

An extended abstract of this paper appears in *Advances in Cryptology – EUROCRYPT '01*, Lecture Notes in Computer Science Vol. 2045, B. Pfitzmann ed., Springer-Verlag, 2001. This is the full version.

## Does encryption with redundancy provide authenticity?

JEE HEA AN\*

MIHIR BELLARE<sup>†</sup>

September 11, 2001

### Abstract

A popular paradigm for achieving privacy plus authenticity is to append some “redundancy” to the data before encrypting. The redundancy is computed by applying a redundancy function to the data. We investigate the security of this paradigm at both a general and a specific level. We consider various possible notions of privacy for the base encryption scheme, and for each such notion we provide a condition on the redundancy function that is necessary and sufficient to ensure authenticity of the encryption-with-redundancy scheme. We then consider the case where the base encryption scheme is a variant of CBC called NCBC, and find sufficient conditions on the redundancy functions for NCBC encryption-with-redundancy to provide authenticity. Our results highlight an important distinction between public redundancy functions, meaning those that the adversary can compute, and secret ones, meaning those that depend on the shared key between the legitimate parties.

**Keywords:** Symmetric encryption, Redundancy, Public redundancy, Secret redundancy, Authenticity, Integrity, Concrete security.

---

\*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-mail: [jeehea@cs.ucsd.edu](mailto:jeehea@cs.ucsd.edu). URL: <http://www-cse.ucsd.edu/users/jeehea>. Supported in part by grants of the second author and an NSF Graduate Fellowship.

<sup>†</sup>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](mailto:mihir@cs.ucsd.edu). URL: <http://www-cse.ucsd.edu/users/mihir>. Supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	General results . . . . .	3
1.2	Encryption with NCBC . . . . .	5
1.3	Related work . . . . .	6
<b>2</b>	<b>Definitions</b>	<b>6</b>
<b>3</b>	<b>The encryption-with-redundancy paradigm</b>	<b>9</b>
<b>4</b>	<b>Encryption with public redundancy</b>	<b>10</b>
<b>5</b>	<b>Encryption with secret redundancy</b>	<b>14</b>
<b>6</b>	<b>Nested CBC (NCBC) with redundancy</b>	<b>18</b>
6.1	NCBC with secret redundancy . . . . .	20
6.2	NCBC with public redundancy . . . . .	24
	<b>References</b>	<b>28</b>
<b>A</b>	<b>Attack on CBC with public redundancy</b>	<b>29</b>
<b>B</b>	<b>Proofs</b>	<b>30</b>
B.1	Proof of Theorem 3.3 . . . . .	30
B.2	Proof of Lemma 6.10 . . . . .	31

# 1 Introduction

The idea that authenticity can be easily obtained as a consequence of the privacy conferred by encryption has long attracted designers. Encryption-with-redundancy is the most popular paradigm to this end. Say that parties sharing key  $K$  are encrypting data via some encryption function  $\mathcal{E}$ . (Typically this is some block-cipher mode of operation.) To obtain authenticity, the sender computes some function  $h$  of the data  $M$  to get a “checksum”  $\tau = h(M)$ .<sup>1</sup> It then computes a ciphertext  $C \leftarrow \mathcal{E}_K(M\|\tau)$  and sends  $C$  to the receiver. The latter decrypts to get  $M\|\tau$  and then checks whether  $\tau = h(M)$ . If not, it rejects the ciphertext as unauthentic.

The attraction of the paradigm is clear: the added cost of providing authenticity is small, amounting to computation of the checksum function plus perhaps one or two extra block-cipher invocations in order to encrypt the now longer message. (Designers attempt to use simple and fast checksum functions.) However, the paradigm has a poor security record. For example, using CBC encryption with the checksum being the XOR of the message blocks (called CBCC) was proposed by the U.S. National Bureau of Standards, and was subsequently found to not provide authenticity, as discussed in [24, 17]. If the encryption algorithm is an additive stream cipher (e.g. CTR-mode encryption) where the adversary knows the plaintext, forgery attacks by [16, 17] apply. An attack attributed to Wagner on a large class of CBC-mode encryption-with-redundancy schemes is described in [26].

## 1.1 General results

The many and continuing efforts to achieve authenticity via the encryption-with-redundancy paradigm point to the existence of some intuition that leads designers to think that it should work. The intuition appears to be that the privacy conveyed by the encryption makes attacks on the integrity harder. The first goal of our work is to assess the correctness of this intuition, and the security of the paradigm, at a general level. We are not concerned so much with the security of specific schemes as with trying to understand how the authenticity of the encryption-with-redundancy scheme relates to the security properties of the underlying primitives and to what extent the paradigm can be validated at a general level.

We denote the base encryption scheme by  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ . (It is specified by its key-generation, encryption, and decryption algorithms.) We are general with regard to the form of the redundancy computation method, allowing it to be key-based. A choice of method is given by a *redundancy code*  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  where  $\mathcal{K}_r$  is an algorithm responsible for generating a key  $K_r$  while  $\mathcal{H}$  takes  $K_r$  and the text  $M$  to return the redundancy or checksum  $\tau = \mathcal{H}_{K_r}(M)$ . Associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is the encryption-with-redundancy scheme  $\mathcal{ER}$  in which one encrypts message  $M$  via  $C \leftarrow \mathcal{E}_{K_e}(M\|\mathcal{H}_{K_r}(M))$ . Upon receipt of ciphertext  $C$ , the receiver applies  $\mathcal{D}_K$  to get back  $M\|\tau$  and accepts iff  $\tau = \mathcal{H}_{K_r}(M)$ . Here  $K_e$  is the (secret) encryption key for  $\mathcal{SE}$ .

We distinguish *public redundancy* and *secret redundancy*. In the first case,  $K_r$  is public information. (For example,  $\mathcal{H}_{K_r}(\cdot)$  might be a public hash function like SHA-1, or simply return the XOR of the message blocks.) In this case,  $K_r$  is known to the adversary, who is thus capable of computing the redundancy function. In the case of secret redundancy,  $K_r$  is part of the secret key shared between the parties. (It might for example be a key for a universal hash function [11] or a message authentication code.) In this case the key  $K_r$  is not given to the adversary.

The desired authenticity property of the encryption-with-redundancy scheme  $\mathcal{ER}$  is integrity of ciphertexts [7, 20, 8]: it should be computationally infeasible for an adversary to produce a

---

<sup>1</sup>Other names for the checksum include MDC —Manipulation Detection Code— and “redundancy,” whence the name of the paradigm.

Type of base encryption	Condition on redundancy code	
	For public redundancy	For secret redundancy
IND-CPA	None	None
NM-CPA	None	UF-NMA
IND-CCA	None	UF-NMA

Figure 1: For each possible privacy attribute SSS-AAA of the base encryption scheme, we indicate a condition on the redundancy code that is *necessary and sufficient* for it to be integrity-providing with respect to SSS-AAA. We distinguish the cases where the redundancy is public (anyone can compute it) and secret (depends on the shared secret key). “None” means that the corresponding class of redundancy codes is empty: *No* redundancy code is integrity-providing.

ciphertext that is valid but different from any created by the sender.

We allow the assumed privacy attribute of the base encryption scheme to range across the various well-established notions of privacy used in the literature: IND-CPA, NM-CPA, IND-CCA. (Indistinguishability under chosen-plaintext attack [14, 4], non-malleability under chosen-plaintext attack [12], and indistinguishability under chosen-ciphertext attack [29], respectively. Recall that non-malleability under chosen-ciphertext attack is equivalent to IND-CCA [5, 19] so we don’t need to consider it separately.)

We say that a redundancy code  $\mathcal{RC}$  is *integrity-providing with respect to security notion SSS-AAA* if for *all* base encryption schemes  $\mathcal{SE}$  that are SSS-AAA secure, the encryption-with-redundancy scheme  $\mathcal{ER}$  obtained from  $\mathcal{SE}$  and  $\mathcal{RC}$  is secure in the sense of integrity of ciphertexts. (This property of a redundancy code is attractive from the design viewpoint, since a redundancy code having this property may be used in conjunction with *any* SSS-AAA-secure base encryption scheme, and authenticity of the resulting encryption-with-redundancy scheme is guaranteed.) The question we ask is the following. Given a notion of security SSS-AAA, what security attribute of the redundancy code  $\mathcal{RC}$  will ensure that  $\mathcal{RC}$  is integrity-providing with respect to security notion SSS-AAA?

We find that an important distinction to be made in answering this question is whether or not the redundancy computation is secret-key based. Figure 1 summarizes the results we expand on below.

ENCRYPTION WITH PUBLIC REDUNDANCY. We show that there is *no* choice of public redundancy code  $\mathcal{RC}$  which is integrity-providing with respect to notions of security IND-CPA, NM-CPA or IND-CCA. This is a powerful indication that the intuition that privacy helps provide integrity via encryption-with-redundancy is wrong in the case where the adversary can compute the redundancy function.

This conclusion is not surprising when the base encryption scheme meets only a weak notion of privacy like IND-CPA. But one might have thought that there are redundancy codes for which a condition like NM-CPA on the base encryption scheme would suffice to prove integrity of ciphertexts for the resulting encryption-with-redundancy scheme. Not only is this false, but it stays false when the base encryption scheme has even a stronger privacy attribute like IND-CCA.

Note that the most popular methods for providing redundancy are public, typically involving computing a keyless checksum of the message, and our result applies to these.

The result is proved by giving an example of a base encryption scheme meeting the notion of privacy in question such that for any redundancy code the corresponding encryption with public redundancy scheme can be attacked. (This assumes there exists some base encryption scheme

meeting the notion of privacy in question, else the issue is moot.)

ENCRYPTION WITH SECRET REDUNDANCY. As Figure 1 indicates, allowing the computation of the redundancy to depend on a secret key does not help if the base encryption scheme meets only a weak notion of privacy like IND-CPA— *no* secret redundancy code is integrity-providing with respect to IND-CPA.

However secret redundancy does help if the base encryption scheme has stronger privacy attributes. We characterize the requirement on the redundancy code in this case. We say that it is UF-NMA (UnForgeable under No-Message Attack) if it is a MAC for which forgery is infeasible for an adversary that is not allowed to see the MACs of any messages before it must output its forgery. We show that this condition on the redundancy code is sufficient to ensure that it is integrity-providing with respect to NM-CPA and IND-CCA. We also show that this condition is necessary in the sense that for any redundancy code that is not UF-NMA secure, there exists a NM-CPA (resp. IND-CCA) secure encryption scheme such that the associated encryption-with-redundancy scheme is not INT-CTXT secure.

We stress that UF-NMA is a very weak security requirement, so the implication is that allowing the redundancy computation to depend on a secret key greatly increases security as long as the base encryption scheme is strong enough. We also stress that our condition on the redundancy code is both necessary and sufficient. Still in practice, the implication is largely negative because common encryption schemes (such as standard block-cipher modes of operation) do not meet notions like NM-CPA or IND-CCA.

PERSPECTIVE. The above results do not rule out obtaining secure schemes from the encryption-with-redundancy paradigm. The results refer to the ability to prove authenticity of the encryption-with-redundancy scheme *in general*, meaning based *solely* on assumed privacy attributes of the base encryption scheme and attributes of the redundancy code.

One might consider encryption with some specific redundancy code using as base encryption scheme a block-cipher based mode of operation that is only IND-CPA secure, and yet be able to prove authenticity by analyzing the encryption-with-redundancy scheme directly based on the assumption that the block-cipher is a pseudorandom permutation. This would not contradict the above results. What the above results do is show that the intuition that privacy helps integrity is flawed. Encryption-with-redundancy might work, but not for that reason. If a specific scheme such as the example we just mentioned works, it is not because of the privacy provided by the encryption, but, say, because of the pseudorandomness of the block-cipher. In practice this tells us that to get secure encryption-with-redundancy schemes we must look at specific constructions and analyze them directly. This is what we do next.

## 1.2 Encryption with NCBC

We consider a variant of (random-IV) CBC mode encryption in which the enciphering corresponding to the last message block is done under a key different from that used for the other blocks. We call this mode NCBC. Here we are able to obtain positive results for both public and secret redundancy functions.

We show that if secret redundancy is used, quite simple and efficient redundancy codes suffice for the NCBC with redundancy scheme to provide authenticity. The redundancy code should satisfy the property called *AXU* (*Almost Xor Universal*) in [21, 27]. (Any Universal-2 function [30] has this property and there are other efficient constructs as well [15, 10, 1].) On the other hand we show that if the redundancy is public, then authenticity of the NCBC with redundancy scheme is guaranteed if the redundancy code is *XOR-collision-resistant*. (The latter, a cryptographic property we define, can be viewed either as a variant of the standard collision-resistance property, or as an extension

of the AXU property to the case where the key underlying the function is public.) These results assume the underlying block-cipher is a strong pseudorandom permutation in the sense of [23].

These results should be contrasted with what we know about encryption with redundancy using the standard CBC mode as the base encryption scheme. Wagner’s attack, pointed out in [26], and discussed in more depth in Section 6, implies that *no* public redundancy code will, in conjunction with CBC encryption, yield an encryption-with-redundancy scheme possessing integrity of ciphertexts. In the case where the redundancy is secret, Krawczyk [22] shows that it suffices for the redundancy code to be a MAC secure against chosen-message attack, but this is a strong condition on the redundancy code compared to the AXU property that suffices for NCBC. Thus, the simple modification consisting of enciphering under a different key for the last block substantially enhances CBC with regard to its ability to provide authenticity under the encryption-with-redundancy paradigm.

### 1.3 Related work

Preneel gives an overview of existing authentication methods [26] that includes much relevant background. A comprehensive treatment of authenticated encryption—the goal of joint privacy and authenticity—is provided in [7]. They relate different notions of privacy and authenticity to compare their relative strengths.

Encryption-with-redundancy is one of many approaches to the design of authenticated encryption schemes. Another general approach is “generic composition:” combine an encryption scheme with a MAC in some way. This is analyzed in [7], who consider the following generic composition methods: *Encrypt-and-mac*, *Mac-then-encrypt*, *Encrypt-then-mac*. For each of these methods they consider two notions of integrity, namely integrity of ciphertexts and a weaker notion of integrity of plaintexts, and then, assuming the base encryption scheme is IND-CPA and the MAC is secure against chosen-message attack, indicate whether or not the method has the integrity property in question. Krawczyk’s recent work [22] considers the same methods from the point of view of building “secure channels” over insecure networks. The drawback of the generic composition approach compared to the encryption-with-redundancy approach is that some MACs might be less efficient than redundancy codes, and that public redundancy avoids the additional independent key that is required for MACs.

Another general paradigm is “encode then encipher” [8]—add randomness and redundancy and then encipher rather than encrypt. Encode then encipher requires a variable-input length strong pseudorandom permutation, which can be relatively expensive to construct.

Let  $SNCBC[F, \mathcal{R}C]$  denote NCBC encryption with block-cipher  $F$  and secret redundancy provided by an efficient AXU redundancy code  $\mathcal{R}C$ . We compare this to other authenticated encryption schemes such as RPC mode [20], IACBC [18], and OCB [28]. RPC is computation and space inefficient compared to all the other methods. IACBC and OCB have cost comparable to that of  $SNCBC[F, \mathcal{R}C]$ , but OCB is parallelizable.

Encryption-with-redundancy is one of many approaches to simultaneously achieving privacy and authenticity. Our goal was to analyze and better understand this approach. We do not suggest it is superior to other approaches.

## 2 Definitions

A *string* is a member of  $\{0, 1\}^*$ . The notation  $\parallel$  denotes the concatenation.

EXTENDED ENCRYPTION SCHEMES. The usual syntax of a symmetric encryption scheme (cf. [4]) is that encryption and decryption depend on a key shared between sender and receiver but not given to the adversary. We wish to consider a setting where operations depend, in addition to the shared

key, on some public information, such as a hash function. The latter may be key based. (Think of the key as having been chosen at random at design time and embedded in the hash function.) All parties including the adversary have access to this key, which we call the *common key*. We need to model it explicitly because security depends on the random choice of this key even though it is public. This requires a change in encryption scheme syntax. Accordingly we define an *extended encryption scheme* which extends the usual symmetric encryption scheme by addition of another key generation algorithm. Specifically an extended encryption scheme  $\mathcal{EE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{E}, \mathcal{D})$  consists of four algorithms as follows. The randomized *common key generation* algorithm  $\mathcal{K}_c$  takes input a security parameter  $k \in \mathbb{N}$  and in time  $\text{poly}(k)$  returns a key  $K_c$ ; we write  $K_c \stackrel{R}{\leftarrow} \mathcal{K}_c(k)$ . The randomized *secret key generation* algorithm  $\mathcal{K}_s$  also takes input  $k \in \mathbb{N}$  and in time  $\text{poly}(k)$  returns a key  $K_s$ ; we write  $K_s \stackrel{R}{\leftarrow} \mathcal{K}_s(k)$ . We let  $K = (K_c, K_s)$ . The *encryption* algorithm  $\mathcal{E}$  is either randomized or stateful. It takes  $K$  and a *plaintext*  $M$  and in time  $\text{poly}(k, |M|)$  returns a *ciphertext*  $C = \mathcal{E}_K(M)$ ; we write  $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$ . (If randomized, it flips coins, anew upon each invocation. If stateful, it maintains a state which it updates upon each invocation.) The deterministic and stateless *decryption* algorithm  $\mathcal{D}$  takes the key  $K$  and a string  $C$  and in time  $\text{poly}(k, |C|)$  returns either the corresponding plaintext  $M$  or the distinguished symbol  $\perp$ ; we write  $x \leftarrow \mathcal{D}_K(C)$ . We require that  $\mathcal{D}_K(\mathcal{E}_K(M)) = M$  for all  $M \in \{0, 1\}^*$ .

