A Subliminal Channel in Secret Block Ciphers

Adam Young¹ and Moti Yung²

¹ Cigital, Inc.

ayoung@cigital.com

² Dept. of Computer Science, Columbia University

moti@cs.columbia.edu

Abstract. In this paper we present the first general purpose subliminal channel that can be built into a secret symmetric cipher by a malicious designer. Subliminal channels traditionally exploit randomness that is used in probabilistic cryptosystems. In contrast, our channel is built into a *deterministic* block cipher, and thus it is based on a new principle. It is a broadcast channel that assumes that the sender and the receiver know the subliminal message m_s (i.e., something derived from their common key). We show that the designer can expect to be able to read m_s when $O(|m_s| \log |m_s|)$ plaintext/ciphertext pairs are obtained. Here $|m_s|$ is the length of m_s in bits. We show how to turn the channel into a narrowcast channel using a deterministic asymmetric cipher and then present an application of the narrowcast channel. In this application, the secret block cipher securely and subliminally transmits the symmetric key of the sender and receiver to the malicious designer and confidentiality holds even when the cipher is made public.

1 Introduction

One of the central concerns, from a user's standpoint, of using a black-box cryptographic device is the possibility that the output might maliciously expose private data. Cryptosystems that contain subliminal channels allow this type of information leakage to occur. Subliminal channels have been shown to exist in digital signing algorithms, asymmetric key generation algorithms, and so forth. These channels traditionally exploit randomness in the algorithms they are built into, and more recently it has been shown how to exploit redundancies in plaintext to create such channels. The methodology behind identifying a subliminal channel within a particular cryptosystem involves identifying these features wherever they are present. However, to date, no general-purpose subliminal channel has been shown to exist in a block cipher which is a deterministic function (this holds for published as well as secret designs). In this paper we present the first general-purpose subliminal channel in secret block ciphers.

We stress again that this channel is built into a *deterministic* function, which contrasts with traditional subliminal channels that have been identified in *probabilistic* cryptosystems. Subliminal channels typically use acceptance/rejection on the random bits that are available to the cryptosystem in order to transmit

the subliminal message m_s in the output. A deterministic channel within a deterministic cryptosystem does not have the luxury of using random bits in this fashion, so it may seem somewhat counter-intuitive that subliminal information can be encoded in the output ciphertext.

This brings us to the nature of our channel, which is atypical and suggests that the definition of a subliminal channel should be broadened. A typical subliminal channel allows, e.g., a sender to send a subliminal message (through digital signatures for example) to a receiver unbeknownst to a passive eavesdropper. Our channel allows a sending device to send a subliminal message to a receiving device *that already knows the subliminal message* such that the subliminal message can be read by a known-plaintext attacker that is privy to secret information (a particular private key). Our main application is a backdoor attack against a secret block cipher that covertly leaks the *asymmetric* encryption of the sender and receiver's symmetric key to the cipher designer. Furthermore, the channel employed this way is robust against reverse-engineering (robust meaning that the confidentiality of encryptions still holds even if the cipher becomes public).

Due to space limitations only an overview of the other applications will be given. A second application is an attack on operating systems that share the same decryption private key within the security kernel (e.g., to decrypt encrypted programs before running them). The attack subliminally leaks the private key. A third application is an attack that subliminally leaks the private signing symmetric key of Russia to the U.S. within the Salt II Treaty verification protocol [23].

1.1 Modern Motivations for Backdoor Research

In the past, the motivation for studying backdoors in symmetric secret ciphers was simple. The U.S. government had endorsed Skipjack, a classified block cipher [16]. RC4 is another cipher that was initially a trade secret (of RSA Data Security Inc.). Companies still engage in the practice of marketing secret ciphers under the premise that they are proprietary. So, the chief motivation for this research has not changed with time.

Also, digital rights management efforts have sought to utilize software obfuscation and hardware implementations to protect digital content ([4] obfuscated a published cipher). In fact, recently Boneh et al. described a technique akin to differential fault analysis to cryptanalyze a simplified version of a software obfuscation package [10]. They suggested countermeasures to their attack and mentioned the possibility of using secret designs prior to obfuscating the code. So, the hazards of secret ciphers are still not fully observed.

1.2 Subliminal Channels and Backdoor Ciphers

To show that the notion of a subliminal channel was applicable in practice, the *prisoner's problem* was devised [21]. In this problem, two prisoners want to communicate *subliminal messages* to each other within digital signatures so that the

warden, who verifies all signatures, will not know this is taking place. Solutions to this problem have been given for DSA, ElGamal, and others. Progress has been slow in subliminal channel research [6, 1] in both identifying attack possibilities as well as in formally defining the subject. Progress has also been slow, yet steady, in researching robust backdoors in cryptosystems [24, 26]. Previously, no subliminal channel of a general nature was known to exist in secret block ciphers nor in publishable block ciphers.

