Hybrid Encryption in the Multi-User Setting

G.M. Zaverucha

Research In Motion
4701 Tahoe Boulevard
Mississauga ON, L4W 0B5

March 20, 2012

Abstract

This paper presents an attack in the multi-user setting on various public-key encryption schemes standardized in IEEE 1363a [20], SECG SEC 1 [27] and ISO 18033-2 [21]. The multi-user setting is a security model proposed by Bellare et al., which allows adversaries to simultaneously attack multiple ciphertexts created by one or more users. An attack is considered successful if the attacker learns information about any of the plaintexts. We show that many standardized public-key encryption schemes are vulnerable in this model, and give ways to prevent the attack. We also show that the key derivation function and pseudorandom generator used to implement a hybrid encryption scheme must be secure in the multi-user setting, in order for the overall primitive to be secure in the multi-user setting. As an illustration of the former, we show that using HKDF (as standardized in NIST SP 800-56C) as a key derivation function for certain standardized hybrid public-key encryption schemes is insecure in the multi-user setting.

1 Introduction

The security of public-key encryption (PKE) schemes is most commonly analyzed in a simplified model where an attacker must distinguish between the encryption of two chosen messages. In reality, there will likely be multiple ciphertexts, potentially created with multiple public keys, and an attacker may be simultaneously attacking some or all of the ciphertexts. In terms of the security games used to define security,

the difference equates to having one vs. many "challenge ciphertexts", i.e., ciphertexts given to the adversary. The adversary is successful if he learns any information about the plaintext. In the most common security definitions (e.g., IND-CPA and IND-CCA) there is a single challenge ciphertext.

Security in the second scenario has been studied under the name *multi-user security* [6, 11], where the traditional, more limited scenario is named *single-user security*. In [6], Bellare, Boldyreva and Micali prove that any PKE scheme secure in the single-user setting is also secure in the multi-user setting. This is a generic result, in that it works for any scheme. Unfortunately, the reduction loses a factor of N, where N is the number of challenge ciphertexts. Since in practice N can be quite large, the theorem does not ensure security with the parameter sizes in common use. For example, a scheme designed to provide 80 bits of single-user security is only guaranteed to provide 60 bits of multi-user security if $N = 2^{20}$.

Most practical examples of PKE use a special type of PKE scheme, called a *hybrid public-key encryption* (HPKE) scheme. Practitioners use a two step-approach to encrypt long messages, since a public-key encryption operation is orders of magnitude more expensive than a symmetric-key operation. First, a symmetric key is encrypted using an expensive public-key operation, then the symmetric key is used to efficiently encrypt the (long) message. The first step is called a *key encapsulation mechanism* (KEM) and the second step is called a *data encapsulation mechanism* (DEM).

Multi-user security of some common KEMs is also studied in [6], where it is shown that two common KEMs (Elgamal and Cramer-Shoup) do provide good concrete security in the multi-user setting. Boldyreva does a similar analysis of some RSA-OAEP variants in [11], and does consider one hybrid scheme. However, the multi-user security of common HPKE schemes has not been analyzed (i.e., the multi-user security of the combined KEM/DEM operations). In particular, the multi-user security of the standardized and deployed HPKE schemes is not supported by published analysis. Previous work has also not considered whether the key derivation functions and pseudorandom generators used in a multi-user HPKE scheme must also be secure in the multi-user setting.

Contributions and Outline In this paper, we present an attack which reduces multi-user security by a factor of N (where N is the number of ciphertexts created) (§3). The attack recovers one of the N symmetric (DEM) keys (but the attacker cannot control which one). The attack is successful against multiple schemes from various standards, such as IEEE 1363a [20], SECG SEC 1 [27] and ISO 18033-2 [21]. The attack is a simple time-memory tradeoff (TMTO) attack on the DEM, a type of attack that has been well-studied in the literature in the context of symmetric-key

primitives (it is also sometimes called a *key-collision attack*). This is possible since the DEM is deterministic – which isn't a security issue in the single-user setting. Preventing the attack is also rather simple, and is accomplished by randomizing the DEM. We discuss a few options and make a recommendation for preventing this attack.

Finally, we look at the other primitives used to implement HPKE in practice, namely a *key derivation function* (KDF) and a *pseudorandom generator* (PRG) (§5). We will see it is necessary that these building blocks must have multi-user security in order for the overall HPKE primitive to provide multi-user security. For PRGs we give an example of a toy PRG that provides single-user but not multi-user security, and apply a TMTO attack to an HPKE scheme using this PRG (§5.2).

For KDFs, we consider using a class of recently standardized [14, 24] "extract-then-expand" functions (including HKDF [23]) in the context of HPKE, and show that a TMTO attack is possible for many of the allowed parameters (§5.1). This class of KDF works by extracting a seed from the input, then expanding the seed to create the output. For some of the standardized parameter choices, using an extract-then-expand KDF with some of the standard HPKE schemes listed above will be insecure. A multi-user setting TMTO attack, similar to the one described for HPKE, can recover the seed, and expand it to recover the DEM key and decrypt the message. Our recommendation is to choose parameters according to a new requirement, or to employ a salt value to prevent the attack.

2 Hybrid Public-Key Encryption

In this section we review public-key encryption (PKE), then turn our attention to a specific type of PKE, hybrid public-key encryption (HPKE). We give the standard security definitions of PKE and define security in the multi-user setting, and then discuss known results in the single-user and multi-user settings.

Throughout, κ will be a security parameter. The notation $x \leftarrow y$ means that x is assigned the value y, and $x \xleftarrow{\$} X$ means that x is assigned a value chosen uniformly at random from a set X.

2.1 Public-Key Encryption

A *public-key encryption scheme* (PKE) is a four-tuple of probabilistic polynomial-time (PPT) algorithms $\mathcal{E} = (\mathsf{P}, \mathsf{K}, \mathsf{E}, \mathsf{D})$, with the following properties.

- Algorithm P is the *parameter generation* algorithm. It takes as input a *secu-*

rity parameter κ, an integer indicating the desired security level, and outputs the *domain parameters* as a string *par*, which is a description of the class of public keys that should achieve the desired security level and interoperability. For example, P will typically determine the bit lengths of the public key, and perhaps other information such as an elliptic curve selection. We will write $par \leftarrow \mathsf{P}(\kappa)$.

- Algorithm K is the *key generation* algorithm. It takes as input domain parameters *par* and generates as output a *key pair* (sk, pk). The value sk is the *secret key* and the value pk is the *public key*. Algorithm K will need to be probabilistic in order to achieve security. We will write $(sk, pk) \leftarrow \mathsf{K}(par)$.

- Algorithm E is the *encryption* algorithm. E will usually be probabilistic. It takes as input a *plaintext* m in $\{0,1\}^*$ and a public key pk. It generates as output a *ciphertext* c in $\{0,1\}^*$. We will write $c \leftarrow \mathsf{E}(pk, m)$.

- Algorithm D is the *decryption* algorithm. It takes as input a ciphertext c in $\{0,1\}^*$ and the secret key sk. It generates as output a plaintext m in $\{0,1\}^*$, or error. For the schemes we consider, Algorithm D will be deterministic. We will write $m \leftarrow \mathsf{D}(sk, c)$.