Notice that it is not apparent from the syntax why there are two keys because they are treated identically. The difference will surface when we consider security: we will view the legitimate users as possessing  $K_s$  while both they and the adversary have  $K_c$ . (It also surfaces in something we don't consider explicitly here, which is a multi-user setting. In that case, although  $K_s$  will be generated anew for each pair of users,  $K_c$  may be the same across the whole system.)

A standard symmetric encryption scheme, namely one where there is no common key, can be recovered as the special case where the common key generation algorithm  $\mathcal{K}_c$  returns the empty string. Formally, we say that  $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  is a (symmetric) encryption scheme if  $\mathcal{EE} = (\mathcal{K}_c, \mathcal{K}, \mathcal{E}, \mathcal{D})$  is an extended encryption scheme where  $\mathcal{K}_c$  is the algorithm which on any input returns the empty string. When the common key  $K_c$  is the empty string we may also omit it in the input given to the adversary.

NOTIONS OF SECURITY. Notions of security for symmetric encryption schemes are easily adapted to extended encryption schemes by giving the adversary the common key as input. Via the formal definitions shown below and this discussion we will summarize the definitions we need.

We let  $\mathcal{EE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{E}, \mathcal{D})$  be the extended encryption scheme whose security we are defining. The formalizations, given in Definition 2.1 and Definition 2.2, associate to each notion of security and each adversary an *experiment*, and based on that, an *advantage*. The latter is a function of the security parameter that measures the success probability of the adversary. Asymptotic notions of security result by asking this function to be negligible for adversaries of time-complexity polynomial in the security parameter. Concrete security assessments can be made by associating to the scheme another advantage function that for each value of the security parameter and given resources for an adversary returns the maximum, over all adversaries limited to the given resources, of the advantage of the adversary.

Note that these definitions apply to standard symmetric encryption schemes too, since as per our conventions the latter are simply the special case of extended encryption schemes in which the common key generation algorithm returns the empty string.

PRIVACY. The basic and weakest natural notion of privacy is IND-CPA. We use one of the formalizations of [4] which adapts that of [14] to the symmetric setting. A challenge bit  $b$  is chosen, the adversary is given  $K_c$ , and can query, adaptively and as often as it likes, the left-or-right encryption oracle. The adversary wins if it can guess  $b$ . For IND-CCA the adversary gets in addition a

decryption oracle but loses if it queries it on any ciphertext returned by the left-or-right encryption oracle.

Non-malleability captures, intuitively, the inability of an adversary to change a ciphertext into another one such that the underlying plaintexts are meaningfully related [12]. We do not formalize it directly as per [12, 5] but rather via the equivalent indistinguishability under parallel chosen-ciphertext attack characterization of [9, 19]. (This facilitates our proofs.) The adversary gets the left-or-right encryption oracle and must then decide on a vector of ciphertexts  $\mathbf{c}$ . (It loses if they contain an output of the left-or-right encryption oracle.) It is given their corresponding decryptions  $\mathbf{p}$  and then wins if it guesses the challenge bit.

The formal definition of privacy is below with the associated experiments.

**Definition 2.1 [Privacy]** Let  $\mathcal{EE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{E}, \mathcal{D})$  be an extended encryption scheme,  $b \in \{0, 1\}$  a challenge bit and  $k \in \mathbb{N}$  the security parameter. Let  $A$  be an adversary that outputs a bit  $d$ . The *left-or-right* encryption oracle  $\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))$ , given to the adversary  $A$ , takes input a pair  $(x_0, x_1)$  of equal-length messages, computes ciphertext  $X \leftarrow \mathcal{E}_K(x_b)$ , and returns  $X$  to the adversary. (It flips coins, or updates state for the encryption function, as necessary. If the input messages are not of equal length it returns the empty string.) Now consider the following experiments each of which returns a bit.

<p>Experiment <math>\mathbf{Exp}_{\mathcal{EE}, A}^{\text{ind-cpa-}b}(k)</math></p> <p><math>K_c \xleftarrow{R} \mathcal{K}_c(k)</math></p> <p><math>K_s \xleftarrow{R} \mathcal{K}_s(k)</math></p> <p><math>K \leftarrow (K_c, K_s)</math></p> <p><math>d \leftarrow A^{\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))}(k, K_c)</math></p> <p>return <math>d</math></p>	<p>Experiment <math>\mathbf{Exp}_{\mathcal{EE}, A}^{\text{ind-cca-}b}(k)</math></p> <p><math>K_c \xleftarrow{R} \mathcal{K}_c(k); K_s \xleftarrow{R} \mathcal{K}_s(k)</math></p> <p><math>K \leftarrow (K_c, K_s)</math></p> <p><math>d \leftarrow A^{\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b)), \mathcal{D}_K(\cdot)}(k, K_c)</math></p> <p>If <math>\mathcal{D}_K(\cdot)</math> was never queried on an output of <math>\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))</math> then return <math>d</math> else return 0</p>	<p>Experiment <math>\mathbf{Exp}_{\mathcal{EE}, A}^{\text{nm-cpa-}b}(k)</math></p> <p><math>K_c \xleftarrow{R} \mathcal{K}_c(k); K_s \xleftarrow{R} \mathcal{K}_s(k)</math></p> <p><math>K \leftarrow (K_c, K_s)</math></p> <p><math>(\mathbf{c}, s) \leftarrow A_1^{\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))}(k, K_c)</math></p> <p><math>\mathbf{p} \leftarrow (\mathcal{D}_K(c_1), \dots, \mathcal{D}_K(c_n))</math></p> <p><math>d \leftarrow A_2(\mathbf{p}, \mathbf{c}, s)</math></p> <p>If <math>\mathbf{c}</math> contains no ciphertext output by <math>\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))</math> then return <math>d</math> else return 0</p>
--	---	---

For each notion of privacy  $\text{sss-aaa} \in \{\text{ind-cpa}, \text{ind-cca}, \text{nm-cpa}\}$  we associate to the adversary  $A$  a corresponding advantage defined via

$$\mathbf{Adv}_{\mathcal{EE}, A}^{\text{sss-aaa}}(k) = \Pr \left[ \mathbf{Exp}_{\mathcal{EE}, A}^{\text{sss-aaa-}1}(k) = 1 \right] - \Pr \left[ \mathbf{Exp}_{\mathcal{EE}, A}^{\text{sss-aaa-}0}(k) = 1 \right].$$

For each security notion  $\text{SSS-AAA} \in \{\text{IND-CPA}, \text{IND-CCA}, \text{NM-CPA}\}$ , the scheme  $\mathcal{EE}$  is said to be *SSS-AAA secure* if the corresponding advantage function,  $\mathbf{Adv}_{\mathcal{EE}, F}^{\text{sss-aaa}}(\cdot)$  of any adversary  $F$  whose time-complexity is polynomial in  $k$ , is negligible. ■

**INTEGRITY.** The formalization of integrity follows [7]. The adversary is allowed to mount a chosen-message attack on the scheme, modeled by giving it access to an encryption oracle. Success is measured by its ability to output a “new” ciphertext that makes the decryption algorithm output a plaintext rather than reject by outputting  $\perp$ . Different interpretations of the “new” ciphertext give rise to two different notions — *integrity of plaintexts (INT-PTXT)* and *integrity of ciphertexts (INT-CTXT)*. The formal definition of integrity is below with the associated experiments.

**Definition 2.2 [Integrity]** Let  $\mathcal{EE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{E}, \mathcal{D})$  be an extended encryption scheme, and  $k \in \mathbb{N}$  the security parameter. Let  $B$  be an adversary that has access to the encryption oracle and outputs a ciphertext. Now consider the following experiments.

Experiment $\mathbf{Exp}_{\mathcal{E}\mathcal{E},B}^{\text{int-ctxt}}(k)$ $K_c \stackrel{R}{\leftarrow} \mathcal{K}_c; K_s \stackrel{R}{\leftarrow} \mathcal{K}_s; K \leftarrow (K_c, K_s)$ $C \leftarrow B^{\mathcal{E}_K(\cdot)}(k, K_c)$ If $\mathcal{D}_K(C) \neq \perp$ and $C$ was never a response of $\mathcal{E}_K(\cdot)$ then return 1 else return 0	Experiment $\mathbf{Exp}_{\mathcal{E}\mathcal{E},B}^{\text{int-ptxt}}(k)$ $K_c \stackrel{R}{\leftarrow} \mathcal{K}_c; K_s \stackrel{R}{\leftarrow} \mathcal{K}_s; K \leftarrow (K_c, K_s)$ $C \leftarrow B^{\mathcal{E}_K(\cdot)}(k, K_c)$ If $M \stackrel{\text{def}}{=} \mathcal{D}_K(C) \neq \perp$ and $M$ was never a query to $\mathcal{E}_K(\cdot)$ then return 1 else return 0
--	--

We associate to the adversary  $B$  corresponding advantages defined via,

$$\mathbf{Adv}_{\mathcal{E}\mathcal{E},B}^{\text{int-ctxt}}(k) = \Pr[\mathbf{Exp}_{\mathcal{E}\mathcal{E},B}^{\text{int-ctxt}}(k) = 1] \quad \Bigg| \quad \mathbf{Adv}_{\mathcal{E}\mathcal{E},B}^{\text{int-ptxt}}(k) = \Pr[\mathbf{Exp}_{\mathcal{E}\mathcal{E},B}^{\text{int-ptxt}}(k) = 1]$$

The scheme  $\mathcal{E}\mathcal{E}$  is said to be *INT-CTXT secure* (resp. *INT-PTXT secure*) if the advantage function  $\mathbf{Adv}_{\mathcal{E}\mathcal{E},F}^{\text{int-ctxt}}(\cdot)$  (resp.  $\mathbf{Adv}_{\mathcal{E}\mathcal{E},F}^{\text{int-ptxt}}(\cdot)$ ) of any adversary  $F$  whose time-complexity is polynomial in  $k$ , is negligible.  $\blacksquare$

Note from the above definition that INT-CTXT is a stronger notion of integrity than INT-PTXT (i.e. if a scheme is INT-CTXT secure, it is also INT-PTXT secure) [7].

NOTIONS FOR ADVERSARY EXECUTION. In proofs using reductions, we will have one adversary  $A$  running another adversary  $B$ , and  $A$  will provide the execution environment for  $B$ . In the code for such reductions, the notation  $B \Rightarrow x$  means that  $B$  is making an oracle query  $x$ , and the notation  $B \Leftarrow y$  means that  $B$  is being provided  $y$  as its query response.

### 3 The encryption-with-redundancy paradigm

We describe the paradigm in a general setting, as a transform that associates to any given symmetric encryption scheme and any given “redundancy code” an extended encryption scheme. We first define the syntax for redundancy codes, then detail the constructions, separating the cases of public and secret redundancy, and conclude by observing that the transform always preserves privacy. This leaves later sections to investigate the difficult issue, namely the integrity of the extended encryption scheme with redundancy.

REDUNDANCY CODES. A *redundancy code*  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  consists of two algorithms  $\mathcal{K}_r$  and  $\mathcal{H}$ . The randomized key generation algorithm  $\mathcal{K}_r$  takes a security parameter  $k$  and in time  $\text{poly}(k)$  returns a key  $K_r$ ; we write  $K_r \stackrel{R}{\leftarrow} \mathcal{K}_r(k)$ . The deterministic redundancy computation algorithm  $\mathcal{H}$  takes  $K_r$  and a string  $M \in \{0,1\}^*$  and in time  $\text{poly}(k, |M|)$  returns a string  $\tau$ ; we write  $\tau \leftarrow \mathcal{H}_{K_r}(M)$ . Usually the length of  $\tau$  is  $\ell(k)$  where  $\ell(\cdot)$ , an integer valued function that depends only on the security parameter, is called the *output length* of the redundancy code. We say that the redundancy is *public* if the key  $K_r$  is public and known to the adversary. We say the redundancy is *secret* if  $K_r$  is part of the shared secret key.

EXTENDED ENCRYPTION SCHEMES WITH REDUNDANCY. Let  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be a given (symmetric) encryption scheme, which we will call the *base* encryption scheme. Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a given redundancy code as above. We define an associated *extended encryption scheme with public redundancy* and an associated *extended encryption scheme with secret redundancy*.

**Construction 3.1** The extended encryption scheme with public redundancy  $\mathcal{EPR} = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ , associated to base encryption scheme  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  and redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$ , is defined as follows:

Algorithm $\mathcal{K}_c(k)$ $K_r \xleftarrow{R} \mathcal{K}_r(k)$ return $K_r$	Algorithm $\mathcal{K}_s(k)$ $K_e \xleftarrow{R} \mathcal{K}_e(k)$ return $K_e$	Algorithm $\overline{\mathcal{E}}_{\langle K_e, K_r \rangle}(M)$ $\tau \leftarrow \mathcal{H}_{K_r}(M)$ $C \xleftarrow{R} \mathcal{E}_{K_e}(M \parallel \tau)$ return $C$	Algorithm $\overline{\mathcal{D}}_{\langle K_e, K_r \rangle}(C)$ $P \leftarrow \mathcal{D}_{K_e}(C)$ Parse $P$ as $M \parallel \tau$ if $\tau \neq \mathcal{H}_{K_r}(M)$ then return $\perp$ else return $M$
---	---	--	--

Note that the common-key generation algorithm returns the key for the redundancy function, which is thus available to the adversary. That is why we say the redundancy is public. ■

**Construction 3.2** The extended encryption scheme with secret redundancy  $\mathcal{ESR} = (\mathcal{K}_c, \mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ , associated to base encryption scheme  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  and redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$ , is defined as follows:

Algorithm $\mathcal{K}_c(k)$ return $\varepsilon$	Algorithm $\mathcal{K}_s(k)$ $K_e \xleftarrow{R} \mathcal{K}_e(k)$ $K_r \xleftarrow{R} \mathcal{K}_r(k)$ return $\langle K_e, K_r \rangle$	Algorithm $\overline{\mathcal{E}}_{\langle K_e, K_r \rangle}(M)$ $\tau \leftarrow \mathcal{H}_{K_r}(M)$ $C \xleftarrow{R} \mathcal{E}_{K_e}(M \parallel \tau)$ return $C$	Algorithm $\overline{\mathcal{D}}_{\langle K_e, K_r \rangle}(C)$ $N \leftarrow \mathcal{D}_{K_e}(C)$ Parse $N$ as $M \parallel \tau$ if $\tau \neq \mathcal{H}_{K_r}(M)$ then return $\perp$ else return $M$
--	---	--	---

Note that the common key generation algorithm  $\mathcal{K}_c$  returns the empty string  $\varepsilon$ . We may omit the algorithm  $\mathcal{K}_c$  and write  $\mathcal{ESR} = (\mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ . The key for the redundancy function is part of the secret key not available to the adversary. ■

The symbol  $\perp$  is a distinct symbol that indicates that the ciphertext is not valid. When we refer to an extended encryption scheme with redundancy in general we mean either of the above, and denote it by  $\mathcal{ER}$ .

**PRIVACY IS PRESERVED.** We now present a theorem regarding the privacy of an extended encryption scheme with redundancy. It applies both to the case of public and to the case of secret redundancy. The theorem below says that the encryption scheme with redundancy inherits the privacy of the base symmetric encryption scheme regardless of the redundancy code being used. This means that privacy depends only on the underlying encryption scheme, not on the redundancy code. The proof is straightforward and can be found in Appendix B.1.

**Theorem 3.3 [Privacy of an extended encryption scheme with redundancy]** Let  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be a symmetric encryption scheme and let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code. Let  $\mathcal{ER} = (\mathcal{K}_c, \mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$  be an associated extended encryption scheme with redundancy, either public or secret. Then if  $\mathcal{SE}$  is IND-CPA (resp. IND-CCA, NM-CPA) secure, so is  $\mathcal{ER}$ . ■

For simplicity we have stated the theorem with reference to asymptotic notions of security but we remark that the reduction in the proof is tight, and a concrete security statement reflecting this can be derived from the proof.

## 4 Encryption with public redundancy

Here we will show that in general the encryption with public redundancy paradigm fails in a strong way, meaning there is a base encryption scheme such that for *all* choices of public redundancy code, the associated extended encryption scheme with public redundancy (cf. Construction 3.1) fails to provide integrity. This is true regardless of the security property of the base encryption scheme (i.e. IND-CPA, NM-CCA, or IND-CCA).

