How to Encrypt with a Malicious Random Number Generator

Seny Kamara[1⋆] and Jonathan Katz[2⋆⋆]

[1] Johns Hopkins University
seny@cs.jhu.edu
[2] University of Maryland
jkatz@cs.umd.edu

Abstract. Chosen-plaintext attacks on private-key encryption schemes are currently modeled by giving an adversary access to an oracle that encrypts a given message m using random coins that are generated *uniformly at random* and *independently* of anything else. This leaves open the possibility of attacks in case the random coins are poorly generated (e.g., using a faulty random number generator), or are under partial adversarial control (e.g., when encryption is done by lightweight devices that may be captured and tampered with).

We introduce new notions of security modeling such attacks, propose two concrete schemes meeting our definitions, and show generic transformations for achieving security in this context.

Key words: Private-key encryption, random number generation

1 Introduction

Security against chosen-plaintext attacks (CPA-security) [10, 2, 12] is, nowadays, considered a minimal notion of security that any private-key encryption scheme deployed in practice should satisfy. (We defer for now any discussion of security against chosen-ciphertext attacks, though we will consider such attacks later.) Very roughly speaking, CPA-security means that given a challenge ciphertext generated using an unknown key K, a computationally-bounded adversary cannot recover any partial information about the underlying plaintext even if it is given access to an encryption oracle that returns an encryption (using the same key K) of any message m provided by the adversary. This "encryption oracle" is meant, in part, to model potential real-world actions of an adversary that might influence the honest sender (holding the key K) to encrypt certain messages that are (partially or entirely) under the adversary's control.

It is not hard to see that any scheme secure with respect to chosen-plaintext attacks must be *probabilistic*. Furthermore, it is by now well-understood how to construct CPA-secure schemes under the assumption that the sender is able

⋆ Supported in part by the Phillips and Camille Bradford Fellowship.
⋆⋆ Research supported in part by the U.S. Army Research Laboratory.

to generate a fresh set of uniformly random coins each time a message is encrypted. In practice, such coins might be generated by using a combination of randomness extractors and pseudorandom number generators (PRNGs) to distill pseudorandom coins from a high-entropy source available to the sender.

The above, however, neglects the possibility that the random coins used to encrypt may sometimes be "less than perfect". For example, the sender may be using a faulty PRNG that produces biased or partially predictable outputs. Or, the random source used to seed the PRNG may have less entropy than originally thought. More malicious scenarios include the possibilities that an adversary may have tampered with the PRNG used by the sender, or may be able to effect some control over the random source used to seed the PRNG. In the most extreme case, the adversary may have physical access to the device performing the encryption (as might be the case if, e.g., encryption is carried out on a lightweight device captured by the adversary), and may then have complete control over the "random coins" that will actually be used to encrypt. We refer to such attacks as *chosen-randomness attacks*.

In this work, we introduce new definitions of security that offer protection against the attacks just described. Our definitions assume the worst possible case: that the randomness used by the encryption oracle is under the complete control of the adversary. In fact, the only random coins that are not under the adversary's control (other than those used to generate the key) are those that are used to encrypt the challenge ciphertext; we assume those coins are truly random.[1] Our definition, then, can be viewed (informally) as offering *semantic security* for any messages that are encrypted using "good" random coins, even if the adversary is able to "probe" the system repeatedly and thereby cause the sender to use "poor" random coins when encrypting other messages.

Summary of our contributions. We formally define security against chosen-randomness attacks (CRA-security), both with and without the additional presence of a decryption oracle. We then show two secure constructions that can be based on any block cipher. The first is a relatively simple fixed-length construction, while the second is a scheme that can encrypt arbitrary-length messages. We also show a generic transformation converting any CPA-secure scheme into a CRA-secure scheme. Finally, we propose a simple way to extend any CRA-secure scheme so as to also achieve security against chosen-ciphertext attacks.

1.1 Related Work

The most relevant prior work is perhaps Rogaway's notion of *nonce-based* private-key encryption [15], which treats the encryption algorithm as a deterministic

[1] It is not hard to see that *some* assumption regarding those coins is necessary in our setting (if the adversary has complete control over *all* coins, then the scheme degenerates to a deterministic one that cannot be secure); our assumption that the coins used to generate the challenge ciphertext are truly random is made for simplicity, and can be relaxed by using randomness extractors and assuming only access to a high-entropy source when encrypting the challenge ciphertext.

function of the message and a user-provided nonce. (With respect to this viewpoint, it is the responsibility of the user — not the encryption algorithm — to ensure, e.g., that nonces are chosen at random.) In this context, Rogaway defines a notion of security that, roughly speaking, guarantees semantic security *as long as nonces never repeat*. While this definition is somewhat similar to our own, we show in Section 3.1 that the notion considered by Rogaway is *incomparable* to the notion of CRA-security considered here; i.e., there are schemes satisfying his definition and not ours, and vice versa. We remark further that the motivations for our work and Rogaway's work are very different: as argued by Rogaway [15], nonce-based security is best understood as a usability requirement, whereas we are interested in examining a stronger attack model (within the conventional framework for encryption).