Rijmen and Preneel gave a methodology for designing a backdoor cipher where the cipher that results has a *public* specification [19]. The recovery ability is based on a specific trapdoor that allows the designer to break the encryption using linear cryptanalysis. Security issues with this methodology were subsequently addressed [25]. Related work includes [17] which was cryptanalyzed in [3, 7].

Earlier work on building backdoors into symmetric ciphers showed that a shared string between the sender and receiver (the symmetric encryption key) could be subliminally leaked. However, these designs left room for improvement. The cipher in [27] reveals multiple plaintext bits (in particular bit positions) in every ciphertext to the reverse-engineer. The backdoor presented in [28] only leaks subliminal information when the input plaintext is highly redundant and also reveals an upper or lower bound on the entropy of every plaintext to the reverse-engineer. The non-trivial security goal that we achieve in this paper is the construction of a backdoor cipher that does not leak plaintext bits to the reverse-engineer nor one that requires redundancy in plaintexts to operate correctly.

A fundamental difference between the cipher we present here and Monkey [27] is the use of *pre-processing symmetric decryption* in the block encryption function (Monkey uses only a post-processing encryption during block encryption). Thus, the block encryption function presented here involves *both* post and pre-processing transformations in the block encryption function. This is an integral part of the mechanism that prevents the reverse-engineer from learning plaintext bits in individual bit positions.

2 Notation and Definitions

$A || B$ denotes the concatenation of strings A and B . $A \oplus B$ denotes the bitwise XOR of A with B . The symbol $\hat{=}$ is used to define a function. For example, $f(x) \hat{=} x^2 + 3x - 7$. $\{0, 1\}^\infty$ is the set of all bit strings that are each countably infinite in length. A random oracle is defined [2] as:

Definition 1. A random oracle R is a function from $\{0, 1\}^*$ to $\{0, 1\}^\infty$ such that for a given query s to R , each and every output bit of $R(s)$ is chosen uniformly at random and independent of every bit in s .

If R is given query s twice, R will respond with the same bit string.

A random function is similar to a random oracle except that the range is defined to be a finite set. It is possible to instantiate a random function F_θ :

$\{0, 1\}^* \rightarrow S$ where $S = \{0, 1, 2, \dots, \theta - 1\}$ using a random oracle R as follows. Let s be a binary string. Let t be the first $|\theta|$ bits of the infinitely long string $R(s)$. If $t \in S$ then $F_\theta(s) = t$. Otherwise, consider the next $|\theta|$ bits of $R(s)$. If this string is contained in S then this string is $F_\theta(s)$, and so on. So, $F_\theta(s)$ maps to an element drawn uniformly at random from S .

In practice, cryptographic hash functions are used to instantiate random oracles (and random functions). However, when a public and private string are supplied to a random function it may be desirable to use a pseudorandom function [9] instead.

A *block cipher* is a pair of algorithms (ENC, DEC) that is used to encrypt and decrypt plaintext messages m that are w bits in length. The encryption is performed using a key k such that for all m , the equality $m = DEC(k, ENC(k, m))$ holds. The function ENC must therefore be injective to allow unambiguous decryption. In a block cipher, $\{0, 1\}^w$ is both the set of plaintexts as well as the set of ciphertexts. The encryption function for a block cipher is therefore a bijection (and must be deterministic). There are $2^w!$ possible bijections in total over $\{0, 1\}^w$. This implies that about $w2^w$ bits are needed to represent each possible bijection. An *ideal random cipher* is a w -bit block cipher that implements all $2^w!$ bijections on 2^w elements. Each of the $2^w!$ keys specifies one such permutation [15]. However, a key size of about $w2^w$ bits is highly impractical. So, a weaker definition of a cipher is needed for practical purposes.

An *ideal classic cipher* implements a randomly chosen subset of all $2^w!$ permutations from the message space onto the ciphertext space. They are secure against chosen-plaintext attacks. It is standard practice to make the cardinality of the key space exponential in some security parameter. A design principle for a block cipher is to make the cipher as close to an ideal classic cipher as possible. A block cipher is “secure” in some sense, if the encryption function corresponding to a randomly selected key appears to be a randomly chosen invertible function (pseudorandom permutation). The notion of choosing a block cipher at random is as follows. If the subset of permutations can be chosen randomly, then a w -bit ideal classic cipher can be chosen randomly.

Kerckhoffs’ basic principle is that the adversary will somehow learn the details of the cipher. So in principle, all of the secrecy should reside in the key. A *secret block cipher*, in contrast, is a block cipher in which ENC and DEC are known only to the designer, unless an implementation of the cipher is reverse-engineered. However, the key space, message space, and ciphertext space are publicly known. A secret block cipher is a gross violation of Kerckhoffs’ principle, and it is this that motivates the present investigation. Namely, we strive to justify this principle from the perspective of information leakage attacks.

Recall that IND-CPA stands for *Indistinguishability - chosen-plaintext attack*. In an IND-CPA attack, the adversary first mounts a CPA attack. The adversary then submits two plaintexts of his choice to the encryption oracle and is then given the encryption of one of the two at random and is asked to guess which plaintext was encrypted. A cipher is IND-CPA secure if the adversary guesses correctly with probability negligibly greater than $1/2$.

