More on Key Wrapping

Rosario Gennaro and Shai Halevi

IBM T.J. Watson Research Center
Hawthorne, NY 10532, USA
rosario@us.ibm.com, shaih@alum.mit.edu

Abstract. We address the practice of key-wrapping, where one symmetric cryptographic key is used to encrypt another. This practice is used extensively in key-management architectures, often to create an "adapter layer" between incompatible legacy systems. Although in principle any secure encryption scheme can be used for key wrapping, practical constraints (which are commonplace when dealing with legacy systems) may severely limit the possible implementations, sometimes to the point of ruling out any "secure general-purpose encryption." It is therefore desirable to identify the security requirements that are "really needed" for the key-wrapping application, and have a large variety of implementations that satisfy these requirements.

This approach was developed in a work by Rogaway and Shrimpton at EUROCRYPT 2006. They focused on allowing deterministic encryption, and defined a notion of *deterministic authenticated encryption* (DAE), which roughly formalizes "the strongest security that one can get without randomness." Although DAE is weaker than full blown authenticated encryption, it seems to suffice for the case of key wrapping (since keys are random and therefore the encryption itself can be deterministic). Rogaway and Shrimpton also described a mode of operation for block ciphers (called SIV) that realizes this notion.

We continue in the direction initiated by Rogaway and Shirmpton. We first observe that the notion of DAE still rules out many practical and "seemingly secure" implementations. We thus look for even weaker notions of security that may still suffice. Specifically we consider notions that mirror the usual security requirements for symmetric encryption, except that the inputs to be encrypted are random rather than adversarially chosen. These notions are all strictly weaker than DAE, yet we argue that they suffice for most applications of key wrapping.

As for implementations, we consider the key-wrapping notion that mirrors authenticated encryption, and investigate a template of Hash-then-Encrypt (HtE), which seems practically appealing: In this method the key is first "hashed" into a short nonce, and then the nonce and key are encrypted using some standard encryption mode. We consider a wide array of "hash functions", ranging from a simple XOR to collision-resistant hashing, and examine what "hash function" can be used with what encryption mode.

Keywords: Deterministic Encryption, Key Wrapping, Modes of Operation, Symmetric Encryption.

M.J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini (Eds.): SAC 2009, LNCS 5867, pp. 53–70, 2009.

1 Introduction

Key-wrapping roughly refers to encrypting one cryptographic key with another. In this paper we focus on the use of this practice for symmetric encryption, where the main application is key-management. Key-management architectures often include a hierarchy (or tree) of keys, with a master key encrypting several lower keys, which in turn encrypt even lower keys, and with the leaf keys used to encrypt "the real data" (cf. [17, Chapter 6]). Another typical case where key-wrapping is used is retrofitting an encryption system to work with an incompatible key-management architecture, for example an AES encryption system with a 3DES key-management. In such cases, one can add a "glue layer" in between the encryption system and the key management architecture, that generates *data keys* as expected by the encryption system (e.g., AES keys) and uses the keys from the key-management architecture to wrap these data keys (e.g., using 3DES).

A similar situation arises when the encryption system must use its keys in a restricted manner, but the key-management architecture is not designed to keep track of these restrictions. For example, one system that we encountered was using the GCM encryption mode, and needed to comply with the following requirement from the NIST standard for GCM [9]:

> *The total number of invocations of the authenticated encryption function shall not exceed* 2^{32}*, including all IV lengths and all instances of the authenticated encryption function with the given key.*

However, that system was using a key-management architecture that did not keep track of the number of times that any single key is being served, and hence was not able to certify that the requirement from above is being met. Here too, the solution was to add a key-wrapping adapter layer that generates a new GCM key every time and wraps it with the given key from key-management.

1.1 What Is a Secure Key-Wrapping?

It is clear that any secure encryption scheme is in particular also a secure key-wrapping scheme. But using secure encryption may be an overkill for the application to key-wrapping. In particular, the usage of key-wrapping as an adapter between legacy systems sometimes imply severe practical limitations on its implementation, perhaps to the point of excluding general-purpose secure encryption. We therefore seek weaker notions of security that can be implemented even in cases where standard secure encryption is impossible, but are still strong enough for the purpose of key-wrapping.

