorithm is executed.

Definition 5.3 Let $H: \{0, 1\}^{hk} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a family of hash functions, and let hK be a string of length hk . We define the following stateful algorithm D . It maintains a counter i and arrays X, Y , and takes n -bit strings x, y as inputs. Then, we let B be an adversary with oracle access to D_{hK} and define an experiment in which B executes.

Algorithm $D_{hK}(x, y)$

$i \leftarrow i + 1 ; r \leftarrow 0 ; X[i] \leftarrow x ; Y[i] \leftarrow y$

For $j = 1, \dots, i-1$ do

 If $(H(hK, X[j]) \oplus Y[j] = H(hK, X[i]) \oplus Y[i])$ and $(X[j] \neq X[i])$ then $r \leftarrow j$

EndFor

Return r

Experiment $\mathbf{Exp}_H^{\text{axu-cr}}(B)$

$hK \stackrel{R}{\leftarrow} \{0, 1\}^{hk}$

Initialize D_{hK} with $i = 0$ and X, Y empty

Run $B^{D_{hK}(\cdot, \cdot)}$ until it halts

If B made some oracle query that received a non-zero response,

 then return 1, else return 0.

We define the *advantage* of the adversary B and the *AXU-Collision advantage function* of H as follows. For any integer q ,

$$\mathbf{Adv}_H^{\text{axu-cr}}(B) = \Pr[\mathbf{Exp}_H^{\text{axu-cr}}(B) = 1]$$

$$\mathbf{Adv}_H^{\text{axu-cr}}(q) = \max_B \{ \mathbf{Adv}_H^{\text{axu-cr}}(B) \}$$

where the maximum is taken over all adversaries making q queries. ■

The following lemma states the relationship between Definition 5.1 and Definition 5.3.

Lemma 5.4 Let $H: \{0, 1\}^{hk} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a family of hash functions. Then,

$$\mathbf{Adv}_H^{\text{axu-cr}}(q) \leq q(q-1)\mathbf{Adv}_H^{\text{axu}}.$$

Proof of Lemma 5.4: Let B be an adversary with advantage $\mathbf{Adv}_H^{\text{axu-cr}}(q)$. We will define an adversary A to which we can apply Lemma 5.2. The adversary A is depicted below. It runs B , and simulates the oracle D_{hK} by guessing the indices l, m where a collision occurs.

Adversary A

Choose l, m at random subject to $1 \leq l < m \leq q$

Run B

For $i = 1, \dots, m-1$ do

 Upon receiving a query (x_i, y_i) , answer with 0.

For $i = m, \dots, q$

 Upon receiving a query (x_i, y_i) from B , answer with l

Return $(x_l, x_m, y_l \oplus y_m)$

Thus, A returns 0 to B until the m -th query, and returns l to it from then on. We know that, in the experiment $\mathbf{Exp}_H^{\text{axu-cr}}(B)$, the adversary B receives a non-zero response to one of its q queries with probability $\mathbf{Adv}_H^{\text{axu-cr}}(B)$. Let j be the smallest integer such that B receives a non-zero response to its j -th query, and let i denote the response received, where $1 \leq i < j$. Then we know that $H(hK, x_i) \oplus H(hK, x_j) = y_i \oplus y_j$. We consider the event that A guesses these values, meaning $l = i$ and $m = j$. In that case, the output $(x_l, x_m, y_l \oplus y_m)$ satisfies $H(hK, x_l) \oplus H(hK, x_m) = y_l \oplus y_m$, and the simulation of B 's oracle is correct until the m -th query. (We note that the simulation of B 's oracle is not correct after the m -th query, since A always returns l , but this does not matter, since A has already accomplished its task.) We let “ A succeeds” denote the event that A 's output satisfies this equation. The probability that $(l, m) = (i, j)$ is $1/[q(q-1)]$ so

$$\Pr[A \text{ succeeds}] \geq \frac{1}{q(q-1)} \cdot \mathbf{Adv}_H^{\text{axu-cr}}(B).$$

Now Lemma 5.2 implies that

$$\Pr[A \text{ succeeds}] \leq \mathbf{Adv}_H^{\text{axu}}.$$

Putting these two inequalities together, we have

$$\mathbf{Adv}_H^{\text{axu-cr}}(q) \leq q(q-1) \cdot \mathbf{Adv}_H^{\text{axu}}$$

as claimed. ■

6 The Hash-CBC-1 cipher

In this section, we suggest a construction of an on-line cipher. We call it HCBC1 and prove its security against chosen-plaintext attacks. This construction is similar to the CBC mode of encryption. The only difference is that each output block passes through a keyed hash function before getting exclusive-or-ed with the next input block. The key of the hash function is kept secret. In the next section, we extend this to HCBC2 and prove its security against chosen-ciphertext attacks.

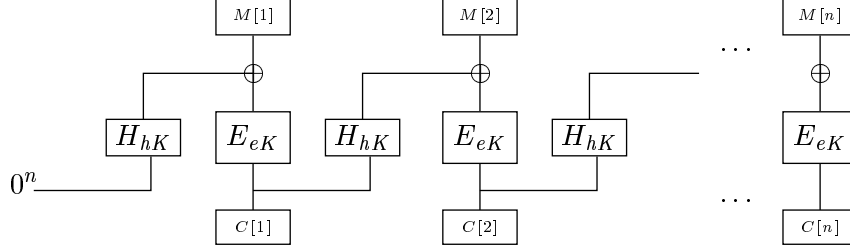


Figure 3: The HCBC1 cipher.

Construction 6.1 Let $n, d \geq 1$ be integers, and let $E: \{0, 1\}^{ek} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher. Let $H: \{0, 1\}^{hk} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a family of hash functions. We associate to them a cipher HCBC1: $\{0, 1\}^{ek+hk} \times D_{d,n} \rightarrow D_{d,n}$. 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 3 illustrates the cipher.