- Scheme $\mathcal{E}$ must be self-consistent in the sense that for all κ and all $m \in \{0,1\}^*$, if

$$\begin{aligned} par &\leftarrow \mathsf{P}(\kappa), \\ (sk, pk) &\leftarrow \mathsf{K}(par), \\ c &\leftarrow \mathsf{E}(pk, m), \\ m' &\leftarrow \mathsf{D}(sk, c), \end{aligned}$$

  then $m = m'$ will be true.

2.2 Symmetric Key Encryption

A symmetric-key encryption (SKE) scheme has two algorithms, encrypt and decrypt, denoted (E, D).[1] Both are keyed with a key K from the keyspace $\{0,1\}^\kappa$. Input to E is a key and an arbitrary length message. The output of E is a ciphertext. Input to D is a key and a ciphertext produced by E. The usual soundness condition applies,

[1] We'll use italics for symmetric-key primitives, i.e., (E, D), and a sans-serif typeface for public-key primitives, i.e., (E, D).

i.e., for any message m and $K \in \{0,1\}^\kappa$, $D(K, E(K, m)) = m$. Encryption may be randomized with an *initialization vector* (IV) depending on the mode of operation.[2] The same IV must be used for decryption.

2.3 Hybrid Public-Key Encryption

Hybrid encryption is a general notion that captures most practical uses of public-key encryption (PKE). A public-key encryption operation is orders of magnitude more expensive than a symmetric-key operation. Therefore, to encrypt long messages, practitioners encrypt a symmetric key K using the expensive public-key operation, $c_1 = \mathsf{E}(pk, K)$, and then use K to encrypt a (possibly long) message, $c_2 = E(K, M)$. The ciphertext is (c_1, c_2). Public-key encryption schemes constructed with a hybrid approach are also referred to as KEM/DEM schemes, since c_1 is produced by a *key encapsulation method* (KEM) and c_2 is produced by a *data encapsulation method* (DEM). A formal treatment of the KEM/DEM paradigm can be found in [15].

A KEM has the same list of algorithms as a PKE scheme (KEM.P, KEM.K, KEM.E, KEM.D), but algorithms KEM.E and KEM.D are a little different.

- Algorithm KEM.E takes as input a public key, and outputs a key $K \in \{0,1\}^\kappa$ (for a parameter κ), and a ciphertext component c_1.
- Algorithm KEM.D takes a private key and ciphertext component c_1 as input, and outputs a key $K' \in \{0,1\}^\kappa$.

A DEM has two algorithms DEM.E and DEM.D, which are symmetric-key encryption and decryption algorithms. The algorithm DEM.E takes K (output by KEM.E) and a plaintext $M \in \{0,1\}^*$, and outputs a second ciphertext component c_2. The algorithm DEM.D takes K' (output by KEM.D) and c_2, and outputs M. The parameter κ is chosen so that the output of KEM.E is suitable for use by DEM.E. Figure 1 shows the encryption and decryption operations in a generic hybrid PKE.

Note that c_1 and c_2 may have one or more components, depending on the scheme, e.g., in DHIES [1] and PKCS #1 [26] c_1 is a single value, while in the Cramer-Shoup scheme CS1 [15] it is a 4-tuple, and likewise c_2 may consist of a ciphertext, or a ciphertext and an authentication tag.

[2] In the case of counter mode, the IV is sometimes called an initial counter value (IC or ICV).

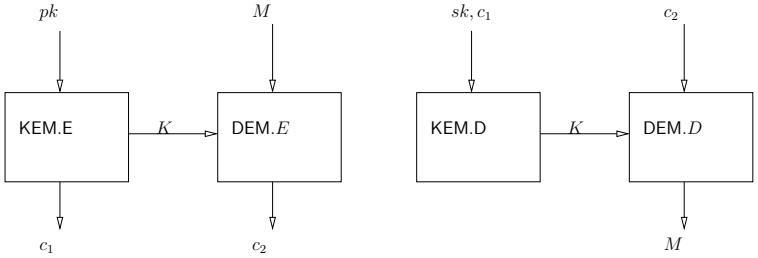

Figure 1: Hybrid encryption with a KEM/DEM construction. The left side shows the encryption operation, and the right side shows the decryption operation.

2.4 Security in the Single-User Setting

First we review the common security notion: *indistinguishability against chosen ciphertext attacks* (IND-CCA) for a public-key encryption scheme $\mathcal{E}$. Let A be an adversary. Define

$$\mathbf{Adv}^{\mathsf{CCA}}_{\mathcal{E},\,A}(\kappa) = 2\Pr\left[\, A^{\mathcal{O}_{sk}}(c^*) = b \;\middle|\; \begin{array}{l} par \leftarrow \mathsf{P}(\kappa); \\ (sk, pk) \leftarrow \mathsf{K}(par); \\ (m_0, m_1) \leftarrow A^{\mathcal{O}_{sk}}(pk); \\ b \xleftarrow{\$} \{0,1\}; \\ c^* \leftarrow \mathsf{E}(pk, m_b) \end{array} \right] - 1$$

where $\mathcal{O}_{sk}$ is a decryption oracle, and there are two conditions: $\mathcal{O}_{sk}$ will not decrypt c^*, and $|m_0| = |m_1|$. We say $\mathcal{E}$ is *IND-CCA secure* if $\mathbf{Adv}^{\mathsf{CCA}}_{\mathcal{E},\,\mathsf{A}}(\kappa)$ is negligible for all stateful PPT adversaries A. Recall the term "left-or-right query" (*LR query* for short), for the step at which A chooses (m_0, m_1) and receives c^*.

Chosen ciphertext security may also be defined for symmetric-key algorithms in an analogous way, but in addition to a decryption oracle, we must provide an encryption oracle which produces ciphertexts with the same key used to create the challenge ciphertext.

Cramer and Shoup [15] define *one-time symmetric-key encryption* to be a restricted type of symmetric-key encryption, where the key is used only to create a single ciphertext, namely, the challenge ciphertext. We write E_1 when the SKE E is restricted to one-time use. One-time SKE is natural in the context of hybrid encryption, since for each encryption, the KEM encapsulates a different key, used only once by the DEM. In practice a traditional (many-time) SKE scheme is often used as the DEM in a hybrid construction, but the notion of one-time SKE is used in security analyses because it is a strictly weaker requirement.

The IND-CCA security game for one-time SKE is as follows. The adversary chooses a pair of equal length plaintexts (m_0, m_1), and submits them to an encryption oracle. The oracle chooses K and b, then outputs $c^* = E_1(K, m_b)$. The adversary may query a decryption oracle any number of times with ciphertexts $c \neq c^*$, and the oracle will output $D(K, c)$. No encryption oracle is provided. Finally the adversary outputs a guess b'. Define adversary A's advantage $\mathbf{Adv}^{\mathsf{CCA}}_{E_1,\,A}(\kappa) = 2\Pr[b = b'] - 1$.

Security of Hybrid PKE (HPKE) Cramer and Shoup [15] proved a tight reduction of the IND-CCA security of an HPKE scheme to the IND-CCA security of the KEM and one-time SKE (used to create the HPKE). Their result is in the single-user setting. This reduces the task of constructing an IND-CCA secure PKE to the construction of a KEM and DEM that are individually secure; Theorem 2.1 says that the composition is secure.