3 New Definitions and Building Blocks

Informally, a *broadcast block cipher* is a block cipher that broadcasts a subliminal message m_s to everyone that knows the algorithm for the cipher. The bit string m_s must be supplied as input (e.g., internally) to the broadcast encryption algorithm as well as the broadcast decryption algorithm. Define $BENC(k, m, k_s, m_s)$ to be a broadcast block cipher with a w -bit block size, where m is the plaintext, k is the symmetric key used to encrypt m , m_s is the subliminal message, and k_s is a secret key. Algorithm $BDEC(k, c, k_s, m_s)$ decrypts the output of $BENC$ using k . Define algorithm $ENC(k, m) \triangleq BENC(k, m, k_s, m_s)$ and define algorithm $DEC(k, c) \triangleq BDEC(k, c, k_s, m_s)$ for some pair (k_s, m_s) .

Our threat model involves four entities that we define informally yet intuitively (a more formal definition will be given in the full version). The role of each of these entities is as follows.

The **designer** is a malicious entity that is permitted to design and deploy the black-box device. Therefore, the designer only has write-once and oracle access to the device. The goal of the designer is to learn m_s . The designer supplies the secret key k_s and the collected plaintext/ciphertext pairs to an algorithm called $BREC$ which then returns the subliminal message m_s .

The **reverse-engineer** has only oracle access to the device (i.e., the cipher can't be changed). The reverse-engineer mounts a chosen-plaintext attack. Upon completion of the attack the reverse-engineer knows α plaintext/ciphertext pairs denoted by $(m_1, c_1), (m_2, c_2), \dots, (m_\alpha, c_\alpha)$ computed using $BENC(k, \cdot, k_s, m_s)$. It is assumed that these ciphertexts subliminally transmit m_s . So, the reverse-engineer knows $BENC$, $BDEC$, m_s , k_s as well as the pairs $(m_1, c_1), \dots, (m_\alpha, c_\alpha)$. The goal of the reverse-engineer is to break the cipher by learning a non-negligible amount of plaintext information.

The **inquirer** and **sampler** have only oracle access to the cipher. The inquirer is a boolean function. An output of true indicates a guess that the cipher is an ideal classic cipher. An output of false indicates a guess that it is not an ideal classic cipher (e.g., that it contains a subliminal channel and as a result appears less than ideal).

There exists a set of weak probability distributions in the broadcast block cipher that breaks the confidentiality of encryptions with respect to the reverse-engineer. The purpose of formalizing the sampler is to show that with overwhelming probability a weak distribution will not be selected (by users in practice). The sampler chooses a probability distribution over the message space and returns plaintexts sampled accordingly.

So, the inquirer is a user (adversary) that tries to distinguish between “good” and “bad” ciphers and the sampler is a user (adversary) that tries to choose a probability distribution that allows the reverse-engineer to violate the confidentiality of encryptions. It is assumed that the reverse-engineer, inquirer, and sampler are computationally bounded.¹

¹ i.e., they are probabilistic poly-time Turing machines.

Definition 2. A secure w -bit **broadcast block cipher** is a 3-tuple $(BENC, BDEC, BREC)$ that satisfies the following:

1. (*inquirer indistinguishability*) It is computationally intractable for the inquirer to distinguish a black-box implementation of (ENC, DEC) from a randomly chosen ideal classic cipher.
2. (*completeness*) For all plaintexts m , for all symmetric keys k , for all subliminal messages m_s , and for all secret keys k_s , the following equality holds, $m = BDEC(k, BENC(k, m, k_s, m_s), k_s, m_s)$.
3. (*reverse-engineer confidentiality*) After completing a chosen-plaintext attack, the reverse-engineer learns at most W pairs of random plaintexts and corresponding ciphertexts (computed using k) where W is bounded by a polynomial in the length of $((m_1, c_1), \dots, (m_\alpha, c_\alpha), k_s)$.
4. (*designer completeness*) For sufficiently large α , for all m_s , and for all distinct plaintexts $(m_1, m_2, \dots, m_\alpha)$, the subliminal message m_s is equal to $BREC((m_1, m_2, \dots, m_\alpha, c_1, c_2, \dots, c_\alpha), k_s)$ with overwhelming probability where each ciphertext $c_i = BENC(k, m_i, k_s, m_s)$ for $i = 1, 2, \dots, \alpha$. The probability is over the random choice of k and k_s .

Property (2) guarantees that decryption will always yield the original plaintext. The secret key k_s is fixed and is contained in all of the black-box devices. This value must be kept secret (hence, the subscript “s”) by the designer to prevent users from being able to read the channel. The reverse-engineer knows k_s and so the reverse-engineer can read the subliminal channel. So, it is a broadcast channel. It is made narrowcast using the deterministic asymmetric encryption function E . In our main application we set $m_s = E(y, k)$ where y is the designer’s public key. Clearly $E(y, k)$ can be reconstructed by $BENC$ and $BDEC$ on input k . Since only the designer knows the decryption private key corresponding to y , only the designer (and not the reverse-engineer) can learn k .