<p>HCBC1($eK \ hK, M$) Parse M as $M[1] \dots M[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow H(hK, C[i-1]) \oplus M[i]$ $C[i] \leftarrow E(eK, P[i])$ EndFor Return $C[1] \dots C[l]$</p>	<p>HCBC1$^{-1}(eK \ hK, C)$ Parse C as $C[1] \dots C[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow E^{-1}(eK, C[i])$ $M[i] \leftarrow H(hK, C[i-1]) \oplus P[i]$ EndFor Return $M[1] \dots M[l]$ ■</p>
---	---

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

Theorem 6.2 Let $E: \{0, 1\}^{ek} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher, and let $H: \{0, 1\}^{hk} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ 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 integers $t, q_e, \mu_e \geq 0$ such that $\mu_e \leq n2^{n-1}$, we have

$$\mathbf{Adv}_{\text{HCBC1}}^{\text{oprp-cpa}}(t, q_e, \mu_e) \leq \mathbf{Adv}_E^{\text{prp-cpa}}(t, \mu_e/n, \mu_e) + \left(\frac{\mu_e^2 - n\mu_e}{n^2} \right) \cdot \mathbf{Adv}_H^{\text{axu}} + \frac{\mu_e^2 + 2n(q_e + 1)\mu_e}{n^2 \cdot 2^n}.$$

The rest of this section is devoted to a proof of Theorem 6.2. We introduce the following definition to facilitate our discussion. This definition simply describes a cipher similar to that defined by Construction 6.1 except that a permutation π is used in place of a permutation drawn from E .

Let $\pi: \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a permutation. We associate to it a cipher HCBC1 $_{\pi}$: $\{0, 1\}^{hk} \times D_{d,n} \rightarrow D_{d,n}$. It is defined as follows for $M \in D_{d,n}$.

$\text{HCBC1}_\pi(hK, M)$ Parse M as $M[1] \dots M[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow H(hK, C[i-1]) \oplus M[i]$ $C[i] \leftarrow \pi(P[i])$ EndFor Return $C[1] \dots C[l]$	$\text{HCBC1}_{\pi^{-1}}(hK, C)$ Parse C as $C[1] \dots C[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow \pi^{-1}(C[i])$ $M[i] \leftarrow H(hK, C[i-1]) \oplus P[i]$ EndFor Return $M[1] \dots M[l]$
--	--

The proof looks at an on-line cipher as a 2^n -ary tree of permutations on $\{0, 1\}^n$, and goes through a hybrid argument involving a sequence of different games that “move” from $\text{OPerm}_{d,n}$ to HCBC1 . Let A be an adversary that has oracle access to a length-preserving function $f: D_{d,n} \rightarrow D_{d,n}$. We assume that A makes at most q_e oracle queries the sum of whose lengths is at most μ_e bits. We define three games associated with the adversary A as follows.

Game 1. Choose a tree of random permutations $T \xleftarrow{R} \text{Tree}(n, \text{Perm}_n, d)$. Run A , replying to its oracle queries via T as described in Section 3. Let P_1 be the probability that A returns 1.

Game 2. Choose a random permutation, $\pi \xleftarrow{R} \text{Perm}_n$, and choose a random key for H via $hK \xleftarrow{R} \{0, 1\}^{hk}$. Run A , replying to its oracle queries via $\text{HCBC1}_\pi(hK, \cdot)$. Let P_2 be the probability that A returns 1.

Game 3. Choose random keys for E and H via $eK \xleftarrow{R} \{0, 1\}^{ek}$ and $hK \xleftarrow{R} \{0, 1\}^{hk}$, respectively. Run A , replying to its oracle queries via $\text{HCBC1}(eK \| hK, \cdot)$. Let P_3 be the probability that A returns 1.

By the definition of $\text{Adv}_{\text{HCBC1}}^{\text{orprp-cpa}}(A)$, we have

$$\text{Adv}_{\text{HCBC1}}^{\text{orprp-cpa}}(A) = P_3 - P_1 = (P_3 - P_2) + (P_2 - P_1). \quad (7)$$

We bound the difference terms via the following lemmas:

Lemma 6.3 $P_3 - P_2 \leq \text{Adv}_E^{\text{prp-cpa}}(t, \mu_e/n, \mu_e)$

Lemma 6.4 $P_2 - P_1 \leq \frac{\mu_e^2 + 2n(q_e + 1)\mu_e}{n^2 \cdot 2^n} + \text{Adv}_H^{\text{axu-cr}}(\mu_e/n)$

Equation (7), Lemma 5.4, and the above lemmas imply the statement of the theorem. We present the proofs of the two lemmas in the next two subsections.

6.1 Proof of Lemma 6.3

We present an adversary B which attacks the pseudorandomness of the block cipher E . B has access to an oracle g which is either a random instance of block cipher E or a random instance of Perm_n . The adversary B uses it to simulate an experiment for A . First, B chooses a random key for H via $hK \xleftarrow{R} \{0, 1\}^{hk}$. Then, it runs A , replying to its oracle queries via $\text{HCBC1}_g(hK, \cdot)$. B can do so since the cipher HCBC1_g makes only oracle use of g . Finally, B outputs the output of A . It is easy to see that, if A succeeds, so does B . Therefore,

$$\Pr \left[g \xleftarrow{R} E : B^g = 1 \right] = P_3$$

$$\Pr \left[g \xleftarrow{R} \text{Perm}_n : B^g = 1 \right] = P_2$$

Subtracting these equations and taking maximums, we get the statement of the claim.

Let hK denote the key of the hash function, and π the choice of permutation from Perm_n , that underly Game 2. Then

```

For each  $j = 1, \dots, q_e$ 
  Let  $C_j[0] = 0^n$ 
  For  $i = 1, \dots, l_j$ 
    Let  $P_j[i] = H(hK, C_j[i-1]) \oplus M_j[i]$ 
    Let  $C_j[i] = \pi(P_j[i])$ 
  EndFor
EndFor

```

We now define some events in this game:

- Event C_2** : There exist (i, j) such that $1 \leq j \leq q_e$ and $1 \leq i \leq l_j$ and $C_j[i] = 0^n$
- Event C_H** : There exist $(i, j), (i', j')$ such that the following are all true:
- $1 \leq j < j' \leq q_e, 1 \leq i \leq l_j$ and $1 \leq i' \leq l_{j'}$
 - $P_j[i] = P_{j'}[i']$ but $C_j[i-1] \neq C_{j'}[i'-1]$

Figure 4: Some definitions for Game 2.