Theorem 2.1. *[15, Theorem 5] Let* HPKE *be a hybrid encryption scheme constructed from a key encapsulation method* KEM *and a one-time symmetric encryption scheme* SKE. *If* KEM *and* SKE *are IND-CCA secure, then* HPKE *is also IND-CCA secure.*

2.5 Security in the Multi-User Setting

In the single user setting there is one public key, one challenge ciphertext (denoted c^* above), and one decryption oracle. The attacker is successful if he distinguishes ciphertexts in this limited world. In practice however, there are many users sending multiple encrypted messages, which may be related. There are therefore multiple public keys, and multiple challenge ciphertexts. The *multi-user setting* attempts to model this aspect of real-world systems. In this section we aim to make precise the notion of multi-user security for public-key encryption. To this end we present and discuss the definition from [6].

The generalization of CCA security in the multi-user setting has n different public keys, and the adversary can make multiple LR queries. These challenge ciphertexts are created with the same (randomly chosen) selector bit b, i.e., all ciphertexts are encryptions of the left input, or all ciphertexts are encryptions of the right input. The adversary is successful if he guesses b. Security theorems in this model are stated for a number of public keys n, and a number of LR queries q_e per public key.

Using a single b value allows the plaintexts between two challenge ciphertexts may be related. For example, the adversary could choose to have $c^*_i = E(pk_i, m_b)$ and $c^*_j = E(pk_j, f(m_b))$ for any f, and any public keys pk_i and pk_j (which might be equal). This captures a class of attack on RSA, where the same message is encrypted under different low-exponent RSA public keys. See [16] for details.

IND-CCA security in the multi-user setting, denoted n-CCA, is formalized as the following advantage:

$$\mathbf{Adv}^{n\text{-}\mathsf{CCA}}_{\mathcal{E},\,A}(\kappa) = 2\Pr\left[\, b' = b \;\middle|\; \begin{array}{l} par \leftarrow \mathsf{P}(k);\ (pk_i, sk_i) \leftarrow \mathsf{K}(par); \\ b \xleftarrow{\$} \{0,1\}; \\ \text{For } i = 1, \ldots, N \\ \quad (m_0^i, m_1^i, pk_i) \leftarrow A^{\mathcal{O}_{sk_1},\ldots\mathcal{O}_{sk_n}}(pk_1, \ldots, pk_n); \\ \quad c_i^* \leftarrow \mathsf{E}(pk_i, m_b) \\ b' \leftarrow A^{\mathcal{O}_{sk_1},\ldots\mathcal{O}_{sk_n}}(c_i^*, \ldots, c_N^*) \end{array} \right] - 1$$

where there are n key pairs created in the second step. As in [6], the key pairs are generated with the same parameters. To some extent our attack relies on this, the KEM may use different parameters but the DEM must have the same parameters.

Multi-user security for one-time SKE has not previously been formally defined. As with n-CCA for PKE, the selector bit b is the same for all LR queries. There are n keys and their associated LR and decryption oracles, but no general encryption oracle is provided. For unrestricted SKE (i.e., without the one-time restriction on the key) multi-user security and the possibility of TMTO attacks are well-known. Two recent papers on this subject cite most of the relevant references [13, 19].

Multi-user HPKE For hybrid encryption, there has been limited analysis done in the multi-user setting. TMTO attacks on PKE are briefly considered in [19], but only the security of the KEM was considered, not the security of hybrid encryption schemes. To our knowledge, [11] is the only paper to date which explicitly considers n-CCA security of HPKE. In [11], Boldyreva studies the security of an RSA-OAEP [5] variant, known as RSA-OAEP++, in the multi-user security model of [6]. Of interest to the present work is one variant of RSA-OAEP++ to encrypt arbitrary-length messages [11, §6]. By modifying the output length of one of the hash functions in the RSA-OAEP++ construction, Boldyreva shows that we may encrypt arbitrary length messages with the same security guarantee as for short, fixed length messages. This result is relevant to the current work because the security proof is tight, and security is considered in the multi-user setting, closing a gap between [6] (which studies only KEM security) and the literature on hybrid encryption (which has largely ignored the multi-user setting). [3]

[3]Note that to securely instantiate the arbitrary length plaintext version of RSA-OAEP++, the parameter k_0 must be twice as long as the security parameter. For an attacker which makes no decryption queries, the reduction [11, Th. 5.1] contains a term: $q_e q_G / 2^{k_0}$. If the attacker can observe $q_e \approx 2^{\kappa/2}$ ciphertexts, and perform $q_G \approx 2^{\kappa/2}$ hash computations, then when $k_0 = \kappa$, (i)

In [6], Bellare et al. prove that any PKE scheme secure in the single-user setting is also secure in the multi-user setting. This is a generic result, in that it works for any scheme. A similar result for stateful PKE is presented in [8, Theorem A.1]. If we apply this general reduction to Theorem 2.1, we lose a factor of N, where N is the number of challenge ciphertexts (created with the n public keys). That is, for a (possibly hybrid) PKE scheme PKE the only guarantee we have is that

$$\mathbf{Adv}^{n\text{-}\mathsf{cca}}_{\mathsf{PKE}} \leq N \cdot \mathbf{Adv}^{\mathsf{cca}}_{\mathsf{PKE}} \ .$$

Put another way, in the worst case an adversary could be N times more successful when attacking the n-CCA security of a scheme than the CCA security of the same scheme.

Bellare et al. also show that their reduction is optimal in the general case, by showing that there exists a PKE scheme with $\mathbf{Adv}^{n\text{-}\mathsf{cca}}_{\mathcal{E}} = N\mathbf{Adv}^{\mathsf{cca}}_{\mathcal{E}}$. The scheme they use to make this argument is contrived, but our attack shows that multi-user attacks exist on practical and deployed schemes as well. We show that the standardized schemes discussed in §A lose a factor of N in the multi-user setting.

2.6 Discussion

An interesting consequence of the one-time restriction on symmetric-key IND-CCA security is that encryption no longer needs to be randomized, or have a changing IV, since the key changes for each encryption operation. Intuitively, random or non-repeating IVs are necessary for IND-CPA security, since if the attacker sees two encryptions of the same message, he may distinguish them. To win the CPA game, first request encryptions of an arbitrary message pair (m_0, m_1), store the ciphertexts, then submit (m_0, m_1) to the LR oracle to get c^*. Compare c^* to the stored ciphertexts to learn b. This attack, which is prevented by randomizing encryption, is not possible with one-time SKE, since it requires multiple encryptions to be created with the same key.

Perhaps this was the reasoning that led to most hybrid encryption standards allowing deterministic DEMs (details in Appendix A). If so, it is an example of a provable security analysis leading to decreased practical security: avoiding the use of a stronger assumption led to an attack in a slightly broader model. [4] Once we move to the multi-user setting, a TMTO attack becomes possible against deterministic

no security is guaranteed by the theorem, and (ii) a TMTO attack along the same lines as the one we discuss in §3 is possible. It is unclear why $k_0 = 128$ is given as a "typical" choice in [11]. For 128-bit security we need $k_0 \geq 256$.

[4] A similar example is given by Koblitz and Menezes (see [22], Remark 1).

one-time encryption, and succeeds even if (deterministic) authenticated encryption is being used.