We stress the strength of the conclusion. Privacy, even in its strongest form, namely IND-CCA, is not sufficient to endow a redundancy-augmented encryption scheme with integrity, even if you choose any redundancy code you like, however complex, as long as it is not secret-key based.

Note that this is fairly easy to see in the public key setting, namely in asymmetric encryption schemes. Recall that in an asymmetric encryption scheme, the key generation algorithm generates a public, secret key pair and the encryption algorithm uses the public key to encrypt a message and the decryption algorithm uses a secret key to decrypt a ciphertext. If an asymmetric encryption scheme is used as the base encryption scheme of an extended encryption scheme, clearly the latter does not provide authenticity if the underlying redundancy code is not secret-key based because anybody who knows the public key for the encryption algorithm will be able to produce a “valid” ciphertext. This kind of attack can also be applied to the symmetric key setting because any “secure” asymmetric encryption scheme can be transformed into a “secure” symmetric key encryption scheme where the public, secret key pair  $(pk, sk)$  generated by the key generation algorithm of the asymmetric encryption scheme is used as the symmetric key for the encryption scheme.

The result follows the paradigm of similar negative results in [4, 7]. We must make the minimal assumption that some asymmetric encryption scheme  $\mathcal{AE}$  secure in the given sense exists, else the question is moot. We then modify the given asymmetric encryption scheme  $\mathcal{AE}$  to a symmetric encryption scheme  $\mathcal{SE}$  so that when  $\mathcal{SE}$  becomes the base encryption scheme of the extended encryption scheme with public redundancy, we can provide an attack on the integrity of the latter. The following theorem states the result more formally and the proof follows.

**Theorem 4.1** Suppose there exists an asymmetric encryption scheme  $\mathcal{AE}$  which is IND-CCA (resp. IND-CPA, NM-CPA) secure. Then there exists a symmetric encryption scheme  $\mathcal{SE}$  which is also IND-CCA (resp. IND-CPA, NM-CPA) secure but, for any redundancy code  $\mathcal{RC}$ , the extended encryption scheme with public redundancy  $\mathcal{EPR}$  associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is not INT-CTXT secure. ■

We remark that the proof actually shows something stronger, namely that  $\mathcal{EPR}$  is not even INT-PTXT secure.

**Proof of Theorem 4.1:** The proof of Theorem 4.1 consists of two parts: in the first part, we present an encryption scheme  $\mathcal{SE}$  (constructed based on  $\mathcal{AE}$ ) and show that the associated extended encryption scheme with public redundancy  $\mathcal{EPR}$  is not INT-PTXT secure, and in the second part, we show that the constructed encryption scheme  $\mathcal{SE}$  inherits the privacy attribute of  $\mathcal{AE}$ . We now show the first part below.

Let  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$  be the given asymmetric encryption scheme. Let  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be defined as follows:

$$\begin{array}{l|l|l} \text{Algorithm } \mathcal{K}_e(k) & \text{Algorithm } \mathcal{E}_{(pk,sk)}(M) & \text{Algorithm } \mathcal{D}_{(pk,sk)}(C) \\ (pk, sk) \stackrel{R}{\leftarrow} \mathcal{K}(k) & C \leftarrow (pk, \mathcal{E}'_{pk}(M)) & \text{Parse } C \text{ as } (x, C') \\ \text{Return } (pk, sk) & \text{return } C & \text{If } x \neq pk \text{ then return } \perp \\ & & \text{else return } \mathcal{D}'_{sk}(C') \end{array}$$

Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be any redundancy code and let  $\mathcal{EPR} = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  be the extended encryption scheme with public redundancy associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  as per Construction 3.1. We now show that the scheme  $\mathcal{EPR}$  is not INT-PTXT secure. Consider the following adversary  $F$  attacking  $\mathcal{EPR}$ . It gets an oracle for encryption  $\bar{\mathcal{E}}_{((pk,sk),K_r)}(\cdot)$  and invokes it just once to get the public key  $pk$  (which is in effect the encryption key). Note that as per Definition 2.2 it gets the common key  $K_r$  as input.

Algorithm  $F_{\langle (pk, sk), K_r \rangle}^{\bar{\mathcal{E}}(\cdot)}(k, K_r)$

$M_1 \leftarrow 0$ ;  $M_2 \leftarrow 1$   
 $(pk, C_1) \leftarrow \bar{\mathcal{E}}_{\langle (pk, sk), K_r \rangle}(M_1)$   
 $\tau \leftarrow \mathcal{H}_{K_r}(M_2)$   
 $C_2 \leftarrow \mathcal{E}'_{pk}(M_2 || \tau)$   
 return  $(pk, C_2)$

We claim that  $\mathbf{Adv}_{\mathcal{EPR}_F}^{\text{int-ptxt}}(k) = 1$ . Indeed when the ciphertext  $C = (pk, C_2)$  output by  $F$  is given to the decryption algorithm  $\bar{\mathcal{D}}_{\langle (pk, sk), K_r \rangle}(\cdot)$ , the latter will return  $M_2$  since  $\mathcal{D}_{\langle pk, sk \rangle}(pk, C_2)$  will output  $M_2 || \tau$ . In particular, the ciphertext is accepted because it is “valid” (the decryption algorithm does not return  $\perp$ ) and its corresponding plaintext  $M_2$  is “new” (was never a query to the encryption oracle).

It now remains to show that  $\mathcal{SE}$  inherits the privacy attribute of  $\mathcal{AE}$ , be it IND-CCA, IND-CPA or NM-CPA. We will show it explicitly for the IND-CCA case only here, but it can be shown for the other two cases in an analogous manner.

Recall that the standard attack model against the IND-CCA security of asymmetric encryption schemes differs from that of symmetric encryption schemes in that the adversary does not get any encryption oracle in the former. However, rather than recalling the precise definition of the attack model for asymmetric encryption scheme here, we adopt the attack model for the symmetric encryption schemes to that for the asymmetric schemes because it suffices to illustrate our point. Let  $A$  be an adversary attacking  $\mathcal{SE}$  in the IND-CCA sense having time-complexity  $\text{poly}(k)$ . We construct an adversary  $B$  that runs  $A$  as a subroutine in order to attack  $\mathcal{AE}$  in the IND-CCA sense. The algorithm for  $B$  is shown below:

Algorithm  $B_{\mathcal{E}'_{pk}(\mathcal{LR}(\cdot, \cdot, b)), \mathcal{D}'_{sk}(\cdot)}(k, pk)$

Run  $A$  on input  $k$  //  $k$  is the security parameter  
 When  $A$  queries the encryption oracle with  $(x_0, x_1)$  do:  
 $A \leftarrow (pk, \mathcal{E}'_{pk}(\mathcal{LR}(x_0, x_1, b)))$   
 When  $A$  queries the decryption oracle with  $y$  do:  
 Parse  $y$  as  $(x, y')$   
 If  $x \neq pk$  then  $A \leftarrow \perp$  else  $A \leftarrow \mathcal{D}'_{sk}(y')$   
 Until  $A$  outputs  $d$   
 return  $d$

We claim that  $\mathbf{Adv}_{\mathcal{AE}, B}^{\text{ind-cca}}(k) = \mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cca}}(k)$ . Note that using its own encryption oracle  $\mathcal{E}'_{pk}(\mathcal{LR}(x_0, x_1, b))$  and decryption oracle  $\mathcal{D}'_{sk}(y')$ ,  $B$  simulates the responses to  $A$ 's encryption and decryption queries exactly in the same way as  $A$ 's real encryption oracle  $\mathcal{E}_{\langle pk, sk \rangle}(\mathcal{LR}(\cdot, \cdot), b)$  and decryption oracle  $\mathcal{D}_{\langle pk, sk \rangle}(\cdot)$  would compute. Since  $B$  outputs the same bit as  $A$  does, the following equation holds for both  $b = 0$  and  $b = 1$ :

$$\Pr \left[ \mathbf{Exp}_{\mathcal{AE}, B}^{\text{ind-cca-}b}(k) = 1 \right] = \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cca-}b}(k) = 1 \right],$$

Hence,  $\mathbf{Adv}_{\mathcal{AE}, B}^{\text{ind-cca}}(k) = \mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cca}}(k)$ , which means that as long as the advantage of  $B$  attacking IND-CCA of  $\mathcal{AE}$  is negligible, so is that of  $A$  attacking IND-CCA of  $\mathcal{SE}$ .  $\blacksquare$

Recall that the assumption we made for Theorem 4.1 is that some asymmetric encryption scheme  $\mathcal{AE}$  secure in the given sense exists. In the following theorem, we make a different assumption, namely that some *symmetric* encryption scheme  $\mathcal{SE}'$  secure in the given sense exists, and show

that we can modify the given symmetric encryption scheme  $\mathcal{SE}'$  to a new symmetric scheme  $\mathcal{SE}$  so that when  $\mathcal{SE}$  becomes the base encryption scheme of the extended encryption scheme with public redundancy, we can provide an attack on the integrity of the latter.

**Theorem 4.2 [Encryption with public redundancy]** Suppose there exists a symmetric encryption scheme  $\mathcal{SE}'$  which is IND-CCA (resp. IND-CPA, NM-CPA) secure. Then there exists a symmetric encryption scheme  $\mathcal{SE}$  which is also IND-CCA (resp. IND-CPA, NM-CPA) secure but, for any redundancy code  $\mathcal{RC}$ , the extended encryption scheme with public redundancy  $\mathcal{EPR}$  associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is not INT-CTXT secure. ■

We remark that as in the proof of Theorem 4.1, the proof here also shows something stronger, namely that  $\mathcal{EPR}$  is not even INT-PTXT secure.

**Proof of Theorem 4.2:** The proof of Theorem 4.2 consists of two parts: in the first part, we present an encryption scheme  $\mathcal{SE}$  (constructed based on  $\mathcal{SE}'$ ) and show that the associated extended encryption scheme with public redundancy  $\mathcal{EPR}$  is not INT-PTXT secure, and in the second part, we show that the constructed encryption scheme  $\mathcal{SE}$  inherits the privacy of  $\mathcal{SE}'$ . We now show the first part below.

Let  $\mathcal{SE}' = (\mathcal{K}_e, \mathcal{E}', \mathcal{D}')$  be the given encryption scheme. Let  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be the encryption scheme in which the key generation algorithm is unchanged and the other two algorithms are defined as follows:

$$\begin{array}{l|l} \text{Algorithm } \mathcal{E}_{K_e}(M) & \text{Algorithm } \mathcal{D}_{K_e}(C) \\ \hline C \leftarrow 0 \parallel \mathcal{E}'_{K_e}(M) & \text{Parse } C \text{ as } b \parallel C' \text{ where } b \text{ is a bit} \\ \text{return } C & \text{If } b = 0 \text{ then return } \mathcal{D}'_{K_e}(C') \\ & \text{else return } C' \end{array}$$

Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be any redundancy code and let  $\mathcal{EPR} = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  be the extended encryption scheme with public redundancy associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  as per Construction 3.1. We now show that the scheme  $\mathcal{EPR}$  is not INT-PTXT secure. Consider the following adversary  $F$  attacking  $\mathcal{EPR}$ . It gets an oracle for encryption  $\bar{\mathcal{E}}_{(K_e, K_r)}(\cdot)$  but does not need to invoke it. Note that as per Definition 2.2 it gets the common key  $K_r$  as input.

Adversary  $F^{\bar{\mathcal{E}}_{(K_e, K_r)}(\cdot)}(k, K_r)$   
 $M \leftarrow 0$  // Choose a message  
 $\tau \leftarrow \mathcal{H}_{K_r}(M)$  // Compute the redundancy  
return  $1 \parallel M \parallel \tau$  // Return the forgery ciphertext

We claim that  $\mathbf{Adv}_{\mathcal{EPR}, F}^{\text{int-ptxt}}(k) = 1$ . Indeed when the ciphertext  $C = 1 \parallel M \parallel \tau$  output by  $F$  is given to the decryption algorithm  $\bar{\mathcal{D}}_{(K_e, K_r)}(\cdot)$ , the latter will return  $M$  since  $\mathcal{D}_{K_e}(1 \parallel M \parallel \tau)$  will output  $M \parallel \tau$ . In particular, the ciphertext is accepted because it is “valid” (the decryption algorithm does not return  $\perp$ ) and its corresponding plaintext  $M$  is “new” (was never a query to the encryption oracle).

Having shown the first part of the theorem, it now remains to show the second part of the theorem—that is,  $\mathcal{SE}$  inherits the privacy of  $\mathcal{SE}'$ , be it IND-CCA, IND-CPA or NM-CPA. This is a simple reduction argument. We will show it explicitly for the IND-CCA case only, but for the other two cases (i.e. IND-CPA, NM-CPA) it can be also shown in a similar manner.

Let  $A$  be an adversary attacking  $\mathcal{SE}$  in the IND-CCA sense. We construct an adversary  $B$  that runs  $A$  as a subroutine in order to attack  $\mathcal{SE}'$  in the IND-CCA sense. The algorithm for  $B$  is shown below:

Algorithm  $B^{\mathcal{E}'_{K_e}(\mathcal{LR}(\cdot, \cdot, b)), \mathcal{D}'_{K_e}(\cdot)}(k)$

Run  $A$  on input  $k$  //  $k$  is the security parameter  
 When  $A$  queries the encryption oracle with  $(x_0, x_1)$  do:  
    $A \leftarrow 0 \parallel \mathcal{E}'_{K_e}(\mathcal{LR}(x_0, x_1, b))$   
 When  $A$  queries the decryption oracle with  $y$  do:  
   Parse  $y$  as  $a \parallel y'$  where  $a$  is a bit  
   If  $a = 0$  then  $A \leftarrow \mathcal{D}'_{K_e}(y')$  else  $A \leftarrow y'$   
 Until  $A$  outputs  $d$   
 return  $d$

We claim that  $\mathbf{Adv}_{\mathcal{SE}', B}^{\text{ind-cca}}(k) = \mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cca}}(k)$ . Notice that  $B$  is simulating the outputs of the oracles  $\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$  and  $\mathcal{D}_{K_e}(\cdot)$  using its own oracles  $\mathcal{E}'_{K_e}(\mathcal{LR}(\cdot, \cdot, b)), \mathcal{D}'_{K_e}(\cdot)$  exactly in the same way as the actual encryption and decryption algorithms for the scheme  $\mathcal{SE}$ .  $B$  never queries its decryption oracle on an output of its (left-or-right) encryption oracle as long as  $A$  never queries its decryption oracle on its encryption oracle's output. This is because  $B$  queries its decryption oracle only when  $A$ 's decryption query has a valid format (i.e. it starts with the bit 0), and if  $B$  queried the decryption oracle on its encryption oracle's output,  $A$ 's decryption oracle query must have been also an output of  $A$ 's encryption oracle since their encryption oracle outputs differ by just the prepended bit 0. Since  $B$  outputs the same bit as  $A$  does, the following equation holds for both  $b = 0$  and  $b = 1$ :

$$\Pr \left[ \mathbf{Exp}_{\mathcal{SE}', B}^{\text{ind-cca-}b}(k) = 1 \right] = \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cca-}b}(k) = 1 \right]$$

Hence,  $\mathbf{Adv}_{\mathcal{SE}', B}^{\text{ind-cca}}(k) = \mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cca}}(k)$ , which means that as long as the advantage of  $B$  attacking IND-CCA of  $\mathcal{SE}'$  is negligible, so is that of  $A$  attacking IND-CCA of  $\mathcal{SE}$ .  $\blacksquare$

## 5 Encryption with secret redundancy

In this section, we examine encryption schemes with secret redundancy in general so as to whether or not they provide integrity. We have both negative and positive results for this depending on the security property of the base encryption scheme.

The negative result shows that in general the encryption with secret redundancy paradigm does not provide INT-CTXT if the base encryption is only IND-CPA secure, meaning there is an IND-CPA secure base encryption scheme such that for *all* choices of secret redundancy code, the associated extended encryption scheme with secret redundancy (cf. Construction 3.2) fails to provide INT-CTXT. The positive result shows that the paradigm provides integrity in general if the base encryption schemes have a stronger privacy property (i.e. NM-CPA or IND-CCA) and certain properties are satisfied by the redundancy codes.

The following theorem states the negative result where the base encryption scheme is IND-CPA secure.

**Theorem 5.1 [IND-CPA encryption with secret redundancy]** Suppose there exists a symmetric encryption scheme  $\mathcal{SE}'$  which is IND-CPA secure. Then there exists a symmetric encryption scheme  $\mathcal{SE}$  which is also IND-CPA secure but, for any redundancy code  $\mathcal{RC}$ , the extended encryption scheme with secret redundancy  $\mathcal{ESR}$  associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is not INT-CTXT secure.  $\blacksquare$

We remark that unlike the encryption with public redundancy case, the encryption with secret redundancy *is* INT-PTXT secure even when the base encryption scheme is only IND-CPA secure, if the redundancy code is unforgeable under chosen-message attack (e.g. a MAC) [7].