Property (4) implies that a sufficient number of known plaintext-ciphertext pairs under a common key k are required in order to read m_s . Observe that this construction constitutes a cipher within a cipher. The secret key k_s acts like a symmetric key that decrypts the “ciphertext” $(m_1, m_2, \dots, m_\alpha, c_1, c_2, \dots, c_\alpha)$ to reveal m_s .

Building Blocks. Let s be a bit string. The function $GetBit(s, i)$ returns the bit at position i of s where $0 \leq i \leq |s| - 1$. The bits are ordered from right to left starting with 0. For example, if $s = 0001$ then $GetBit(s, 0) = 1$, $GetBit(s, 1) = 0$, $GetBit(s, 2) = 0$, and $GetBit(s, 3) = 0$. Let $H_{\delta-1} : \{0, 1\}^* \rightarrow \{0, 1\}^{\delta-1}$ be a random function. Also, let $F_1 : \{0, 1\}^* \rightarrow \{0, 1\}$ and $GetRandPos_\theta : \{0, 1\}^* \rightarrow \{0, 1, 2, \dots, \theta - 1\}$ be random functions. These random functions are publicly known.

The pair $(ENC1, DEC1)$ is a secret ideal classic cipher with a w -bit block size. Let δ be a constant. The pair $(ENC2, DEC2)$ is a secret ideal classic cipher with block size $w - \delta$ bits. We are somewhat lax about specifying the set from which the symmetric keys k are drawn. The set is typically, say $\{0, 1\}^{128}$ or an even larger set.

These ciphers can be instantiated using a pseudorandom invertible permutation generator that will make the outputs secure against chosen plaintext attacks [13]. In practice, symmetric ciphers often utilize Feistel transformations. Luby and Rackoff have provided some theoretical justification for Feistel’s construction [12, 11].

4 The Broadcast Block Cipher

The broadcast block encryption algorithm is given in Fig. 1.

Fig. 1. Subliminal Channel in Block Encryption Algorithm

(Intuitive Description). The channel transmits one pseudorandomly chosen bit of the subliminal message in each ciphertext block that is output. First, a large portion of the block is simply encrypted using a secure block cipher. The resulting ciphertext is recoverable under a known plaintext attack by the reverse-engineer. This block is used as the “public” input to a random function.² This

² It is not really a public input, but when pseudorandom functions are used to instantiate these random functions, they correspond to the “public input.”

public input is used to select a bit position randomly in the subliminal message. The problem that remains is to display the subliminal bit in this bit position.

This is accomplished by embedding this bit in the encryption of the remaining plaintext that has not been enciphered. To do so, the public input is again supplied to a random function, but this time the user's symmetric key is supplied to the random function as well. The result is a random pad that is used to XOR encrypt all but the last remaining plaintext bit. This bit is also XOR encrypted. This is accomplished by supplying the pad along with the public input to yet another random function, to obtain a one bit pad. The idea is that the larger pad is secret due to the secrecy of the user's symmetric key, and so this larger pad can be used to derive a secret one-bit pad to XOR encrypt the final plaintext bit. An initial permutation and a final permutation are also performed for reasons that will become clear later on.

(Detailed Description). The pair $k_s = (k_\alpha, k_\beta)$ is randomly chosen by the designer and is placed in the black-box device that implements *BENC* and *BDEC* (so k_s may become known to a reverse-engineer but the symmetric key k that the user supplies to the device will not). For simplicity it is assumed that w is even. The value δ is an integer constant used in the broadcast block cipher. The function f is defined as follows.

$$f(b, \beta_u, k_\ell) \triangleq H_{\delta-1}(k_\ell \parallel \beta_u) \parallel (b \oplus F_1(H_{\delta-1}(k_\ell \parallel \beta_u) \parallel \beta_u))$$

$\Pi_1(k, m, k_\alpha, m_s)$: /* subroutine for encryption algorithm */

1. let k_u and k_ℓ be strings such that $k = k_u \parallel k_\ell$ and $|k_u| = |k_\ell|$
2. compute $\alpha = DEC1(k_\alpha, m)$ /* yes, we want to decrypt */
3. let α_u and α_ℓ be strings such that $\alpha = \alpha_u \parallel \alpha_\ell$ and $|\alpha_\ell| = \delta$
4. $\beta_u = ENC2(k_u, \alpha_u)$
5. compute $i = GetRandPos_{|m_s|}(\beta_u)$ and then $b = GetBit(m_s, i)$
6. compute $pad = f(b, \beta_u, k_\ell)$, $\beta_\ell = pad \oplus \alpha_\ell$, and return $\beta = \beta_u \parallel \beta_\ell$

BENC($k, m, (k_\alpha, k_\beta), m_s$):

