Encryption-Scheme Security in the Presence of Key-Dependent Messages

J. Black [*]   P. Rogaway [†]   T. Shrimpton [‡]

May 2, 2002

Abstract

Encryption that is only semantically secure should not be used on messages that depend on the underlying secret key; all bets are off when, for example, one encrypts using a shared key K the value K. Here we introduce a new notion of security, KDM security, appropriate for key-dependent messages. The notion makes sense in both the public-key and shared-key settings. For the latter we show that KDM security is easily achievable within the random-oracle model. By developing and achieving stronger notions of encryption-scheme security it is hoped that protocols which are proven secure under "formal" models of security can, in time, be safely realized by generically instantiating their primitives.

Key words: Definitions, "formal" cryptography, semantic security, symmetric encryption.

1 Introduction

Background. This paper defines and begins to investigate a new notion for encryption-scheme security: KDM security. KDM stands for *key-dependent messages*. This is a new attack model, one that allows requested plaintexts to depend on the underlying decryption key. Our definitions are strictly stronger than indistinguishability under chosen-plaintext attacks (IND-CPA) [3, 11], and the idea is orthogonal to strengthenings in the direction of non-malleability [8], chosen-ciphertext security [18], and anonymity [10]. We show that, in the symmetric-key setting, KDM security is achievable, and by simple means, within the random-oracle model [5].

Why, at this time, do we put forth a new definition for encryption-scheme security? Our goal has been to nudge the computational treatment of encryption in a direction that makes it closer to simplistic intuition about what an *idealized* encryption scheme does. We have done this to facilitate smoothly linking up "real" encryption with "formal" encryption. Let us elaborate.

The formal views vs. the computational view. In designing high-level protocols many users of cryptography prefer to take an "abstract" or "formal" view of what they're given. They don't view encryption as a transformation on strings at all. Instead, typically, they see encryption as a formal

[*] Department of Computer Science UCB 430, University of Colorado, Boulder, Colorado, 80309, USA. E-mail: jrblack@cs.colorado.edu WWW: www.cs.colorado.edu/~jrblack/

[†] Department of Computer Science, University of California, Davis, California, 95616, USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, 50200 Thailand. E-mail: rogaway@cs.ucdavis.edu WWW: www.cs.ucdavis.edu/~rogaway/

[‡] Department of Electrical and Computer Engineering, University of California, Davis, California, 95616, USA. E-mail: teshrim@ucdavis.edu WWW: www.ece.ucdavis.edu/~teshrim/

symbol, the properties of which are modeled (not defined) by the manner in which this formal symbol may be manipulated. There are many such formal views towards cryptography; see, for example, [1, 6, 9, 15].

Quite recently, some work has emerged which starts to bridge the computational view and the formal one. Lincoln, Mitchell, Mitchell and Scedrov [13] develop a formal model that blends in computational aspects. Pfitzmann, Schunter and Waidner [16], and then Pfitzmann and Waidner [17], consider security, in the computational sense, of general reactive systems, which are modeled formally. Abadi and Rogaway [2] give a formal treatment of encryption and then prove that formally-equivalent expressions give rise to computationally indistinguishable ensembles, as long as the expressions contain no "encryption cycles" (a generalization of encrypting ones own key) and as long as the encryption scheme has the right (computational) properties.

Consider the last of these works. To the computational cryptographer the technical restriction just mentioned might be understood as a warning: formal cryptography easily permits its users to ignore an essential restriction (the necessity of avoiding encryption cycles). But there is another viewpoint—and the one that motivates us. Maybe the formal view of cryptography is already doing a nice job to capture strong intuition and it is incumbent on computational cryptography to find definitions and schemes which support formal cryptography. From this viewpoint, there is nothing obviously and intuitively "wrong" with encrypting ones decryption key, and so one had better find definitions and realizations that make this allowable. In this view, a mandate to avoid encryption cycles might seem to be an artifact of inessential definitional choices.

Contributions. At a technical level, the current paper does several things:

- We present a new, and very strong, definition of security—a definition that allows the adversary indirect access to hidden keys. One can think of this as a new attack model, KDM, and we leave the goal, IND, alone. We define KDM security for both the symmetric and asymmetric settings.
- We separate KDM security from other notions of security. We show that KDM security implies but is not implied by CPA security. We show that some conventional means to achieve CPA security, in particular those that employ stateful encryption (e.g., counter-mode with a fixed initial counter), do *not* achieve KDM security.
- At the same time, we show that KDM security is easy to achieve within the random-oracle model [5]: we give a simple scheme that is KDM secure.

Most of this paper focuses on the symmetric model for KDM security, but there would seem to be no essential differences between the symmetric and asymmetric settings. We define KDM security in the public-key trust model and discuss achieving it.

Concurrent work. A recent paper by Camenisch and Lysyanskaya defines a notion of *circular security* for the asymmetric setting [7, p. 17]. Their notion would seem to be strictly weaker than our (asymmetric-model) notion of KDM security. Camenisch and Lysyanskaya were interested in anonymous credential systems and saw encryption schemes satisfying circular security as a tool. Their work highlights the possibility that KDM-secure encryption schemes may be a useful primitive in designing higher-level protocols. Our own work and that of [7] were independent and cross-reference each other.