Adversarial manipulation of a PRNG was mentioned as motivation for our work. While there has been prior work developing "forward-" and "backward-secure" pseudorandom number generators [4, 1, 8], simply composing such generators with a standard CPA- or CCA-secure encryption scheme does *not* defend against the attacks considered here. The reason is that these prior works consider only adversaries that *learn* the internal state of the PRNG, whereas our notions consider stronger adversaries that may *control* the state of the PRNG. One can therefore view our notion of CRA-security as achieving a strong variant of backward- and forward-security with respect to the underlying source of randomness. In other words, our definitions guarantee that a plaintext encrypted using high-quality randomness is protected even against adversaries that can control the source after the present plaintext is encrypted (i.e., strong forward-security), or that have controlled it in the past (i.e., strong backward-security).

Work of McInnes and Pinkas [13] and Dodis et al. [7, 5] has also considered the security of encryption when truly random coins are not available to the sender. Although these works are superficially related to our own, the problems being considered — as well as the motivation — are very different. The work of [13, 7, 5] is unwilling to assume *any* truly random coins, even during generation of the secret key, and is interested in exploring what can be achieved in such an extreme setting. For this reason, they are primarily concerned with information-theoretic security (although later work [6, 5] treats computational security) and do not consider security against chosen-plaintext attacks at all. In this work, in contrast, we are willing to assume that truly random coins *exist* (e.g., during key generation and, at least once, when encrypting), but are concerned that the adversary may periodically be able to tamper with the honest user's ability to generate true random coins. We are then interested in the question of whether the analogue of CPA-security is achievable.

2 Notation and Preliminaries

We use standard cryptographic notation and terminology. We use $\langle a, b \rangle$ or $a \| b$ interchangeably for the concatenation of strings a and b. We let $\mathsf{Func}[n, m]$ denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$, and let $\mathsf{Perm}[n, n]$ denote the set of all permutations over $\{0,1\}^n$. A function $f : \mathbb{N} \to \mathbb{N}$ is negligible in k, if for every polynomial p and sufficiently large k, $f(k) < 1/p(k)$. We write $\mathsf{negl}(k)$ and $\mathsf{poly}(k)$ to refer to unspecified negligible and polynomial functions in k.

If $\mathcal{O}$ is a probabilistic algorithm, then $\mathcal{O}(x)$ denotes an execution of $\mathcal{O}$ on input x with uniformly chosen random coins, and $\mathcal{O}(x;r)$ denotes an execution of $\mathcal{O}$ on input x with random coins r. Given a probabilistic algorithm $\mathcal{O}$, we will consider adversaries $\mathcal{A}$ given access to an oracle that on input $\langle x, r\rangle$ returns $\mathcal{O}(x;r)$ (this is different from the usual case where $\mathcal{A}$ is given access to an oracle that on input x returns $\mathcal{O}(x;r)$ for uniformly chosen random coins r).

2.1 Cryptographic Tools

We use standard cryptographic tools which are reviewed here to fix notation.

Pseudo-random functions. Let F be an efficiently-computable keyed function, where for a fixed key K of length k we have $F_K : \{0,1\}^{\ell_{in}(k)} \to \{0,1\}^{\ell_{out}(k)}$ with ℓ_{in}, ℓ_{out} polynomial in k. We say F is a *pseudorandom function* (PRF) if, informally, the function F_K (for a random key $K \in \{0,1\}^k$) is indistinguishable from a function f chosen uniformly at random from $\mathsf{Func}[\ell_{in}(k), \ell_{out}(k)]$. If F is a PRF and F_K is an efficiently-invertible permutation for each choice of K, then we call F a *pseudorandom permutation* (PRP). We refer to [11] for the formal definitions, which are standard.

Encryption. A private-key encryption scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three polynomial-time algorithms with the following functionality:

- Gen takes as input a security parameter 1^k and returns a key K.
- Enc is a probabilistic algorithm that takes as input the key K and a message m from some associated message space. It returns a ciphertext c, and we denote this by $c \leftarrow \mathsf{Enc}_K(m)$.
- Dec takes as input the key K and a ciphertext c; it returns either a message m in the message space or a special failure symbol $\perp$. We write this as $m := \mathsf{Dec}_K(c)$, and assume without loss of generality that Dec is deterministic.

We assume perfect correctness: for all $k \in \mathbb{N}$, all K output by $\mathsf{Gen}(1^k)$, and all messages m in the message space, $\mathsf{Dec}_K(\mathsf{Enc}_K(m)) = m$.

In most schemes, $\mathsf{Gen}(1^k)$ simply outputs a random key of length k; when this is the case, we write $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$.

We use the standard notions of security against chosen-plaintext attacks (CPA-security) and security against (adaptive) chosen-ciphertext attacks (CCA-security); see, e.g., [11].