6.2 Proof of Lemma 6.4

Let

$$\begin{aligned}
M_1 &= M_1[1] \dots M_1[l_1] \\
&\dots \\
M_{q_e} &= M_{q_e}[1] \dots M_{q_e}[l_{q_e}]
\end{aligned}$$

denote A 's queries. Figure 4 presents some notation and events related to Game 2, and Figure 5 presents some notation and events related to Game 1. We begin by providing some intuition about the proof and then proceed to the actual analysis.

Let $\text{Pr}_1[\cdot]$ denote the probability function underlying Game 1, namely that created by the random choice $T \stackrel{R}{\leftarrow} \text{Tree}(n, \text{Perm}_n, d)$, and let $\text{Pr}_2[\cdot]$ denote the probability function underlying Game 2, namely that created by the random choices of π and hK . Let F denote $\text{HCBC1}_\pi(hK, \cdot)$. As is common in such proofs, we wish to isolate “bad” events B_1, B_2 in Games 1 and 2, respectively, such that the “view” of the adversary, and hence the probability that it returns 1, is the same in the two games conditioned on the absence of the corresponding bad events, that is,

$$\text{Pr}_1[A^T = 1 \mid \overline{B}_1] = \text{Pr}_2[A^F = 1 \mid \overline{B}_2].$$

In that case, a conditioning argument will show that the difference between P_2 and P_1 can be bounded by the sum of the probabilities of the bad events in their respective games. We find the bad events by looking at Game 2 and asking under what conditions we can make the distribution of ciphertexts returned to the adversary there mimic that in Game 1. To explain the intuition, we introduce the following terminology:

Definition. Suppose $1 \leq j, j' \leq q, 1 \leq i \leq l_j$ and $1 \leq i' \leq l_{j'}$. We say that $(i, j) \prec (i', j')$ if: either $j = j'$ and $i < i'$, or $j < j'$. We say that (i', j') is *trivial* if there exists some $j < j'$ such that $ni' \leq |\text{LCP}_n(M_j, M_{j'})|$.

Let T denote the random choice of tree from $\text{Tree}(n, \text{Perm}_n, d)$ that underlies this game. Then

```

For each  $j = 1, \dots, q_e$ 
  Let  $x_j[0] = \varepsilon$ 
  For  $i = 1, \dots, l_j$ 
    Let  $C_j[i] = T_{x_j[i-1]}(M_j[i])$ 
    Let  $x_j[i] = x_j[i-1] || C_j[i]$ 
  EndFor
EndFor

```

We now define some events in this game:

- Event $C_{1,1}$** : There exist (i, j) such that $1 \leq j \leq q_e$ and $1 \leq i \leq l_j$ and $C_j[i] = 0^n$
- Event $C_{1,2}$** : There exist $(i, j), (i', j')$ such that the following are all true:
- $1 \leq j < j' \leq q_e, 1 \leq i \leq l_j$ and $1 \leq i' \leq l_{j'}$
 - $x_j[i-1] \neq x_{j'}[i'-1]$ but $C_j[i] = C_{j'}[i']$
- Event C_1** : $C_{1,1} \vee C_{1,2}$

Figure 5: Some definitions for Game 1.

We claim that the bad event has been chosen so that, in its absence, the following is true for every non-trivial (i', j') : If $(i, j) \prec (i', j')$ then $P_j[i] \neq P_{j'}[i']$. In other words, any two input points to the function π are unequal unless they are equal for the trivial reason that the corresponding message prefixes are equal. We will set $B_2 = C_H \vee C_2$, which says that either there is a collision for $H(hK, \cdot)$ or some returned ciphertext block is 0^n , and show in Claim 6.5 that this choice implies the statement we just made. This means that in the absence of the bad event, ciphertext blocks whose value is not “forced” by message prefix conditions are random but distinct, being outputs of a random permutation. We now choose an event B_1 in Game 1 which makes the output distribution here the same. The choice is $B_1 = C_1 = C_{1,1} \vee C_{1,2}$. This says that either some tree node returns 0^n , or there are two different tree nodes that return the same output. Now we provide the detailed analysis.

We upper bound $P_2 - P_1$ as follows.

$$\begin{aligned}
P_2 - P_1 &= \Pr_2[A^F = 1] - \Pr_1[A^T = 1] \\
&= \Pr_2[A^F = 1 | C_H \vee C_2] \cdot \Pr_2[C_H \vee C_2] + \Pr_2[A^F = 1 | \bar{C}_H \wedge \bar{C}_2] \cdot \Pr_2[\bar{C}_H \wedge \bar{C}_2] \\
&\quad - \Pr_1[A^T = 1 | C_1] \cdot \Pr_1[C_1] - \Pr_1[A^T = 1 | \bar{C}_1] \cdot \Pr_1[\bar{C}_1] \\
&\leq \Pr_2[C_H \vee C_2] + \Pr_2[A^F = 1 | \bar{C}_H \wedge \bar{C}_2] \cdot \Pr_2[\bar{C}_H \wedge \bar{C}_2] \\
&\quad - \Pr_1[A^T = 1 | \bar{C}_1] \cdot \Pr_1[\bar{C}_1]
\end{aligned} \tag{8}$$

We use the following claim to simplify Equation (8).