**Proof of Theorem 5.1:** The proof of Theorem 5.1 consists of two parts: in the first part, we present an encryption scheme  $\mathcal{SE}$  (constructed based on  $\mathcal{SE}'$ ) and show that the associated extended encryption scheme with secret redundancy  $\mathcal{ESR}$  is not INT-CTXT secure, and in the second part, we show that the constructed encryption scheme  $\mathcal{SE}$  is IND-CPA secure if  $\mathcal{SE}'$  is IND-CPA secure. We now show the first part below.

Let  $\mathcal{SE}' = (\mathcal{K}_e, \mathcal{E}', \mathcal{D}')$  be the given encryption scheme. We construct an encryption scheme  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  based on  $\mathcal{SE}'$ , in which the key generation algorithm is unchanged and the other two algorithms are defined as follows:

$$\begin{array}{l|l} \text{Algorithm } \mathcal{E}_{K_e}(M) & \text{Algorithm } \mathcal{D}_{K_e}(C) \\ C \leftarrow 0 \parallel \mathcal{E}'_{K_e}(M) & \text{Parse } C \text{ as } b \parallel C' \text{ where } b \text{ is a bit} \\ \text{return } C & \text{return } \mathcal{D}'_{K_e}(C') \end{array}$$

Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be any redundancy code, and let  $\mathcal{ESR} = (\mathcal{K}_e, \mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$  be the extended encryption scheme with secret redundancy associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  as per Construction 3.2. We now show that the scheme  $\mathcal{ESR}$  is not INT-CTXT secure. Consider the following adversary  $F$  attacking  $\mathcal{ESR}$ . It gets an oracle for encryption  $\overline{\mathcal{E}}_{\langle K_e, K_r \rangle}(\cdot)$  and invokes it once with a chosen message. Note that as per Construction 3.2 it does *not* get  $K_r$  (the key for the redundancy function) as input because  $K_r$  is *not* a common key but part of the secret key.

Adversary  $F^{\overline{\mathcal{E}}_{\langle K_e, K_r \rangle}(\cdot)}(k)$   
 $M \leftarrow 0$  // Choose a message  
 $y \leftarrow \overline{\mathcal{E}}_{\langle K_e, K_r \rangle}(M)$   
Parse  $y$  as  $0 \parallel y'$   
return  $1 \parallel y'$  // Return the forgery ciphertext

We claim that  $\mathbf{Adv}_{\mathcal{ESR}, F}^{\text{int-ctxt}}(k) = 1$ . Indeed when the ciphertext  $C = 1 \parallel y'$  output by  $F$  is given to the decryption algorithm  $\overline{\mathcal{D}}_{\langle K_e, K_r \rangle}(\cdot)$ , the latter will return  $M$  since  $\mathcal{D}_{K_e}(1 \parallel y')$  will output  $\mathcal{D}'_{K_e}(y')$ , which is  $M \parallel \tau$ . Note that the first bit of the ciphertext  $C$  is different from that of the oracle response  $y$  which is the only query response obtained from the encryption oracle. Hence, the ciphertext is accepted because it is “valid” (the decryption algorithm does not return  $\perp$ ) and “new” (was never a response of the encryption oracle).

It now remains to show that if  $\mathcal{SE}'$  is IND-CPA secure, so is  $\mathcal{SE}$ . Let  $A$  be an adversary attacking IND-CPA of  $\mathcal{SE}$ . Using  $A$ , we can construct  $A'$  attacking IND-CPA of  $\mathcal{SE}'$  in a straightforward way. Recall that in the attack model of IND-CPA, no decryption oracle is given to the adversary. Hence, the adversary  $A'$  attacking  $\mathcal{SE}'$  needs to only simulate the left-or-right encryption oracle  $\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$  using its own left-or-right encryption oracle  $\mathcal{E}'_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$ . This can be easily done by prepending the bit 0 to the output of its encryption oracle. When  $A$  outputs a bit  $d$ ,  $A'$  returns the same bit. It is easy to see that the advantages of both adversaries are the same, and this concludes the proof. ■

For the positive result, we define below the security property required of the redundancy code.

We define a notion of *unforgeability under no message attack (UF-NMA)*, which is the weakest form of security required of a MAC (message authentication code) —roughly, the adversary wins if

it outputs a valid message and tag pair without seeing any legitimately produced message and tag pairs. A MAC and a redundancy code are syntactically identical, so we can use the same security notion for them. The formal definition is given below. Note that, in the attack model, the key to the redundancy code is not given to the adversary, indicating that the redundancy is secret.

**Definition 5.2 [Unforgeability under no message attack (UF-NMA)]** Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code. Let  $k \in \mathbb{N}$ . Let  $F$  be an adversary. Consider the following experiment:

Experiment  $\mathbf{Exp}_{\mathcal{RC}, F}^{\text{uf-nma}}(k)$   
 $K_r \xleftarrow{R} \mathcal{K}_r(k); (M, \tau) \leftarrow F(k)$   
 If  $\tau = \mathcal{H}_{K_r}(M)$  then return 1 else return 0

We define the *advantage* of the adversary via,

$$\mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(k) = \Pr \left[ \mathbf{Exp}_{\mathcal{RC}, F}^{\text{uf-nma}}(k) = 1 \right]$$

The redundancy code  $\mathcal{RC}$  is said to be *UF-NMA secure* if the function  $\mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(\cdot)$  is negligible for any adversary  $F$  whose time-complexity is polynomial in  $k$ . ■

The following theorem states the positive result, saying that if the base encryption scheme is NM-CPA or IND-CCA secure, the associated extended encryption scheme with secret redundancy provides integrity if the redundancy code is UF-NMA secure.

**Theorem 5.3** Let  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be a symmetric encryption scheme which is NM-CPA or IND-CCA secure, and let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a UF-NMA secure redundancy code. Then the extended encryption scheme with secret redundancy  $\mathcal{ESR} = (\mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is INT-CTXT secure. ■

**Proof of Theorem 5.3:** Since any IND-CCA secure encryption scheme is also NM-CPA secure [5, 19], it suffices to prove the theorem for the case where  $\mathcal{SE}$  is assumed NM-CPA secure.

Let  $B$  be an adversary having time-complexity  $\text{poly}(k)$ , and attacking INT-CTXT of  $\mathcal{ESR}$ . Using the adversary  $B$ , we will construct an adversary  $A = (A_1, A_2)$  attacking NM-CPA of  $\mathcal{SE}$  and an adversary  $F$  attacking UF-NMA of  $\mathcal{RC}$ , and show

$$\mathbf{Adv}_{\mathcal{RC}, B}^{\text{int-ctxt}}(k) \leq \mathbf{Adv}_{\mathcal{SE}, A}^{\text{nm-cpa}}(k) + \mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(k). \quad (1)$$

Furthermore, the time-complexity of both  $A$  and  $F$  will be  $\text{poly}(k)$ . By assumption, the quantities on the right-hand side of the above equation are negligible, and hence so is the quantity on the left-hand side of the equation. Hence, the theorem follows.

Algorithms for  $A$  and  $F$  will both use  $B$  as a subroutine, themselves providing answers to the oracle queries of  $B$ . The adversary  $F$  attacking UF-CMA of  $\mathcal{RC}$  replies to  $B$ 's encryption oracle queries by using random values in place of the tags that should be computed by applying  $\mathcal{H}_{K_r}(\cdot)$ . The fact that  $\mathcal{SE}$  is NM-CPA secure means that  $B$  will be unable to exploit this to increase its forgery probability. The algorithms for  $A = (A_1, A_2)$  and  $F$  are depicted in full in Figure 2.

In the algorithm, the adversary  $A_1$  queries the oracle  $\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$  with a pair  $(x||r, x||\sigma)$ , where  $x$  is the query string obtained from the adversary  $B$ ,  $\sigma$  is the redundancy function of  $x$ , and  $r$  is a random string of length  $|\sigma|$ . It is easy to see that if the bit  $b$  is 1 in the oracle  $\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$  given to  $A_1$  (meaning the right-hand side input  $(x||\sigma)$  is encrypted), then the algorithm for  $A_1$  simulates the correct oracle  $\bar{\mathcal{E}}_{K_e}(\cdot)$  for  $B$ . Since  $A_2$  returns 1 if the ciphertext  $C$  output by  $B$  is “new” and a

<p>Algorithm <math>A_1^{\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))}(k)</math></p> <p><math>K_r \xleftarrow{R} \mathcal{K}_r(k); Y \leftarrow \emptyset</math></p> <p>Run <math>B</math> on input <math>k</math></p> <p><math>B \Rightarrow x</math></p> <p><math>\sigma \leftarrow \mathcal{H}_{K_r}(x); r \xleftarrow{R} \{0, 1\}^{ \sigma }</math></p> <p><math>y \leftarrow \mathcal{E}_{K_e}(\mathcal{LR}(x  r, x  \sigma, b)); Y \leftarrow Y \cup \{y\}</math></p> <p><math>B \Leftarrow y</math></p> <p>Until <math>B</math> outputs <math>C</math> as a forgery</p> <p><math>\mathbf{c} \leftarrow (C); s \leftarrow (K_r, Y)</math></p> <p>return <math>(\mathbf{c}, s)</math></p> <p>Algorithm <math>A_2(\mathbf{p}, \mathbf{c}, s)</math></p> <p>Parse <math>\mathbf{p}</math> as <math>(M  \tau)</math></p> <p>Parse <math>\mathbf{c}</math> as <math>(C);</math> Parse <math>s</math> as <math>(K_r, Y)</math></p> <p>If <math>C \notin Y</math> and <math>\tau = \mathcal{H}_{K_r}(M)</math></p> <p>then return 1 else return 0</p>	<p>Algorithm <math>F(k)</math></p> <p><math>K_e \xleftarrow{R} \mathcal{K}_e(k)</math></p> <p>Run <math>B</math> on input <math>(k)</math></p> <p><math>B \Rightarrow x</math></p> <p><math>r \xleftarrow{R} \{0, 1\}^{\ell(k)}</math></p> <p><math>y \leftarrow \mathcal{E}_{K_e}(x  r)</math></p> <p><math>B \Leftarrow y</math></p> <p>Until <math>B</math> outputs <math>C</math> as a forgery</p> <p><math>P \leftarrow \mathcal{D}_{K_e}(C)</math></p> <p>Parse <math>P</math> as <math>M  \tau</math></p> <p>return <math>(M, \tau)</math></p>
--	---

Figure 2: The algorithms for the adversaries  $A = (A_1, A_2)$  and  $F$  attacking NM-CPA of  $\mathcal{SE}$  and UF-NMA of  $\mathcal{RC}$ , respectively, running an adversary  $B$  attacking INT-CTXT of  $\mathcal{ESR}$ .

valid forgery, the probability of  $A$  returning 1 when  $b = 1$  is equal to the success probability of  $B$ . Thus,

$$\Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{nm-cpa-1}}(k) = 1 \right] = \mathbf{Adv}_{\mathcal{ESR}, B}^{\text{int-ctxt}}(k).$$

If the bit  $b$  is 0 and the input is  $(x||r, x||\sigma)$ , the oracle  $\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, \cdot, b))$  outputs the encryption of the left-hand side of the input  $x||r$ , where  $r$  is a random string of length the same as the output length of the redundancy code. In that case, it is easy to see that the algorithms  $A_1$  and  $F$  simulate the responses to  $B$ 's queries to the encryption oracle in the same way (i.e.  $\mathcal{E}_{K_e}(x||r)$ ). The algorithm  $A_2$  effectively performs the verification of the forgery output of  $F$  when it checks the validity of the input  $\mathbf{p} = (M||\tau)$ . However, there is an additional condition that  $A_2$  checks on top of the validity check of  $\mathbf{p}$  —namely, the “newness” of the ciphertext  $C$ , which is missing in the experiment  $\mathbf{Exp}_{\mathcal{RC}, F}^{\text{uf-nma}}(k)$  for the adversary  $F$ . With only that difference,  $\mathbf{Exp}_{\mathcal{SE}, A}^{\text{nm-cpa-0}}(k)$  essentially performs the same thing as  $\mathbf{Exp}_{\mathcal{RC}, F}^{\text{uf-nma}}(k)$  when running the same adversary  $B$ . Hence,

$$\Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{nm-cpa-0}}(k) = 1 \right] \leq \mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(k).$$

Combining the results shown above, we have the following equations:

$$\begin{aligned} \mathbf{Adv}_{\mathcal{SE}, A}^{\text{nm-cpa}}(k) &= \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{nm-cpa-1}}(k) = 1 \right] - \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{nm-cpa-0}}(k) = 1 \right] \\ &\geq \mathbf{Adv}_{\mathcal{ESR}, B}^{\text{int-ctxt}}(k) - \mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(k) \end{aligned}$$

Equation (1) follows from this.  $\blacksquare$

To illustrate the tightness of our result, we show that if a given redundancy code is not UF-NMA secure, there exists a base encryption scheme such that for the given UF-NMA insecure redundancy code, the associated extended encryption scheme with secret redundancy fails to provide integrity. The following theorem states this result more formally.

**Theorem 5.4** Given a redundancy code  $\mathcal{RC}$  that is *not* UF-NMA secure and an encryption scheme  $\mathcal{SE}'$  that is IND-CCA (resp. NM-CPA) secure, we can construct an encryption scheme  $\mathcal{SE}$  that is also IND-CCA (resp. NM-CPA) secure, but the extended encryption scheme with secret redundancy  $\mathcal{ESR}$  associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  is not INT-CTXT secure. ■

We remark the proof here shows something stronger, namely that  $\mathcal{ESR}$  is not even INT-PTXT secure if the redundancy code is not UF-NMA secure.

**Proof of Theorem 5.4:** The proof is very similar to that of Theorem 4.2. In fact, we construct  $\mathcal{SE}$  from the given encryption scheme  $\mathcal{SE}'$  exactly in the same way as that in Theorem 4.2 and show that the associated extended extended encryption scheme with secret redundancy  $\mathcal{ESR}$  is not INT-PTXT secure assuming the redundancy code is not UF-NMA secure. Since the second part of the theorem (i.e.  $\mathcal{SE}$  inherits the privacy property of  $\mathcal{SE}'$ ) is already shown in the proof of Theorem 4.2, we omit the proof of the second part.

Let  $\mathcal{SE}' = (\mathcal{K}'_e, \mathcal{E}', \mathcal{D}')$  be the given encryption scheme. The encryption scheme  $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  is constructed based on  $\mathcal{SE}' = (\mathcal{K}'_e, \mathcal{E}', \mathcal{D}')$  as in the proof of Theorem 4.2. We recall their definitions here. The key generation algorithm  $\mathcal{K}_e$  is unchanged and the other two algorithms are shown below:

$$\begin{array}{l|l} \text{Algorithm } \mathcal{E}_{K_e}(M) & \text{Algorithm } \mathcal{D}_{K_e}(C) \\ C \leftarrow 0 \parallel \mathcal{E}'_{K'_e}(M) & \text{Parse } C \text{ as } b \parallel C' \text{ where } b \text{ is a bit} \\ \text{return } C & \text{If } b = 0 \text{ then return } \mathcal{D}'_{K'_e}(C') \\ & \text{else return } C' \end{array}$$

Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code and let  $\mathcal{ESR} = (\mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  be the extended encryption scheme with secret redundancy associated to  $\mathcal{SE}$  and  $\mathcal{RC}$  as per Construction 3.2. We now show that, assuming  $\mathcal{RC}$  is *not* UF-NMA secure, the encryption scheme with secret redundancy  $\mathcal{ESR}$  is not INT-PTXT secure. If  $\mathcal{RC}$  is not UF-NMA secure, there exists an adversary  $F$  whose advantage function  $\mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(\cdot)$  is not negligible. Using the adversary  $F$ , we can construct an adversary  $B$  attacking INT-CTXT of  $\mathcal{ESR}$  as follows. It gets an oracle for encryption  $\bar{\mathcal{E}}_{(K_e, K_r)}(\cdot)$  but does not need to invoke it.

$$\begin{array}{l} \text{Algorithm } B^{\bar{\mathcal{E}}_{(K_e, K_r)}(\cdot)}(k) \\ (M, \tau) \leftarrow F(k) \quad // \text{ Obtain the forgery output by } F \\ \text{return } 1 \parallel M \parallel \tau \quad // \text{ Return the forgery ciphertext formed based on the forgery output by } F \end{array}$$

It is easy to see that  $\mathbf{Adv}_{\mathcal{ESR}, B}^{\text{int-ptxt}}(k) \geq \mathbf{Adv}_{\mathcal{RC}, F}^{\text{uf-nma}}(k)$ . Thus  $\mathbf{Adv}_{\mathcal{ESR}, B}^{\text{int-ptxt}}(\cdot)$  is not negligible, meaning  $\mathcal{ESR}$  is not INT-PTXT secure. ■

## 6 Nested CBC (NCBC) with redundancy

Let  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  be a family of permutations (i.e. a block cipher). We let  $F_a(\cdot) = F(a, \cdot)$  and we let  $F_a^{-1}$  denote the inverse of  $F_a$ , for any key  $a \in \{0, 1\}^\kappa$ .