An interesting open problem. This paper proposes a new privacy notion and then answers some simple questions about it. We leave unanswered the question as to whether KDM security, either for the symmetric or the asymmetric trust model, can be provably achieved in the *standard* model of computation (no random oracle) under a standard complexity assumption.

2 The Peril of Encrypting Ones Key

Goldwasser and Micali were the first to recognize the potential problems surrounding encryption cycles [11, 14]. Let us assume the symmetric setting. Given an encryption scheme that is secure in any of a number of standard senses (say IND-CPA), one can trivially modify it to construct a similarly secure encryption scheme (still IND-CPA secure, say) that is rendered completely *insecure* were the adversary somehow handed an encryption of the underlying key. (For example, modify $\mathcal{E}_K(M,R)$ to $0 \parallel \mathcal{E}_K(M,R)$ if $M \neq K$, and $1 \parallel K$ otherwise.) We call an encryption of the decryption key *an encryption cycle of length one.*

More generally, an encryption cycle involves the encryption of some function of the key. For example, the encryption the bitwise complement of K and the encryption of $M \parallel K$ (for some plaintext M) are both cycles. Some cycles pose no threat to security; other cycles do. There is no known characterization of the cases that cause problems for the standard definitions.

Informally, an encryption scheme will be called "KDM secure" if it is secure (in the sense of indistinguishability) despite an adversary's ability to obtain the encryption under key K_i of some function $g(\mathbf{K})$ of the vector $\mathbf{K}$ of underlying secret keys. (We have to think of a vector of keys because the assumption that there is just a single key would seem to entail a loss of generality.) Through the function g the adversary has indirect access to the keys, but this information is only surfaced after encryption. As examples, function g might request a particular key K_i, or it might xor various keys.

We note that one can also give a notion for key-dependent chosen-ciphertext attack, KDC, where a function of the keys can be surfaced through decryption as well as through encryption. We do not pursue this at this time.

Well-designed protocols rarely employ encryption cycles (a fact that certainly does limit the applicability of this paper!). Some encrypting backup systems may encrypt their own key (blithely encrypting the entire system image, which may include the encryption key). Some key-distribution protocols allow the adversary to effectively create an encryption cycle: having recorded a message M which includes the encryption of a session key K, the adversary may now be able to replay this message M (perhaps pretending it is a nonce) in a context in which the principal will now encrypt it under K. This gives a cycle, though not a troublesome one. The recently proposed credential system of [7] uses encryption cycles in the asymmetric setting.

We comment that there are multiple properties (beyond the ability to safely encrypt ones key) implicit in formal models and not guaranteed by IND-CPA security. These properties include protecting the integrity of messages, concealing the length of a plaintext, and concealing which pairs of ciphertexts were encrypted under the same key. But the most bothersome and most ignored problem seems to be key-dependent plaintexts.

3 Background Notions

NOTATION. Let $\{0,1\}^*$ be the space of (finite) binary strings, let $\{0,1\}^\infty$ be the space of infinite binary strings, and let S^∞ be the set of tuples $(K_1, K_2, \ldots)$ where each $K_i \in S$. If $Pad \in \{0,1\}^\infty$ and $M \in \{0,1\}^*$ then $Pad \oplus M$ and $M \oplus Pad$ are the xor of M and the first $|M|$ bits of Pad. Let $x_1, x_2, \ldots \xleftarrow{\$} \Omega$ mean that $x_1, x_2, \ldots$ are assigned values independently from the distribution Ω. (Use the uniform distribution when Ω is a finite set.) If A is a probabilistic algorithm then $a \xleftarrow{\$} A(x_1, x_2, \ldots)$ denotes running A on the given inputs and letting a be the outcome. An *adversary* is a probabilistic algorithm that may access one or more oracles. The running time of A is the actual running time of A plus its description size. If A makes oracle queries that specify functions $g_1, g_2, \ldots$ then the running time of A also includes the (worst case) time to compute each queried function g_i.

Symmetric encryption. We recall the syntax for a symmetric-encryption scheme as given (in a slightly modified form) by [3]. Let $\mathsf{Plaintext}$ and $\mathsf{Ciphertext}$ be nonempty sets of strings. We assume that $\mathsf{Plaintext}$ has the property that if $M \in \mathsf{Plaintext}$ then $M' \in \mathsf{Plaintext}$ for every M' of the same length as M. Let $\mathsf{Key} = \{0,1\}^k$ for some $k \geq 1$ and let $\mathsf{Coins} = \{0,1\}^r$ for some $r \geq 1$. Let $\mathsf{String} = \{0,1\}^*$ be the set of all (finite) strings. Then a *symmetric encryption scheme* is a triple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where: $\mathcal{K}$: $\mathsf{Coins} \to \mathsf{Key}$ and $\mathcal{E}$: $\mathsf{Key} \times \mathsf{String} \times \mathsf{Coins} \to \mathsf{Ciphertext} \cup \{*\}$ and $\mathcal{D}$: $\mathsf{Key} \times \mathsf{String} \to \mathsf{Plaintext} \cup \{*\}$. Algorithm $\mathcal{K}$ is called the *key-generation* algorithm, $\mathcal{E}$ is called the *encryption* algorithm, and $\mathcal{D}$ is the *decryption* algorithm. We call k (the length of any string K output by $\mathcal{K}$) the *key length* of Π and we call r (the number of random bits used by $\mathcal{E}$) the *coin length* of Π. We usually write the first argument to $\mathcal{E}$ and $\mathcal{D}$ as a subscript. We call $\mathcal{E}_K(M, R)$ the encryption of the *plaintext* M under key K and coins R. Usually we omit specific reference to the final argument to $\mathcal{E}$, the random string R, thinking of $\mathcal{E}_K$ as a probabilistic algorithm. Likewise, we usually omit mention of the argument to $\mathcal{K}$, thinking of $\mathcal{K}$ as a probabilistic algorithm, or else the induced probability space. We require that for all $K \in \mathsf{Key}$ and $R \in \mathsf{Coins}$, if $M \notin \mathsf{Plaintext}$ then $\mathcal{E}_K(M, R) = *$ and if $M \in \mathsf{Plaintext}$ then $\mathcal{E}_K(M, R) \in \mathsf{Ciphertext}$ and, what's more, $\mathcal{D}_K(\mathcal{E}_K(M, R)) = M$.