This may also be an example of the omission of a standard safety margin (i.e., randomizing encryption) because the provable security analysis did not require it. See Section 5 of [22] for a discussion of this point.

In contrast, there was a submission to the IEEE P1363 working group, which made clear that the DEM should be randomized, at least in the case of DHIES [2].

3 An Attack on Some Hybrid Encryption Schemes

The core of the attack is a time-memory trade-off (TMTO) attack on the symmetric-key encryption used in the DEM. The attack is also called a *key collision attack*, a term first coined by Biham [9] while applying Hellman's time/memory trade-off (TMTO) methodology [18] to block ciphers. TMTO attacks on symmetric-key encryption and other primitives are also examined in detail in [10] and [19].

In this section we review a key collision attack on symmetric-key encryption, as described by Biham. From this description, it will be easy to describe the attack in the context of hybrid PKE.

We will consider the attack on block ciphers with κ-bit keys (where κ is also the security level). The attack is generic in that it treats the encryption function as a black box. Informally, the attack works by first seeing a large number of ciphertexts that are the encryption of a partially known plaintext under different keys. Then the attacker chooses random keys and encrypts the known plaintext until finding a match. Essentially it is a birthday attack on the keyspace, and the complexity is $O(2^{\kappa/2})$ in time and space. The attack allows one of roughly $O(2^{\kappa/2})$ ciphertexts to be decrypted, but does not allow the attacker to choose which ciphertext will be decrypted (or equivalently, which key will be compromised). This is why it is effective in the multi-user setting, but not in the single-user setting. We first describe the attack for a deterministic E (e.g., with a fixed initialization vector (IV)), then discuss how it may be applied when E is randomized.

Basic Key Collision Attack. Let $E : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^*$ be a symmetric-key encryption function denoted $E(k, M)$, with the message given by the concatenation $M = M_0 \| \ldots \| M_\ell$. Let M_0 be one or more fixed plaintext block(s) known to the attacker.

1. Obtain a set L_1 containing ciphertexts $\{C_1, \ldots, C_{\ell_1}\}$ where each C_i is an encryption of a plaintext of the form $M_0 \| \ldots$, meaning M_0 followed by arbitrary

plaintext blocks. The adversary obtains the ciphertexts by observing honest parties. We assume here that the keys are chosen at random (without replacement). In many practical applications M_0 may be a standard protocol header, and therefore be known. Let k_i denote the (unknown) key used to create C_i.

2. Compute a set L_2 as follows. For $j = 1, \ldots, \ell_2$ choose a distinct key $k_j \in \{0,1\}^\kappa$ and compute $C'_j = E(k_j, M_0)$. Store (k_j, C'_j) in L_2.

3. Compare the ciphertexts $C'_j \in L_2$ to the first parts of $C_i \in L_1$. If $C'_j = \texttt{FirstPart}(C_i)$, output k_j as k_i (the key used to encrypt C_i). For this step to be efficient, L_1 should be stored in a data structure providing constant-time look-up, for example, in a hash table keyed by the first block of C_i. [5]

4. Decrypt C_i, and optionally use k_j for other attacks (depending on the application).

Remark 3.1. It is not necessary to store the set L_2, items in it may be computed, compared against L_1, then deleted.

Remark 3.2. If the length of M_0 is less than the length of the key, then there may be multiple keys found in Step 3 that are different from the true key. Since these keys will incorrectly decrypt the remaining blocks of C_i in Step 4, the attack fails. However, once M_0 becomes larger than the key, the probability of this type of failure decreases rapidly. See [13] for a precise analysis. When $|M_0|$ and the key length are equal, the attack may still be conducted, provided there is some way to verify whether the key is correct. For example, if a MAC of the ciphertext is included it may be verified, or if the unknown plaintext satisfies some redundancy criteria that may be efficiently verified (e.g., English text, or HTML file).

For AES, the block size is 128-bits and the key is 128, 192, or 256 bits. The attack therefore needs at least one, and up to three known blocks to be effective.

Remark 3.3. The attack requires that block i in a ciphertext not depend on blocks j, where $j > i$. Most (if not all) modes in use have this property.

Analysis Our analysis is informal. The more detailed analysis of [13], which considers TMTO attacks on MAC schemes is easily adapted to our setting.

[5] This is oversimplified, since this storage system will likely be distributed among many nodes when κ is large. However, distributed hash tables (DHTs) are practical, and a system optimized for this application may give even better performance than a general-purpose DHT.

$\log_2(\ell_1)$	$\log_2(\ell_2)$	p
62	62	0.060
63	63	0.221
64	64	0.632
65	65	0.982
66	66	0.999
10	118	0.632
20	108	0.632
48	80	0.632
56	70	0.221
51	75	0.221

Table 1: A table of values for $\kappa = 128$, with $|L_1| = |L_2|$ in the upper half.

When $\ell_1 = \ell_2 = \kappa/2$ it is expected, by the birthday paradox, that we recover one of the 2^{ℓ_1} keys. For varying ℓ_1 and ℓ_2, the probability of finding one of the 2^{ℓ_1} keys is given by:

$$p = 1 - \left(1 - \frac{|L_2|}{2^\kappa}\right)^{|L_1|} \geq 1 - e^{-|L_1| \cdot |L_2|/2^\kappa} .$$

Therefore, if $\ell_1 + \ell_2 \geq \kappa$, we expect to recover one of the keys with nontrivial probability. A table of values is given in [9] for various choices for ℓ_1 and ℓ_2 when $\kappa = 56$ (corresponding to DES). Table 1 gives some possible values when $\kappa = 128$ (corresponding to AES-128).

As the name suggests, the attacker may trade off computation time (required to create L_2) for memory (required to store L_1), since any ℓ_1 and ℓ_2 which sum to κ give a successful attack. This flexibility may also allow the attack to proceed when the adversary may observe only a limited amount of ciphertexts, for example, if $\kappa = 128$ and the honest parties produce only 2^{20} ciphertexts, the attacker may perform 2^{108} work to mount the attack.

The flexibility may also be used to reduce online computation, which depends on the honest parties. With this tradeoff, L_2 is constructed first, at the attacker's leisure (offline), then the observed ciphertexts in L_1 are compared to L_2, as they arrive (online). The online vs. offline aspects of TMTO attacks are discussed further in [10, 13].

Extension to randomized E. Suppose the IV is not constant, is either public or private, and is r bits long. The attack still works, by treating the IV and key as an

$r + \kappa$ bit key. The complexity increases to $2^{(r+\kappa)/2}$. First suppose that the IV has r bits of entropy. To neutralize the attack, we need $r = \kappa$, however, for most modes of operation (e.g., CTR, CBC, CFB, CCM, GCM), r is equal to the block length. Therefore, with AES, r is at most 128, which provides security when $\kappa = 128$, but is insufficient[6] when $\kappa > 128$. When IV has $r_0 < r$ bits of entropy, the cost of the attack drops to $O(2^{(r_0+\kappa)/2})$. The recent survey of IV generation practices in IETF protocols by McGrew [25] gives some example protocols that use deterministic IVs (and others that use low-entropy IVs).