NESTED CBC. In this section, we will consider a variant of the CBC mode of operation that involves the use of two keys instead of just one. The additional key is used for the last iteration of the block cipher. We call this the *Nested CBC (NCBC)* mode, and denote by  $\text{NCBC}[F] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  the NCBC mode of operation with block-cipher  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ . The algorithms for  $\text{NCBC}[F]$  viewed as an encryption scheme are shown in Figure 3. We assume that the messages

Algorithm $\mathcal{K}_e(k)$ $a_1 \xleftarrow{R} \{0, 1\}^\kappa$ $a_2 \xleftarrow{R} \{0, 1\}^\kappa$ Return $(a_1 \  a_2)$	Algorithm $\mathcal{E}_{a_1 \  a_2}(X)$ Parse $X$ as $x_1 \cdots x_{n+1}$ $y_0 \xleftarrow{R} \{0, 1\}^l$ For $i = 1, \dots, n$ do $y_i \leftarrow F_{a_1}(y_{i-1} \oplus x_i)$ $y_{n+1} \leftarrow F_{a_2}(y_n \oplus x_{n+1})$ Return $y_0 y_1 \cdots y_{n+1}$	Algorithm $\mathcal{D}_{a_1 \  a_2}(Y)$ Parse $Y$ as $y_0 y_1 \cdots y_{n+1}$ For $i = 1, \dots, n$ do $x_i \leftarrow F_{a_1}^{-1}(y_i) \oplus y_{i-1}$ $x_{n+1} \leftarrow F_{a_2}^{-1}(y_{n+1}) \oplus y_n$ $X \leftarrow x_1 \cdots x_{n+1}$ Return $X$
---	--	---

Figure 3: Nested CBC encryption scheme  $NCBC[F] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ .

have length a multiple of the block length  $l$ . If this is not the case, the message is appropriately padded.

We examine what kinds of security properties for the redundancy code will provide integrity of ciphertexts for the encryption with redundancy scheme which uses  $NCBC[F]$  as the base encryption scheme. We examine this for both public redundancy and secret redundancy. In order to facilitate the practical security analyses, we will make concrete security assessments for the schemes examined in this section.

Since the security of the  $NCBC[F]$  scheme is based on the security of the underlying block-cipher  $F$ , we first define the security property of the underlying block-cipher on which our security analysis will be based.

Block-ciphers are usually modeled as “pseudorandom permutations” (sometimes even as “pseudorandom functions”) [4]. However, we use a stronger notion called *pseudorandom permutation under chosen-ciphertext attack (PRP-CCA)* [23], where the adversary gets access to both forward and inverse permutation oracles in the attack model. This was called “super pseudorandom permutation” in [23] and “strong pseudorandom permutation” in [25]. Intuitively, the stronger security is necessary because verification of a forgery requires decryption. (This point has arisen before in the design of cipher based authenticated encryption schemes [8], and the latter showed that the stronger requirement is necessary for integrity under the encode-then-encipher paradigm.)

**Definition 6.1 [PRP-CCA]** Let  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  be a block cipher with key-length  $\kappa$  and block-length  $l$ . Let  $P^l$  be the family of all permutations on  $l$ -bits. Let  $k \in \mathbb{N}$  and  $b \in \{0, 1\}$ . Let  $D$  be an adversary that has access to oracles  $g(\cdot)$  and  $g^{-1}(\cdot)$ . Consider the following experiment:

Experiment  $\mathbf{Exp}_{F,D}^{\text{prp-cca-b}}(k)$   
 If  $b = 0$  then  $g \xleftarrow{R} P^l$  else  $K \xleftarrow{R} \{0, 1\}^l$ ;  $g \leftarrow F_K$   
 $d \leftarrow D^{g(\cdot), g^{-1}(\cdot)}(k)$ ; Return  $d$

We define the *advantage* and the *advantage function* of the adversary as follows. For any integers  $t, q \geq 0$ ,

$$\mathbf{Adv}_{F,D}^{\text{prp-cca}}(k) = \Pr \left[ \mathbf{Exp}_{F,D}^{\text{prp-cca-1}}(k) = 1 \right] - \Pr \left[ \mathbf{Exp}_{F,D}^{\text{prp-cca-0}}(k) = 1 \right]$$

$$\mathbf{Adv}_F^{\text{prp-cca}}(k, t, q) = \max_D \left\{ \mathbf{Adv}_{F,D}^{\text{prp-cca}}(k) \right\}$$

where the maximum is over all  $D$  with time-complexity  $t$ , making at most  $q$  queries to the oracles  $g(\cdot)$  and  $g^{-1}(\cdot)$ . The block-cipher  $F$  is said to be *PRP-CCA secure* if the function  $\mathbf{Adv}_{F,D}^{\text{prp-cca}}(\cdot)$  is negligible for any adversary  $D$  whose time-complexity is polynomial in  $k$ . ■

The “time-complexity” refers to that of the entire experiment. Here, the choice of a random permutation  $g$  is not made all at once, but rather  $g$  is simulated in the natural way.

## 6.1 NCBC with secret redundancy

We denote by  $SNCBC[F, \mathcal{RC}] = (\mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$  the extended encryption scheme with secret redundancy associated to the encryption scheme  $NCBC[F] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  and a redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  as per Construction 3.2.

It turns out that  $SNCBC[F, \mathcal{RC}]$  is INT-CTXT secure if  $\mathcal{RC}$  satisfies the relatively weak security requirement of being “almost XOR universal” (AXU) introduced in [21].

**Definition 6.2 [Almost XOR Universal (AXU)]** Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code whose output length is  $\ell(\cdot)$ , and let  $k \in \mathbb{N}$ . We define the *axu-advantage function* of  $\mathcal{RC}$  as follows.

$$\begin{aligned} \text{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu) &= \max_{x, x' \in \{0,1\}^*, r \in \{0,1\}^{\ell(k)}} \left\{ \Pr \left[ \mathcal{H}_{K_r}(x) \oplus \mathcal{H}_{K_r}(x') = r : K_r \xleftarrow{R} \mathcal{K}_r(k) \right] \right\} \end{aligned}$$

where maximum is taken over all *distinct*  $x, x'$  of length at most  $\mu$  each, and all  $r \in \{0,1\}^{\ell(k)}$ .  $\blacksquare$

Note that we do not use  $\epsilon$  to indicate the probability bound, but instead, we use the advantage function notation for consistency and clarity of notations. Note also that the property AXU is not cryptographic, but, combinatoric.

We define another property that is useful in relating the combinatoric property of AXU for a redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  to a cryptographic attack model. The adversary has access to a stateful oracle  $\mathcal{O}$  that holds a key  $K_r$  for the redundancy function. The adversary makes queries of the form  $(X_i, y_i)$  for  $i = 1, \dots, q$ , and in response to the  $i$ -th query, the oracle returns 1 if there is an “axu-collision”, meaning there is some  $j < i$  such that  $\mathcal{H}_{K_r}(X_j) \oplus y_j = \mathcal{H}_{K_r}(X_i) \oplus y_i$  and  $X_j \neq X_i$ . If not, the oracle returns 0.

**Definition 6.3 [AXU-Collision [2]]** Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code with output length  $\ell(\cdot)$ , and let  $k \in \mathbb{N}$ . Let  $B$  be an adversary that has access to a stateful oracle  $\mathcal{O}$  that takes inputs of the form  $(X, y)$  and returns a bit. Consider the following experiment.

Experiment  $\mathbf{Exp}_{\mathcal{RC}, B}^{\text{axu-col}}(k)$

$K_r \xleftarrow{R} \mathcal{K}_r(k); i = 0$

Run  $B$  replying to its oracle queries as follows:

$B \Rightarrow (X, y)$

$i \leftarrow i + 1; X_i \leftarrow X; y_i \leftarrow y$

If  $\exists j \in \{1, \dots, i-1\}$  such that

–  $\mathcal{H}_{K_r}(X) \oplus y = \mathcal{H}_{K_r}(X_j) \oplus y_j$ , and

–  $X \neq X_j$

then  $a_i \leftarrow 1$  else  $a_i \leftarrow 0$

$B \Leftarrow a_i$

Until  $B$  halts

If  $\exists i$  such that  $a_i = 1$

then return 1 else return 0

Note that the code for the oracle  $\mathcal{O}$  is embedded in the above experiment.

We define the *advantage* and the *advantage function* of the adversary as follows. For any integers  $q, \mu \geq 0$ ,

$$\text{Adv}_{\mathcal{RC}, B}^{\text{axu-col}}(k) = \Pr \left[ \mathbf{Exp}_{\mathcal{RC}, B}^{\text{axu-col}}(k) = 1 \right] \quad \Bigg| \quad \text{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q, \mu) = \max_B \left\{ \text{Adv}_{\mathcal{RC}, B}^{\text{axu-col}}(k) \right\}$$

Algorithm $\mathcal{K}_e(k)$ $a_1 \xleftarrow{R} \{0, 1\}^\kappa$ $f \xleftarrow{R} P^l$ Return $(a_1 \  f)$	Algorithm $\mathcal{E}_{a_1}^f(X)$ Parse $X$ as $x_1 \cdots x_{n+1}$ $y_0 \xleftarrow{R} \{0, 1\}^l$ For $i = 1, \dots, n$ do $y_i \leftarrow F_{a_1}(y_{i-1} \oplus x_i)$ $y_{n+1} \leftarrow f(y_n \oplus x_{n+1})$ Return $y_0 y_1 \cdots y_{n+1}$	Algorithm $\mathcal{D}_{a_1}^{f^{-1}}(Y)$ Parse $Y$ as $y_0 y_1 \cdots y_{n+1}$ For $i = 1, \dots, n$ do $x_i \leftarrow F_{a_1}^{-1}(y_i) \oplus y_{i-1}$ $x_{n+1} \leftarrow f^{-1}(y_{n+1}) \oplus y_n$ $X \leftarrow x_1 \cdots x_{n+1}$ Return $X$
---	---	---

Figure 4: Nested CBC encryption scheme  $NCBC[F, P^l] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  constructed based on a block-cipher  $F$  and a random permutation  $P^l$ .

where the maximum is over all  $B$  making at most  $q$  oracle queries, the sum of whose length is at most  $\mu$ . We adopt the convention that the length of a query  $(X, y)$  is  $|X|$ .  $\blacksquare$

In the following claim, we relate the probability that there is an “axu-collision” (i.e.  $\mathbf{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q, \mu)$ ) to the axu-advantage function of  $\mathcal{RC}$  (i.e.  $\mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu)$ ).

**Claim 6.4** [2] Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code. Then,

$$\mathbf{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q, \mu) \leq \frac{q(q-1)}{2} \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu) \quad \blacksquare$$

The above claim will be used in the proof of the following theorem.

**Theorem 6.5 [Integrity of NCBC with secret redundancy]** Let  $\mathcal{RC}$  be a redundancy code whose output length is  $l$ -bits. Let  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  be a block-cipher, and let  $NCBC[F]$  be the NCBC encryption scheme based on  $F$ . Let  $SNCBC[F, \mathcal{RC}]$  be the extended encryption scheme with secret redundancy associated to  $NCBC[F]$  and  $\mathcal{RC}$ . Let  $k \in \mathbb{N}$ . Then

$$\begin{aligned} & \mathbf{Adv}_{SNCBC[F, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \\ & \leq \frac{q(q+1)}{2} \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu) + \frac{1}{2^l - q} + \mathbf{Adv}_F^{\text{prp-cca}}(k, t, q + \mu/l) \quad \blacksquare \end{aligned}$$

We take the standard approach taken from [6], namely considering the information theoretic case first and then computational case.

Let  $P^l$  denote the family of all permutations of  $l$ -bits. For the analysis in the information theoretic case, we model the second instance of the block-cipher in  $NCBC[F]$  as a permutation chosen at random from  $P^l$ . We denote by  $NCBC[F, P^l] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  the  $NCBC[F]$  encryption scheme where the second instance of the block-cipher is chosen from  $P^l$ . The algorithms for  $NCBC[F, P^l]$  are shown in Figure 4. The encryption algorithm  $\mathcal{E}_{a_1}^f(\cdot)$  is given an oracle for a permutation  $f$  from  $P^l$ , while the decryption algorithm  $\mathcal{D}_{a_1}^{f^{-1}}(\cdot)$  is given an oracle for  $f^{-1}$ , where  $f, a_1$  are chosen by the key generation algorithm  $\mathcal{K}_e$ .

**Lemma 6.6** [Information theoretic case] Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code whose output length is  $l$ -bits, and let  $NCBC[F, P^l] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  be the NCBC encryption scheme as defined in Figure 4. Let  $SNCBC[F, P^l, \mathcal{RC}] = (\mathcal{K}_s, \overline{\mathcal{E}}, \overline{\mathcal{D}})$  be the extended encryption scheme with secret redundancy associated to  $NCBC[F, P^l]$  and  $\mathcal{RC}$ . Then,

$$\mathbf{Adv}_{SNCBC[F, P^l, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \leq \frac{q(q+1)}{2} \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu) + \frac{1}{2^l - q} \quad \blacksquare \quad (2)$$

Algorithm  $B^{\mathcal{O}_{K_r}(\cdot, \cdot)}(k)$

```

 $a_1 \xleftarrow{R} \{0, 1\}^\kappa$  // Pick the key  $a_1$  for the first instance of the block-cipher
Initialize  $S \leftarrow \emptyset$  //  $S$  is a set containing the current range values
// for the simulated random permutation being used for the last block
For  $i = 1, \dots, q$  do
   $A \Rightarrow X_i$ ; //  $A$  makes a query  $X_i$  to the encryption oracle
  Parse  $X_i$  as  $x_i[1] \dots x_i[n_i]$ 
   $y_i[0] \dots y_i[n_i] \leftarrow CBC\text{-}\mathcal{E}_{a_1}(x_i[1] \dots x_i[n_i])$  // Encrypt the message blocks
   $b_i \leftarrow \mathcal{O}_{K_r}(X_i, y_i[n_i])$  // Make a query  $(X_i, y_i[n_i])$  to the oracle
  If  $b_i = 1$  then halt //  $B$  succeeds in achieving its goal.
   $R_i \xleftarrow{R} (\{0, 1\}^l - S)$ ;  $y_i[n_i + 1] \leftarrow R_i$ 
  // Pick an unused value  $R_i$  at random from the range of the simulated permutation
  // and use the value  $R_i$  as  $y_i[n_i + 1]$ 
   $S \leftarrow S \cup \{R_i\}$  // Update the set of the used values for the simulated permutation
   $Y_i \leftarrow y_i[0] \dots y_i[n_i]y_i[n_i + 1]$ 
   $A \Leftarrow Y_i$  // return  $Y_i$  to  $A$  as the query response
EndFor
 $A$  outputs  $C = c[0] \dots c[n]c[n + 1]$  as its forgery
 $X \leftarrow CBC\text{-}\mathcal{D}_{a_1}(c[0] \dots c[n])$  // Obtain the corresponding plaintext
 $b \leftarrow \mathcal{O}_{K_r}(X, c[n])$  // Make a query  $(X, c[n])$  to the oracle

```

Figure 5: The algorithm for the adversary attacking the AXU-Collision security of the redundancy code  $\mathcal{RC}$  using the adversary attacking the INT-CTXT of  $SNCBC[F, P^l, \mathcal{RC}]$ .  $CBC\text{-}\mathcal{E}_{a_1}(\cdot)$  denotes the CBC mode of encryption with a random IV, and  $CBC\text{-}\mathcal{D}_{a_1}(\cdot)$  denotes the corresponding decryption.

---

### Proof of Lemma 6.6:

We prove this by the standard reduction method. Let  $A$  be an adversary violating the INT-CTXT of the  $SNCBC[F, P^l, \mathcal{RC}]$  scheme. Using the adversary  $A$ , we will construct an adversary  $B$  violating the AXU-Collision security of  $\mathcal{RC}$ .

The goal of the adversary  $B$  is to find an axu-collision without knowing the key  $K_r$  for the redundancy function  $\mathcal{H}$  (i.e. make the given oracle  $\mathcal{O}_{K_r}(\cdot, \cdot)$  return 1). In order to achieve its goal,  $B$  runs the adversary  $A$  answering  $A$ 's queries to the encryption oracle  $\bar{\mathcal{E}}_{a_1}^f(\cdot)$  by picking a key  $a_1$  for the block-cipher and simulating the random permutation  $f$ . A difficulty in simulating  $f$  comes from the fact that  $B$  does not know the key  $K_r$  for  $\mathcal{H}$ , whose output values are needed to compute the inputs to  $f$ . However, this can be solved by observing that  $B$  does not need to know the input values to  $f$  in order to simulate the outputs of  $f$ . As long as there is no collision on the inputs to  $f$ ,  $B$  can simulate the outputs by just picking strings at random from the available range of the permutation and the distribution of the simulated output values will be correct (because the permutation is modeled as a random permutation). In order to find out whether there exists a collision on the inputs to  $f$ ,  $B$  enlists the help of the oracle  $\mathcal{O}_{K_r}(\cdot, \cdot)$  by querying it with appropriately formed string pairs such that if the oracle returns 0 it means there is no collision on inputs to  $f$  and if the oracle returns 1, then  $B$  wins the game.