The syntax above is for encryption schemes in the *standard model*. To define encryption schemes in the *random-oracle model* we allow that $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ are given an oracle $H \in \Omega$ where Ω is the set of all functions from $\{0,1\}^*$ to $\{0,1\}^\infty$.

IND-CPA security. We review the notion of indistinguishability under a chosen-plaintext attack [3, 11]. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. For $K \in \mathsf{Key}$ let

Real_K be the oracle that on input M returns $C \xleftarrow{\$} \mathcal{E}_K(M)$ and let

Fake_K be the oracle that on input M returns $C \xleftarrow{\$} \mathcal{E}_K(0^{|M|})$

The **IND-CPA advantage** of A is defined as

$$\mathsf{Adv}_{\Pi}^{\mathrm{cpa}}(A) \stackrel{\mathrm{def}}{=} \left| \Pr\left[K \xleftarrow{\$} \mathcal{K} :\ A^{\mathsf{Real}_K} = 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} :\ A^{\mathsf{Fake}_K} = 1\right] \right|$$

4 The Notion of Symmetric KDM Security

The notion of IND-KDM security strengthens the IND-CPA notion just given. We again imagine two experiments. Both experiments begin by selecting a vector of keys $\mathbf{K} = (K_1, K_2, \ldots)$ where each key K_i is determined by running the key-generation algorithm $\mathcal{K}$. Independent coins are used in each run. We denote this step by $\mathbf{K} \xleftarrow{\$} \mathcal{K}$ and it is the same as: **for** $i \in \{1, 2, \ldots\}$ **do** $K_i \xleftarrow{\$} \mathcal{K}$. Then:

- **Real**: the adversary is given an oracle $\mathsf{Real}_{\mathbf{K}}$ that takes two inputs: a number $j \in \{1, 2, \ldots\}$ and a function g. The function g is described by a (deterministic) RAM program, encoded according to some fixed conventions. Function g maps $\mathbf{K} \in (\{0,1\}^*)^\infty$ to $g(\mathbf{K}) \in \{0,1\}^*$. When oracle $\mathsf{Real}_{\mathbf{K}}$ is asked (j, g) it probabilistically encrypts $M = g(\mathbf{K})$ under key K_j.
- **Fake**: the adversary is given an oracle $\mathsf{Fake}_{\mathbf{K}}$ that, when similarly asked (j, g), probabilistically encrypts $|g(\mathbf{K})|$ zero-bits under key K_j.

The adversary should output a 1 if it believes it has participated in experiment **Real** and 0 if it believes it has participated in experiment **Fake**. We insist that for any (j, g) queried by an adversary, $|g(\mathbf{K})|$ does not depend on $\mathbf{K}$. To describe this condition we say that g is *fixed-length*. We call g the *plaintext-construction function*.

Definition 4.1 [IND-KDM — standard model — symmetric setting] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. For $\mathbf{K} \in (\{0,1\}^*)^\infty$, let

$\mathsf{Real}_{\mathbf{K}}$ be the oracle that on input (j, g) returns $C \xleftarrow{\$} \mathcal{E}_{K_j}(g(\mathbf{K}))$ and let

$\mathsf{Fake}_{\mathbf{K}}$ be the oracle that on input (j, g) returns $C \xleftarrow{\$} \mathcal{E}_{K_j}(0^{|g(\mathbf{K})|})$

The **IND-KDM advantage** of an adversary A is defined as

$$\mathsf{Adv}_{\Pi}^{\mathrm{kdm}}(A) \ \stackrel{\mathrm{def}}{=}\ \left|\Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K}:\ A^{\mathsf{Real}_{\mathbf{K}}} = 1\right] - \Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K}:\ A^{\mathsf{Fake}_{\mathbf{K}}} = 1\right]\right| \qquad \diamondsuit$$

We often omit the quantifier IND in IND-KDM.