This approach was taken by Rogaway and Shrimpton in [22], where they focused on allowing deterministic procedures. Specifically, they investigated *deterministic authenticated encryption* (DAE), which roughly formalizes "the strongest security that one can get from deterministic procedures." Although achieving less than standard authenticated encryption, DAE appears to be sufficient for key-wrapping: since the key itself is already random, it seems that randomness is not

really needed in the encryption procedure itself. Indeed, Rogaway and Shrimpton included in the full version of their work an appendix in which they prove that DAE is good enough for applications that encrypt high entropy plaintext. (See more discussion in our Appendix A.)

However, even DAE may sometimes be too much to ask for. In this paper we show several examples of practical "seemingly secure" schemes that nevertheless fail to meet the notion of DAE. We thus aim lower, looking for even weaker notions that still suffice for key wrapping. Noting that the difference between key wrapping and general-purpose encryption is that the plaintext to be encrypted is a symmetric cryptographic key (and therefore is random), we consider notions of security that mirror the usual notions for symmetric encryption, except that the attack model postulates that the plaintext to be encrypted is random rather than adversarially controlled. Specifically, in Section 2 we present notions that mirror CPA-security, CCA-security, and integrity of ciphertext. We argue that these notions suffice for many application of key-wrapping. We also prove formally that they suffice for the typical application in which a master-key is used to encrypt data keys, which themselves are used to encrypt real data.

1.2 How to Achieve Secure Key-Wrapping?

Implementing secure key wrapping can be done using standard secure symmetric encryption, perhaps using generic composition techniques such as encrypt-then-authenticate [5, 6, 14] (which work for key-wrap just as well as for regular encryption). Another solution was given by Rogaway and Shrimpton [22], who designed a mode of operation called SIV that they prove to meet the stronger notion of DAE. However, applications of key wrapping sometimes place restrictions on the implementation. (For example, being deterministic, or using a specific encryption mode because that mode is already implemented in hardware, etc.) The thrust of this paper is therefore to examine many different plausible constructions, trying to separate secure constructions from insecure ones.

We focus specifically on an approach for achieving authenticated key-wrap that we call *Hash-then-Encrypt* (HtE). In this method, the key is first "hashed" into a short nonce, and then the nonce and key are encrypted using some standard encryption mode. There are several reasons to look at this approach: First, we may be able to get away with using a very simple "hash function" (maybe as simple as just XOR), which could be very efficient. Perhaps more importantly, this template could allow re-use of components that is already implemented in existing systems.

In this work we consider a wide array of "hash functions", and examine what "hash function" can be used with what encryption mode. We show that all the modes that we considered can be turned into a secure authenticated key-wrapping scheme by using a second-preimage-resistant function for the hash function, and many of them (except ECB and maybe CBC) can also use universal hashing. But resisting collisions is not really necessary in most cases. We show that for all modes except CTR, a simple fixed linear function is already enough to get some level of security (but this level of security deteriorates quickly with the

Table 1. Security of various Hash-then-Encrypt constructions

Encryption/Hash	XOR	Linear	2nd-preimage resistant	universal hashing
CTR	broken	broken	secure	secure
ECB	broken	somewhat*	secure	broken
CBC	broken	somewhat*	secure	?
masked ECB/CBC	somewhat*	somewhat*	secure	secure
XEX	secure	secure	secure	secure

*"somewhat" means concrete security that is worse than the birthday bound

length of the data key), and when using masked versions of ECB and CBC then even a simple XOR of the key blocks suffices to get the same level of security. Finally, when using a tweakable encryption mode [16] such as XEX [20], a simple XOR suffices to get security upto the birthday bound, regardless of the length of the data keys.[1] These results are summarized in Table 1.

1.3 Related Work

We already discussed the work of Rogaway and Shrimpton [22] on the key-wrap problem. A somewhat similar definition to DAE was later formulated by Amanatidis et al. in the context of searchable encryption [2]. An and Bellare studied authentication via redundancy in the context of standard symmetric encryption [3]. They argued that public redundancy function is not very useful for achieving authenticated encryption (although work by Gligor and Donescu [10], Jutla [12], and Rogaway [19] demonstrated that simple public redundancy is sufficient when using masked CBC or masked ECB for encryption). In our case we show that even very simple public redundancy is sufficient in most cases. Also Bellare and Namprempre [5] and Krawczyk [14] deal with generic composition techniques of encryption and authentication, and some further results were described by Canetti et al. [6].