Message authentication codes (MACs). A message authentication code $\mathsf{MAC} = (\mathsf{Mac}, \mathsf{Vrfy})$ is a pair of polynomial-time algorithms. Mac takes as input a key $K \in \{0,1\}^k$ and a message $m \in \{0,1\}^*$ and outputs a tag t; we assume[2] that Mac is deterministic and denote this by $t := \mathsf{Mac}_K(m)$. The deterministic verification algorithm Vrfy takes as input a key $K \in \{0,1\}^k$, a message

[2] This is without loss of generality, and anyway holds for many common constructions.

$m \in \{0,1\}^*$, and a tag t; it outputs a bit $b := \mathsf{Vrfy}_K(m,t)$ where a '1' indicates acceptance and a '0' indicates rejection. We assume that for all $k \in \mathbb{N}$, all $K \in \{0,1\}^k$, and all $m \in \{0,1\}^*$ it holds that $\mathsf{Vrfy}_K(m, \mathsf{Mac}_K(m)) = 1$.

Note that we assume trivial key generation, where on security parameter 1^k the key is chosen uniformly from $\{0,1\}^k$. We also assume MACs for arbitrary-length messages. (Neither assumption is essential, but making these assumptions simplifies the presentation.)

We use the standard definition of existential unforgeability under an adaptive chosen message attack; see [11]. A MAC has *unique tags* if for all $k \in \mathbb{N}$, all $K \in \{0,1\}^k$, and all $m \in \{0,1\}^*$, there is a unique t such that $\mathsf{Vrfy}_K(m,t) = 1$.

3 Definitions

We now present our definitions of CRA and CCRA-security. Intuitively, CRA-security guarantees that, given a ciphertext, no polynomially-bounded adversary can recover any partial information about the plaintext even if it has access to an encryption oracle *and complete control over its source of randomness.*

Definition 1 (CRA-security). *Let* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a private-key encryption scheme.* SKE *is* CRA-secure *if the advantage of any polynomial-time adversary* $\mathcal{A}$ *in the following experiment is negligible (in* k*):*

1. *First, a key* $K \leftarrow \mathsf{Gen}(1^k)$ *is generated.*
2. $\mathcal{A}$ *is allowed to interact with an oracle* $\mathsf{Enc}_K(\cdot\,;\cdot)$. *We stress that, here,* $\mathcal{A}$ *submits pairs* $\langle m, r\rangle$ *and, in return, is given* $\mathsf{Enc}_K(m;r)$. *(Since* $\mathcal{A}$ *can choose* r *uniformly at random, this is at least as strong as a chosen-plaintext attack.)*
3. $\mathcal{A}$ *outputs two equal-length messages* m_0, m_1. *A bit* b *is chosen at random, and a "challenge ciphertext"* $c \leftarrow \mathsf{Enc}_K(m_b)$ *is computed and given to* $\mathcal{A}$. *We stress that encryption here uses uniform coins that are not known to* $\mathcal{A}$.
4. $\mathcal{A}$ *may continue to interact with its oracle as before. Eventually, it outputs a bit* b'*; the experiment evaluates to 1 if* $b' = b$.

We denote the above experiment by $\mathsf{CRA}_{\mathcal{A},\mathsf{SKE}}(k)$*, and define the advantage of* $\mathcal{A}$ *in the experiment as* $\left|\Pr[\mathsf{CRA}_{\mathcal{A},\mathsf{SKE}}(k) = 1] - \frac{1}{2}\right|$.

The stronger notion of CCRA-security guarantees that, given a ciphertext, no polynomially-bounded adversary can recover any partial information about the plaintext, even if it has access to both an encryption and a decryption oracle and complete control over the encryption oracle's source of randomness. This is defined in the natural way, and we denote the experiment in this case by $\mathsf{CCRA}_{\mathcal{A},\mathsf{SKE}}(k)$. We remark that since Dec is deterministic, there is no analogue of the adversary's being able to "control the randomness" used during decryption.

3.1 Comparison to Previous Definitions

In this section we compare our new definitions to previous security notions for private-key encryption, including CPA- and CCA-security, and the more closely related notions of nonce-based CPA and CCA-security from [15].

Theorem 1. *CRA-security is strictly stronger than CPA-security.*

Proof. It is easy to see that CRA-security implies CPA-security. We show that the converse is not true. Let F be a pseudorandom function, and consider the standard CPA-secure private-key encryption scheme with encryption given by $\mathsf{Enc}_K(m; r) = \langle r, F_K(r) \oplus m \rangle$. We claim that this scheme is not CRA-secure. To see this, note that an adversary given a challenge ciphertext $\langle r, c \rangle$ can submit $\langle 0^k, r \rangle$ to its oracle and will receive in return the ciphertext

$$\mathsf{Enc}_K(0^k; r) = \langle r, F_K(r) \rangle.$$

It is then trivial for the adversary to determine the message that was encrypted.