The details of the algorithm for the adversary  $B$  is given in Figure 5.

From the algorithm  $B$ , note that as long as the adversary  $A$  succeeds (i.e.  $A$ 's output forgery ciphertext  $C = c[0] \cdots c[n]c[n+1]$  is “valid” and “new” as per Definition 2.2) and the last block  $c[n+1]$  matches one (say  $y_j[n_j+1]$ ) of the simulated last blocks in the response blocks, the adversary  $B$  will succeed in making its oracle output 1. Indeed, in this case, the equation  $\mathcal{H}_{K_r}(X) \oplus c[n] = \mathcal{H}_{K_r}(X_j) \oplus y_j[n_j]$  holds (note that a valid forgery implies that  $f^{-1}(c[n+1]) = \mathcal{H}_{K_r}(X) \oplus c[n]$ ). Also note that  $X \neq X_j$  in this case. (Suppose to the contrary that  $X = X_j$ . Then  $n = n_j$  and  $\mathcal{H}_{K_r}(X_j) = \mathcal{H}_{K_r}(X)$ . Since  $c[n+1] = y_j[n+1]$  we now get  $c[0] \cdots c[n+1] = y_j[0] \cdots y_j[n+1]$ , meaning that  $C$  was not “new.”) Hence,  $B$  is guaranteed to make its oracle output 1. However, if the adversary  $A$  succeeds in forgery but the last block  $c[n+1]$  of the forgery ciphertext does not match any of the simulated last blocks in the response blocks (i.e.  $c[n+1]$  was never returned to  $A$  by  $B$  as a last block of a query response), then  $B$  will not succeed in making its oracle output 1 even if  $A$  makes a successful forgery output. Let **Bad** denote this event. It is easy to see that the probability that the event **Bad** occurs is at most  $1/(2^l - q)$ . Note that as long as the event **Bad** does not occur,  $B$  will succeed if  $A$  succeeds. Moreover,  $B$  will be able to succeed even if  $A$  doesn't succeed if an axu-collision (i.e. a collision on the inputs to  $f$ ) occurs while  $B$  is simulating the responses to  $A$ 's queries. Hence, the probability that  $B$  succeeds is at least the probability that  $A$  succeeds under the absence of the event **Bad**.

Combining the results, we have

$$\begin{aligned} \Pr \left[ \mathbf{Exp}_{\mathcal{RC},B}^{\text{axu-col}}(k) = 1 \right] &\geq \Pr \left[ \mathbf{Exp}_{\text{SNCBC}[F,P^l,\mathcal{RC}],A}^{\text{int-ctxt}}(k) = 1 \wedge \overline{\mathbf{Bad}} \right] \\ &\geq \Pr \left[ \mathbf{Exp}_{\text{SNCBC}[F,P^l,\mathcal{RC}],A}^{\text{int-ctxt}}(k) = 1 \right] - \Pr [\mathbf{Bad}] \\ &\geq \Pr \left[ \mathbf{Exp}_{\text{SNCBC}[F,P^l,\mathcal{RC}],A}^{\text{int-ctxt}}(k) = 1 \right] - \frac{1}{2^l - q} \end{aligned}$$

Expressing the result in terms of the advantage functions of the adversaries, we get:

$$\mathbf{Adv}_{\text{SNCBC}[F,P^l,\mathcal{RC}],A}^{\text{int-ctxt}}(k) \leq \mathbf{Adv}_{\mathcal{RC},B}^{\text{axu-col}}(k) + \frac{1}{2^l - q}$$

The number of oracle queries made by the adversary  $B$  is at most  $q + 1$ , which is one more than the number of queries made by  $A$  (because  $B$  makes an additional query to the AXU oracle after  $A$  finishes making its encryption oracle queries and outputs its forgery). Taking into account the resources (which is at most in  $\text{poly}(k)$ ) used by the adversaries, we obtain the following equation:

$$\mathbf{Adv}_{\text{SNCBC}[F,P,\mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \leq \mathbf{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q + 1, \mu) + \frac{1}{2^l - q} \quad (3)$$

Above, the value  $\mu$  in  $\mathbf{Adv}_{\text{SNCBC}[F,P,\mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu)$  is defined as the maximum value of the sum of the lengths of the oracle queries and the decrypted plaintext corresponding to the forgery ciphertext.

We now upper-bound the advantage function  $\mathbf{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q + 1, \mu)$  in terms of the advantage function  $\mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu)$  using Claim 6.4 as follows.

$$\mathbf{Adv}_{\mathcal{RC}}^{\text{axu-col}}(k, q + 1, \mu) \leq \frac{q(q+1)}{2} \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{axu}}(k, \mu) \quad (4)$$

Combining Equation (3) and Equation (4), we obtain the conclusion of Lemma 6.6.  $\blacksquare$

We now consider the computational case, where the block-ciphers are modeled as strong pseudorandom permutations (PRP-CCA). The following lemma states this case. The proof, being standard, is omitted.

**Lemma 6.7** [Computational case] Let  $NCBC[F]$  be the NCBC encryption scheme based on a block-cipher  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  and let  $NCBC[F, P^l]$  be the NCBC encryption scheme as defined in Figure 4. Let  $\mathcal{RC}$  be a redundancy code. Let  $SNCBC[F, \mathcal{RC}]$  and  $NCBC[F, P^l, \mathcal{RC}]$  be the extended encryption schemes with secret redundancy ( $\mathcal{RC}$ ) associated to  $NCBC[F]$  and  $NCBC[F, P^l]$ , respectively. Then,

$$\mathbf{Adv}_{SNCBC[F, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \leq \mathbf{Adv}_{SNCBC[F, P^l, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) + \mathbf{Adv}_F^{\text{prp-cca}}(k, t, q + \mu/l) \quad \blacksquare$$

Given the above two lemmas (i.e. Lemma 6.6 and Lemma 6.7), Theorem 6.5 follows.

## 6.2 NCBC with public redundancy

We denote by  $PNCBC[F, \mathcal{RC}] = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  the extended encryption scheme with public redundancy associated to the NCBC encryption scheme  $NCBC[F] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  and a redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  as per Construction 3.1.

We want to examine what kind of security property for the underlying redundancy code  $\mathcal{RC}$  suffices to make the NCBC with public redundancy scheme provide integrity. It turns out that a cryptographic property called “XOR-collision-resistance” suffices. XOR-collision-resistance is slightly stronger than “collision-resistance”. Roughly, a redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  is said to be *XOR-collision-resistant (XCR)* if it is “hard” to find strings  $x, x'$  where  $x \neq x'$  such that  $\mathcal{H}_{K_r}(x) \oplus \mathcal{H}_{K_r}(x') = r$  for any committed value  $r$  and any given key  $K_r$ . We define XOR-collision-resistance (XCR) more formally as follows.

**Definition 6.8 [XOR-Collision-Resistance (XCR)]** Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be a redundancy code whose output length is  $\ell(\cdot)$ . Let  $k \in \mathbb{N}$ . Let  $B = (B_1, B_2)$  be an adversary. Consider the following experiment:

Experiment  $\mathbf{Exp}_{\mathcal{RC}, B}^{\text{XCR}}(k)$

$(r, s) \leftarrow B_1(k); K \xleftarrow{R} \mathcal{K}_r(k); (x, x') \leftarrow B_2(K, r, s)$   
 if  $\mathcal{H}_K(x) \oplus \mathcal{H}_K(x') = r$  and  $x \neq x'$  then return 1 else return 0

Above, the variable  $s$  denotes state information. We define the *XCR-advantage* of the adversary and the *XCR-advantage function* of  $\mathcal{RC}$  via,

$$\mathbf{Adv}_{\mathcal{RC}, B}^{\text{XCR}}(k) = \Pr [\mathbf{Exp}_{\mathcal{RC}, B}^{\text{XCR}}(k) = 1] \quad \Bigg| \quad \mathbf{Adv}_{\mathcal{RC}}^{\text{XCR}}(k, t) = \max_B \{ \mathbf{Adv}_{\mathcal{RC}, B}^{\text{XCR}}(k) \}$$

where the maximum is over all  $B$  with time-complexity  $t$ . The scheme  $\mathcal{RC}$  is said to be *XCR secure* if the function  $\mathbf{Adv}_{\mathcal{RC}, A}^{\text{XCR}}(k)$  is negligible for any adversary  $A$  whose time-complexity is polynomial in  $k$ .  $\blacksquare$

XOR-collision-resistance (XCR) as defined above is a new notion that has not been explicitly studied in the literature. In XCR, the adversary first outputs a string  $r$  and then obtains the key to the function. The adversary’s goal is to find a pair of strings  $x, x'$  (called an “XOR-collision” pair) such that the XOR of their images equals  $r$ .

Given the definitions for the security properties of the underlying primitives, we now state the theorem regarding the security of the *PNCBC* scheme. Following that we will further discuss XCR redundancy codes.

**Theorem 6.9 [Integrity of NCBC with public redundancy]** Let  $\mathcal{RC}$  be a redundancy code whose output length is  $l$ -bits. Let  $F: \{0, 1\}^\kappa \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  be a block-cipher, and let  $NCBC[F]$

be the NCBC encryption scheme based on  $F$ . Let  $PNCBC[F, \mathcal{RC}]$  be the extended encryption scheme with public redundancy associated to  $NCBC[F]$  and  $\mathcal{RC}$ . Let  $k \in \mathbb{N}$ . Then

$$\begin{aligned} & \mathbf{Adv}_{PNCBC[F, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \\ & \leq mq \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{xcr}}(k, t') + \frac{2}{2^l - m} + \frac{m^2}{2(2^l - m)} + 2 \cdot \mathbf{Adv}_F^{\text{prp-cca}}(k, t, q + m) \end{aligned}$$

where  $m = \mu/l$  and  $t' = t + O(\mu + ql)$ .  $\blacksquare$

As is done in the proof for the  $SNCBC$  scheme, we consider the information theoretic case first and then add on the computational case.

The following lemma states the information theoretic case where the block-ciphers underlying the scheme  $PNCBC$  are modeled as random permutations. Unlike the NCBC with secret redundancy case,  $NCBC[P^l]$  here denotes the NCBC encryption scheme where *both* underlying block-ciphers are modeled as random permutations chosen from the family of all permutations  $P^l$ . The proof of the following lemma is given in Appendix B.2.

**Lemma 6.10** [Information theoretic case] Let  $PNCBC[P^l, \mathcal{RC}] = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  be the extended encryption scheme with public redundancy associated to  $NCBC[P^l] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  and  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$ . Then,

$$\mathbf{Adv}_{PNCBC[P^l, \mathcal{RC}]}^{\text{int-ctxt}}(k, t, q, \mu) \leq mq \cdot \mathbf{Adv}_{\mathcal{RC}}^{\text{xcr}}(k, t') + \frac{1}{2^l - q} + \frac{1}{2^l - m} + \frac{m^2}{2(2^l - m)} \quad (5)$$

where  $m = \mu/l$  and  $t' = t + O(\mu + ql)$ .  $\blacksquare$

The analysis on the computational case gives rise to a similar result as in Lemma 6.7, except that the factor of one on the advantage function  $\mathbf{Adv}_F^{\text{prp-cca}}(\cdot, \cdot, \cdot)$  for PRP-CCA is increased to two. This is because both of the two block-ciphers are modeled as random permutations in the  $PNCBC$  case, whereas, in the  $SNCBC$  case, only the second block-cipher is modeled as a random permutation. In general, the factor multiplied to the term  $\mathbf{Adv}_F^{\text{prp-cca}}(\cdot, \cdot, \cdot)$  indicates the number of the underlying permutations that are modeled as random permutations in the information theoretic case. Incorporating the computational case with the information theoretic case, we obtain the statement of Theorem 6.9.

**XCR REDUNDANCY CODES.** We now further discuss XCR redundancy codes. Note that the XCR property can be thought of as a cryptographic counterpart of the AXU property described in the previous section. The combinatorial property of AXU (for secret redundancy) is weaker, and therefore, easier to implement than the cryptographic property of XCR (for public redundancy).

What are candidates for XCR redundancy codes? Note that an unkeyed hash function like SHA-1 does not yield an XCR redundancy code. Indeed, an adversary can choose any distinct  $x, x'$ , and let  $r = \text{SHA-1}(x) \oplus \text{SHA-1}(x')$ . It can output  $r$  in its first stage, and  $x, x'$  in its second, and win the game. An XCR redundancy code must be keyed. A keyed hash function is a good candidate. Specifically, we suggest that HMAC [3] is a candidate for a XCR redundancy code. Furthermore, we show a general way to transform any collision-resistant function into an XCR redundancy code.

For notational convenience in the remaining section, we treat “a function family” of the form  $\mathcal{H}: \{0, 1\}^\kappa \times \{0, 1\}^b \rightarrow \{0, 1\}^\ell$  (where  $\kappa(\cdot)$  is the key-length function) like a redundancy code  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$ . Note that they are syntactically identical in that we can represent the function family as  $\mathcal{RC}$  that consists of the following two algorithms: the key generation algorithm  $\mathcal{K}_r(\cdot)$  that

returns a key chosen at random from  $\{0, 1\}^{\kappa(\cdot)}$  and the redundancy function  $\mathcal{H}$  that returns a string of length  $\ell(\cdot)$  given as input a string of length  $b(\cdot)$ .

Before we show the transformation method, we first recall the collision-resistance property of a function family below:

**Definition 6.11 [Collision-resistance]** Let  $H: \{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^l$  be a function family. Let  $k \in \mathbb{N}$ . Let  $A$  be an adversary. Consider the following experiment:

Experiment  $\mathbf{Exp}_{H,A}^{\text{cr}}(k)$

$K \xleftarrow{R} \{0, 1\}^{\kappa(k)}$

$(x, x') \leftarrow A(k, K)$

If  $x \neq x'$  and  $H_K(x) = H_K(x')$  then return 1 else return 0

We define the advantage and the advantage function of the adversary via,

$$\mathbf{Adv}_{H,A}^{\text{cr}}(k) = \Pr [\mathbf{Exp}_{H,A}^{\text{cr}}(k) = 1] \quad \Bigg| \quad \mathbf{Adv}_H^{\text{cr}}(k, t) = \max_A \{ \mathbf{Adv}_{H,A}^{\text{cr}}(k) \}$$

where the maximum is over all  $A$  with time-complexity  $t$ .  $\blacksquare$

We now describe how to transform a collision-resistant function family into an XCR function family. The transformation involves an application of an AXU function family on the collision-resistant function family.

**Construction 6.12** Given a collision-resistant function family  $A: \{0, 1\}^{\kappa_A} \times \{0, 1\}^* \rightarrow \{0, 1\}^m$  and an AXU function family  $B: \{0, 1\}^{\kappa_B} \times \{0, 1\}^m \rightarrow \{0, 1\}^{3m}$ , we can construct an XCR function family  $H: \{0, 1\}^{\kappa_A + \kappa_B} \times \{0, 1\}^* \rightarrow \{0, 1\}^{3m}$  as follows:

$$H_{k_1 \| k_2}(X) = B_{k_2}(A_{k_1}(X)),$$

where  $X \in \{0, 1\}^*$  is the input string, and  $k_1 \| k_2 \xleftarrow{R} \{0, 1\}^{\kappa_A + \kappa_B}$  is the key string.  $\blacksquare$

We claim that the function family  $H$  constructed above is indeed XCR if the underlying function families  $A$  and  $B$  are collision-resistant and AXU, respectively, with  $\mathbf{Adv}_B^{\text{axu}}(k, m) \leq 2^{-3m}$ .

**Claim 6.13** Let  $A: \{0, 1\}^{\kappa_A} \times \{0, 1\}^* \rightarrow \{0, 1\}^m$  be a collision-resistant function family and let  $B: \{0, 1\}^{\kappa_B} \times \{0, 1\}^m \rightarrow \{0, 1\}^{3m}$  be an AXU function family. Let  $H: \{0, 1\}^{\kappa_A + \kappa_B} \times \{0, 1\}^* \rightarrow \{0, 1\}^{3m}$  be a function family as per Construction 6.12. Then,

$$\mathbf{Adv}_H^{\text{xcr}}(k, t) \leq \mathbf{Adv}_A^{\text{cr}}(k, t) + 2^{2m} \cdot \mathbf{Adv}_B^{\text{axu}}(k, m) \quad \blacksquare$$

**Corollary 6.14** Let  $A: \{0, 1\}^{\kappa_A} \times \{0, 1\}^* \rightarrow \{0, 1\}^m$  be a collision-resistant function family and let  $B: \{0, 1\}^{\kappa_B} \times \{0, 1\}^m \rightarrow \{0, 1\}^{3m}$  be an AXU function family with  $\mathbf{Adv}_B^{\text{axu}}(k, m) \leq 2^{-3m}$ . Let  $H: \{0, 1\}^{\kappa_A + \kappa_B} \times \{0, 1\}^* \rightarrow \{0, 1\}^{3m}$  be a function family as per Construction 6.12. Then,

$$\mathbf{Adv}_H^{\text{xcr}}(k, t) \leq \mathbf{Adv}_A^{\text{cr}}(k, t) + 2^{-m} \quad \blacksquare$$

**Proof of Claim 6.13:** Let  $F = (F_1, F_2)$  be an adversary against the XCR property of  $H$ . Then, we can construct  $F'$  that attacks the collision-resistance of  $A$  running the adversary  $F$  as follows.