Informally, an encryption scheme Π is KDM secure if for any "reasonable" adversary A we have that $\mathsf{Adv}_{\Pi}^{\mathrm{kdm}}(A)$ is "small." We shall make precise statements along these lines. However, there is also an asymptotic way of defining this notion. For it one parameterizes the encryption scheme by a security parameter k and provides k to the adversary, in unary. Then one demands that for any polynomial time adversary A we have that $\mathsf{Adv}_{\Pi}^{\mathrm{kdm}}(A)$ is negligible (i.e., it vanishes faster than the inverse of any polynomial). One includes in A's running time the running time of the plaintext-construction functions g.

IND-KDM IN THE RO MODEL. KDM security is an extremely strong notion. It is easily seen to imply "standard" notions, like IND-CPA, and it is easily seen not to be implied by IND-CPA (that is, if there exists an IND-CPA secure encryption scheme then there exists an IND-CPA secure encryption scheme that is not IND-KDM secure). The IND-KDM notion is so strong, in fact, that we are unaware of any scheme, regardless of complexity, that demonstrably meets the definition we have given (for the standard model of computation). Here we will instead be giving an efficient scheme in the random-oracle model. Before giving our scheme, we must define KDM security in the random-oracle model.

Recall that Ω is the set of all functions H from $\{0,1\}^*$ to $\{0,1\}^\infty$. This set is provided with a probability measure by saying that a random H from Ω assigns to each $X \in \{0,1\}^*$ an infinite sequence of bits each of which is selected uniformly at random. Though $H(X)$ is defined to be an infinite string, algorithms that use H necessarily make use of only a finite-length prefix. We provide algorithms $\mathcal{K}$, $\mathcal{E}$, $\mathcal{D}$, A, and g with oracle access to $H \in \Omega$. We write a question-mark superscript, as in $g^?$, if we wish to emphasize that an algorithm calls out to an (unspecified) oracle.

For the RO model we must update the notion of g being fixed-length: for any (j, g) asked by an adversary, $|g^H(\mathbf{K})|$ must be independent of both H and $\mathbf{K}$. This means that $|g^H(\mathbf{K})| = |g^{H'}(\mathbf{K}')|$ for any $H, H' \in \Omega$ and $\mathbf{K}, \mathbf{K}' \in (\{0,1\}^*)^\infty$.

Definition 4.2 [IND-KDM — RO-model — symmetric setting] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme for the RO model, and let A be an adversary. For $\mathbf{K} \in (\{0,1\}^*)^\infty$, let

$\mathsf{Real}_{\mathbf{K}}^H$ be the oracle that on input $(j, g^?)$ returns $C \xleftarrow{\$} \mathcal{E}_{K_j}^H(g^H(\mathbf{K}))$ and let

$\mathsf{Fake}_{\mathbf{K}}^H$ be the oracle that on input $(j, g^?)$ returns $C \xleftarrow{\$} \mathcal{E}_{K_j}^H(0^{|g^H(\mathbf{K})|})$

The **KDM-advantage** of A in the random-oracle model is defined as

$$\mathsf{Adv}_{\Pi}^{\mathrm{kdm}}(A) \stackrel{\mathrm{def}}{=} \left|\Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K};\, H \xleftarrow{\$} \Omega:\ A^{H,\,\mathsf{Real}_{\mathbf{K}}^H} = 1\right] - \Pr\left[\mathbf{K} \xleftarrow{\$} \mathcal{K};\, H \xleftarrow{\$} \Omega:\ A^{H,\,\mathsf{Fake}_{\mathbf{K}}^H} = 1\right]\right| \quad \diamondsuit$$

COMMENTS. We emphasize that for KDM security in the RO model, the adversary A can effectively access the random oracle H in two different ways: in a *direct* query the adversary asks X and receives $H(X)$; while in an *indirect* query the adversary asks for the encryption under key K_i of $M = g^H(\mathbf{K})$ and the computation of $g^H(\mathbf{K})$ does *itself* involve some number of calls to H.

The assumption that the plaintext-construction function g is fixed-length is a necessary one; without making this assumption, KDM-secure schemes would not exist. Consider, for example, the function $g_{i,j}$

where $g_{i,j}(\mathbf{K}) = 1$ if the i-th bit of K_j is 1 and $g_{i,j}(\mathbf{K}) = 00$ if the i-th bit of K_j is 0. If encryption function $\mathcal{E}$ reveals the length of the plaintext (which our definition allows) then an adversary can ask for $g_{i,j}(\mathbf{K})$-values as a way to learn any key, bit by bit. Once a key K_j is determined by the adversary it can use this knowledge to distinguish experiments **Real** and **Fake**. Basically, having allowed encryption to be the length-revealing, one must not permit the length of $M = g^H(\mathbf{K})$ to act as "covert channel" for the adversary to learn information about $\mathbf{K}$.

The role of the plaintext-construction function g can be thought of like this: it's as though A and g are in cahoots (after all, A chooses g) and g knows all of the keys. If only g could get some information back to its buddy A, then A would win. But g's output is forced to go through an encryption operation before it is sent back to A. In a KDM-secure scheme, the plaintext-construction function g, poor thing, has no way to get anything useful back to A.

5 Achieving KDM Security in the Symmetric Setting

We now provide a simple and efficient technique to achieve an IND-KDM secure symmetric encryption scheme in the RO model. Several natural methods would seem to accomplish this goal (see the discussion at the end of this section); we give a particularly simple one.