Other related work was done in the area of KEM/DEM schemes for public key encryption, where different conditions on the KEM and/or DEM parts were investigated (e.g., [1, 7, 15]). Also, some recent work addressed public-key deterministic encryption [4]. Finally, we mention that encryption of "random messages" was also considered in the very different context of "Entropic security" by Russell and Wang [23] and Dodis and Smith [8]: in these works they attempted to provide statistical security for random messages using as little key material per message as possible.

Organization. Due to space limitations, some of the results and proofs were deferred to the final version.

[1] Observe that the masked modes and XEX are obtained by adding very simple masking to ECB and CBC modes, so it makes sense to talk about them here, even though our paper is focused on dealing with legacy systems.

2 Defining Security for Key Wrapping

Below we adapt the usual notions of security for symmetric encryption to the case of key-wrapping. The only difference between our notions and the standard ones is that the plaintext is chosen at random rather than being controlled by the attacker. We focus on the simplest case of fixed input length and no associated data, extensions and variations are discussed in the long version.

Syntactically, a key-wrapping scheme is identical to an encryption scheme. Namely, it includes a wrapping procedure Wrap that takes plaintext and wrapping-key and returns ciphertext, and an unwrapping procedure Unwrap that takes ciphertext and wrapping key and returns the plaintext (or an error symbol $\perp$). We have the usual validity condition, asserting that for any wrapping key K and plaintext D it holds that $\mathsf{Unwrap}_K(\mathsf{Wrap}_K(\mathsf{D})) = \mathsf{D}$. Below we usually refer to the plaintext as a *data key*. We insist that wrapping keys (as well as data keys) are uniformly random bit strings of some given length. Hence key-generation is implicitly specified as choosing a random key of the appropriate length. We denote the length of the wrapping key by k, and the length of the plaintext/data-keys by ℓ. (One can think of k as the security parameter and ℓ is typically also a parameter.)

2.1 Security for Key-Wrap

Let $\mathcal{KW} = (\mathsf{Wrap}, \mathsf{Unwrap})$ be a key-wrapping scheme. All the security definitions are based on probabilistic games, involving an attacker A and the procedures Wrap and Unwrap. Our definitions use the "left-or-right" style. The basic game is the $\mathsf{Random\text{-}Plaintext\ Attack}$ (RPA), mirroring the usual chosen-plaintext attack: First a wrapping key W is chosen uniformly at random in $\{0,1\}^k$, together with random "challenge bit" b. Then the attacker interacts with the wrapping procedure as follows: whenever A invokes the wrapping procedure, two data keys $\mathsf{D}_0, \mathsf{D}_1$ are chosen uniformly at random in $\{0,1\}^\ell$, and A receives both D_0, D_1, and also the ciphertext $C = \mathsf{Wrap}_{\mathsf{W}}(\mathsf{D}_b)$. The attacker A can keep making such queries, and eventually it halts and outputs a guess for the value of the challenge bit b. The *RPA-advantage* of A is defined as

$$\mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathcal{KW}}(A) \stackrel{\text{def}}{=} \Pr[A^{LR\$\mathsf{Wrap}_{\mathsf{W}}} \Rightarrow 1 | b = 1] - \Pr[A^{LR\$\mathsf{Wrap}_{\mathsf{W}}} \Rightarrow 1 | b = 0] \quad (1)$$

where $LR\$\mathsf{Wrap}$ is the "left-or-right" procedure described above, $A \Rightarrow 1$ is the event where A outputs the bit '1', and the probability is taken over all the probabilistic choices in this game.