Low entropy plaintext block vs. known plaintext block. In a similar manner as IVs, if the first plaintext block is not known, but has only t_0 bits of entropy, the attack runs with complexity $O(2^{(\kappa+t_0)/2})$, which is better than brute force for $t_0 < \kappa$. The complexity $O(2^{(\kappa+t_0)/2})$ is obtained by treating the unknown first block as part of the key. Instead of guessing a key in $\{0,1\}^\kappa$ when forming L_2 we are guessing a value in $\{0,1\}^\kappa \times \mathcal{M}$, for some set $\mathcal{M}$ of possible first plaintext blocks, having size 2^{t_0}.

Extension to non-first known plaintext blocks With some modes of operation, for example counter mode or ECB mode, the attack may be mounted by an attacker who knows any fixed block(s). This is possible since in these modes all blocks are independent.

3.1 Applying the Attack to HPKE Schemes

In the context of HPKE, we may mount the attack from Section 3 on the output of the DEM, and ignore the KEM component of the ciphertext, whenever the DEM is deterministic (or poorly randomized). As we will see, current PKE standards specify a deterministic DEM. So the attack, like most TMTO attacks, is quite simple. Also note that the attack is applicable regardless of the KEM, even if the KEM were ideal, providing information-theoretic security in the multi-user setting. We give two example HPKE schemes (figures 2 and 3) and discuss the practicality of the attack. In Appendix A we discuss some HPKE standards, and the applicability of this attack.

In Figure 2 we give an informal description of the ECIES hybrid encryption scheme, to serve as a concrete example. We describe a simplified variant of the ECIES

[6]We are unaware of a block cipher mode of operation which allows an IV as long as the key, for keys longer than the blocksize. This may make an interesting topic for future work.

Let G be a generator of the elliptic curve group $\mathbb{G}$ having prime order n. The KEM step takes as input the recipient's public key $A = aG$, and a parameter *keyLen*, then proceeds as follows:

1. Choose a random integer $r \in [1, \ldots, n]$.
2. Compute $c_1 = rG$ and $Z = rA = raG$.
3. Compute $K = \mathsf{KDF}(Z)$, where $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^{keyLen}$ is a key derivation function
4. Output: c_1 (public KEM component), K (private component, input to DEM)

The DEM step takes as input K and a message M to be encrypted.

1. Encrypt M as $c_2 = E_K(M)$ where E is a symmetric encryption algorithm, such as AES-CBC. A variety of possible choices for E are present in the standards listed at the beginning of §3.1, and many are deterministic, for example, AES-CBC with an all-zero IV, or AES-CTR with an all-zero initial counter value.
2. Output c_2

The overall output of ECIES is (c_1, c_2). In practice, *keyLen* is the length of two symmetric keys, an encryption key K_1 and a MAC key K_2, and the DEM outputs a third component $c_3 = \mathrm{MAC}_{K_2}(c_2) = \mathrm{MAC}_{K_2}(E_{K_1}(M))$.

Figure 2: A KEM/DEM description of ECIES encryption.

scheme in the KEM/DEM framework. ECIES is standardized in IEEE 1363a [20], SECG SEC 1 [27] and ISO 18033-2 [21].

Note that not all DEMs in SEC 1 and ISO 18033-2 are vulnerable to this attack. In particular the *XOR encryption* scheme (see [27, §3.8] and [21, §6.5.3]) does not appear to be vulnerable. Referring to Figure 2, the XOR encryption DEM uses the KDF to derive, from Z, a key K with the same length as M. The output is $c_2 = M \oplus K$. In Section 5.1, when we look at KDFs in the context of multi-user HPKE, we will see an example of a standardized KDF which is insecure for use in this DEM.

As a second example, Figure 3 gives an RSA-based HPKE scheme. Our description does not follow one particular standard but takes a general approach to RSA-based hybrid encryption. For example, the KEM here is close to the KEM in PKCS-1.5 [26], and ISO 18033-2 [21, §11.4].

Practicality of the Attack. While not a devastating attack, the attack is not strictly theoretical either, for the following reasons.

1. Typically the first block of an encrypted message is a standard protocol header that is either public or has low-entropy.

Let (N, e) be an RSA public key. The integer N is a large composite number having two prime factors p and q, i.e., $N = pq$. The encryption exponent e is a small integer (e.g., 3 or $2^{16} - 1$). The input to the KEM is (N, e), and a parameter $keyLen < \lfloor \log_2 N \rfloor$.

1. Choose $K \in_R \{0,1\}^{keyLen}$.
2. Encode K as an integer k in $\mathbb{Z}_N$ and compute $c_1 = k^e \pmod N$.
3. Output c_1 (public KEM component) and K (private component, input to DEM).

The DEM step takes as input K and a message M to be encrypted.

1. Encrypt M as $c_2 = E(K, M)$, where E is a symmetric-key DEM (as in Figure 2).
2. Output c_2.

As in ECIES (Fig. 2), the DEM may also output a MAC.

Figure 3: A KEM/DEM description of RSA encryption.

2. While most users would never produce $2^{\kappa/2}$ ciphertexts under different keys, a group of 2^{20} users might, and an attacker observing a large network could observe sufficiently many ciphertexts to succeed.

3. The attack may still be performed with fewer than $2^{\kappa/2}$ ciphertexts at the expense of additional computation.

4. For applications with strict security requirements, creating 2^β ciphertexts reduces collective security by β bits, causing the application to fall short of its stated security goal.

5. The storage needed in the attack does not need to be fast (e.g., it may be disk (cheap) instead of more RAM (expensive)), and both storage and computation may be efficiently parallelized.

Given these points, and since the attack may be prevented without a significant increase in computation or bandwidth (as described in §4), standards should be amended to prevent it.

4 Realizing n-CCA Secure HPKE

It is reasonable to assume that as a minimum, both the KEM and DEM must have multi-user security for an HPKE construction to have multi-user security. The n-CCA security of KEMs was studied in [6] and [11]. For DH-based schemes, a tight proof of the n-CCA security of the ElGamal KEM is given in [6]. A proof is given

for Cramer-Shoup KEM scheme is also given in [6] (but the reduction is less tight than for ElGamal).

Here we list a few possible countermeasures to prevent the attack of §3, to improve current standards. We leave proofs of multi-user security of standardized KEMs with the DEMs suggested below to other work.

1. **Choose a random IV.** Choose an IV at random and include it with the ciphertext. This has the drawback of increasing the size of the ciphertext, and an IV must be generated during encryption.

2. **Derive an IV from the KEM component output.** Recall that the KEM generally has two outputs: a private key and a public output that forms part of the hybrid ciphertext. When the public KEM output contains sufficient entropy, we may derive a high-entropy IV from it.

   In ECIES (refer to the description in §3.1) the KEM component is a point of the form rG for a random integer r, and point G. Simply using the low order bits of the binary representation of rG as the IV should suffice. Alternatively, the IV could be derived from rG using a hash function.

   In the RSA example of §3.1, the encryption function applied to K is deterministic. For this approach to be effective, K must be padded with random bits.

3. **Derive an IV from the shared secret.** During the KEM step, while deriving (or generating) the key K used for symmetric encryption in the DEM, also derive (or generate) an IV. Since the intended recipient may recompute K from the public output of the KEM, they may recompute the IV as well. In ECIES the shared secret is rA, and a KDF is applied to it to generate a key for the DEM. In the RSA example of §3.1 there is no key derivation step, but the analog would be to choose an IV at the same time as K and encrypt it along with K.