Input: w -bit plaintext m , symmetric key k where $|k|$ is even,
subliminal message m_s , secret key (k_α, k_β)

Output: w -bit ciphertext c

1. $\beta = \Pi_1(k, m, k_\alpha, m_s)$
2. compute $c = ENC1(k_\beta, \beta)$ and then output c

$\Pi_{-1}(k, c, k_\alpha, m_s)$: /* subroutine for decryption algorithm */

1. let k_u and k_ℓ be strings such that $k = k_u \parallel k_\ell$ and $|k_u| = |k_\ell|$
2. compute $\beta = DEC1(k_\beta, c)$
3. let β_u and β_ℓ be strings such that $\beta = \beta_u \parallel \beta_\ell$ and $|\beta_\ell| = \delta$
4. $\alpha_u = DEC2(k_u, \beta_u)$
5. compute $i = GetRandPos_{|m_s|}(\beta_u)$ and then $b = GetBit(m_s, i)$
6. compute $pad = f(b, \beta_u, k_\ell)$, $\alpha_\ell = pad \oplus \beta_\ell$, and return $\alpha = \alpha_u \parallel \alpha_\ell$

$BDEC(k, c, (k_\alpha, k_\beta), m_s)$:

Input: w -bit ciphertext c , symmetric key k where $|k|$ is even,
subliminal message m_s , secret key (k_α, k_β)

Output: w -bit plaintext m

1. $\alpha = \Pi_{-1}(k, c, k_\alpha, m_s)$
2. compute $m = ENC1(k_\alpha, \alpha)$ and then output m

It is clear that property (2) of Definition 2 holds (completeness).

BRECBIT recovers a single bit of m_s from a plaintext/ciphertext pair. *BREC* invokes this subroutine for each plaintext/ciphertext pair that it is given in order to recover m_s .

$BRECBIT((k_\alpha, k_\beta), m, c)$:

Input: w -bit plaintext m , w -bit ciphertext c ,
secret key (k_α, k_β)

Output: (i, b) where b is the bit at bit position i of m_s

1. compute $\alpha = DEC1(k_\alpha, m)$
2. let α_u and α_ℓ be strings such that $\alpha = \alpha_u || \alpha_\ell$ and $|\alpha_\ell| = \delta$
3. compute $\beta = DEC1(k_\beta, c)$
4. let β_u and β_ℓ be strings such that $\beta = \beta_u || \beta_\ell$ and $|\beta_\ell| = \delta$
5. compute $pad = \alpha_\ell \oplus \beta_\ell$
6. let z and t be strings such that $pad = z || t$ and $|t| = 1$
7. compute $r = F_1(z || \beta_u)$ and then set $b = t \oplus r$
8. compute $i = GetRandPos_{|m_s|}(\beta_u)$ and then output (i, b)

Since each bit is selected uniformly at random from the $|m_s|$ bit positions, the designer (and the reverse-engineer) can expect to have to obtain $O(|m_s| \log |m_s|)$ plaintext/ciphertext pairs under a common key k in order to recover m_s . This results from analyzing the first moment of the *coupon collector's problem* [8]. It follows that property (4) of Definition 2 holds (designer completeness). We call this the insignis channel, named after the carnivorous plant *Nepenthes Insignis* Danser.

5 Security

The following claim is used to show that indistinguishability (i.e., property (1) of Definition 2) holds.

Claim 1. $\forall k, m_s$, and k_α , $\Pi_1(k, \cdot, k_\alpha, m_s)$ is a permutation over $\{0, 1\}^w$.

Proof. Assume for the sake of contradiction that this does not hold. Then there exists k, m_s, k_α, m_1 , and m_2 where $m_1 \neq m_2$ such that $\beta_1 = \Pi_1(k, m_1, k_\alpha, m_s) = \Pi_1(k, m_2, k_\alpha, m_s) = \beta_2$.

Observe that α is a permutation of the input message. So, m_1 maps to α_1 and m_2 maps to α_2 where $\alpha_1 \neq \alpha_2$. Define $\alpha_{u,1}, \alpha_{\ell,1}, \alpha_{u,2}, \alpha_{\ell,2}, \beta_{u,1}, \beta_{\ell,1}, \beta_{u,2}$, and $\beta_{\ell,2}$ as follows:

$$\begin{array}{lll} \alpha_1 = \alpha_{u,1} \parallel \alpha_{\ell,1} & \alpha_2 = \alpha_{u,2} \parallel \alpha_{\ell,2} & |\alpha_{\ell,1}| = |\alpha_{\ell,2}| = \delta \\ \beta_1 = \beta_{u,1} \parallel \beta_{\ell,1} & \beta_2 = \beta_{u,2} \parallel \beta_{\ell,2} & |\beta_{\ell,1}| = |\beta_{\ell,2}| = \delta \end{array}$$

Suppose that $\alpha_{u,1} \neq \alpha_{u,2}$. Since $ENC2(k_u, \cdot)$ is a permutation it follows that $\beta_{u,1} \neq \beta_{u,2}$. Hence, $\beta_1 \neq \beta_2$ in this case. So, it remains to consider the case that $\alpha_{u,1} = \alpha_{u,2}$.