The *Chosen-Ciphertext Attack* (CCA) game is similar, except that the attacker is also given access to the unwrapping procedure that on query C returns $\mathsf{Unwrap}_{\mathsf{W}}(C)$, but the attacker is prevented from querying it on ciphertexts that were previously returned from the procedure Wrap. Then the *CCA-advantage* of A is defined as

$$\mathbf{Adv}^{\mathsf{kw.cca}}_{\mathcal{KW}}(A) \stackrel{\text{def}}{=} \Pr[A^{LR\$\mathsf{Wrap}_{\mathsf{W}},\mathsf{Unwrap}_{\mathsf{W}}} \Rightarrow 1 | b = 1] - \Pr[A^{LR\$\mathsf{Wrap}_{\mathsf{W}},\mathsf{Unwrap}_{\mathsf{W}}} \Rightarrow 1 | b = 0] \quad (2)$$

The *integrity of ciphertext* game (INT) is defined similarly to the RPA game, except that wrapping queries return only one random data key D and the corresponding ciphertext $C = \mathsf{Wrap}_{\mathsf{W}}(\mathsf{D})$, and the attacker's goal is to output any ciphertext C^*, different than the ones that were returned from the Wrap procedure, that the unwrap procedure does not reject. Namely, the advantage of A is defined as

$$\mathbf{Adv}^{\mathsf{kw.int}}_{\mathcal{KW}}(A) \stackrel{\text{def}}{=} \Pr[A^{\$\mathsf{Wrap}} \Rightarrow C^* : \ C^* \text{ is "new" and } \mathsf{Unwrap}(C^*) \neq \perp] \quad (3)$$

where $\$\mathsf{Wrap}$ is the procedure above that returns both a random data key and its encryption.

As usual, we extend the advantage notations to talk about the advantage of "any attacker within the given limited resources". For example, $\mathbf{Adv}(\text{enc} = q)$ means any attacker that makes at most q queries to its encryption oracle. We will explicitly specify the relevant resources whenever we use this convention. We informally say that a scheme is "secure" when the advantage of an attacker is no more than the birthday bound (i.e., $O(q^2/2^n)$ where q is the resource bound and n is a relevant security parameter). We say that a scheme is "somewhat secure" where the advantage is exponentially small in n but larger than the birthday bound, and otherwise we say that the scheme is "broken." Clearly, RPA-security captures the notion of secrecy for the key against eavesdroppers, CCA-security ensures key secrecy also against active attackers, and the last notion adds explicit authentication.

Below we refer to a scheme which is both RPA-secure and has integrity of ciphertext as *authenticated key-wrap*. Just as for encryption [5, 13], an easy argument shows that RPA-security and integrity-of-ciphertext imply CCA-security. It is also easy to see that requiring both is strictly stronger than requiring CCA-security (e.g., a random permutation is CCA-secure but does not provide integrity of ciphertext).

In the long version we discuss some extensions of these definitions, e.g., to handle variable-input-length, associated data, etc.

2.2 Key-Wrapping Is Weaker Than DAE

We note that the security notions from above are all strictly weaker than the notion of deterministic authenticated encryption (DAE) of Rogaway and Shrimpton [22]: DAE requires that an attacker that interacts with the Wrap and Unwrap procedures (with some fixed random secret key W) cannot distinguish them from a dummy Wrap that returns only random bits, and a dummy Unwrap that rejects any "new" ciphertext (i.e., any ciphertext that was not returned by the Wrap procedure). Obviously, when interacting with the dummy procedures we have $\mathbf{Adv}^{\mathsf{kw.rpa}}(A) = \mathbf{Adv}^{\mathsf{kw.int}}(A) = 0$ (and therefore also $\mathbf{Adv}^{\mathsf{kw.cca}}(A) = 0$), so DAE implies all of our notions. In fact, Rogaway and Shrimpton proved in an appendix of the long version of [22] that DAE implies a similar (but stronger) notion of security, for a case where part of the plaintext is random and another part is chosen by the attacker. See discussion in our Appendix A.

On the other hand, in the DAE game the attacker can query the wrapping procedure on inputs of its choice, so it is easy to find examples of schemes that satisfy our notions but are not DAE. (In fact, some of our "provably secure" constructions from Section 3 below fail to meet the notion of DAE.) One easy example is a wrapping procedure based on a block cipher, that wraps a one-block data-key D using the wrapping key W by setting $C = \langle E_{\mathsf{W}}(\mathsf{D}), E_{\mathsf{W}}(\mathsf{D}+1)\rangle$. It is clear that in all the games as described above, an attacker making at most q queries to the wrapping procedure has advantage at most $O(q^2/2^n)$, since the inputs to the block ciphers will be all disjoint except with that probability. On the other hand, a DAE attacker that can specify the data keys only needs to encrypt two keys $\mathsf{D}_0, \mathsf{D}_1$ such that $\mathsf{D}_1 = \mathsf{D}_0 + 1$ and check that the first block in C_1 is the same as the second block in C_0.