Theorem 2. *CCRA-security is strictly stronger than CCA-security.*

A proof is very similar to the proof of the previous theorem, and is omitted.

Nonce-based encryption [15] is a formalization of private-key encryption where the encryption algorithm is a deterministic function of a message and a nonce, and the user (or, more generally, the program calling the encryption algorithm as a sub-routine) is responsible for providing the nonce. E.g., in the case of CBC-mode encryption the IV would be an additional input provided to the encryption algorithm as opposed to being generated "internally". This formulation gives more flexibility with respect to how the nonce is chosen: by assuming the nonce is chosen uniformly each time the encryption algorithm is called, the standard notion of probabilistic encryption is recovered, but another option is to assume only that nonces never repeat (but are not necessarily random).

Rogaway [15] considers definitions of security for nonce-based schemes in which the adversary is given some control over the nonce that is used to encrypt *at all times*, i.e., both when interacting with an encryption oracle as well as when the challenge ciphertext is computed. (A definition of nonce-based CPA-security is given in Appendix A.) Intuitively, these definitions are incomparable to our own because:

- On one hand, we assume the adversary has *no control* over the randomness used to encrypt the challenge ciphertext, whereas Rogaway allows the adversary to have some control over the randomness even in this case.
- On the other hand, we give the adversary *full control* over the randomness used by the encryption oracle, whereas Rogaway restricts the adversary to never using the same nonce twice.

We formally prove that the notions are incomparable now.

Theorem 3. *Nonce-based CPA-security and CRA-security are incomparable.*

The theorem is a consequence of the following two lemmas. In proving them, we rely on the definition of nonce-based security given in Appendix A; the definition is weaker than that given in [15], but the difference is inessential and unimportant for the present discussion.

Lemma 1. *Assuming the existence of one-way functions, there is a private-key encryption scheme that is nonce-based CPA-secure but that is not CRA-secure.*

Proof. We take the standard encryption scheme used in the proof of Theorem 1. Recall, F is a pseudorandom function, which may be constructed from any one-way function. Encryption is given by $\mathsf{Enc}_K(m; r) = \langle r, F_K(r) \oplus m \rangle$, where we treat r as a nonce, and decryption is given by $\mathsf{Dec}_K(\langle r, c \rangle) = F_K(r) \oplus c$.

We have already shown in the proof of Theorem 1 that this scheme is not CRA-secure. On the other hand, it is not hard to see that it is nonce-based CPA-secure: since the adversary is not allowed to use the same nonce twice, it holds in particular that the nonce r used when encrypting the challenge ciphertext is distinct from any nonce used in answering any queries to the encryption oracle. It then follows easily from the pseudorandomness of F that the scheme is nonce-based CPA-secure.

Lemma 2. *Assuming the existence of a private-key encryption scheme that is CRA-secure, there is a CRA-secure scheme that is not nonce-based CPA-secure.*

Proof. Let $\mathsf{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ be a CRA-secure private-key encryption scheme. Assume without loss of generality that, on security parameter k, the encryption algorithm uses k bits of randomness. (We will again treat the random coins used by Enc as a nonce.) Define a modified encryption scheme $\mathsf{SKE}' = (\mathsf{Enc}', \mathsf{Dec})$ (decryption remains unchanged) as follows:

$$\mathsf{Enc}'_K(m; r\|b) = \mathsf{Enc}_K(m; r),$$

where b is a bit and $r \in \{0,1\}^k$. It is easy to see that SKE' is not nonce-based CPA secure: an adversary can simply request to have the challenge ciphertext encrypted using the nonce $r\|0$ and then query its encryption oracle using the (distinct) nonce $r\|1$. (Further details omitted.) It is similarly easy to see that SKE' remains CRA-secure: oracle queries with respect to the modified scheme SKE' are no more powerful than oracle queries with respect to the original scheme SKE; when the challenge ciphertext is encrypted, it will be encrypted using algorithm Enc with uniform random coins.

In the sections that follow, we will show constructions of CRA-secure encryption schemes that may be based on any one-way function.

Using ideas as above, we can similarly show that the notions of CCRA-security and nonce-based CCA-security are incomparable.

4 Achieving CRA-Security

In this section we propose two CRA-secure private-key encryption schemes based on PRPs; our first construction handles fixed-length messages only, while our second construction handles messages of variable length. We then show a general transformation from any CPA-secure scheme to a CRA-secure one.

4.1 A Fixed-Length CRA-Secure Construction

Our first construction is a modification of the standard CPA-secure encryption scheme that we have seen before in the proof of Theorem 1. Let P be a pseudorandom *permutation* on k-bit strings, and let F be a pseudorandom function mapping k-bit inputs to k-bit outputs. Our scheme is defined as follows:

$\mathsf{Gen}(1^k)$: Choose $K_1, K_2 \leftarrow \{0,1\}^k$ and output $K = \langle K_1, K_2 \rangle$.

$\mathsf{Enc}_K(m; r)$: Compute $c_2 = F_{K_2}(r) \oplus m$, then output the ciphertext $\langle P_{K_1}(r), c_2 \rangle$.