Claim 6.5 $\Pr_2[A^F = 1 | \bar{C}_H \wedge \bar{C}_2] = \Pr_1[A^T = 1 | \bar{C}_1]$ ■

We denote the probability of the above claim by α and simplify Equation (8) as follows.

$$\begin{aligned}
P_2 - P_1 &\leq \Pr_2[\mathbf{C}_H \vee \mathbf{C}_2] + \alpha \cdot \left(\Pr_2[\overline{\mathbf{C}}_H \wedge \overline{\mathbf{C}}_2] - \Pr_1[\overline{\mathbf{C}}_1] \right) \\
&= \Pr_2[\mathbf{C}_H \vee \mathbf{C}_2] + \alpha \cdot (1 - \Pr_2[\mathbf{C}_H \vee \mathbf{C}_2] - 1 + \Pr_1[\mathbf{C}_1]) \\
&\leq \Pr_2[\mathbf{C}_H \vee \mathbf{C}_2] + \Pr_1[\mathbf{C}_1] \\
&\leq \Pr_2[\mathbf{C}_H] + \Pr_2[\mathbf{C}_2] + \Pr_1[\mathbf{C}_1].
\end{aligned}$$

The terms are bounded via the following claims:

Claim 6.6 $\Pr_2[\mathbf{C}_H] \leq \mathbf{Adv}_H^{\text{axu-cr}}(\mu_e/n)$ ■

Claim 6.7 $\Pr_2[\mathbf{C}_2] \leq \frac{2\mu_e}{n \cdot 2^n}$ ■

Claim 6.8 $\Pr_1[\mathbf{C}_1] \leq \frac{\mu_e^2 + 2nq_e\mu_e}{n^2 \cdot 2^n}$ ■

Given the claims and Lemma 5.4, the proof of Lemma 6.4 is complete. Now, we prove the claims.

Proof of Claim 6.7: All ciphertext blocks are outputs of the random permutation π . The probability that at least one of them equals 0^n is thus at most

$$\sum_{i=1}^{\mu_e/n} \frac{1}{2^n - (i-1)} \leq \sum_{i=1}^{\mu_e/n} \frac{1}{2^n - \mu/n} \leq \frac{\mu_e}{n} \cdot \frac{1}{2^n - \mu/n}.$$

The assumption $\mu_e/n \leq 2^{n-1}$ made in the theorem statement implies that the above is at most

$$\frac{\mu_e}{n} \cdot \frac{1}{2^n - 2^{n-1}} = \frac{\mu_e}{n} \cdot \frac{1}{2^{n-1}},$$

as claimed. ■

Proof of Claim 6.8: We first provide the computation and then an explanation. Letting $l_0 = 0$ we claim that

$$\begin{aligned}
\Pr_1[\mathbf{C}_1] &\leq \sum_{j=0}^{q_e-1} \left(\frac{l_0 + \dots + l_j + 1}{2^n - j} + \sum_{i=2}^{l_{j+1}} \frac{l_0 + \dots + l_j + i}{2^n} \right) \\
&\leq \sum_{j=0}^{q_e-1} \frac{l_0 + \dots + l_j + 1}{2^n - j} + \sum_{j=0}^{q_e-1} \sum_{i=2}^{l_{j+1}} \frac{l_0 + \dots + l_j + i}{2^n} \\
&\leq \sum_{j=0}^{q_e-1} \frac{l_0 + \dots + l_j + 1}{2^n - j} + \sum_{j=0}^{q_e-1} \sum_{i=2}^{l_{j+1}} \frac{l_0 + \dots + l_j + i}{2^n} \\
&\leq \frac{1}{2^n - q_e} \cdot \sum_{j=0}^{q_e-1} (l_0 + \dots + l_j + 1) + \frac{1}{2^n} \cdot \sum_{j=0}^{q_e-1} \sum_{i=2}^{l_{j+1}} (l_0 + \dots + l_j + i) \\
&\leq \frac{1}{2^n - q_e} \cdot q_e \cdot \frac{\mu_e}{n} + \frac{1}{2^n} \cdot \sum_{j=0}^{q_e-1} \sum_{i=2}^{l_{j+1}} \frac{\mu_e}{n}
\end{aligned}$$

$$\begin{aligned}
&\leq \frac{q_e \mu_e}{n \cdot (2^n - q_e)} + \frac{1}{2^n} \cdot \frac{\mu_e}{n} \sum_{j=0}^{q_e-1} (l_{j+1} - 1) \\
&\leq \frac{q_e \mu_e}{n \cdot (2^n - q_e)} + \frac{\mu_e}{n \cdot 2^n} \cdot \frac{\mu_e}{n}
\end{aligned}$$

The assumption $\mu_e/n \leq 2^{n-1}$ made in the theorem statement implies that $q_e \leq 2^{n-1}$, so the above is at most

$$\frac{q_e \mu_e}{n \cdot (2^n - 2^{n-1})} + \frac{\mu_e^2}{n^2 \cdot 2^n} \leq \frac{\mu_e^2 + 2nq_e \mu_e}{n^2 \cdot 2^n},$$

as claimed. We now proceed to the explanation.

We have a tree whose nodes are independently chosen random permutations. We imagine the queries being made one by one, ask what is the probability that event C_1 happens with each added query. Suppose j queries have been made, and consider the $(j+1)$ -th query. We call *crucial* the i -th node reached by this query, where

$$i = \frac{1}{n} \cdot \max_{1 \leq j' \leq j} \{|\text{LCP}_n(M_{j'}, M_{j+1})|\}.$$

The permutation at the i -th node is getting a new input, and we ask that the output not collide with previous outputs or have value 0^n . The number of points at which this permutation has been previously evaluated is at most $2^n - j$, since each query can go through this node at most once. Given that there are no collisions, the rest of the nodes that query M_j incurs are all new, so, for each of them, the corresponding permutation has a 2^{-n} probability of hitting any of the previous output points or 0^n . Finally, we claim that the worst case is when the crucial node is the root node of the tree. Our first equation above is based on all this. ■

Proof of Claim 6.5: Let us first look at Game 2. We define the event

Event G_2 : For every non-trivial (i', j') : If $(i, j) \prec (i', j')$ then $P_j[i] \neq P_{j'}[i']$.

We claim that

$$\overline{C}_H \wedge \overline{C}_2 \text{ iff } G_2 \wedge \overline{C}_2. \quad (9)$$

We postpone a justification of this claim and proceed. This claim says that an input block $P_{j'}[i']$ to the permutation π is new unless (i', j') is trivial. Since π is a random permutation, this means that the output ciphertexts corresponding to all non-trivial pairs (i', j') are random, distinct and (since we are assuming \overline{C}_2) also non- 0^n . Now, we note that the ciphertexts in the replies to A 's queries in Game 1 have the same distribution assuming event \overline{C}_1 is true there. It remains to justify Equation (9).