2.3 Key-Wrapping Is Sufficient for Applications

Although weaker than DAE, we claim that our notions are sufficiently strong for most application of key wrapping. That is, for any application that uses a key-wrapping procedure to wrap random keys, it is sufficient for the key-wrapping to satisfy the notions that we defined above. In some sense this statement is true by definition: our notions ensure that an attacker cannot distinguish the "real keys" from random unrelated keys, which means that even after seeing (and perhaps even manipulating) the wrapped keys, they are still just random secret keys from the attacker's perspective. Below we demonstrate formally that secure key-wrapping is sufficient to get secure symmetric encryption.

Specifically, let $\mathcal{KW} = (\mathsf{Wrap}, \mathsf{Unwrap})$ be a key-wrapping scheme and let $\mathcal{SE} = (\mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme. Consider the composite symmetric encryption scheme $\mathcal{C}$, whose key space is that of $\mathcal{KW}$ and whose message space is that of $\mathcal{SE}$. On a given key W and plaintext message M, the composite encryption chooses a new random data-key D from the key-space of $\mathcal{SE}$, wraps it using W to get $C_1 \leftarrow \mathsf{Wrap}_{\mathsf{W}}(\mathsf{D})$, uses it to encrypt the message, getting $C_2 \leftarrow \mathsf{Enc}_{\mathsf{D}}(M)$, and outputs the composite ciphertext $C = (C_1, C_2)$. The composite decryption first recovers $\mathsf{D} \leftarrow \mathsf{Unwrap}_{\mathsf{W}}(C_1)$ and if $\mathsf{D} \neq \perp$ then computes and returns $M \leftarrow \mathsf{Dec}_{\mathsf{D}}(C_2)$.

The following lemma asserts that if $\mathcal{KW}$ and $\mathcal{SE}$ are secure then so is $\mathcal{C}$. More specifically, if $\mathcal{KW}$ is RPA-secure and $\mathcal{SE}$ is CPA-secure then the composite is CPA-secure, if both are CCA-secure then so is the composite, and if both have integrity of ciphertext and the key-wrapping is RPA-secure then the composite also has integrity of ciphertext. One point to note is that since the application uses each data key only once, then the underlying encryption $\mathcal{SE}$ need only be secure under encryption of a single message.

Lemma 1. *Let q, q' be bounds on the number of encryption/wrapping queries and the number of decryption/unwrapping, respectively. Then*

$$\mathbf{Adv}^{\mathsf{enc.cpa}}_{\mathcal{C}}(enc = q) \leq \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathcal{KW}}(wrap = q) \;+\; q \cdot \mathbf{Adv}^{\mathsf{enc.cpa}}_{\mathcal{SE}}(enc = 1),$$

$$\mathbf{Adv}^{\mathsf{enc.cca}}_{\mathcal{C}}(enc = q, dec = q') \leq$$

$$\mathbf{Adv}^{\mathsf{kw.cca}}_{\mathcal{KW}}(wrap = q, unwrap = q') \ + \ q \cdot \mathbf{Adv}^{\mathsf{enc.cca}}_{\mathcal{SE}}(enc = 1, dec = q'),$$

$$\mathbf{Adv}^{\mathsf{enc.int}}_{\mathcal{C}}(enc = q) \leq$$
$$\mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathcal{KW}}(wrap = q) + \mathbf{Adv}^{\mathsf{kw.int}}_{\mathcal{KW}}(wrap = q) \ + \ q \cdot \mathbf{Adv}^{\mathsf{enc.int}}_{\mathcal{SE}}(enc = 1).$$

The running-time bounds on the various attackers that are hidden in the expressions above are all about equal: they differ by at most the time that it takes to compute q encryptions/wrappings and q' decryptions/unwrappings.