Our preference is for Option 3, since the changes are minimal, and the ciphertext size and encoding do not change. Computationally, Option 3 is efficient; for typical KDFs only one or two additional hash function evaluations will be required.

5 Multi-User Security of HPKE Building Blocks

So far, we have focused on the KEM and DEM components. In practice, hybrid public-key encryption is realized with two other important components: a key deriva-

tion function (KDF), and a pseudorandom generator (PRG). Do these primitives need to provide security in the multi-user setting as well?

We describe some TMTO attacks on some KDFs and PRGs in the multi-user setting, and show how they compromise HPKE security. Our example in the PRG case is a toy example, but our key derivation example uses HKDF, a recently standardized KDF. We conclude that the KDFs and PRGs used for HPKE in the multi-user setting should also provide multi-user security. However, we do not formally define security of these primitives in the multi-user setting here (this is a good topic for future work).

5.1 Key Derivation

With a few exceptions, KDFs have not received a lot of (published) analysis by cryptographers, and most applications use the following KDF. Let Z_0 be some secret key material, like the output of a key agreement protocol such as Diffie-Hellman or MQV and let H be a cryptographic hash function. To derive a key of length ℓ, compute $K_0 = H(Z_0\|1)\|H(Z_0\|2)\|\ldots$ such that $|K_0| \geq \ell$. Output K, the truncation of K_0 to ℓ bits. Sometimes an auxiliary input parameter is included as input to the hash function. This construction, and variants of it, are used in standards such as ANSI X9.42 [4], IEEE 1363 [20], ANSI X9.63 [3], and ISO-18033-2 [21]. TLS 1.0 uses a variant which replaces H with HMAC keyed by Z_0, applied to a counter along with session-specific randomness.

In [23], Krawczyk argues that cryptographic applications should move to a single, well-studied, rigorously analyzed family of KDFs. To this end, he formally defines security for KDFs, presents a general construction that uses any keyed pseudorandom function (PRF), and proves the security of his construction in the new model. The approach espoused by the construction is called *extract-then-expand*. First, an *extract*or is applied to Z_0, to produce a fixed-length pseudorandom key. Second, the extracted key is *expand*ed into keying material of the desired length. Figure 4 describes the approach.

The HKDF scheme is a concrete instantiation of this general construction when HMAC is used for both extraction and expansion. To use HMAC as an extractor, the *salt* value is used as a key, and if no salt value is given, a constant value is used instead.

Krawczyk's security definition [23, Definition 7] for KDFs is a single-user definition, because there is only one Z_0 value under attack. Taking key agreement as an example, there may be in reality many instances of the protocol under observation (or recorded) by the adversary, who should be considered successful if he can attack

Let XTR be an extractor and PRF^* be a variable-length pseudorandom function. Let L be the desired length of the output K. There are two optional values: *ContextInfo* is some (possibly public) context information, and *salt* is a random value (also possibly public). The value Z_0 is the *initial key material*, from which a key will be derived.

1. Compute $Z_1 \leftarrow \mathsf{XTR}(Z_0, [salt])$.
2. Compute $K \leftarrow \mathsf{PRF}^*(Z_1, ContextInfo, L)$

A common way to construct a variable length PRF from a fixed-length PRF is as follows: $\mathsf{PRF}^*(Z_1, input) = \mathsf{PRF}(Z_1, input\|0)\|\mathsf{PRF}(Z_1, input\|1)\ldots$. Krawczyk recommends a "chained" variant, i.e., the input to form the i-th block of the output includes the $(i-1)$-th block.

Figure 4: A generic extract-then-expand KDF.

any single instance. KDF security should also be modelled in the multi-user setting.

Standardized Extract-then-Expand KDFs Fifteen months after the publication of [23] at CRYPTO 2010, NIST published SP 800-56C [14] standardizing extract-then-expand KDFs. Even before publication of [23], the IETF published RFC 5869 [24] specifying HKDF. The first public appearance of HKDF seems to be around June of 2009, with the publication of the IETF Internet Draft which became RFC 5869. RFC 5869 does not give guidance on parameter choices, but the NIST document gives detailed recommendations. RFC 5869 gives a list of applications which might use HKDF, including "derivation of symmetric keys from a hybrid public-key encryption scheme"[24, §4]. We now consider this use case in detail.

HKDF in HPKE Example Suppose we want to use HKDF as the KDF in an HPKE scheme. For simplicity, we'll focus on sources of key material used in ECC at the 128-bit security level. Inputs to the KDF are therefore 256-bit or larger values (refer to Step 3 in Figure 2). NIST approves the following MAC functions as extractors (i.e., XTR in Figure 4) for 128-bit security, keyed with a public or private salt value (or a constant if no salt is given).

- AES-CMAC (128-bit output)
- HMAC-SHA-1 (160-bit output)
- HMAC-SHA-224 and HMAC-SHA-512/224 (224-bit outputs)
- HMAC-SHA-256 and HMAC-SHA-512/256 (256-bit outputs)
- HMAC-SHA-384, HMAC-SHA-512 (384 and 512-bit outputs, respectively)

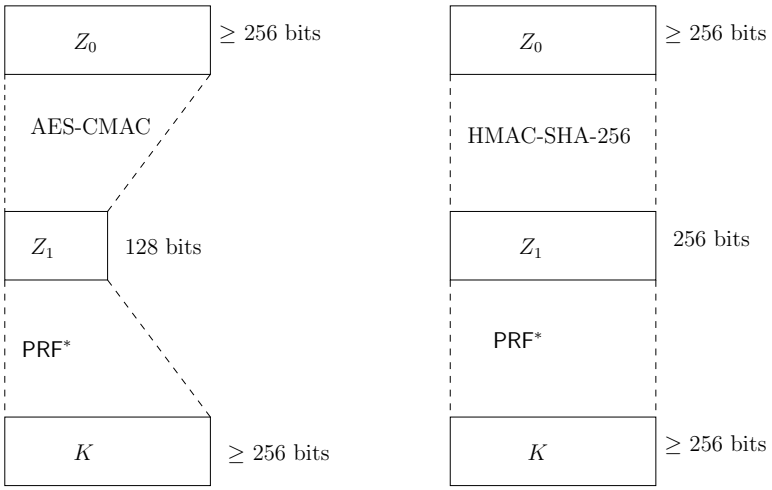

Figure 5: Extract-then-expand KDFs for ECC at the 128-bit security level. On the left the extract function is AES-CMAC and on the right it is HMAC-SHA-256. The choice of PRF^* is not important for this our example.

At the 128-bit security level a TMTO attack in the multi-user setting is possible when the output length of XTR is less than 256 bits. In this setting, the extract-then-expand construction has an hourglass shape, as shown in Figure 5.

Now consider ECIES where the KEM is implemented with a 256-bit prime curve (e.g., nistp256), and the DEM is the XOR DEM.[7] For an ℓ-bit plaintext, the XOR DEM requires that the KEM output an ℓ-bit key, which is XOR'ed with the plaintext during encryption. Common KEMs use a KDF to generate this key. With an ideal KDF, the attack from Section 3 fails against this DEM, since if we guess the first part of the keystream (encrypting the known blocks), this doesn't tell us anything about the key material used to encrypt the unknown blocks.