It is easy to see that $G_2 \wedge \overline{C}_2 \Rightarrow \overline{C}_H \wedge \overline{C}_2$. We now want to show that $\overline{C}_H \wedge \overline{C}_2 \Rightarrow G_2 \wedge \overline{C}_2$. So assume that event $\overline{C}_H \wedge \overline{C}_2$ is true and that (i', j') is non-trivial. Assume $(i, j) \prec (i', j')$. We want to show that $P_j[i] \neq P_{j'}[i']$. Assume towards a contradiction that this is not true, meaning $P_j[i] = P_{j'}[i']$. We know that

$$\begin{aligned}
P_j[i] &= H(hK, C_j[i-1]) \oplus M_j[i] \\
P_{j'}[i'] &= H(hK, C_{j'}[i'-1]) \oplus M_{j'}[i'].
\end{aligned}$$

Adversary $B^{D_{hK}}$

- (1) $j \leftarrow 0; S \leftarrow \emptyset$
- (2) Run A , replying to its oracle queries as follows:
- (3) $j \leftarrow j + 1$
- (4) Let $M_j[1] \dots M_j[l_j]$ be A 's next oracle query
- (5) $j' \leftarrow 0; k \leftarrow 0$
- (6) For $s = 1, \dots, j - 1$ do
- (7) If $|\text{LCP}_n(M_s, M_j)| > nk$ then $j' \leftarrow s; k \leftarrow |\text{LCP}_n(M_{j'}, M_j)|/n$ EndIf
- (8) EndFor
- (9) $C_j[0] \leftarrow 0^n$
- (10) For $i = 1, \dots, k$ do
- (11) $C_j[i] \leftarrow C_{j'}[i]$
- (12) EndFor
- (13) For $i = k + 1, \dots, l_j$ do
- (14) $c \leftarrow D_{hK}(C_j[i - 1], M_j[i])$
- (15) If $c \neq 0$ then
- (16) Let j', i' be the unique integers such that $1 \leq i' \leq l_{j'}$ and $c = l_1 + \dots + l_{j'-1} + i'$
- (17) $C_j[i] \leftarrow C_{j'}[i']$
- (18) Else
- (19) $C_j[i] \stackrel{R}{\leftarrow} \{0, 1\}^n - S; S \leftarrow S \cup \{C_j[i]\}$
- (20) EndIf
- (21) EndFor
- (22) Return $C_j[1] \dots C_j[l_j]$ to A as the response to its oracle query
- (23) Until A halts

Figure 6: Adversary B for proof of Claim 6.6.

Since $\overline{\mathbf{C}}_H$ is true, the only way we could have $P_j[i] = P_{j'}[i']$ is that $C_j[i - 1] = C_{j'}[i' - 1]$ and $M_j[i] = M_{j'}[i' - 1]$. Since π is a permutation, it must be that $P_j[i - 1] = P_{j'}[i - 1]$. We iterate this argument, each time decrementing both i and i' by one, until $\min(i, i') = 0$. Now consider two cases. The first is that $j = j'$ and $i < i'$. In that case we have $C_j[0] = C_{j'}[i' - i]$. But $C_j[0] = 0^n$, contradicting the assumption that event $\overline{\mathbf{C}}_2$ is true. The second case is that $j < j'$. If $i \neq i'$ we contradict the assumption that event $\overline{\mathbf{C}}_2$ is true just as before. If $i = i'$ we contradict the assumption that (i', j') is non-trivial. ■

Proof of Claim 6.6: We present an adversary B such that

$$\Pr[\mathbf{C}_H] \leq \mathbf{Adv}_H^{\text{axu-cr}}(B),$$

and B submits μ_e/n queries to the oracle D_{hK} . The adversary B simulates the adversary A . In order to answer A 's queries, B simulates the random permutation π . The adversary B is shown in Figure 6.

If A can trigger the event \mathbf{C}_H , then the following is true: there exist $1 \leq j, j' \leq q_e$ such that the distinct queries $M_j = M_j[1] \dots M_j[l_j]$ and $M_{j'} = M_{j'}[1] \dots M_{j'}[l_{j'}]$ result in the responses $C_j = C_j[1] \dots C_j[l_j]$ and $C_{j'} = C_{j'}[1] \dots C_{j'}[l_{j'}]$, respectively, and there exist integers i, i' where

Distinguisher $A^{g,g^{-1}}$

Let $C[1]$ be any n -bit string not equal to 0^n

Let $C[2], \dots, C[l]$ be any n -bit strings

Let $C_1 = 1^n C[2] \dots C[l]$ and let $C_2 = 0^n 1^n C[2] \dots C[l-1]$

Let $M_1[1] \dots M_1[l] \leftarrow g^{-1}(C_1)$ and let $M_2[1] \dots M_2[l] \leftarrow g^{-1}(C_2)$

If $M_1[2] = M_2[3]$, then return 1 else return 0.

Figure 7: Chosen-ciphertext attack on HCBC1.

$1 \leq i \leq l_j$ and $1 \leq i' \leq l_{j'}$ such that if $j = j'$, then $i \neq i'$ and

$$H(hK, C_j[i-1]) \oplus M_j[i] = H(hK, C_{j'}[i'-1]) \oplus M_{j'}[i'] .$$

This means that B must have submitted the pair $C_j[i-1], M_j[i]$ and the pair $C_{j'}[i'-1], M_{j'}[i']$ to the oracle D_{hK} at some point. Since $C_j[i-1] \neq C_{j'}[i'-1]$, the oracle D_{hK} will return a non-zero value, and the experiment $\mathbf{Exp}_H^{\text{axu-cr}}(B)$ will return 1.

It remains to justify that B simulates A in the exact same environment as that of Game 2. When a collision occurs, B uses the answer from the oracle D_{hK} to figure out which previous output block was output as a result of the collision and submits it as an answer to A . When no collisions occur, B simply returns a new random value. It does so by memorizing used values in the set S . Thus, B appropriately simulates HCBC1_π where π is a random permutation in this case. We point out that lines 6-12 allow B to answer queries with common prefixes in a consistent manner. In particular, in processing the j th query M_j , B looks for a previous query that has the largest longest common prefix with M_j . Let us denote such a query with M'_j . Then, B simply uses the sequence of ciphertexts that it has used before for M'_j , upto the end of the longest common prefix, as part of the answer to the query M_j .