Since $\alpha_1 \neq \alpha_2$ it must be the case that $\alpha_{\ell,1} \neq \alpha_{\ell,2}$. It is not hard to show that pad is the same for m_1 and m_2 . Since $\alpha_{\ell,1} \neq \alpha_{\ell,2}$ it follows that the strings resulting from the bitwise XOR operation differ. Therefore, $\beta_{\ell,1} \neq \beta_{\ell,2}$. It follows that $\beta_1 \neq \beta_2$. Therefore, in all cases $\beta_1 \neq \beta_2$ which is a contradiction. $\diamond$

Let m_s be any subliminal message and let k_s be any secret key. Claim 1 shows that $ENC(k, m)$ maps each $m \in \{0, 1\}^w$ under k to one and only one $\beta \in \{0, 1\}^w$. ENC then encrypts β using a randomly chosen key k_β in a randomly chosen ideal classic cipher. Hence, this last operation is a randomly chosen permutation from among the set of possible random permutations in the ideal classical cipher. Since the composition of two permutations is a permutation and since k_β is secret, we have therefore shown the following.

Claim 2. *A secret implementation of (ENC, DEC) is indistinguishable from an ideal classic cipher.*

It follows that property 1 of Definition 2 holds. The following corollary follows from Claim 2 and the notion of security for an ideal classic cipher.

Corollary 1. *With only oracle access, ENC appears like a randomly chosen invertible function.*

(ENC, DEC) is as secure as $ENC1$ against attacks mounted by a user since the key k_β is secret from the user. Therefore, since $(ENC1, DEC1)$ is secure against chosen-plaintext attacks mounted by the user (note that it is first [14]) (ENC, DEC) is secure against chosen-plaintext attacks mounted by the user. We will now consider the security of (ENC, DEC) when attacks are carried out by the reverse-engineer who knows k_β , among other things. Thus, the remainder of this section is devoted to showing that property (3) of Definition 2 holds.

First, note that β_u is the block encryption of α_u . So, the encryption using $ENC2$ will not compromise k_u unless $ENC2$ is itself vulnerable to a chosen plaintext attack. Now consider the confidentiality of α_ℓ .

Observe that in (ENC, DEC) there exist non-trivial distributions M_p that compromise plaintexts. These M_p 's lead to a non-negligible probability of collision in β_u . A collision in β_u implies a collision in pad . So, we must first show that the chances that the sampler compromises its own plaintexts is negligible.

Define p_c to be the probability that two messages m_1 and m_2 that are chosen according to M_p lead to the same value for β_u in the corresponding encryptions c_1 and c_2 . If p_c is not negligible then the sampler may produce messages that are compromised. This results from the fact that the random functions in f would be given the same β_u thereby resulting in a selection of pads for c_1 and c_2 that are not only dependent, but identical.

Claim 3 utilizes a fact that is related to the *birthday paradox*. An urn has u balls numbered from 1 to u . Suppose that v balls are drawn from the urn one at a time, with replacement, and their numbers are listed. **(Fact 1)** A well-known fact is that as $u \rightarrow \infty$, the expected number of draws before a coincidence is $\sqrt{(\pi u)/2}$.

Claim 3. (*random oracle model*) *If $ENC2$ is an ideal classic cipher and $w - \delta$ is sufficiently large then p_c is negligible.*

Proof. Assume that $ENC1$ is an ideal classic cipher for sufficiently large w . It follows that $DEC1$ is a bijection and so every message maps to one and only one α . Also, assume that $w - \delta$ is sufficiently large.

Since $ENC2$ is an ideal classic cipher, each α_u is assigned randomly to a β_u under the random permutation $ENC2(k_u, \cdot)$. It follows from Corollary 1 that the sampler can do no better than guess a value for M_p that has a non-negligible probability of yielding a collision in β_u when sampled. So, it remains to consider the probability of a collision for randomly chosen β_u 's.

The value β_u is selected randomly from a set with cardinality $2^{w-\delta}$. It follows from Fact 1 that the expected number of ciphertexts needed to have a collision in β_u is close to $2^{(w-\delta)/2} \sqrt{\pi/2}$. Since $w - \delta$ is sufficiently large it follows that with overwhelming probability the β_u 's will be unique. Hence, p_c is negligible. $\diamond$

This analysis implies that $w - \delta$ should be at least 128.

Observe that the function f is known to the reverse-engineer since $H_{\delta-1}$ and F_1 are public. From property (4) of Definition 2 it follows that m_s is known to the reverse-engineer. It is not hard to show that b and β_u are known to the reverse-engineer for every ciphertext. This implies that k_ℓ is a private input and (b, β_u) is a public input (from the reverse-engineer's perspective) to f .