Algorithm  $F'(k, K_a)$

$(r, s) \leftarrow F_1(k)$

$K_b \xleftarrow{R} \{0, 1\}^{\kappa_B}$

$(x, x') \leftarrow F_2(K_a \| K_b, r, s)$

return  $(x, x')$

We want to bound the advantage of  $F$  attacking the XCR property of  $H$  in terms of the advantage of  $F'$  attacking the CR property of  $A$  and the advantage function of  $B$  as an AXU function.

Recall that  $F$  succeeds (i.e.  $\mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1$ ) in violating the XCR property of  $H$  if its output pair  $(x, x')$  is a “correct” XOR-collision pair —meaning,  $x \neq x'$  and  $H_{K_a \| K_b}(x) \oplus H_{K_a \| K_b}(x') = r$ , which we can rewrite as  $B_{K_b}(A_{K_a}(x)) \oplus B_{K_b}(A_{K_a}(x'))$ .

Depending on the distinctness of the inputs to the function  $B_{K_b}(\cdot)$  which is also the outputs of the function  $A_{K_a}(\cdot)$ , the event  $\mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1$  can be divided into the following two cases:  $A_{K_a}(x) = A_{K_a}(x')$  or  $A_{K_a}(x) \neq A_{K_a}(x')$ . Note that if  $A_{K_a}(x) = A_{K_a}(x')$  with  $x \neq x'$ , it implies that  $F$  implicitly found a collision pair, and therefore, by outputting the pair  $(x, x')$ ,  $F'$  will succeed in attacking against the collision-resistance of  $B$ . On the other hand, if  $A_{K_a}(x) \neq A_{K_a}(x')$ , then we can bound the probability that  $B_{K_b}(y) \oplus B_{K_b}(y') = r$  (where,  $y = A_{K_a}(x)$  and  $y' = A_{K_a}(x')$ ) as the probability that the key  $K_b$  is “bad” in that there exist distinct strings  $y, y' \in \{0, 1\}^m$  such that  $B_{K_b}(y) \oplus B_{K_b}(y') = r$  for a given  $r \in \{0, 1\}^{3m}$ . We define the event where the key is “bad” more formally as follows: given  $r \in \{0, 1\}^{3m}$ , let  $K_b \in \{0, 1\}^{\kappa_B}$  be  $r$ -Bad if there exist  $y, y' \in \{0, 1\}^m$  such that  $y \neq y'$  and  $B_{K_b}(y) \oplus B_{K_b}(y') = r$ . Then,

$$\begin{aligned}
& \Pr \left[ K_b \text{ is } r\text{-Bad} : K_b \xleftarrow{R} \{0, 1\}^{\kappa_B} \right] \\
&= \Pr \left[ \exists y, y' \text{ such that } y \neq y' \wedge B_{K_b}(y) \oplus B_{K_b}(y') = r : K_b \xleftarrow{R} \{0, 1\}^{\kappa_B} \right] \\
&\leq \sum_{y, y': y \neq y'} \Pr \left[ B_{K_b}(y) \oplus B_{K_b}(y') = r : K_b \xleftarrow{R} \{0, 1\}^{\kappa_B} \right] \\
&\leq 2^{2m} \cdot \mathbf{Adv}_B^{\text{axu}}(k, m)
\end{aligned}$$

Note the the event “ $K_b$  is  $r$ -Bad” upper-bounds the event  $\mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1$  and  $A_{K_a}(x) \neq A_{K_a}(x')$ . This is because in the attack model, the adversary  $F$  against  $H$  is required to *search for* distinct strings  $x, x'$  such that  $A_{K_a}(x) \neq A_{K_a}(x')$  and  $B_{K_b}(A_{K_a}(x)) \oplus B_{K_b}(A_{K_a}(x')) = r$  after it picks  $r$  first in the first stage, and given the key  $K_a \| K_b$  for  $H$ , whereas the event “ $K_b$  is  $r$ -Bad” is about the *existence* of distinct strings  $y, y'$  such that  $B_{K_b}(y) \oplus B_{K_b}(y') = r$  for the given  $r$  and over the random choice of  $K_b$ . If  $F$  finds such a pair  $x, x'$  for a random choice of a key  $K_b$ , then the event “ $K_b$  is  $r$ -Bad” is by definition true, while the opposite is not necessarily true — even if there exists a pair  $y, y'$  that satisfies the mentioned property for a randomly chosen key  $K_b$ ,  $F$  might not be able to “find” a pair  $x, x'$  such that the above-mentioned property holds. Hence,  $\Pr \left[ \mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1 \wedge A_{K_a}(x) \neq A_{K_a}(x') \right] \leq \Pr \left[ K_b \text{ is } r\text{-Bad} : K_b \xleftarrow{R} \{0, 1\}^{\kappa_B} \right] \leq 2^{2m} \cdot \mathbf{Adv}_B^{\text{axu}}(k, m)$ .

Combining the results obtained above, we now bound the probability that the advantage of  $F$  in terms of the advantage of  $F'$  and the advantage function of  $B$  as an AXU function as follows:

$$\begin{aligned}
\mathbf{Adv}_{H,F}^{\text{xcr}}(k) &= \Pr \left[ \mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1 \right] \\
&= \Pr \left[ \mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1 \wedge A_{K_a}(x) = A_{K_a}(x') \right] + \\
&\quad \Pr \left[ \mathbf{Exp}_{H,F}^{\text{xcr}}(k) = 1 \wedge A_{K_a}(x) \neq A_{K_a}(x') \right] \\
&\leq \mathbf{Adv}_{A,F'}^{\text{cr}}(k) + 2^{2m} \cdot \mathbf{Adv}_B^{\text{axu}}(k, m)
\end{aligned}$$

Taking into account the resources used by the adversaries, we obtain the conclusion of Claim 6.13.

Note that Corollary 6.14 can be obtained by taking into account the bound on the advantage function  $\mathbf{Adv}_B^{\text{axu}}(k, m) \leq 2^{-3m}$ . ■

## Acknowledgments

We thank Hugo Krawczyk for helpful comments on a previous version of this paper. We thank Daniele Micciancio for helpful discussions.

## References

- [1] M. ATICI AND D. STINSON, “Universal Hashing and Multiple Authentication,” *Advances in Cryptology – CRYPTO ’96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
- [2] M. BELLARE, A. BOLDYREVA, L. KNUDSEN, AND C. NAMPREMPRE, “On-line ciphers and the Hash-CBC constructions,” *Advances in Cryptology – CRYPTO ’01*, Lecture Notes in Computer Science Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.
- [3] M. BELLARE, R. CANETTI AND H. KRAWCZYK, “Keying hash functions for message authentication,” *Advances in Cryptology – CRYPTO ’96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
- [4] M. BELLARE, A. DESAI, E. JOKIPII AND P. ROGAWAY, “A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation,” *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.
- [5] M. BELLARE, A. DESAI, D. POINTCHEVAL AND P. ROGAWAY, “Relations among notions of security for public-key encryption schemes,” *Advances in Cryptology – CRYPTO ’98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
- [6] M. BELLARE, J. KILIAN AND P. ROGAWAY, “The Security of the Cipher Block Chaining Message Authentication Code,” *Journal of Computer and System Sciences*, Vol. 61, No. 3, December 2000, pp. 362–399.
- [7] M. BELLARE AND C. NAMPREMPRE, “Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm,” *Advances in Cryptology – ASIACRYPT ’00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.
- [8] M. BELLARE AND P. ROGAWAY, “Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography,” *Advances in Cryptology – ASIACRYPT ’00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.
- [9] M. BELLARE AND A. SAHAI, “Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization,” *Advances in Cryptology – CRYPTO ’99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
- [10] J. BLACK, S. HALEVI, H. KRAWCZYK, T. KROVETZ AND P. ROGAWAY, “UMAC: Fast and secure message authentication,” *Advances in Cryptology – CRYPTO ’99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
- [11] L. CARTER AND M. WEGMAN, “Universal Classes of Hash Functions,” *Journal of Computer and System Sciences*, Vol. 18, 1979, pp. 143–154.
- [12] D. DOLEV, C. DWORK AND M. NAOR, “Non-malleable cryptography,” *SIAM Journal on Computing*, Vol. 30, No. 2, April 2000, pp. 391–437.
- [13] V. GLIGOR AND P. DONESCU, “Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes,” *Fast Software Encryption ’01*, Lecture Notes in Computer Science Vol. ??, ?? ed., Springer-Verlag, 2001.
- [14] S. GOLDWASSER AND S. MICALI, “Probabilistic encryption,” *Journal of Computer and System Sciences*, Vol. 28, 1984, pp. 270–299.

- [15] S. HALEVI AND H. KRAWCZYK, “MMH: Software Message Authentication in the Gbit/Second Rates,” *Fast Software Encryption — 4th International Workshop, FSE’97 Proceedings*, Lecture Notes in Computer Science, vol. 1267, E. Biham ed., Springer, 1997.
- [16] R. JUENEMAN, “A high speed manipulation detection code,” *Advances in Cryptology – CRYPTO ’86*, Lecture Notes in Computer Science Vol. 263, A. Odlyzko ed., Springer-Verlag, 1986.
- [17] R. JUENEMAN, C. MEYER AND S. MATYAS, “Message Authentication with Manipulation Detection Codes,” in *Proceedings of the 1983 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1984, pp.33-54.
- [18] C. JUTLA, “Encryption modes with almost free message integrity,” *Advances in Cryptology – EUROCRYPT ’01*, Lecture Notes in Computer Science Vol. 2045, B. Pfitzmann ed., Springer-Verlag, 2001.
- [19] J. KATZ AND M. YUNG, “Complete characterization of security notions for probabilistic private-key encryption,” *Proceedings of the 32nd Annual Symposium on the Theory of Computing*, ACM, 2000.
- [20] J. KATZ AND M. YUNG, “Unforgeable Encryption and Adaptively Secure Modes of Operation,” *Fast Software Encryption ’00*, Lecture Notes in Computer Science, B. Schneier ed., Springer-Verlag, 2000.
- [21] H. KRAWCZYK, “LFSR-based Hashing and Authentication,” *Advances in Cryptology – CRYPTO ’94*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
- [22] H. KRAWCZYK, “The order of encryption and authentication for protecting communications (Or: how secure is SSL?),” *Advances in Cryptology – CRYPTO ’01*, Lecture Notes in Computer Science Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.
- [23] M. LUBY AND C. RACKOFF, “How to Construct Pseudorandom Permutations from Pseudorandom Functions,” *SIAM Journal of Computing*, Vol. 17, No. 2, pp. 373–386, April 1988.
- [24] A. MENEZES, P. VAN OORSHOT AND S. VANSTONE, “Handbook of applied cryptography,” CRC Press LLC, 1997.
- [25] M. NAOR AND O. REINGOLD, “On the construction of pseudorandom permutations: Luby-Rackoff revisited,” *J. of Cryptology*, 12(1), 1999.
- [26] B. PRENEEL, “Cryptographic Primitives for Information Authentication — State of the Art,” *State of the Art in Applied Cryptography*, COSIC’97, LNCS 1528, B. Preneel and V. Rijmen eds., Springer-Verlag, pp. 49-104, 1998.
- [27] P. ROGAWAY, “Bucket Hashing and its Application to Fast Message Authentication,” *Advances in Cryptology – CRYPTO ’95*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
- [28] P. ROGAWAY, M. BELLARE, J. BLACK AND T. KROVETZ, “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption,” *Proceedings of the 8th Annual Conference on Computer and Communications Security*, ACM, 2001.
- [29] C. RACKOFF AND D. SIMON, “Non-Interactive zero-knowledge proof of knowledge and chosen ciphertext attack,” *Advances in Cryptology – CRYPTO ’91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.
- [30] M. WEGMAN AND L. CARTER, “New hash functions and their use in authentication and set equality,” *Journal of Computer and System Sciences*, Vol. 22, 1981, pp. 265–279.

## A Attack on CBC with public redundancy

Let  $F: \{0, 1\}^k \times \{0, 1\}^l \rightarrow \{0, 1\}^l$  be a family of permutations (i.e. a block-cipher). Note that by permutations we mean that they are invertible —for each permutation  $F_a$  specified by a key  $a$ , the inverse is denoted as  $F_a^{-1}$ . The scheme  $CBC[F] = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$  has the key generation algorithm  $\mathcal{K}_e$ , where the key for encryption is the key  $a$  specifying  $f = F_a$ . Given the message  $X$  to be encrypted, which is a sequence of  $l$ -bit blocks,  $X = x_1 \cdots x_n$ , the algorithms for  $\mathcal{E}_a(x)$  and  $\mathcal{D}_a(x)$  for the CBC encryption scheme are defined as follows:

Algorithm $\mathcal{E}_a(x)$ Parse $X$ as $x_1 \cdots x_n$ ; $y_0 \xleftarrow{R} \{0, 1\}^l$ For $i = 1, \dots, n$ do $y_i \leftarrow F_a(y_{i-1} \oplus x_i)$ Return $y_0 \  y_1 y_2 \cdots y_n$	Algorithm $\mathcal{D}_a(Y)$ Parse $Y$ as $y_0 \  y_1 \cdots y_n$ For $i = 1, \dots, n$ do $x_i \leftarrow F_a^{-1}(y_i) \oplus y_{i-1}$ Return $x_1 \cdots x_n$
--	---

Let  $\mathcal{RC} = (\mathcal{K}_r, \mathcal{H})$  be any redundancy code whose output length is  $l$ -bits. Let  $PCBC[F, \mathcal{RC}] = (\mathcal{K}_c, \mathcal{K}_s, \bar{\mathcal{E}}, \bar{\mathcal{D}})$  be the extended encryption with public redundancy associated to  $CBC[F]$  and  $\mathcal{RC}$ . The attack against  $PCBC[F, \mathcal{RC}]$  is based on the length variability. The attacker chooses as its query a string followed by the redundancy of the string (such that the redundancy is placed in the last block), and obtains the corresponding ciphertext. It then outputs all but the last block of the ciphertext as its forged ciphertext. The forgery is successful because the ciphertext is new with respect to the query response, and the recovered plaintext is the original query, which is composed of a string and its redundancy.

For completeness, we present the attack in more detail below. The adversary has access to the encryption oracle  $\bar{\mathcal{E}}_{\langle a, K_r \rangle}(\cdot)$  and is given as input the key  $K_r$  for the redundancy code  $\mathcal{RC}$ .

Adversary  $A^{\bar{\mathcal{E}}_{\langle a, K_r \rangle}(\cdot)}(k, K_r)$   
 $x_1 \leftarrow 0$ ;  $x_2 \leftarrow \mathcal{H}_{K_r}(x_1)$ ;  $X \leftarrow x_1 \| x_2$   
 $Y \leftarrow \bar{\mathcal{E}}_{\langle a, K_r \rangle}(X)$ ; Parse  $Y$  as  $y_0 y_1 y_2 y_3$   
 $Y' \leftarrow y_0 y_1 y_2$ ; Return  $Y'$

It is easy to see that the ciphertext  $Y'$  output by the adversary  $A$  is “new”, meaning the ciphertext was never output by the encryption oracle and “valid”, meaning the decryption algorithm  $\bar{\mathcal{D}}_{\langle a, K_r \rangle}(\cdot)$  will return  $x_1 \neq \perp$  on input  $y_0 y_1 y_2$  because the corresponding plaintext  $x_1 \| x_2$  returned by the base decryption algorithm  $\mathcal{D}_a(\cdot)$  will satisfy the relationship  $\mathcal{H}_{K_r}(x_1) = x_2$ .