Now suppose the keystream for the XOR DEM is generated with HKDF-HMAC-SHA-1, so in the attack $\mathsf{DEM}.E(K, M)$ is HMAC-SHA-1$^*(Z_1) \oplus M$, where $Z_1 =$ HMAC-SHA-1(Z_0, salt) (the situation is similar to the left hand side of Figure 5, but $|Z_1| = 160$). The attack from Section 3 applies, since the attacker can make guesses at Z_1, the narrow part of the hourglass. Instead of finding a collision in the encryption key space, he searches for a collision in the PRF key space (the output of the extract function). In light of this, one can mount the same attack against ECIES

[7]The XOR DEM is specified in X9.63, IEEE 1363 and SECG SEC1 and ISO 18033-2. We discuss the XOR DEM in the context of ECIES in Section 3.1.

when the DEM is a randomized block cipher, provided the IV is generated from the shared secret using the HKDF.

Therefore, after observing 2^β ciphertexts, the scheme provides $160 - \beta$ bits of security, so when $\beta > 32$, the overall scheme fails to meet the 128-bit security level. The situation is even worse when HMAC-SHA-1 is replaced with AES-CMAC. Here the security is $128-\beta$ bits, so security is below 128 bits after as few as two ciphertexts are created. We note that this attack does not seem to apply when the KDF described at the beginning of this section is used. In this example, replacing a commonly used KDF in favor of a provably secure one causes a decrease in practical security.

The use of salt in HKDF is interesting. The salt value is used in the extract step, since every deterministic extractor fails for some high entropy distributions (this is discussed in [23]). But no salt is included as input to the expand step. In the multi-user setting, there is always a TMTO attack on deterministic PRFs (see Hellman's original paper [18], and [13] for a recent example with deterministic MACs), and it is easy to see that salting the PRF would also prevent the attack described above.

Recommendation SP 800-56C and RFC 5869 should be amended to either prevent the use of parameters leading to hourglass-shaped extract-then-expand KDFs (when the output of extract is less than twice the security parameter), or to require that the expand step be salted in these cases. Salting the expand step can be done by including a salt value in the ContextInfo field. For many applications this will happen "naturally", since the context information will have sufficient entropy. Reusing the same salt value for extraction and expansion is appealing from an implementation perspective, and seems to prevent the attack described here. However, further analysis is needed, so we recommend using different salt values for both the extract and expand steps.

5.2 Pseudorandom Generation

Let $F : \{0,1\}^\kappa \to \{0,1\}^\kappa \times \{0,1\}^{\geq 2\kappa}$. Consider the following common PRG operation.

$$\begin{array}{ccccccccc} s_0 & \to & F(s_0) & \to & s_1 & \to & F(s_1) & \to & s_2 \ldots \\ & & \downarrow & & & & \downarrow & & \\ & & r_1 & & & & r_2 & & \end{array}$$

Here the state s_i is κ bits and the outputs r_i are at least 2κ bits.

If we set κ to the desired security level, a TMTO attack allows us to attack the generator: observe a large number of outputs (from one or many users), then guess

s' and compare $F(s')$ to the observed outputs. If we have a match, the output is distinguishable from random, and we've recovered the state, allowing us to distinguish subsequent outputs. As usual, the attacker does not get to target a specific output to distinguish. Note the attack is missed by a more limited security definition, where the attacker is only trying to distinguish a *single* r_i from random. So while in a single user security definition there may be one or multiple "target" r_i values, in a multi-user definition there will certainly be multiple target values.

We give an example in the context of HPKE, again using ECIES with a 256-bit prime curve. Since we're aiming for $\kappa = 128$-bit security, the PRG construction above has a 128-bit seed and state values. Referring to the notation of Section 3.1, the KEM generates r_i, then computes $c_1 = r_i A$. An attacker does not directly see r_i, but stores c_1. He then guesses values s', computes $r'_i||s'_i \leftarrow F(s')$ and $c'_1 = r'_i A$ and compares c'_1 to the set of observed values. If there is a match, the attacker has recovered the ephemeral secret and can recover the secret key material used to derive keys for the DEM, and hence decrypt the ciphertext.

A second issue related to PRGs is the amount of entropy used for seeding. Suppose now that the state is large enough to prevent TMTO attacks, but that the entropy of the value used to initialize the PRG is still only κ bits. Further, assume all users (e.g., a set of devices produced by the same manufacturer) use the first PRG output during key generation. For an ECC key pair this is a 2κ-bit value. Let e_i be the initial value used by user i. Then the secret key of user i is $sk_i = R(e_i)$ where e_i has only κ bits of entropy, and R initializes the PRG and generates the first output. A TMTO attack is possible here as well, if an attacker observes 2^β public keys, he may learn one of the secret keys with $O(2^{\kappa-\beta})$ work and $O(2^\beta)$ storage.

Fortunately, the designers of practical PRGs are aware of these issues, and standardized PRGs ensure the state size is at least 2κ for κ-bit security, and that the PRG is seeded with more than κ-bits of entropy. The examples we have given here serve to illustrate the importance of using primitives that provide multi-user security when building HPKE schemes.[8]

Acknowledgments I thank Dan Brown and Alfred Menezes for helpful discussions, and gratefully acknowledge Dan Brown, Eoin Buckley, Matt Campagna, Nevine Ebeid, Rob Lambert and Alfred Menezes for providing feedback on an earlier draft of this paper.

[8]That said, there has been at least one proposed PRG which *is* vulnerable to the multi-user attack described above (see the instantiations of Construction 2.4 in [7]).

References

[1] M. Abdalla, M. Bellare and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES (Extended abstract) *Proceedings of Topics in Cryptology (CT-RSA'01), LNCS,* **2020** (2001). Earlier version was submitted to IEEE P1363. Full paper available at `http://cseweb.ucsd.edu/~mihir/papers/dhaes.pdf`

[2] M. Abdalla, M. Bellare and P. Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. Submission to the IEEE P1363 working group. August 1998 (Updated March 1999). Presented at the August 1998 meeting. `http://grouper.ieee.org/groups/1363/P1363a/Encryption.html#ABR`

[3] American National Standards Institute. *Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography*, ANS X9.63, 2011.

[4] American National Standards Institute. *Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography*, ANS X9.42, 2003.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to encrypt with RSA. *Proceedings of EUROCRYPT'94, LNCS* **950** (1995), 92-111. Full version and updates online at: `http://cseweb.ucsd.edu/~mihir/papers/oaep.html`.

[6] M. Bellare, A. Boldyreva and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. *Proceedings of EUROCRYPT'00, LNCS* **1807** (2000), 259–274.

[7] M. Bellare and B. Yee. Forward-Security in Private-Key Cryptography. *Proceedings of the Cryptographer's Track at the RSA Conference (CT-RSA'03) LNCS* **2612** (2003), 1–18.

[8] M. Bellare, T. Kohno and V. Shoup. Stateful Public Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation. *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, ACM Press, New York, 2005, 380–389.

[9] E. Biham. How to Forge DES-Encrypted Messages in 2^{28} Steps. Technion Department of Computer Science Technical Report Number CS0884-1996, 1996.