Since μ_e denotes the total number of bits that A queries, it follows that

$$\mathbf{Adv}_H^{\text{axu-cr}}(\mu_e/n) \geq \Pr[\mathbf{C}_H] .$$

And Claim 6.6 follows. ■

7 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 7 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 $\text{OPerm}_{n,d}$. We claim that

$$\mathbf{Adv}_{\text{HCBC1}}^{\text{oprp-cpa}}(A) \geq 1 - 2^{-n} . \quad (10)$$

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

The idea of the attack is to exploit the oracle access to g^{-1} and the fact that, under HCBC1, two fixed, consecutive ciphertext blocks yield two consecutive message blocks where the second message block is also fixed.

We claim that

$$\Pr \left[g \stackrel{R}{\leftarrow} \text{HCBC1} : A^g = 1 \right] = 1 \quad \text{and} \quad \Pr \left[g \stackrel{R}{\leftarrow} \text{OPerm}_{n,d} : A^g = 1 \right] \leq 2^{-n} ,$$

from which Equation (10) 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

$$\begin{aligned} M_1[2] &= E^{-1}(eK, C[2]) \oplus H(hK, 1^n) \\ &= M_2[3]. \end{aligned}$$

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 $\text{OPerm}_{n,d}$. Here, $M_1[2] = M_2[3]$ holds with probability at most 2^{-n} . Therefore, the distinguisher A^g outputs 1 with the probability at most 2^{-n} , and this justifies the second equation.

8 The Hash-CBC-2 cipher

We investigate the possibility of constructing an on-line cipher secure that that is secure against chosen-ciphertext attacks. We begin by looking at the following plausible extension of HCBC1.

Construction 8.1 Let $E: \{0,1\}^{ek} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. Let $H: \{0,1\}^{hk} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a family of functions. We associate to them a cipher HCBC2: $\{0,1\}^{ek+hk} \times D_{d,n} \rightarrow D_{d,n}$. 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 8 illustrates the cipher.

<pre> HCBC2($eK hK, M$) Parse M as $M[1] \dots M[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$; $M[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow H(hK, C[i-1]) \oplus M[i]$ $C[i] \leftarrow E(eK, P[i]) \oplus H(hK, M[i-1])$ EndFor Return $C[1] \dots C[l]$ </pre>	<pre> HCBC2⁻¹($eK hK, C$) Parse C as $C[1] \dots C[l]$ with $l \geq 1$ $C[0] \leftarrow 0^n$; $M[0] \leftarrow 0^n$ For $i = 1, \dots, l$ do $P[i] \leftarrow E^{-1}(eK, C[i] \oplus H(hK, M[i-1]))$ $M[i] \leftarrow H(hK, C[i-1]) \oplus P[i]$ EndFor Return $M[1] \dots M[l]$ ■ </pre>
---	---

It turns out that there are examples of AXU families H for which the HCBC2 construct is insecure against chosen-ciphertext attack. However we know of no attacks if H is PRF. The design of on-line ciphers provably secure against chosen-ciphertext attack seems to be an open question.

9 Usage of on-line ciphers

The use of an on-line ciphers can provide strong privacy and authenticity properties, even though the cipher itself is weak compared to a standard one, if the plaintext space has appropriate properties. This follows via the the encode-then-encipher paradigm of [5], under which we imagine an explicit encoding step applied to the raw data before enciphering. While [5] say that randomness and redundancy anywhere in the message suffices, we have to be more constrained: we *prepend* randomness and *append* redundancy.

Construction 9.1 Let n, d be integers, and let $F: \text{Keys}(F) \times D_{d,n} \rightarrow D_{d,n}$ be a cipher. We associate to them the following symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

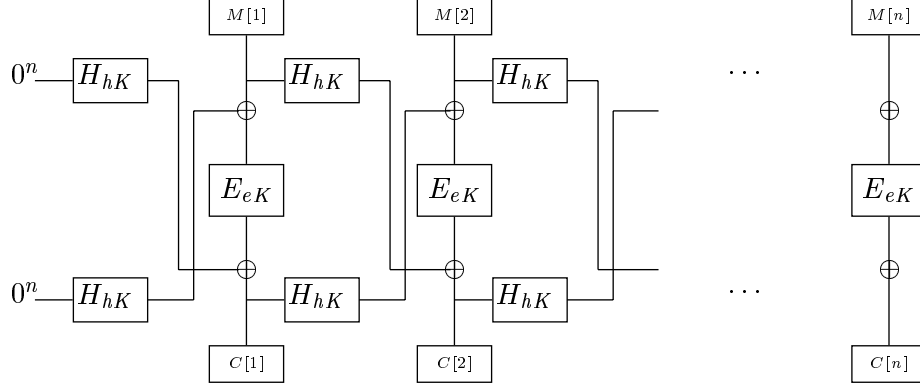


Figure 8: The HCBC2 cipher.

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

We want to show that this scheme provides privacy, when F is an n -on-line cipher secure against chosen-plaintext attacks, and authenticity, when F is an n -on-line cipher secure against chosen-ciphertext attacks. Definitions for these privacy and authenticity notions are standard (see for example [4]) and are recalled in Appendix A. Briefly, the symmetric encryption scheme achieves privacy and is called IND-CPA-secure if no polynomial time adversary, which gets to see ciphertexts for plaintexts of its choice and is given a challenge ciphertext, can get “any” information about the underlying plaintext. The symmetric encryption scheme achieves integrity and is called INT-CTXT-secure if no polynomial time adversary, which gets to see ciphertexts of plaintexts of its choice, can create a “new” valid ciphertext. The following claims state our results.