3 Authenticated Key-Wrap

As we said before, in principle one can use any authenticated encryption scheme to achieve authenticated key-wrap. Another "clean solution" for obtaining authenticated key-wrap is *wrap-then-authenticate*, where one first employs any RPA-secure scheme for wrapping and then authenticates the ciphertext with any secure MAC. As for encryption [5, 6, 14], here too one gets RCCA-security from any MAC and authenticated key-wrap when the MAC is "strongly unforgeable".[2]

Yet another option is to use the carefully-engineered SIV mode of Rogaway and Shrimpton [22]: In SIV the data key D (and associated data A) are first fed into a pseudorandom function to get a nonce, $N = PRF_w(\mathsf{D}, \mathsf{A})$, and then the data key is encrypted with an IV-based encryption scheme (such as CTR mode or CBC mode, with a key which is independent of the PRF key). Shrimpton and Rogaway proved that SIV realizes their notion of DAE (and therefore is also an authenticated key-wrap) for any PRF and any "pseudorandom IV-based encryption."[3] They suggested implementing the pseudorandom function using a variant of CBC-MAC, and the encryption using CTR mode.

But as we argued in the introduction, there are still cases where one may want to use other implementation strategies. Below we analyze a wide range of solutions that may be appealing in practice, with a goal to determine what works and what doesn't.

3.1 Simplified SIV May Not Work

We remind the reader that the main difference between Rogaway and Shrimpton's notion of DAE and our notions of security is that the attacker in their model can choose the plaintext, whereas in our model the data-key is always chosen at random. One could therefore hope that we can get a secure scheme even if we weaken SIV by replacing the pseudorandom function with a "weak pseudorandom function." (Recall that a weak pseudorandom function [18] is a

[2] A MAC is "strongly unforgeable" if the attacker cannot even produce a new valid authentication tag for a previously-authenticated message.

[3] A "pseudorandom IV-based encryption" is one where the ciphertext in a chosen-plaintext attack is indistinguishable from random. Shrimpton and Rogaway called this notion "conventional IV-based encryption".

function F, such that no attacker can distinguish $F(x)$ from random as long as the points x themselves are chosen at random.)

Unfortunately, this intuition fails: For example, for an n-bit block cipher E and $2n$-bit data keys K_1, K_2, it is easy to see that the function $F_w(K_1, K_2) = E_w(K_1 \oplus K_2)$ is a weak pseudorandom function (upto the birthday bound). But implementing a key-wrap using this function and CTR mode is completely broken: an attacker that sees the ciphertext C_0, C_1, C_2 that corresponds to the key K_1, K_2 can trivially obtain a ciphertext for related keys simply by XORing the same non-zero block Δ into both C_1 and C_2, which would be a valid ciphertext for the key $K_1 \oplus \Delta, K_2 \oplus \Delta$. Similarly, using this function F with CBC encryption is insecure, since if C_0, C_1, C_2, C_3 is a valid ciphertext for the key K_1, K_2, K_3, then C_0, C_2, C_1, C_3 is a valid ciphertext for the key $K_1, K_2 \oplus C_1 \oplus C_2, K_3 \oplus C_1 \oplus C_2$.

3.2 Hash-then-Encrypt

Next we examine the solution template of "Hash-then-Encrypt" (HtE). That is, the data-key K to be wrapped is first compressed into a one-block nonce using some "hash functions", $N = H(K)$, and then the nonce and key together are encrypted using a standard encryption mode. Here we consider using CTR mode, ECB, and CBC. (In the long version we also explore the masked variants ECB-X and CBC-X, and "narrow block tweakable modes" [16] such as Rogaway's XEX [20].) Hash-then-Encrypt is similar to SIV when one thinks of $E(H(K))$ as the pseudorandom function of SIV. However, below we also consider weak versions of H for which $E(H(K))$ is not a PRF.[4]

For any "hash function" H, given a wrapping key W (that includes a cipher key w) and a data key $K = \langle K[1], \ldots, K[\ell] \rangle$ (where each $K[i]$ is an n-bit block), compute $N = H(K)$ and $C[0] = E_w(N)$ and then for $i = 1, \ldots, \ell$ set:

HtCTR. $C[i] = K[i] \oplus E_w(C[0] + i - 1)$ where the addition is modulo 2^n (say).
HtECB. $C[i] = E_w(K[i])$.
HtCBC. $C[i] = E_w(C[i-1] \oplus K[i])$.
HtECB-X. $C[i] = E_w(X[i] \oplus K[i]) \oplus X[i]$, where the $X[i]$'s are "XOR universal" and derived from a different part of the wrapping key W.
HtCBC-X. $C[i] = E_w(C[i-1] \oplus K[i]) \oplus X[i]$, where the $X[i]$'s are "XOR universal" and derived from a different part of the wrapping key W.
HtXEX. $C[i] = E_w(X[i] \oplus K[i]) \oplus X[i]$, where the $X[i]$'s are computed as $X_i \leftarrow \alpha^{i-1} \cdot E_w(C[0])$, with α a primitive element of $GF(2^n)$.