[10] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Proceedings of ASIACRYPT'00, LNCS* **1976** (2000), 1–13.

[11] A. Boldyreva. Strengthening Security of RSA-OAEP. *Proceedings of CT-RSA 2009, LNCS* **5473** (2009), 399–413.

[12] X. Boyen and L. Martin. Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems. IETF Request for Comments, RFC 5091. Available online `http://tools.ietf.org/html/rfc5091`

[13] S. Chatterjee, A. Menezes and P. Sarkar. Another Look at Tightness. *Proceedings of Selected Areas in Cryptography (SAC'11), LNCS*. **7118** (2012), 293-319.

[14] L. Chen. NIST Special Publication 800-56C: Recommendation for Key Derivation through Extraction-then-Expansion. National Institute of Standards and Technology, November 2011. `http://csrc.nist.gov/publications/nistpubs/800-56C/SP-800-56C.pdf`

[15] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing* **33** (2003), 167-226.

[16] J. Håstad. Solving simultaneous modular equations of low degree. *SIAM Journal of Computing* **17** (1988), 336–341.

[17] S. Halevi and H. Krawczyk One-Pass HMQV and Asymmetric Key-Wrapping. *Proceedings of Public Key Cryptography (PKC'11), LNCS* **6571** (2011), 317–334.

[18] M.E. Hellman. A Cryptanalytic Time–Memory Trade-Off. *IEEE Transactions on Information Theory* **26** (1980), 401–406.

[19] J. Hong and P. Sarkar. New Applications of Time Memory Data Tradeoffs. *Proceedings of ASIACRYPT'05, LNCS* **3788** (2005), 353–372.

[20] Institute of Electrical and Electronics Engineers. *Specifications for Public-Key Cryptography*, IEEE Standard 1363-2000, Aug. 2000. See also *Amendment 1: Additional Techniques*, IEEE Standard 1363A-2004, Oct. 2004.

[21] International Standards Organization. *Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers*. ISO/IEC Standard 18033-2, 2006. Draft version available online: `http://shoup.net/iso/std6.pdf`.

[22] N. Koblitz and A. Menezes. Another Look at Security Definitions. Preprint, September 2011. Available at `http://www.anotherlook.ca`

[23] H. Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme.- *Proceedings of CRYPTO'10, LNCS* **6223** (2010), 631–648.

[24] H. Krawczyk and P. Eronen. Internet Engineering Task Force (IETF) RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). May 2010. `http://tools.ietf.org/html/rfc5869`

[25] D. McGrew. Generation of Deterministic Initialization Vectors (IVs) and Nonces. IETF Internet Draft, July 2011. Available online `http://tools.ietf.org/html/draft-mcgrew-iv-gen-00`

[26] Public-Key Cryptography Standards (PKCS) PKCS #1: RSA Cryptography Standard, version 2.1. Available online `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`

[27] Standards for Efficient Cryptography Group. *SEC 1: Elliptic Curve Cryptography*, Mar. 2009. Version 2.0. `http://www.secg.org/download/aid-780/sec1-v2.pdf`.

A Standards

The attack of Section 3 applies to the following standards with time and space $O(2^{\kappa/2})$ (or one of the allowable tradeoffs, as discussed in Section 3).

A.1 Vulnerable Standards

IEEE 1363 [20] 1363a (amendment specifying additional techniques) specifies various DHIES-like[9] hybrid encryption schemes, and includes ECIES. Section 14.3 specifies the symmetric key algorithms that may be used as a DEM: 3DES-CBC-IV0 and AES-CBC-IV0. Both specify a constant IV, and so the techniques in this standard are vulnerable to the attack of §3.

[9]DHIES stands for *Diffie-Hellman Integrated Encryption Scheme*, see [1].

ISO 18033-2 [21] specifies multiple hybrid PKE schemes, and specifies that the IV used in the symmetric component should be a string of zero bytes (See the description of SC1 in §6.5.3, and the commentary in §B.3.1). The XOR symmetric encryption (SC2) mode is not vulnerable. The attack applies equally to any of the KEMs specified in this standard, including the RSA-based methods.

NIST SP 800-56C , *Recommendation for Key Derivation through Extraction-then-Expansion* specifies HKDF implemented with the SHA family of hash functions, and variants allowing the use of AES-CMAC as an extractor and PRF. As detailed in §5.1, the allowed parameters do not prevent a TMTO attack.

SECG SEC 1 specifies the Elliptic Curve Integrated Encryption Scheme (ECIES). In SEC 1, §B.2.8 allows the IV or initial counter to be zero for ECIES, and in Section 3.8 it recommends that all IVs should be zero. The XOR symmetric encryption mode is not vulnerable to this attack.

A.2 Possibly Affected Standards

Many PKE standards specify a KEM only (sometimes called a *key transport*), and leave applications to specify a DEM, if required. The following standards should use a randomized DEM, to avoid being vulnerable to the attack of §3.

RFC 5091 [12] This standard describes how to use the BF (Boneh-Franklin) and BB1 (Boneh-Boyen 1) IBE (identity-based encryption) schemes. The standard only describes how to encrypt a session key (i.e., a KEM), but the attack is possible if a deterministic DEM is used (e.g., if one of the DEMs from SEC 1, IEEE 1363a, or ISO 18033-2 is used with the KEM described in RFC 5091).

PKCS#1 [26] Like RFC 5091, the PKCS#1 standard only specifies a KEM, based on RSA. The standard states "For typical applications, the message to be encrypted is short (e.g., a 128-bit symmetric key)", confirming the intended use of KEM/DEM constructions. There are two variants, RSAES-PKCS1-v1_5, and RSAES-OAEP. If implementations use, e.g., RSAES-OAEP to encrypt a κ-bit symmetric key k, then use k with a deterministic DEM, then the key collision attack requires $O(2^{\kappa/2})$ work.

RFC 5869 specifies HKDF, an extract-then-expand KDF where HMAC is used in both the extract and expand functions. Since RFC 5869 does not give recommendations for choosing a suitable hash function at common security levels, use of RFC

5869 may or may not be vulnerable. As detailed in § 5.1, some parameter choices given in SP 800-56C (which are also applicable to RFC 5869) are unsuitable for use in the HPKE setting.

A.3 Other hybrid encryption proposals

The KEM/DEM paradigm is very popular, and PKE schemes based on it abound. These should be implemented with caution. A recent example: if the HOMQV [17] scheme is implemented with a deterministic DEM, the attack from §3 applies.

S/MIME and CMS The S/MIME (secure multipart message encoding) and CMS (RFC 5652, cryptographic message syntax) standards specify (highly flexible) ways of encrypting content for one or multiple recipients. The CMS standards follow the KEM/DEM paradigm, where the KEM is called "key encryption" and the DEM is called "content encryption". Use of deterministic DEMs may allow the attack we describe. For example, using RSA-KEM (RFC 5990) with a deterministic DEM (AES-CBC with a fixed IV), is vulnerable.

ANSI X9.63 The lone PKE standard we found which resists the attack completely is ANSI X9.63. This standard only specifies a single DEM, namely the "XOR encryption" DEM. The attack does not appear to apply in this case, because the key is as long as the message, and is derived from a 2κ-bit Diffie-Hellman value using a secure KDF.