Proposition 9.2 Let $F: \text{Keys}(F) \times D_{d,n} \rightarrow D_{d,n}$ be an n -on-line cipher, and let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the symmetric encryption scheme defined in Construction 9.1. Then, for any integers $t, q_e, \mu_e \geq 0$,

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(t, q_e, \mu_e) \leq 2\mathbf{Adv}_F^{\text{opr-cpa}}(t, q_e, \mu_e) + \frac{q_e^2}{2^n}.$$

Also, for any integers $t, q_e, q_d, \mu_e, \mu_d \geq 0$,

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{int-ctxt}}(t, q_e, q_d, \mu_e, \mu_d) \leq 2\mathbf{Adv}_F^{\text{opr-cca}}(t, q_e, \mu_e, q_d, \mu_d) + \frac{q_d}{2^n}.$$

That is, if F is an n -on-line cipher secure against chosen-plaintext attacks, then \mathcal{SE} is IND-CPA secure, and if F is also secure against chosen-ciphertext attacks, then \mathcal{SE} is INT-CTXT secure.

The proof of Proposition 9.2 is simple and follows [5].

Proof of Proposition 9.2: Assume that there exists an adversary A that attacks privacy of \mathcal{SE} . We present an adversary B that attacks the on-line cipher F . B has an oracle g which is either a random instance of the on-line cipher F or a random instance of $\text{OPerm}_{n,d}$. B uses it to simulate an experiment $\mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cpa-}b}$ for A where b is a bit chosen at random by B . Then, for each query (M_1, M_2) made by A , the adversary B chooses a random element $r \xleftarrow{R} \{0, 1\}^n$ and submits

$\mathcal{E}_g(r\|M_b\|0^n)$ as a reply. B can do so since the encryption algorithm \mathcal{E}_g makes only oracle use of g . If A guesses the bit b correctly, then B returns 1. Otherwise, it returns 0. Analyzing the algorithm, we have

$$\begin{aligned} \Pr \left[g \stackrel{R}{\leftarrow} F : B^g = 1 \right] &= \frac{1}{2} \Pr[\mathbf{Exp}_{\mathcal{SE},A}^{\text{ind-cpa-1}} = 1] + \frac{1}{2} (1 - \Pr[\mathbf{Exp}_{\mathcal{SE},A}^{\text{ind-cpa-0}} = 1]) \quad (11) \\ &= \frac{1}{2} + \frac{1}{2} \mathbf{Adv}_{\mathcal{SE},A}^{\text{ind-cpa}} \\ \Pr \left[g \stackrel{R}{\leftarrow} \text{OPerm}_{n,d} : B^g = 1 \right] &\geq \frac{1}{2} + \frac{q_e^2}{2^{n+1}} \quad (12) \end{aligned}$$

Equation (11) holds since, if g is an instance of F , then B simulates a valid experiment for A . For Equation (12), 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 random elements r_1 and r_2 chosen by B happen to be the same. The reason is that, if they are different, g behaves like an ideal pseudorandom permutation. The additional term in Equation (12) comes from the probability of any two of q_e random elements of length n being equal.

Subtracting these equations and taking maximums, we get the statement of the first inequality of the proposition. We now move to the second.

Assume that there exists an adversary A that attacks integrity of \mathcal{SE} . We show how to construct an adversary B that attacks the pseudorandomness of the on-line cipher F . B 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 $\text{OPerm}_{n,d}$ and its inverse. B uses them to simulate an experiment $\mathbf{Exp}_{\mathcal{SE},A}^{\text{int-ctxt}}$ for A as follows. For each encryption query $\mathcal{E}_K(M)$ made by A , the adversary B chooses a random element $r \stackrel{R}{\leftarrow} \{0, 1\}^n$ and submits $\mathcal{E}_g(r\|M\|0^n)$ as a reply. For each verification query $\mathcal{D}_K^*(M)$ made by A , the adversary B submits $\mathcal{D}_g(M)$ as a reply. B can do so since the algorithms $\mathcal{E}_g, \mathcal{D}_g$ make only oracle use of g, g^{-1} . If A obtains 1 after querying a new ciphertext, then B returns 1. Otherwise, it returns 0. Analyzing the algorithm, we have

$$\Pr \left[g \stackrel{R}{\leftarrow} F : B^{g,g^{-1}} = 1 \right] = \Pr[\mathbf{Exp}_{\mathcal{SE},A_{\text{ctxt}}}^{\text{int-ctxt}} = 1] \quad (13)$$

$$\Pr \left[g \stackrel{R}{\leftarrow} \text{OPerm}_{n,d} : B^{g,g^{-1}} = 1 \right] = \frac{q_d}{2^n} \quad (14)$$

Equation (13) holds since, if g is an instance of F , then B simulates a valid experiment for A . For Equation (14), we observe that, if g is an ideal on-line cipher, then A cannot succeed unless it randomly guesses the valid ciphertext. To see this, we view the answers to A 's queries as outputs of a function T , where $T \stackrel{R}{\leftarrow} \text{Tree}(n, \text{Perm}_n, d)$. Recall that the adversary A has to output a *new* ciphertext C in order to succeed. In particular, the ciphertext C has to be different from all ciphertexts that A has seen in at least one block. If we imagine a tree based representation, this would mean that the corresponding message will be determined by the random path in the tree starting from the point defined by the first unique block in the output ciphertext. Therefore, the probability of the last block of this message being 0^n is the probability of guessing at random a valid ciphertext. This probability of guessing the block of n bits making q_d is the term in Equation (14).

Subtracting these equations and taking maximums we get the statement of the second inequality in the proposition. ■

Note that if n -on-line ciphers are used to encrypt messages which by their nature start with at least n random bits and end with some fixed sequence of n bits then we get a symmetric encryption scheme that achieves privacy and integrity and, moreover, is length-preserving.

10 Acknowledgments

We thank Anand Desai and Bogdan Warinschi for their helpful comments.