THE **ver** CONSTRUCTION. For $k \geq 1$ an integer, the symmetric encryption scheme that we denote $\mathbf{ver}[k] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ works as follows. Key-generation algorithm $\mathcal{K}$ outputs a random k-bit string K. Encryption algorithm $\mathcal{E}$ is defined by $\mathcal{E}_K(M, R) = R \,\|\, (H(K \,\|\, R) \oplus M)$ where $R \in \{0,1\}^k$ is the $r = k = |K|$ random coins used for encryption. Decryption algorithm $\mathcal{D}$ is defined by $\mathcal{D}_K(\mathcal{C}) = *$ if $|\mathcal{C}| < |K| = k$ and, otherwise, $\mathcal{D}_K(\mathcal{C}) = H(K \,\|\, R) \oplus C$ where $R = \mathcal{C}[1..k]$ is the first $k = |K|$ bits of $\mathcal{C}$ and $C = \mathcal{C}[k+1..]$ is the rest of $\mathcal{C}$.

We now show that scheme **ver** is KDM secure (in the RO model).

Theorem 5.1 [KDM security of the **ver** construction] Let $k \geq 1$ be a number and let A be an adversary that attacks $\mathbf{ver}[k]$. Suppose that A asks at most q oracle queries (encryption queries + direct RO queries + indirect RO queries). Then $\mathsf{Adv}^{\mathrm{kdm}}_{\mathbf{ver}[k]}(A) \leq q^2/2^{k-2}$. $\Diamond$

Proof: The proof is an application of the game-playing paradigm as used, for example, in [12].

We begin by making some without-loss-of-generality assumptions about the behavior of A. Let $N < q$ be the number of different keys referenced by adversary A (that is, referenced as j-values in $(j, g^?)$-queries). First, we assume that the keys referenced by A are keys $j = 1, 2, \ldots, N$. This entails no loss of generality: an adversary that uses a key with a name other than $j \in [1..N]$ can easily be modified not to do this by renaming to j the j-th new key accessed by A. Such a renaming does not impact A's advantage. Second, we assume that A makes no H-queries, either direct or indirect, having length other than $2k$. Such queries are clearly irrelevant to A's mission and any adversary A making such queries is easily modified to an adversary A which does not. Finally, we assume that A never repeats a direct RO query $K \,\|\, R$. The adversary has no need to repeat RO queries since it already knows the answer.

To prove that $\mathsf{Adv}^{\mathrm{kdm}}_{\mathbf{ver}[k]}(A)$ is small we are going to present A not with a Real or Fake oracle to distinguish, but an R or F oracle, instead, where these oracles are defined in Figure 1. We refer to these oracles as specifying different *games* that the adversary plays. The games (oracle behavior) depends on the parameter k but we omit this from the notation.

We first claim that the R-labeled code in Figure 1 provides A an identical view as a Real $\mathbf{ver}[k]$-oracle, while the F-labeled code provides A an identical view as a Fake $\mathbf{ver}[k]$-oracle. This is apparent by

```
INITIALIZATION:
    for j ← 1 to N do K_j ←$ {0,1}^k
    bad ← false;   𝒳 ← ∅
    for all X ∈ {0,1}^{2k} do H(X) ← undefined

ON RO QUERY K ‖ R:
    if K ‖ R ∈ 𝒳 then bad ← true
    if K ‖ R ∉ Domain(H) then H(K ‖ R) ←$ {0,1}^∞
    return H(K ‖ R)

ON ENCRYPTION QUERY (j, g^?):
    Compute M ← g^?(K). To do this, run g.
    When g makes a RO query of K ‖ R:
        if K ‖ R ∉ Domain(H) then H(K ‖ R) ←$ {0,1}^∞
        Answer g's RO query with H(K ‖ R)
    R ←$ {0,1}^k
    C ←$ {0,1}^{|M|}
    if K_j ‖ R ∈ Domain(H) then
      bad ← true
      C ← H(K_j ‖ R) ⊕ M           ⟵ use this line in game R
      C ← H(K_j ‖ R) ⊕ 0^{|M|}     ⟵ use this line in game F
    else
      H(K_j ‖ R) ← C ⊕ M           ⟵ use this line in game R
      H(K_j ‖ R) ← C ⊕ 0^{|M|}     ⟵ use this line in game F
      𝒳 ← 𝒳 ∪ {K_j ‖ R}
    return R ‖ C
```

Figure 1: Definition for game R and F. We define $\mathrm{Domain}(H)$ to be the set of points X where $H(X)$ is not equal to $\mathsf{undefined}$.

inspecting the code. Ignore all statements involving variable *bad* since these statements do not result in any change visible to adversary A (they are used only for accounting purposes). The code simulates a random oracle H by assigning a uniform and independent range point $H(X)$ to each domain point X for which a value is needed. The code chooses C at random and then defines $H(K_j||R)$ from it, rather than choosing $H(K_j||R)$ at random and then defining C from it. These are clearly equivalent as far as the view provided to adversary A. With the natural abbreviations in notation we then have the following:

Claim 1. $\mathsf{Adv}^{\mathrm{kdm}}_{\mathbf{ver}[k]}(A) = \left|\Pr[A^{\mathsf{R}} = 1] - \Pr[A^{\mathsf{F}} = 1]\right|$