$\mathsf{Dec}_K(\langle c_1, c_2 \rangle)$: Compute $r = P^{-1}_{K_1}(c_1)$. Then output $m := F_{K_2}(r) \oplus c_2$ as the message.

Theorem 4. *If P is a pseudorandom permutation and F is a pseudorandom function, then the scheme described above is CRA-secure.*

Proof. Consider the (imaginary) scheme $\widetilde{\mathsf{SKE}} = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ in which $\widetilde{\mathsf{Gen}}$ samples $p \leftarrow \mathsf{Perm}[k,k]$ and $f \leftarrow \mathsf{Func}[k,k]$ uniformly at random, and $\widetilde{\mathsf{Enc}}(m; r)$ outputs the ciphertext $\langle p(r), f(r) \oplus m \rangle$. We analyze the security of this scheme in an information-theoretic sense; CRA-security of the scheme described above (for polynomial-time adversaries) then follows easily.

Let $\mathcal{A}$ be an adversary making at most $q(k)$ queries to its oracle in experiment $\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k)$. Let r be the randomness used to generate the challenge ciphertext in this experiment, and let query be the event that one of $\mathcal{A}$'s oracle queries uses randomness r. Then:

$$\begin{aligned}\Pr\left[\,\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k) = 1\,\right] &= \Pr\left[\, b' = b \,\right] \\ &= \Pr\left[\, b' = b \wedge \mathsf{query} \,\right] + \Pr\left[\, b' = b \wedge \overline{\mathsf{query}} \,\right]\end{aligned}$$

from which it follows that

$$\left| \Pr\left[\,\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k) = 1\,\right] - \Pr\left[\, b' = b \mid \overline{\mathsf{query}} \,\right] \right| \leq \Pr\left[\,\mathsf{query}\,\right].$$

The following two claims complete the proof of the theorem.

Claim. $\Pr\left[\,\mathsf{query}\,\right] \leq q(k)/2^k$.

This follows from the fact that, after observing the challenge ciphertext and making q' queries to its oracle that do not use randomness r, all $\mathcal{A}$ knows is that r is not equal to any of the random coins used in its queries thus far.

Claim. $\Pr[b' = b \mid \overline{\mathsf{query}}] = 1/2$.

Let m_0, m_1 denote the messages output by $\mathcal{A}$, and let $\langle c_1, c_2 \rangle$ be the challenge ciphertext. If query does not occur, then $f(r)$ is equally likely to be $c_2 \oplus m_0$ or $c_2 \oplus m_1$ (just as in the one-time pad), and thus $\mathcal{A}$ can do no better than guess.

4.2 A CRA-Secure Construction for Variable-Length Messages

Our second construction applies a similar modification as in the previous section, but to CTR-mode encryption. Let P be a pseudorandom permutation and F a pseudorandom function, as in the previous section.

$\mathsf{Gen}(1^k)$: Choose $K_1, K_2 \leftarrow \{0,1\}^k$ and output $K = \langle K_1, K_2 \rangle$.

$\mathsf{Enc}_K(m; r)$: Parse m into ℓ blocks $m = \langle m_1, \ldots, m_\ell \rangle$, each of length k. For $1 \leq i \leq \ell$, compute $c_i = F_{K_2}(r+i) \oplus m_i$. (Here, we are viewing r as a k-bit integer and addition is done modulo 2^k). Output the ciphertext $c = \langle P_{K_1}(r), c_1, \ldots, c_\ell \rangle$.

$\mathsf{Dec}_K(\langle c_0, c_1, \ldots, c_\ell \rangle)$: Compute $r = P^{-1}_{K_1}(c_0)$. For $1 \leq i \leq \ell$, compute $m_i = F_{K_2}(r+i) \oplus c_i$. Output $m = \langle m_1, \ldots, m_\ell \rangle$.

Theorem 5. *If P is a pseudorandom permutation and F is a pseudorandom function then the scheme described above is CRA-secure.*

Proof. Consider the private-key encryption scheme $(\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ such that $\widetilde{\mathsf{Gen}}$ samples $p \leftarrow \mathsf{Perm}[k,k]$ and $f \leftarrow \mathsf{Func}[k,k]$ uniformly at random, and then $\widetilde{\mathsf{Enc}}$ is define in the natural way based on the scheme described above. We analyze the security of this scheme in an information-theoretic sense; security of the scheme described above (for polynomial-time adversaries) then follows easily.