Claim 4. (*random oracle model*) *If p_c is negligible and k_ℓ is secret then with overwhelming probability the values for pad that result (from the sampler's choice of plaintexts) in the resulting ciphertexts are independently random and secret.*

Proof. Assume that k_ℓ is secret from the reverse-engineer. Also, assume that p_c is negligible. It follows from the latter assumption that the β_u 's are different with overwhelming probability. Consider the event that the β_u 's differ.

Since the β_u 's differ, the $\delta - 1$ uppermost bits of pad are selected uniformly at random and independent of every other plaintext. This follows from the fact that β_u is supplied as input to the random function $H_{\delta-1}$. Furthermore, the $\delta - 1$ uppermost bits $H_{\delta-1}(k_\ell || \beta_u)$ of pad are secret due to the secret k_ℓ that is supplied as input to $H_{\delta-1}$ in f .

Since the β_u 's differ, for each β_u , the least significant bit of pad is selected uniformly at random and independent of every other plaintext. This follows from the fact that β_u is supplied as input to the random function F_1 . Furthermore, the least significant bit of pad is secret since it was shown that $H_{\delta-1}(k_\ell || \beta_u)$ is secret and this value is supplied as input to F_1 in f . $\diamond$

This analysis implies that δ should not be less than 64. The assumption here is that 64 is an acceptable security parameter, which even today is arguably

cutting it close. We remark that Rijndael has block sizes of 128, 192, and 256 bits [5]. So, a block size of 192 bits or larger is not unreasonable. Also, the bits that comprise pad are the outputs of a random oracle, so they will not compromise k_ℓ . Claims 3 and 4 show that even if the computationally bounded sampler wants to find an M_p that compromises plaintexts, the probability that the sampler finds such an M_p is negligible.

Now consider the case that the reverse-engineer mounts a known-plaintext attack. Recall that in a known plaintext attack the reverse-engineer queries an encryption oracle (the sampler) and receives $S = \{(m_1, c_1), (m_2, c_2), \dots, (m_\gamma, c_\gamma)\}$. Here c_i is the ciphertext of m_i for $1 \leq i \leq \gamma$. The encryption key k that is generately randomly is used to compute all the ciphertexts, and is kept secret by the sampler. The sampler samples the m_i from the message space according to M_p . The reverse-engineer is not permitted to query the sampler again.

Consider the problem for the reverse-engineer to learn information relating to the plaintext in c where $c \neq c_i$ for $1 \leq i \leq \gamma$. This is possible in the following known plaintext attack. The reverse-engineer computes pad , α_ℓ , β_u , α_u , etc. used in (m_1, c_1) . The reverse-engineer begins to iterate through the possible values for α_ℓ (there are 2^δ possible values in total). The values $\alpha = \alpha_u \parallel \alpha_\ell$ are encrypted using k_α in the cipher $ENC1$. This yields a set of new plaintexts for ENC . The reverse-engineer computes the corresponding ciphertexts in the same way that ENC would. The number of plaintext/ciphertext pairs that can be learned by the reverse-engineer is bounded by a polynomial in the length of $((m_1, c_1), \dots, (m_\gamma, c_\gamma), k_s)$ which is provided as “input” to the computationally bounded reverse-engineer.

This attack also applies to IND-CPA attacks. Note that when the encryption oracle $ENC(k, \cdot)$ is taken away, the reverse-engineer is able to make oracle queries to $ENC1(k_\alpha, \cdot)$ and learn more plaintext/ciphertext pairs. The reverse-engineer can take one of the new plaintext/ciphertext pairs, make the IND-CPA oracle request with the new plaintext, and then distinguish perfectly. For this reason we use a weaker notion of confidentiality than IND-CPA in Definition 2.

However, by taking the probability over the coin tosses of when $ENC1$ was generated, it follows that the new plaintexts that are learned are random ($ENC1$ is an ideal classic cipher). So, the reverse-engineer can only sample the new plaintext/ciphertext pairs *randomly*. Under these arguments, property (3) of Definition 2 holds.

Theorem 1. *(BENC, BDEC, BREC) is a broadcast block cipher.*

6 Kleptographic Attack

This section describes how to carry out a kleptographic attack that leaks k exclusively to the designer. Let (E, D) be a deterministic asymmetric cryptosystem, let y be the public key of the malicious designer and let x be the corresponding private key. The ciphertext $c = E(y, m)$ denotes the encryption of m . To decrypt we compute $m = D(x, c)$.

In the attack, the cipher computes $m_s = E(y, k)$. The designer computes $k = D(x, m_s)$. The reverse-engineer has access to (m_s, y, E) but does not have access to x . So, the broadcast subliminal channel is transformed into a narrowcast channel using y and E . Clearly other information besides k can be leaked this way as well. The notion of a SETUP is given in [26]. We have therefore shown the following.