To proceed with our analysis we introduce in Figure 2 a third game, game C. Let "A^{R} sets *bad*" be the event that flag *bad* gets set to true (at some point in the execution) when A runs with an R oracle, and similarly define events "A^{F} sets *bad*" and "A^{C} sets *bad*". We have the following:

Claim 2. $\left|\Pr[A^{\mathsf{R}} = 1] - \Pr[A^{\mathsf{C}} = 1]\right| = \Pr[A^{\mathsf{C}} \text{ sets } bad\,]$ and
$\left|\Pr[A^{\mathsf{F}} = 1] - \Pr[A^{\mathsf{C}} = 1]\right| = \Pr[A^{\mathsf{C}} \text{ sets } bad\,]$ and so

INITIALIZATION:

10 **for** $j \leftarrow 1$ **to** N **do** $K_j \xleftarrow{\$} \{0,1\}^k$
11 $bad \leftarrow \mathsf{false}$; $\mathcal{X} \leftarrow \emptyset$
12 **for all** $X \in \{0,1\}^{2k}$ **do** $H(X) \leftarrow \mathsf{undefined}$

ON RO QUERY $K \parallel R$:

20 **if** $K \parallel R \in \mathcal{X}$ **then** $bad \leftarrow \mathsf{true}$
21 **if** $K \parallel R \notin \mathrm{Domain}(H)$ **then** $H(K \parallel R) \xleftarrow{\$} \{0,1\}^\infty$
22 **return** $H(K \parallel R)$

ON ENCRYPTION QUERY $(j, g^?)$:

30 Compute $M \leftarrow g^?(\mathbf{K})$. To do this, run g.
31 When g makes a RO query of $K \parallel R$:
32   **if** $K \parallel R \notin \mathrm{Domain}(H)$ **then** $H(K \parallel R) \xleftarrow{\$} \{0,1\}^\infty$
33   Answer g's RO query with $H(K \parallel R)$
34 $R \xleftarrow{\$} \{0,1\}^k$
35 $C \xleftarrow{\$} \{0,1\}^{|M|}$
36 **if** $K_j \parallel R \in \mathrm{Domain}(H)$ **then** $bad \leftarrow \mathsf{true}$
41   $\mathcal{X} \leftarrow \mathcal{X} \cup \{K_j \parallel R\}$
42 **return** $R \parallel C$

Figure 2: Game C. This game has similar behavior, as far as the adversary can tell, to games R and F.

$$\left|\Pr[A^{\mathsf{R}} = 1] - \Pr[A^{\mathsf{F}} = 1]\right| \leq 2 \cdot \Pr[A^{\mathsf{C}} \text{ sets } bad\,]$$

Let us show the the first equality. First imagine eliminating line 38 from the code for game R. Call the new game R'. The change does not impact the probability that flag *bad* is set to true (that is, $\Pr[A^{\mathsf{R}} \text{ sets } bad\,] = \Pr[A^{\mathsf{R}'} \text{ sets } bad\,]$) because *bad* has already been set to true when line 38 executes. Now that line 38 is gone, in game R', also eliminate line 40. This too does not change the probability that *bad* gets set to true. The reason is that the value stored at $H(K_j \parallel R)$ by line 40 can only influence what the adversary sees by a subsequent execution of line 22, and when the assignment statement at line 40 does so influence line 22, the bookkeeping done at line 41 will have caused *bad* to be set to true at line 20 anyway. (Why can't the assigning of a value to $H(K_j \parallel R)$ at line 40 influence values returned at a subsequent execution of line 42 in game R'? Because the use made of the $H(\cdots)$-values in lines 30–42 of game R' is only in defining $M = g^H(\mathbf{K})$, but then all that we use of M (in lines 34–42 of game R') is $|M|$ (at line 35), and we have *assumed* that $|g^H(\mathbf{K})|$ is independent of both H and $\mathbf{K}$, our "fixed-length" assumption on g.) We conclude that $\Pr[A^{\mathsf{R}} \text{ sets } bad\,] = \Pr[A^{\mathsf{C}} \text{ sets } bad\,]$. The exact same reasoning tells us that $\Pr[A^{\mathsf{F}} \text{ sets } bad\,] = \Pr[A^{\mathsf{C}} \text{ sets } bad\,]$. The final inequality of Claim 2 then follows by the triangle inequality.

Next we show the following:

Claim 3. $\Pr[A^{\mathsf{C}} \text{ sets } bad] \leq q^2/2^{k-1}$

To prove this claim we separately consider the two places in game C where *bad* can be set to true. The probability that *bad* gets set to true at line 36 is at most $q^2/2^k$ since during each of the at most q times that line 36 is executed the value R is randomly chosen just before (line 34) from a set of size 2^k and then we test if *something* $\parallel R$ is in a set of size at most q. Similarly, we claim that the probability that *bad* gets set to true at line 20 is at most $qN/2^k \le q^2/2^k$. This statement would be clear if only the adversary were given no information about the values $(K_1, \ldots, K_N)$ selected at line 10; in such a case, we are testing at most q times if $K \parallel$ *something* (where K is supplied by the adversary) is in a random, secret set of size at most $N/2^k$. But, in fact, the view provided to the adversary A in game C *is* independent of $(K_1, \ldots, K_N)$. Note that every encryption query returns a random string whose length is independent of $(K_1, \ldots, K_N)$ (by the fixed-length assumption on g) while every RO query returns a random infinite string, independent of $(K_1, \ldots, K_N)$ (whether that random string is selected in line 21 or in line 32 is irrelevant).