Let $\mathcal{A}$ be an adversary making at most $q = q(k)$ queries to its oracle in experiment $\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k)$, where the messages in these queries have block-length at most $\ell = q(k)$. We also let $q(k)$ be a bound on the block-length of the messages m_0, m_1 output by $\mathcal{A}$. Let r be the randomness used to generate the challenge ciphertext in this experiment, and let query be the event that one of $\mathcal{A}$'s oracle queries uses randomness $r' \in \{r-q+1, \ldots, r+q-1\}$. Then:

$$\begin{aligned}\Pr\left[\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k) = 1\right] &= \Pr\left[b' = b\right] \\ &= \Pr\left[b' = b \wedge \mathsf{query}\right] + \Pr\left[b' = b \wedge \overline{\mathsf{query}}\right]\end{aligned}$$

from which it follows that

$$\left|\Pr\left[\mathsf{CRA}_{\mathcal{A},\widetilde{\mathsf{SKE}}}(k) = 1\right] - \Pr\left[b' = b \mid \overline{\mathsf{query}}\right]\right| \leq \Pr\left[\mathsf{query}\right].$$

The following two claims complete the proof of the theorem.

Claim. $\Pr\left[\mathsf{query}\right] \leq \mathcal{O}(q(k)^2/2^k)$.

Intuitively, the value r used to encrypt the challenge ciphertext is "hidden" from $\mathcal{A}$. Thus, assuming query has not yet occurred, a query made by $\mathcal{A}$ to its encryption oracle can cause query to occur with probability at most

$$\frac{(r+q-1)-(r-q+1)+1}{2^k} = \frac{2q-1}{2^k}.$$

Applying a union bound to the q queries of $\mathcal{A}$ gives the stated result.

Claim. $\Pr[b' = b \mid \overline{\mathsf{query}}] = 1/2$.

This follows by analogy to the one-time pad; conditioned on query not occurring, $F_{K_2}(r+1), \ldots, F_{K_2}(r+q)$ are uniformly distributed from $\mathcal{A}$'s point of view.

4.3 A CPA-to-CRA Transformation

Finally, we present a transformation that turns any CPA-secure private-key encryption scheme into a CRA-secure scheme. The transformation assumes the existence of pseudorandom functions for arbitrary-length inputs; these may be constructed based on any one-way function, whose existence is implied by the existence of a CPA-secure encryption scheme.

Let $\mathsf{SKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a CPA-secure encryption scheme in which encryption uses k random coins (this is not essential, but makes the analysis easier), and let F be a pseudorandom function. Define SKE as follows:

$\mathsf{Gen}(1^k)$: Compute $K_1 \leftarrow \mathsf{Gen}'(1^k)$, and then choose $K_2 \leftarrow \{0,1\}^k$. Output the key $K = \langle K_1, K_2 \rangle$.

$\mathsf{Enc}_K(m; r)$, where $r \in \{0,1\}^k$: Compute "random coins" $r' = F_{K_2}(m \| r)$. Then output the ciphertext $c' = \mathsf{Enc}'_{K_1}(m; r')$.

$\mathsf{Dec}_K(c')$: Output $m = \mathsf{Dec}'_{K_1}(c')$.

Theorem 6. *If SKE' is a CPA-secure private-key encryption scheme and F is a pseudorandom function, then the scheme described above is CRA-secure.*

Proof. Given an adversary $\mathcal{A}$ attacking the constructed scheme (in the sense of CRA-security), we construct an adversary $\mathcal{A}'$ attacking SKE' (in the sense of CPA-security). Our adversary $\mathcal{A}'$ is defined as follows:

1. Run $\mathcal{A}$. When $\mathcal{A}$ makes oracle query $\langle m, r \rangle$ to its oracle, $\mathcal{A}'$ queries m to its own (standard) encryption oracle and returns the result to $\mathcal{A}$. We assume without loss of generality that $\mathcal{A}$ never makes the same oracle query twice.
2. When $\mathcal{A}$ outputs two messages m_0, m_1, these same messages are output by $\mathcal{A}'$. The challenge ciphertext given to $\mathcal{A}'$ is forwarded to $\mathcal{A}$.
3. Oracle queries of $\mathcal{A}$ are handled exactly as before.
4. When $\mathcal{A}$ outputs a bit b', the same bit is output by $\mathcal{A}'$.

It is not hard to see that the view of $\mathcal{A}$ in the above is computationally indistinguishable from its view when attacking the constructed scheme. Thus, the advantage of $\mathcal{A}'$ is negligibly close to the advantage of $\mathcal{A}$. Since SKE' is CPA-secure by assumption, we conclude that the advantage of $\mathcal{A}$ in attacking the constructed scheme (in the sense of CRA-security) is negligible.

5 Achieving CCRA-Security

We now show that the standard "encrypt-then-MAC" transformation [3] from CPA-secure schemes to CCA-secure ones works in our setting also. Let $(\mathsf{Mac}, \mathsf{Vrfy})$ be a secure message authentication code, and let $\mathsf{SKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a CRA-secure encryption scheme. Define SKE as follows:

$\mathsf{Gen}(1^k)$: Compute $K_1 \leftarrow \mathsf{Gen}'(1^k)$, and then choose $K_2 \leftarrow \{0,1\}^k$. Output the key $K = \langle K_1, K_2 \rangle$.