References

- [1] M. Bellare, A. Boldyreva, L. Knudsen, C. Namprempre. On-line ciphers and the Hash-CBC construction. Full version of this paper, available via <http://www-cse.ucsd.edu/users/mihir>.
- [2] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.
- [3] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In *Journal of Computer and System Sciences*, volume 61, No. 3, pages 362–399, Dec 2000. Academic Press.
- [4] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Berlin, Germany, Dec. 2000. Springer-Verlag.
- [5] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330, Berlin, Germany, Dec. 2000. Springer-Verlag.
- [6] C. Campbell. Design and specification of cryptographic capabilities. In D. Brandstad, editor, *Computer Security and the Data Encryption Standard*, National Bureau of Standards Special Publications 500-27, U.S. Department of Commerce, pages 54-66, February 1978.
- [7] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In G. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288, Berlin, Germany, Aug. 1984. Springer-Verlag.
- [8] C. Jutla. Encryption modes with almost free message integrity. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 525–544, Berlin, Germany, 2001. Springer-Verlag.
- [9] L. Knudsen. Block chaining modes of operation. Symmetric Key Block Cipher Modes of Operation Workshop, <http://csrc.nist.gov/encryption/modes/workshop1/>, Oct. 2000.
- [10] H. Krawczyk. LFSR-based hashing and authenticating. In Y. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139, Berlin, Germany, 1994. Springer-Verlag.
- [11] M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In H. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, page 447, Berlin, Germany, Aug. 1985. Springer-Verlag.

- [12] C. Meyer and Matyas. *A new direction in Computer Data Security*. John Wiley & Sons, 1982.
- [13] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff Revisited. In J. Feigenbaum, editor, *Journal of Cryptology*, Volume 12, Number 1, Winter 1999. Springer-Verlag.
- [14] W. Nevelsteen and B. Preneel. Software performance of universal hash functions. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 24–41, Berlin, Germany, 1999. Springer-Verlag.

A Definitions for privacy and integrity of symmetric encryption schemes

PRIVACY OF SYMMETRIC ENCRYPTION SCHEMES. We measure indistinguishability via the “left-or-right” model of [2]. Define the *left-or-right encryption oracle* $\mathcal{E}(K, \mathcal{LR}(\cdot, \cdot, b))$, where $b \in \{0, 1\}$, to take input (x_0, x_1) and returns $\mathcal{E}(K, x_b)$. The adversary makes oracle queries of the form (x_0, x_1) consisting of two equal length messages and must guess the bit b . Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Let $b \in \{0, 1\}$. Let A_{cpa} be an adversary that has access to an oracle. Now, we consider the following experiments:

Experiment $\mathbf{Exp}_{\mathcal{SE}, A_{\text{cpa}}}^{\text{ind-cpa-}b}$

$K \xleftarrow{R} \mathcal{K}$
 $x \leftarrow A_{\text{cpa}}^{\mathcal{E}(K, \mathcal{LR}(\cdot, \cdot, b))}$
 Return x

The *advantage* of the adversary is defined via

$$\mathbf{Adv}_{\mathcal{SE}, A_{\text{cpa}}}^{\text{ind-cpa}} = \Pr[\mathbf{Exp}_{\mathcal{SE}, A_{\text{cpa}}}^{\text{ind-cpa-}1} = 1] - \Pr[\mathbf{Exp}_{\mathcal{SE}, A_{\text{cpa}}}^{\text{ind-cpa-}0} = 1]$$

We define the *advantage function of the scheme* as follows. For any integers t, q_e, μ_e ,

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{ind-cpa}}(t, q_e, \mu_e) = \max_{A_{\text{cpa}}} \left\{ \mathbf{Adv}_{\mathcal{SE}, A_{\text{cpa}}}^{\text{ind-cpa}} \right\}$$

where the maximum is over all A_{cpa} with time-complexity t , each making to the $\mathcal{E}(K, \mathcal{LR}(\cdot, \cdot, b))$ oracle at most q_e queries the sum of whose lengths is at most μ_e bits. The scheme \mathcal{SE} is said to be IND-CPA secure if the function $\mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cpa}}(\cdot)$ is negligible for any polynomial time adversary A .

INTEGRITY OF SYMMETRIC ENCRYPTION SCHEMES. Now we recall security definitions for integrity (authenticity) of a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ from [4]. We first define a verification algorithm $\mathcal{D}_K^*(\cdot)$ as follows:

Algorithm $\mathcal{D}_K^*(C)$

If $\mathcal{D}_K(C) \neq \perp$, then return 1.
 Else return 0.

The adversary has access to an encryption oracle $\mathcal{E}_K(\cdot)$ and also to the verification oracle. It is successful if it makes the verification oracle accept a ciphertext that was never returned by the encryption oracle, even if the corresponding plaintext was queried of the encryption oracle.

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Let A_{ctxt} be an adversary which has access to two oracles. Consider the following experiment:

Experiment $\mathbf{Exp}_{\mathcal{SE}, A_{\text{ctxt}}}^{\text{int-ctxt}}$

$K \xleftarrow{R} \mathcal{K}$

If $A_{\text{ctxt}}^{\mathcal{E}_K(\cdot), \mathcal{D}_K^*(\cdot)}$ makes a query C to

the oracle $\mathcal{D}_K^*(\cdot)$ such that

- $\mathcal{D}_K^*(C)$ returns 1, and
- C was never a response of $\mathcal{E}_K(\cdot)$

then return 1 else return 0.

We define the *advantage* of the adversary via

$$\mathbf{Adv}_{\mathcal{SE}, A_{\text{ctxt}}}^{\text{int-ctxt}} = \Pr[\mathbf{Exp}_{\mathcal{SE}, A_{\text{ctxt}}}^{\text{int-ctxt}} = 1]$$

We define the *advantage function of the scheme* as follows. For any integers $t, q_e, q_d, \mu_e, \mu_d$,

$$\mathbf{Adv}_{\mathcal{SE}}^{\text{int-ctxt}}(t, q_e, q_d, \mu_e, \mu_d) = \max_{A_{\text{ctxt}}} \left\{ \mathbf{Adv}_{\mathcal{SE}, A_{\text{ctxt}}}^{\text{int-ctxt}} \right\}$$

where the maximum is over all A_{ctxt} with time-complexity t , each making to the oracle $\mathcal{E}_K(\cdot)$ at most q_e queries the sum of whose lengths is at most μ_e bits, and each making to the $\mathcal{D}_K^*(\cdot)$ oracle at most q_d queries the sum of whose lengths is at most μ_d bits. The scheme \mathcal{SE} is said to be *INT-CTXT secure* if the function $\mathbf{Adv}_{\mathcal{SE}, A}^{\text{int-ctxt}}(\cdot)$ is negligible for any polynomial time adversary A .