Lemma 1. *The subliminal transmission of $E(y, k)$ is a SETUP attack against the secret w -bit block cipher (ENC, DEC).*

7 Conclusion

The notion of a broadcast subliminal channel in a block cipher was introduced and an instantiation was given for secret block ciphers. The channel broadcasts a subliminal message to the reverse-engineer and the malicious designer. It was shown how to turn the broadcast channel into a narrow cast channel using a deterministic asymmetric cipher. In the narrowcast construction, only the malicious designer can obtain the plaintext message. An application of the narrowcast channel was given that constitutes the first secretly embedded trapdoor attack against a deterministic block cipher.

References

1. R. Anderson, S. Vaudenay, B. Preneel, K. Nyberg. The Newton Channel. In *Workshop on Information Hiding*, pages 151–156, 1996.
2. M. Bellare, P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Conference on Computer and Communications Security*, pages 62–73, ACM, 1993.
3. E. Biham. Cryptanalysis of Patarin’s 2-Round Public Key System S Boxes (2R). In *Advances in Cryptology—Eurocrypt ’00*, pages 408–416, 1999.
4. S. Chow, P. Eisen, H. Johnson, P. C. van Oorshot. A White-Box DES Implementation for DRM Applications. Workshop on Digital Rights Management, ACM, 2002.
5. J. Daemen, V. Rijmen. The Block Cipher Rijndael. In *Smart Card Research and Applications*, pages 288–296, 2000.
6. Y. Desmedt. Abuses in Cryptography and How to Fight Them. In *Advances in Cryptology—Crypto ’88*, pages 375–389, 1988.
7. Y. Ding-Feng, L. Kwok-Yan, D. Zong-Duo. Cryptanalysis of the “2R” schemes. In *Advances in Cryptology—Crypto ’99*, pages 315–325, 1999.
8. W. Feller. An Introduction to Probability Theory and its Applications. John Wiley & Sons, Inc., pages 210–212, 1957.
9. O. Goldreich, S. Goldwasser, S. Micali. How to Construct Random Functions. *J. of the ACM*, 33(4), pages 210–217, 1986.
10. M. Jacob, D. Boneh, E. Felten. Attacking an obfuscated cipher by injecting faults. ACM Workshop on Digital Rights Management, 2002.
11. L. Knudsen. DEAL: A 128-bit block cipher. Technical Report 151, Department of Informatics, University of Bergen, Norway, Feb. 1998.

12. M. Luby, C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. In *SIAM J. Comput.*, v. 17, 1988, pages 373–386.
13. M. Luby. Pseudorandomness and Cryptographic Applications. Princeton Computer Science Notes, Princeton University Press, Lectures 13 & 14, pages 128–145, 1996.
14. U. Maurer, J. Massey. Cascade Ciphers: The Importance of Being First. In *Journal of Cryptology*, vol. 6, no. 1, pages 55–61, 1993.
15. A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, pages 224–225, 1997.
16. Skipjack Symmetric Cipher. Declassified on June 23, 1998. Appeared on NIST website on June 24, 1998 at <http://csrc.nist.gov/encryption/skipjack-1.pdf> and <http://csrc.nist.gov/encryption/skipjack-2.pdf> (no author).
17. J. Patarin, L. Goubin. Asymmetric Cryptography with S-Boxes. In *Proceedings of ICICS*, pages 369–380, 1997.
18. M. Rabin. Digitalized Signatures as Intractable as Factorization. MIT Laboratory for Computer Science, MIT/LCS/TR-212, Jan, 1979.
19. V. Rijmen, B. Preneel, A Family of Trapdoor Ciphers. Fast Software Encryption, pages 139–148, 1997.
20. R. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. CACM, v. 21, n. 2, pages 120–126, Feb. 1978.
21. G. J. Simmons. The Prisoners’ Problem and the Subliminal Channel. In *Advances in Cryptology—Crypto ’83*, pages 51–67, Plenum Press, 1984.
22. G. J. Simmons. Subliminal Communication is Easy Using the DSA. In *Advances in Cryptology—Eurocrypt ’93*, pages 218–232, 1993.
23. G. J. Simmons. The History of Subliminal Channels. *IEEE Journal on selected areas in communication*, v. 16, n. 4, pages 452–462, 1998.
24. R. Weis, S. Lucks. “All Your Key Bit Are Belong to Us” The True Story of Blackbox Cryptography. In *Proceedings of the 3rd International System Administration and Networking Conference—SANE ’02*, Maastricht, 2002.
25. H. Wu, F. Bao, R. Deng, Q. Ye. Cryptanalysis of Rijmen-Preneel Trapdoor Ciphers. In *Advances in Cryptology—Asiacrypt ’98*, pages 126–132, 1998.
26. A. Young, M. Yung. The Dark Side of Black-Box Cryptography, or: Should we trust Capstone? In *Advances in Cryptology—Crypto ’96*, pages 89–103, 1996.
27. A. Young, M. Yung. Monkey: Black-Box Symmetric Ciphers Designed for Monopolizing KEYS. In *Fast Software Encryption*, pages 122–133, 1998.
28. A. Young, M. Yung. Backdoor Attacks on Black-Box Ciphers Exploiting Low-Entropy Plaintexts. In *Proceedings of ACISP*, pages 297–311, 2003.