Putting together the three claims completes the proof. ▮

Alternative constructions. There are a number of natural constructions for a KDM-secure symmetric encryption scheme in the RO model. For example, one might start with an arbitrary symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ for which the coins R used to encrypt can be recovered from the ciphertext. (Standard modes like CBC with a random IV have this property.) Then modify Π to a symmetric encryption scheme $\bar{\Pi} = (\mathcal{K}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ by setting $\bar{\mathcal{E}}_K(M, R) = \mathcal{E}_{H(K \parallel R)}(M, R)$ and correspondingly modifying $\mathcal{D}$. It seems reasonable to expect that $\bar{\Pi}$ will be IND-KDM secure (in the RO model) when Π is IND-CPA secure (in the standard model).

6 KDM Insecurity for Stateful Symmetric Encryption

A *stateful* encryption scheme is like the (probabilistic, symmetric) encryption schemes we have defined except that the encryption function $\mathcal{E}$ is deterministic and takes as input a hidden variable, the *state*. Every time a message M is encrypted under $\mathcal{E}_K$ the current state S of the encryption algorithm is modified to a new state $S' = f(S, M)$. (It is fine to allow this update to depend on K as well.) The initial state, S_0, is specified as part of the scheme. We write $\mathcal{E}_K^S(M)$ for the encryption of message M using key K and state S.

An example of a stateful encryption scheme is the form of CBC mode where the IV is $E_K(S)$ and S, initially the zero-block, is incremented with every message that is encrypted. This scheme is IND-CPA secure.

It is straightforward to adapt the definitions we have given for KDM security to deal with stateful schemes. However, one will soon notice that there can not exist a stateful KDM-secure encryption scheme. Let us explain the problem.

Since the function $\mathcal{E}$ is deterministic and and the plaintext-construction function g depends on $\mathbf{K}$, the plaintext-construction function g "knows" everything that $\mathcal{E}$ knows and so, in particular, g can determine what ciphertext $\mathcal{E}$ will produce next for any given message. The function g can generate a plaintext such that its encryption conveys to A some information about $\mathbf{K}$. For example, one could define a plaintext-construction function $g_{i,j,k,S,V}$ that returns the first k-bit message M (or 0^k if no such message exists) such that the inner product of V and ciphertext $C = \mathcal{E}_K^S(M)$ is the ith bit of K_j. Using such functions the adversary can easily extract an underlying key, bit-by-bit, and use this key to break the scheme's KDM security.

7 KDM Security in the Public-Key Setting

Preliminaries. We recall the syntax of an asymmetric encryption scheme, as given (in a slightly modified form) in [4]. Let String, $\mathsf{Plaintext}$, and $\mathsf{Ciphertext}$ all be as before. Also, let $\mathsf{Parameter} = 1^*$ and let $\mathsf{Coins} = \{0,1\}^\infty$, $\mathsf{PublicKey} \subseteq \mathsf{String}$, and $\mathsf{SecretKey} \subseteq \mathsf{String}$. Then an *asymmetric encryption* *scheme* is a triple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where $\mathcal{K}$: $\mathsf{Parameter} \times \mathsf{Coins} \to \mathsf{PublicKey} \times \mathsf{SecretKey}$ and $\mathcal{E}$: $\mathsf{PublicKey} \times \mathsf{String} \times \mathsf{Coins} \to \mathsf{Ciphertext} \cup \{*\}$ and $\mathcal{D}$: $\mathsf{SecretKey} \times \mathsf{String} \to \mathsf{Plaintext} \cup \{*\}$. As for symmetric encryption, we usually omit the last argument to both $\mathcal{K}$ and $\mathcal{E}$, thinking of both as probabilistic algorithms, and we write the second argument to $\mathcal{E}$ and $\mathcal{D}$ as a subscript. Upon input 1^k the key-generation algorithm returns a pair (pk, sk) of matching public and secret keys. We require that for all (pk, sk) which can be output by $\mathcal{K}(1^k)$, for all $M \in \mathsf{Plaintext}$, and for all strings C that can be output by $\mathcal{E}_{pk}(M)$, we have that $\mathcal{D}_{sk}(C) = M$.

Definition of KDM security in the public-key setting. The intuition behind the definition of KDM security in the asymmetric setting is just like that for the symmetric setting. The adversary A is given access to an oracle that will encrypt under a specific public key pk a specified function g that may depend on a vector of secret keys $\mathbf{sk}$. Notice that A can compute encryptions of standard plaintext messages on its own, but still requires the oracle for computation of g when it accesses $\mathbf{sk}$. After interacting with its oracle, adversary A outputs a "1" if it believes its oracle returns real encryptions, and it outputs a "0" otherwise. If a reasonable adversary can not do a good job at this game, then the scheme is KDM secure. We let $(\mathbf{pk}, \mathbf{sk}) \stackrel{\$}{\leftarrow} 1^k$ denote: **for** $i \in \{1, 2, \ldots\}$ **do** $(pk_i, sk_i) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$.