$\mathsf{Enc}_K(m;r)$: Compute $c' = \mathsf{Enc}'_{K_1}(m;r)$ and $t = \mathsf{Mac}_{K_2}(c')$. Output the ciphertext $\langle c', t\rangle$.

$\mathsf{Dec}_K(\langle c', t\rangle)$: If $\mathsf{Vrfy}_{K_2}(c', t) = 1$ then output $\mathsf{Dec}'_{K_1}(c')$. Otherwise output $\perp$.

Theorem 7. *If* SKE' *is CRA-secure and* $(\mathsf{Mac}, \mathsf{Vrfy})$ *is a secure MAC with unique tags, then the scheme described above is CCRA-secure.*

Proof. Let $\mathcal{A}$ be an adversary attacking SKE in the sense of CCRA-security. Let query be the event that $\mathcal{A}$ submits a decryption query $\langle c, t\rangle$ to its decryption oracle such that $\mathsf{Dec}_K(c,t) \neq \perp$ and $\langle c, t\rangle$ was not the result of a previous encryption query. Clearly,

$$\Pr[\, b' = b\,] = \Pr[\, b' = b \wedge \mathsf{query}\,] + \Pr[\, b' = b \wedge \overline{\mathsf{query}}\,]$$

from which it follows that

$$|\Pr[\, b' = b\,] - \Pr[b' = b \mid \overline{\mathsf{query}}]| \leq \Pr[\,\mathsf{query}\,].$$

The following claims complete the proof.

Claim. For all PPT adversaries $\mathcal{A}$ it holds that $\Pr[\,\mathsf{query}\,]$ is negligible.

We show that if there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\,\mathsf{query}\,]$ is not negligible, then there exists a PPT adversary $\mathcal{B}$ that can win the existential unforgeability experiment against MAC with non-negligible probability.

Consider the adversary $\mathcal{B}$ that, given 1^k and oracle access to $\mathsf{Mac}_K(\cdot)$ and $\mathsf{Vrfy}_K(\cdot,\cdot)$, begins by generating an encryption key $K_1 \leftarrow \mathsf{Gen}'(1^k)$ and runs $\mathcal{A}(1^k)$ as follows,

> Given an encryption query $e = \langle m, r\rangle$, adversary $\mathcal{B}$ computes $c \leftarrow \mathsf{Enc}'_{K_1}(m;r)$ and queries its own Mac oracle with c, receiving t. Finally, it returns the ciphertext $\langle c, t\rangle$ to $\mathcal{A}$.
>
> Given a decryption query $d = \langle c, t\rangle$, adversary $\mathcal{B}$ queries its Vrfy oracle with c and t. If the oracle returns 1 then it computes and returns $m \leftarrow \mathsf{Dec}'_{K_1}(c)$ to $\mathcal{A}$; otherwise it returns $\perp$. Adversary $\mathcal{B}$ stores all of $\mathcal{A}$'s decryption queries. After polynomially-many queries, $\mathcal{A}$ outputs m_0, m_1.

$\mathcal{B}$ samples $b \leftarrow \{0,1\}$, computes $c^* \leftarrow \mathsf{Enc}'_{K_1}(m_b)$, and queries its oracle to receive $t^* \leftarrow \mathsf{Mac}_K(c^*)$. It then runs $\mathcal{A}$ with the challenge ciphertext $\langle c^*, t^*\rangle$, and answers its queries as before. After polynomially many queries, $\mathcal{A}$ outputs a bit b' and halts. Let $q(k)$ be the number of decryption queries made by $\mathcal{A}$. If query has occurred by the end of the game (note that $\mathcal{B}$ can determine if this happens), then $\mathcal{B}$ outputs the appropriate query for which this first occurred.

Notice that $\mathcal{B}$ succeeds if query occurs. Since $\mathcal{A}$'s view is identical to its view when attacking SKE, the claim follows.

Claim. For all PPT adversaries $\mathcal{A}$, it holds that $\Pr[b' = b \mid \overline{\mathsf{query}}] \leq 1/2 + \mathsf{negl}(k)$.

We show that if there exists a PPT adversary $\mathcal{A}$ such that

$$\Pr[b' = b \mid \overline{\mathsf{query}}] \geq 1/2 + 1/\mathsf{poly}(k),$$

then there exists a PPT adversary $\mathcal{B}$ that can succeed in attacking SKE' (in the sense of CRA-security) with non-negligible probability.

Consider $\mathcal{B}$ that, given 1^k, begins by choosing $K_2 \in \{0,1\}^k$ and runs $\mathcal{A}_1(1^k)$ as follows.

> Given an encryption query $\langle m, r \rangle$, adversary $\mathcal{B}$ queries its oracle with $\langle m, r \rangle$ to obtain a ciphertext c. It then computes $t \leftarrow \mathsf{Mac}_{K_2}(c)$, and returns the ciphertext $\langle c, t \rangle$ to $\mathcal{A}$. It stores the tuple $\langle c, t, m \rangle$ in a table T.
>
> Given a decryption query $d = \langle c, t \rangle$, adversary $\mathcal{B}$ looks up the pair $\langle c, t \rangle$ in its table and returns the corresponding plaintext m. If the pair $\langle c, t \rangle$ is not in T, then it returns $\perp$. After polynomially many queries, $\mathcal{A}$ outputs messages m_0, m_1 which $\mathcal{B}$ also outputs.