The modes HtCTR, HtECB, and HtCBC are depicted in Figures 1, 2, and 3, respectively. For the "hashing" part we analyze several different functions, both keyed and un-keyed. For each encryption mode we seek sufficient and/or necessary conditions on the hash function to get authenticated key-wrap.

[4] Another technical difference is that in our case the key used by E in the "PRF part" is the same as the key used by E in the "encryption part."

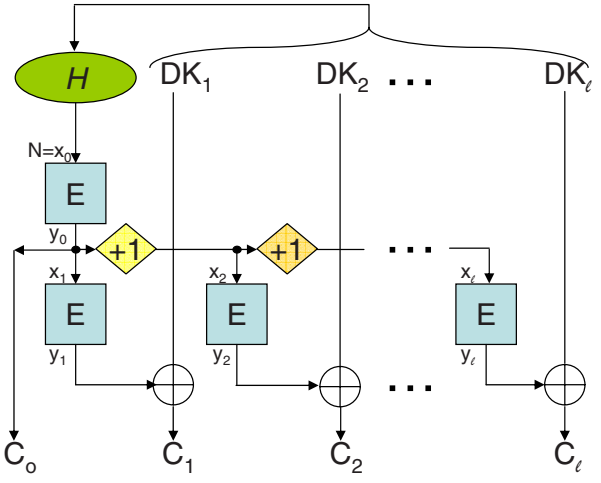

Fig. 1. HtCTR

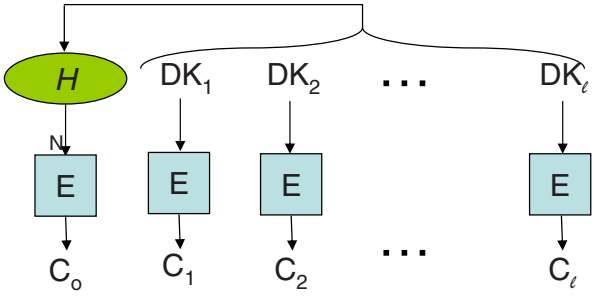

Fig. 2. HtECB

3.3 Hash-then-CTR

When using counter-mode encryption, it turns out that a necessary and sufficient condition on the hash function H is that it resists second-preimage collisions. However, we point out that in our case, the function H is allowed to have a secret key and moreover the attacker does not get to see the hash value, so it is easier to get second preimage resistance than in the usual settings where H is public. (In particular, any universal hash function is second preimage resistant in our setting.) The definition below formalizes the notion of second-preimage-resistance that we use:

Resisting second-preimage collisions. Let H be a function that can depend on a secret key and/or on public parameters. The definition below is formalized for the case of fixed input length, so we have a parameter ℓ that denotes the input length of H. We also have parameters n, k', k'' denoting the output length

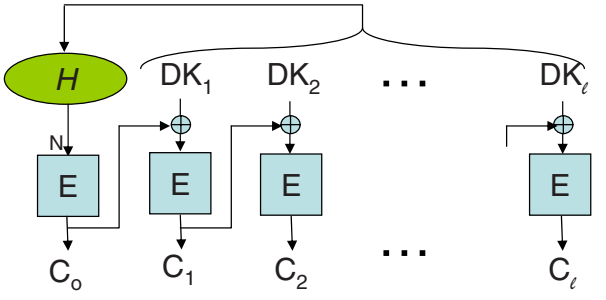

Fig. 3. HtCBC

and the lengths of the secret key and public parameters (if any). The attack scenario that we consider is where the secret key, public parameters, and the first preimage are chosen at random, $\mathsf{sk} \in_R \{0,1\}^{k'}$ $\mathsf{pp} \in_R \{0,1\}^{k''}$, $X \in_R \{0,1\}^{\ell}$, the attacker is given the public parameters and the first preimage, and its goal is to find a second preimage that collides with the first under H. We denote the second-preimage-advantage of an attacker A by

$$\mathbf{Adv}^{\mathsf{spr}}_{H}(A) \stackrel{\text{def}}{=} \Pr_{\mathsf{sk},\mathsf{pp},X}[A(\mathsf{pp},X) \Rightarrow X' \ : \ X' \neq X \text{ and } H(\mathsf{sk},\mathsf{pp},X) = H(\mathsf{sk},\mathsf{pp},X')] \quad (4)$$