## B Proofs

### B.1 Proof of Theorem 3.3

Here we show it for the IND-CPA case only, but in a similar manner, it can be shown for the other cases (i.e. IND-CCA and NM-CPA) as well. We use a standard reduction argument. Let  $B$  be an IND-CPA-adversary for  $\mathcal{ER}$ . We associate to  $B$  an IND-CPA-adversary  $A$  for  $\mathcal{SE}$ . The algorithm for  $A$  is shown below:

Algorithm  $A^{\mathcal{E}_{K_e}(\mathcal{LR}(\cdot, b))}(k)$   
 $K_r \xleftarrow{R} \mathcal{K}_r(k)$   
 If  $\mathcal{ER}$  is a public redundancy scheme then  $K_c \leftarrow K_r$  else  $K_c \leftarrow \varepsilon$   
 Run  $B$  on input  $(k, K_c)$   
 When  $B$  queries  $(x_0, x_1)$  to the LR-encryption oracle do  
 $x'_0 \leftarrow x_0 \| \mathcal{H}_{K_r}(x_0)$ ;  $x'_1 \leftarrow x_1 \| \mathcal{H}_{K_r}(x_1)$   
 $y \leftarrow \mathcal{E}_{K_e}(\mathcal{LR}(x'_0, x'_1, b))$ ; return  $y$  to  $B$   
 Until  $B$  outputs a bit  $d$   
 Return  $d$

By definition, we have

$$\mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cpa}}(k) = \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cpa-1}}(k) = 1 \right] - \Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cpa-0}}(k) = 1 \right].$$

It is easy to see from the above algorithm that  $A$  is effectively simulating the encryption oracle  $\bar{\mathcal{E}}_{\langle K_e, K_r \rangle}(\mathcal{LR}(\cdot, \cdot, b))$  in its response to  $B$ 's queries. Since  $A$  outputs the same output as  $B$  does, the following equation holds for both  $b = 0$  and  $b = 1$ :

$$\Pr \left[ \mathbf{Exp}_{\mathcal{SE}, A}^{\text{ind-cpa-}b}(k) = 1 \right] = \Pr \left[ \mathbf{Exp}_{\mathcal{ER}, B}^{\text{ind-cpa-}b}(k) = 1 \right]$$

Hence,  $\mathbf{Adv}_{\mathcal{SE}, A}^{\text{ind-cpa}}(k) = \mathbf{Adv}_{\mathcal{ER}, B}^{\text{ind-cpa}}(k)$ , which means that as long as the advantage function of  $A$  is negligible, so is that of  $B$ . Hence, the conclusion of the theorem follows.  $\blacksquare$

## B.2 Proof of Lemma 6.10

Given any adversary  $B$  attacking the  $PNCBC[P^l, \mathcal{RC}]$  scheme using resources at most  $t, q, \mu$ , we will construct an adversary (XOR-collision finder)  $A$  with time-complexity  $t' = t + O(\mu + ql)$  and show that

$$\mathbf{Adv}_{\mathcal{RC}, A}^{\text{XCR}}(k) \geq \frac{1}{mq} \cdot \left[ \mathbf{Adv}_{PNCBC[P^l, \mathcal{RC}], B}^{\text{int-ctxt}}(k) - \frac{1}{2^l - q} - \frac{1}{2^l - m} - \frac{m^2}{2(2^l - m)} \right] \quad (6)$$

where  $m = \mu/l$ . Equation (5) follows by transposing terms and taking the maximum, over all adversaries limited to the given resources, of the advantage of the adversary.

**PROOF IDEA.** We will first give a high level intuition of the proof. Recall from the definition of XOR-collision-resistance that the adversary  $A$  is divided into two stages ( $A_1, A_2$ ). In the first stage,  $A_1$  will output the value  $r$  for which it will try to find XOR collision. In the second stage, given the key  $K$  for  $\mathcal{RC}$  (i.e.  $\mathcal{H}$ ),  $A_2$  will output the purported XOR collision pair. As usual, the algorithm  $A$  runs the adversary  $B$  as a subroutine providing answers to the encryption oracle queries, and somehow uses  $B$ 's output forgery ciphertext to find the collision pair. In answering  $B$ 's queries,  $A$  will simulate the two random permutations denoted by  $\pi_1$  and  $\pi_2$ .

Consider a special case first. Suppose  $B$  asks just one query of  $k$  blocks  $Z = z[1] \cdots z[k]$  to the encryption oracle, and gets back the answer  $Y = y[0] \cdots y[k+1]$ .  $B$  then outputs the forgery ciphertext  $C = c[0] \cdots c[n]c[n+1]$  that is “new” and “valid”, meaning  $C \neq Y$  and its decrypted text  $X = x[1] \cdots x[n+1]$  contains the valid redundancy—that is,  $\mathcal{H}_K(X) = x[n+1]$ . Furthermore, suppose that the last two blocks  $c[n]$  and  $c[n+1]$  of  $B$ 's forgery output were the same as the query response blocks  $y[i]$  and  $y[k+1]$ , respectively, for some  $i \in \{1, \dots, k\}$ —that is,  $c[n] = y[i]$  and  $c[n+1] = y[k+1]$ . Then, the equation  $y[k] \oplus \mathcal{H}_K(Z) = y[i] \oplus \mathcal{H}_K(X)$  holds, and if  $A_1$  had somehow output the value  $y[k] \oplus y[i]$  as  $r$ ,  $A_2$  would have been able to succeed by outputting  $(Z, X)$  as its XOR-collision pair. This can be made possible because  $A$  is the one who is providing the answers to  $B$ 's queries. If  $A_1$  picked the two values  $y[i]$  and  $y[k]$  randomly and output  $y[i] \oplus y[k]$  as  $r$ , and then in the second stage, if  $A_2$  positioned the two blocks correctly in the answer to  $B$ 's query,  $A$  would have succeeded in the special scenario described above. Hence, the matter boils down to correctly positioning the two “crucial” blocks ( $y[i]$  and  $y[k]$ ) in the simulated query answers. Since  $A$  does not know ahead of time which two blocks  $B$  will output as its last two ciphertext blocks, the best thing  $A$  can do is to “guess” the positions of the two crucial blocks at random and try to place the blocks at the guessed position in the answers to  $B$ 's queries. Since  $B$  does not know how these blocks were picked, those blocks would not be viewed any different from other blocks by  $B$ . As long as the last two blocks  $B$  outputs came from the response blocks that  $A$  simulated for  $B$ 's query, with some probability  $A$  can guess those positions correctly.

Consider now the general case, where  $B$  queries at most  $q$  queries totaling  $\mu$  bits. The two random permutations  $\pi_1$  and  $\pi_2$  need to be simulated as well as their inverses. Assuming that  $A$  knows the total number of message blocks (bounded by  $\mu/bl$ ) that  $B$  will query,  $A$  can prepare

---

Algorithm  $A_1(k)$

Initialize tables  $T_1$  and  $T_2$  //  $T_1$  and  $T_2$  store values to be used for simulating  $\pi_1$  and  $\pi_2$ .  
Pick  $m = \mu/l$  distinct  $l$ -bit random strings and store them in  $T_1$   
Pick  $q$  distinct  $l$ -bit random strings and store them in  $T_2$   
From the entries of table  $T_1$ , pick a string at random and assign it to  $y_a$   
From the last  $q$  entries of  $T_1$ , pick a string at random and assign it to  $y_b$   
Compute  $y_a \oplus y_b$  and assign it to  $r$   
Return  $(r, (y_a, y_b, T_1, T_2))$

---

Algorithm  $A_2(K, r, s)$

Parse  $s$  as  $(y_a, y_b, T_1, T_2)$  //  $y_a, y_b, T_1, T_2$  are globally accessible  
 $p_1 \leftarrow 1; p_2 \leftarrow m - q + 1; p_3 \leftarrow 1$  //  $p_1, p_2, p_3$  are global pointers to the tables  
For  $i = 1, \dots, q$  do  
     $B \Rightarrow X_i$ ; Parse  $X_i$  as  $x_i[1] \cdots x_i[n_i]$ ;  $\tau_i \leftarrow \mathcal{H}_K(X_i)$   
     $Y_i \leftarrow y_i[0]y_i[1] \cdots y_i[n_i + 1] \leftarrow \mathcal{E}\text{-Sim}(X_i \parallel \tau_i)$   
     $B \Leftarrow Y_i$   
 $B$  outputs  $C = c[0] \cdots c[n + 1]$  as its forgery ciphertext  
 $X \parallel \tau \leftarrow \mathcal{D}\text{-Sim}(C)$  // Decryption simulator  
If  $C$  is *new* and  $\tau = \mathcal{H}_K(X)$  and  $c[n + 1] \in \{y_1[n_1 + 1], y_2[n_2 + 1], \dots, y_q[n_q + 1]\}$  then  
    Let  $i$  be the index such that  $c[n + 1] = y_i[n_i + 1] \in \{y_1[n_1 + 1], y_2[n_2 + 1], \dots, y_q[n_q + 1]\}$   
    Return  $(X, X_i)$

---

Figure 6: The XOR collision finder algorithm  $A$  for the proof of Lemma 6.10. It invokes the subroutine  $\mathcal{E}\text{-Sim}(\cdot)$  of Figure 7. The description of  $\mathcal{D}\text{-Sim}(\cdot)$  is omitted.

---

the output values for the simulated permutations by picking distinct values at random for each permutation. Since in the first stage,  $A_1$  needs to output the target value  $r$  of the XOR-collision,  $A_1$  can prepare the output values for the simulated permutation and pick the two crucial blocks. The prepared output values can be stored in two tables (one for each permutation). In the second stage,  $A_2$  can answer  $B$ 's queries by simulating the permutations using the prepared values in the two tables. For each input to the permutation,  $A_2$  can use the values in the tables one by one in order. Care must be taken, however. If the input to the permutation in question is previously defined, the defined value needs to be returned regardless of what the prepared value in the table, meaning the particular value in the table is discarded. This would be a problem if the discarded value is one of the crucial block values —if either of them is unused in the responses to  $B$ 's queries,  $A$  will bound to fail. Part of the error term in Equation (6) computed from this “bad” event. Other “bad” events are considered in the detailed analysis which will be given shortly.

ALGORITHMS AND ANALYSIS. We now present the actual algorithms for the XOR-collision finder  $A = (A_1, A_2)$ . The algorithms are shown in Figure 6. The algorithm  $A_1$  invokes two subroutines:  $\mathcal{E}\text{-Sim}$  and  $\mathcal{D}\text{-Sim}$ . We present only the  $NCBC[P^l]$  encryption algorithm simulator  $\mathcal{E}\text{-Sim}$  and omit the decryption algorithm simulator  $\mathcal{D}\text{-Sim}$  because the latter performs just the inverse operations of the former. The subroutine  $\mathcal{E}\text{-Sim}$  and its invoked subroutines are shown in Figure 7.

In the algorithm  $A$ ,  $T_1$  and  $T_2$  denote the tables that store the values that are used in the subroutines P1-Sim and P2-Sim in order to simulate the two permutations,  $\pi_1$  and  $\pi_2$  that are chosen at random from  $P^l$ . In the subroutines, P1( $\cdot$ ) and P2( $\cdot$ ) denote the tables that store the actual input and output values that are generated via invoking the subroutines P1-Sim and P2-Sim.

Subroutine $\mathcal{E}\text{-Sim}(X)$ Parse $X$ into $x_1 \cdots x_{n+1}$ $y_0 \stackrel{R}{\leftarrow} \{0, 1\}^l$ For $i = 1, \dots, n - 1$ do $y_i \leftarrow \text{P1-Sim}(y_{i-1} \oplus x_i, \perp)$ $y_n \leftarrow \text{P1-Sim}(y_{n-1} \oplus x_n, \text{LST})$ $y_{n+1} \leftarrow \text{P2-Sim}(y_n \oplus x_{n+1})$ Return $y_0 y_1 \cdots y_{n+1}$	Subroutine $\text{P1-Sim}(x, \text{flag})$ If $(\text{flag} \neq \text{LST})$ then $y \leftarrow \text{T}_1[p_1]; p_1 \leftarrow p_1 + 1$ else $y \leftarrow \text{T}_1[p_2]; p_2 \leftarrow p_2 + 1$ If $\text{P1}(x)$ is defined then If $(y = y_a)$ or $(y = y_b)$ then $\text{Bad} \leftarrow \text{true}$ return $\text{P1}(x)$ $\text{P1}(x) \leftarrow y; \text{Return } y$	Subroutine $\text{P2-Sim}(x)$ $y \leftarrow \text{T}_2[p_3]$ $p_3 \leftarrow p_3 + 1$ If $\text{P2}(x)$ is defined then return $\text{P2}(x)$ $\text{P2}(x) \leftarrow y$ Return $y$
--	--	--

Figure 7: The encryption-oracle and random permutation simulator subroutines for the XOR collision finder algorithm of Figure 6.

The value  $m = \mu/l$  denotes the total number of query message blocks, and  $q$  denotes the number of queries of  $B$ . Since  $\text{T}_1$  is used to simulate  $\pi_1$  that is applied to all of the  $m$  query blocks,  $m$  distinct values are picked for  $\text{T}_1$ . For  $\text{T}_2$ ,  $q$  distinct values are picked because  $\text{T}_2$  is used to simulate  $\pi_2$  which is applied to the redundancy blocks. The pointers  $(p_1, p_2, p_3)$  are used to keep track of the current positions in the tables so that the entries are picked in order. The strings  $y_a$  and  $y_b$  randomly chosen from the table  $\text{T}_1$  denote the two “crucial” blocks described earlier. Notice that  $y_b$  is chosen from the last  $q$  entries in  $\text{T}_1$ , whereas  $y_a$  is chosen from all the entries in  $\text{T}_1$ . The last  $q$  entries in  $\text{T}_1$  are reserved to be used as the images for the last message blocks that are processed by  $\text{P1-Sim}$ . The flag  $\text{LST}$  is used in the subroutines  $\mathcal{E}\text{-Sim}$  and  $\text{P1-Sim}$  to indicate the simulated outputs are to be picked from the last  $q$  entries in  $\text{T}_1$  where the crucial block  $y_b$  belongs to.

Recall that in order for  $A$  to succeed in finding the XOR-collision pair, the last two forgery blocks  $B$  outputs need to come from the query response blocks simulated by  $A$ . In particular, the last forgery block  $c[n+1]$  needs to be chosen from the query response blocks generated by  $\text{P2-Sim}$ , and the second to the last block  $c[n]$  needs to be chosen from the query response blocks generated by  $\text{P1-Sim}$ . If not,  $A$  will not succeed even if  $B$  succeeds. We denote by  $\text{BAD1}$  the event where  $B$  succeeds in forging without using the blocks as described. We claim that the probability  $\text{BAD1}$  occurs is at most  $1/(2^l - q) + 1/(2^l - m)$ . Notice that  $c[n+1]$  and  $c[n]$  are considered “new” if they did not come from the outputs of  $\text{P2-Sim}$  and  $\text{P1-Sim}$ , respectively. This is because of the way the decryption is done.  $c[n+1]$  is processed by the inverse operation of  $\text{P2-Sim}$ , and if it is “new”, then a random value out of  $2^l - q$  available values is chosen as its inverse. In a similar way, for the inverse  $\text{P1-Sim}$  operation on  $c[n]$ , a random value out of  $2^l - m$  available values is chosen.

Let us consider another “bad” event, which we mentioned earlier. We denote by  $\text{BAD2}$  the event where  $\text{Bad}$  is set  $\text{true}$  in the subroutines shown in Figure 7. Notice that if all the inputs to  $\text{P1-Sim}$  are distinct,  $\text{Bad}$  will never be set  $\text{true}$ . When there are any two inputs that are not distinct, it is said that a “collision” occurred. Although the event  $\text{BAD2}$  is a more restricted event than a collision, we upper bound the probability of  $\text{BAD2}$  event by the probability of collision because it is simpler and cleaner to analyze. We can bound the collision on the inputs using the standard method. Recall that the inputs to  $\text{P1-Sim}$  are of the form  $y_{i-1} \oplus x_i$ , where  $x_i$  is the  $i$ 'th message block and  $y_{i-1}$  is the output returned from  $\text{P1-Sim}$  for the  $i - 1$ 'th input. For convenience, let us view all the message blocks in  $B$ 's queries as a sequence of blocks  $x_1 \cdots x_m$ , where  $m$  is the total number of query blocks (i.e.  $m = \mu/l$ ). Let  $\text{Col}_i$  denote the event that there is a collision on the inputs of  $\text{P1-Sim}$  up to the  $i$ -th input block. Then

$$\Pr[\text{BAD2}] \leq \Pr[\text{Col}_m] \leq \sum_{i=1}^m \Pr[\text{Col}_i \mid \overline{\text{Col}_{i-1}}] \leq \frac{m^2}{2(2^l - m)}$$

Finally, from the algorithm for  $A$ , it is easy to see that the probability that  $A$  can guess the “correct” positions of the two crucial blocks is  $1/mq$  if  $B$  succeeds and no “bad” event occurs.

We now combine the results obtained from the above analyses. For notational convenience, let  $\text{BAD}$  denote the event where either  $\text{BAD1}$  or  $\text{BAD2}$  occur, and let  $\text{B succeeds}$  denote the event  $\mathbf{Exp}_{PNCBC[P^l, \mathcal{RC}], B}^{\text{int-ctxt}}(k) = 1$ . Note that if  $B$  succeeds in the absence of the event  $\text{BAD}$ ,  $A$  will also succeed if  $A$  can guess the correct positions of the two crucial blocks. Hence,

$$\begin{aligned} \mathbf{Adv}_{\mathcal{RC}, A}^{\text{scr}}(k) &\geq \frac{1}{mq} \cdot \Pr \left[ \text{B succeeds} \wedge \overline{\text{BAD}} \right] \\ &\geq \frac{1}{mq} \cdot (\Pr [\text{B succeeds}] - \Pr [\text{BAD}]) \\ &\geq \frac{1}{mq} \cdot \left[ \Pr [\text{B succeeds}] - \frac{1}{2^l - q} - \frac{1}{2^l - m} - \frac{m^2}{2(2^l - m)} \right] \end{aligned}$$

This is exactly Equation (6).  $\blacksquare$