Given a challenge ciphertext c^*, adversary $\mathcal{B}$ computes $t^* \leftarrow \mathsf{Mac}_{K_2}(c^*)$ and gives the challenge ciphertext $\langle c^*, t^* \rangle$ to $\mathcal{A}$, answering its oracle queries as before. Eventually, $\mathcal{A}$ outputs a bit b' which $\mathcal{B}$ outputs as well.

It remains to analyze $\mathcal{B}$'s success probability. First, notice that $\mathcal{B}$ can answer $\mathcal{A}$'s encryption queries perfectly. Furthermore, if query does not occur, then the only valid decryption queries $\mathcal{A}$ makes are for ciphertexts that were the result of previous encryption queries. In this case (i.e., conditioned on $\overline{\mathsf{query}}$), $\mathcal{B}$ will also correctly answer all of $\mathcal{A}$'s decryption queries (using its table). It follows then that conditioned on $\overline{\mathsf{query}}$, the view of $\mathcal{A}$ is identical to its view when attacking SKE. The claim follows.

References

1. B. Barak and S. Halevi. A model and architecture for pseudorandom generation and applications to /dev/random. *ACM Conf. on Computer and Communications Security 2005.*
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. *38th Annual Symposium on Foundations of Computer Science (FOCS) 1997.*
3. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Asiacrypt 2000.*
4. M. Bellare and B. Yee. Forward-security in private-key cryptography. *RSA — Cryptographers' Track 2003.*
5. C. Bosley and Y. Dodis. Does privacy require true randomness? *Theory of Cryptography Conference 2007.*
6. Y. Dodis, S.J. Ong, M. Prabhakaran, and A. Sahai. On the (im)possibility of cryptography with imperfect randomness. *45th Annual Symposium on Foundations of Computer Science (FOCS) 2004.*
7. Y. Dodis and J. Spencer. On the (non)universality of the one-time pad. *43rd Annual Symposium on Foundations of Computer Science (FOCS) 2002.*

8. K. Fu, S. Kamara, and T. Kohno. Key regression: Enabling efficient key distribution for secure distributed storage. *NDSS 2006*.
9. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM* 33(4):792–807, 1984.
10. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences* 28(2):270–299, 1984.
11. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC Press, 2007.
12. J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *J. Cryptology* 19(1):67–96, 2006.
13. J. McInnes and B. Pinkas. On the impossibility of private-key cryptography with weakly random keys. *Advances in Cryptology — Crypto '90*.
14. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – Crypto '91*.
15. P. Rogaway. Nonce-based symmetric encryption. *Fast Software Encryption (FSE)* 2004.

A Nonce-based Private-key Encryption

We offer a definition in the spirit of nonce-based security [15], but we do not require that ciphertexts be indistinguishable from random strings. (This extra requirement is irrelevant as far as the results of the present paper are concerned.)

Definition 2 (Nonce-based CPA-security). *Let* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a private-key encryption scheme where encryption uses* k *random coins and we treat these coins as a nonce.* SKE *is* nonce-based CPA-secure *if the advantage of any polynomial-time adversary* $\mathcal{A}$ *in the following experiment is negligible (in* k*):*

1. *First, a key* $K \leftarrow \mathsf{Gen}(1^k)$ *is generated. Set* $\mathsf{Nonces} = \{0,1\}^k$.
2. $\mathcal{A}$ *is allowed to adaptively submit multiple queries of the form* $\langle m, r\rangle$*, subject always to the restriction that* $r \in \mathsf{Nonces}$. *In response to each such a query,* $\mathcal{A}$ *is given* $c = \mathsf{Enc}_K(m; r)$ *and* r *is removed from* Nonces.
3. $\mathcal{A}$ *outputs two equal-length messages* m_0, m_1 *and a nonce* $r \in \mathsf{Nonces}$. *A bit* b *is chosen at random, and a "challenge ciphertext"* $c = \mathsf{Enc}_K(m_b; r)$ *is computed and given to* $\mathcal{A}$*. Also,* r *is removed from* Nonces.
4. $\mathcal{A}$ *may continue to interact with its oracle as before. Eventually, it outputs a bit* b'*; the experiment evaluates to 1 if* $b' = b$.

We denote the above experiment by $\mathsf{NB\text{-}CPA}_{\mathcal{A},\mathsf{SKE}}(k)$*, and define the advantage of* $\mathcal{A}$ *in the experiment as* $\left|\Pr[\mathsf{NB\text{-}CPA}_{\mathcal{A},\mathsf{SKE}}(k) = 1] - \frac{1}{2}\right|$.