Definition 7.1 [IND-KDM — standard model — asymmetric setting] Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme, and let A be an adversary. For $(\mathbf{pk}, \mathbf{sk}) \in (\mathsf{PublicKey} \times \mathsf{SecretKey})^\infty$ let

$\mathsf{Real}_{\mathbf{sk}}$ be the oracle that on input (i, g) returns $C \stackrel{\$}{\leftarrow} \mathcal{E}_{pk_i}(g(\mathbf{sk}))$

$\mathsf{Fake}_{\mathbf{sk}}$ be the oracle that on input (i, g) returns $C \stackrel{\$}{\leftarrow} \mathcal{E}_{pk_i}(0^{|g(\mathbf{sk})|})$.

The **IND-KDM advantage** of an adversary A for security parameter k is

$$\begin{aligned} \mathsf{Adv}^{\mathrm{kdm}}_{\Pi,k}(A) \stackrel{\mathrm{def}}{=} & \Big| \Pr\Big[(\mathbf{pk}, \mathbf{sk}) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k) :\ A^{\mathsf{Real}_{\mathbf{sk}}}(1^k, \mathbf{pk}) = 1\Big] \\ & - \Pr\Big[(\mathbf{pk}, \mathbf{sk}) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k) :\ A^{\mathsf{Fake}_{\mathbf{sk}}}(1^k, \mathbf{pk}) = 1\Big]\Big|. \qquad \diamondsuit \end{aligned}$$

The plaintext-construction function g must, as usual, be fixed-length: $|g(\mathbf{sk})|$ may not depend on $\mathbf{sk}$.

The definition for KDM security within the RO model is exactly analogous to what we have done already. Here one insists that $|g^H(\mathbf{sk})|$ be independent of both H and $\mathbf{sk}$.

The **VER** construction. We recall a simple encryption scheme specified in [5]. Let $\mathcal{F}$ be a trapdoor permutation generator: on input 1^k the probabilistic algorithm $\mathcal{F}$ returns (the encodings of) functions (f, f^{-1}) where f: $\{0,1\}^k \to \{0,1\}^k$ and f^{-1}: $\{0,1\}^k \to \{0,1\}^k$ is its inverse. (We do not bother to distinguish between functions and their encodings.) Mirroring our construction for the symmetric-key setting, define the (random-oracle model) asymmetric encryption scheme $\mathbf{VER}[\mathcal{F}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ as follows. The key-generation algorithm $\mathcal{K}$ is identical to $\mathcal{F}$. Encryption algorithm $\mathcal{E}_f(M, R) = f(R) \,\|\, (H(R) \oplus M)$ where $k = |R|$ is the domain length of f (assumed to be apparent f and f^{-1}) and R is the random coins used by the encryption algorithm and H is the random oracle. Decryption algorithm $\mathcal{D}_{f^{-1}}(\mathcal{C}) = *$ if $|\mathcal{C}| < k$ and $\mathcal{D}_{f^{-1}}(\mathcal{C}) = H(f^{-1}(Y)) \oplus C$ otherwise, where Y is the first k bits of $\mathcal{C}$ and C is the remaining bits.

One expects that $\mathbf{VER}[\mathcal{F}]$ is a KDM-secure encryption scheme if $\mathcal{F}$ is a secure trapdoor permutation. At the time of this writing, we have not written up a proof of this.

The above is only one natural construction; others would seem to work. In [7] Camenisch and Lysyanskaya give a different scheme which they claim is "circular secure" (in the RO model), a notion that they define. One would expect their scheme to be KDM secure as well, though we have not written up a proof of this.

Acknowledgments

Thanks to the anonymous reviewers for their comments.

John Black received support from NSF CAREER award CCR-0133985. Part of this work was carried out while John was at the University of Nevada, Reno. Phil Rogaway and his student Tom Shrimpton received support from NSF grant CCR-0085961 and a gift from CISCO Systems. Many thanks for their kind support.

References

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, January 1999. An extended version appeared as Digital Equipment Corporation Systems Research Center report No. 149, January 1998.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *IFIP International Conference on Theoretical Computer Science*, August 2000.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1998.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989. A preliminary version appeared as Digital Equipment Corporation Systems Research Center report No. 39, February 1989.

[7] J. Camenisch and A. Lysyanskaya. "Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation". In *Advances in Cryptology – EUROCRYPT '01*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[8] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. To appear in *SIAM J. on Computing*. Earlier version in STOC 91, 1998.

[9] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, March 1983.

[10] M. Fischlin. "Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications". In *Advances in Cryptology – EUROCRYPT '99*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.

[12] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search. *Journal of Cryptology*, 14(1):17–35, 2001. Earlier version in CRYPTO '96.

[13] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[14] S. Micali. Personal communication, circa 1985.

[15] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.

[16] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems (extended abstract). *Electronic Notes in Theoretical Computer Science*, 32, April 2000.

[17] B. Pfitzmann and M. Waidner. "Composition and integrity preservation of secure reactive systems". *IBM Research Report RZ 3234, #93280*, June 2000.

[18] C. Rackoff and D.Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Computer Science. Springer-Verlag, 1994.