The case where the secret key is empty corresponds to the usual notion of second preimage resistance for public hash functions (of function families). Below we also denote by α_H the probability that two random inputs collide under a random key, namely $\alpha_H \stackrel{\text{def}}{=} \Pr_{\mathsf{sk},\mathsf{pp},X,X'}[H(\mathsf{sk},\mathsf{pp},X) = H(\mathsf{sk},\mathsf{pp},X')]$. (Clearly $\alpha_H \leq \mathbf{Adv}^{\mathsf{spr}}_{H}$, but α_H could sometimes be much smaller.)

Next we show that the HtCTR construction is secure in the sense of authenticated key-wrap if and only if the hash function H is second-preimage resistant according to the notion above.

Lemma 2. *Let H be a (potentially keyed) hash function as above with input length ℓn and output length n, and consider the HtCTR construction using H for the hash function and with a truly random permutation for the block cipher. Then for any bound q on the number of wrapping queries, we have*

$$\mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtCTR}}(\mathit{wrap} = q) \leq 2\binom{q}{2}\alpha_H + O(q^2\ell/2^n),$$

$$\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCTR}}(\mathit{wrap} = q) \leq q\mathbf{Adv}^{\mathsf{spr}}_{H} + \binom{q}{2}\alpha_H + O(q^2\ell^2/2^n),$$

$$\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCTR}}(\mathit{wrap} = 1) \geq \mathbf{Adv}^{\mathsf{spr}}_{H}$$

The running-time bounds on the various attackers that are hidden in the expressions above are all about equal: they differ by at most the time that it takes to compute q wrappings.

Some constructions that are likely to meet the second-preimage resistant condition that is needed in Lemma 2 include most of the known cryptographic hash functions such as SHA1 or SHA256. Observe that when used with AES as the underlying cipher for encryption, we are limited to using only 128 bits of the output of the hash function. But second-preimage resistance is a very weak requirement, so it is likely that the SHA family meet this notion even if we only take 128 bits of output. Another solution would be to key these functions (e.g., using HMAC).

Another class of practical functions that meet this condition are universal hash functions (e.g., the polynomial-evaluation hash, or linear hash functions). These functions can be proven to meet the condition of second-preimage resistance as formulated in Eq. (4), but we stress that they only meet it if the hashing key is kept secret (as part of the key-wrapping key).

When the data-key K is a key for a block cipher E, it may even be plausible to use $N = E_K(\mathsf{const})$ as a checksum, where const is some public constant. For contemporary ciphers like AES, it may be reasonable to assume that the public function $H(K) = AES_K(\mathsf{const})$ is a second-preimage resistant function.

3.4 Hash-then-ECB and Hash-then-CBC

At first glance, one may suspect that the hash-then-encrypt method cannot be used with ECB encryption, since in ECB the hash value does not influence in any way the encryption of the data key itself. Below we show that this is not really the case, indeed ECB and CBC mode behave very similarly in our context (with one exception that is described below). For example, in Lemma 4 we prove that even a public linear function can result in an authenticated key-wrap when combined with ECB or CBC modes.

We begin with examining a composition of second-preimage resistant hashing with ECB and CBC. For both modes, we prove below that using a *public* second-preimage-resistant hashing is secure (under an additional mild structural condition). Perhaps surprisingly, however, it turns out that at least for ECB, when using a hash function that depends on a secret key, second-preimage resistance (or even universality) is not sufficient. (For CBC we still don't know if universal hashing suffices. We suspect that it is, but so far could not prove it.)

univHash-then-ECB may be insecure. We show a hash function with secret key (from $2n$ to n bits), which is second-preimage resistant and yet has the property that for any X, Y, Z, it holds that $X = H(Y, Z) \Leftrightarrow Y = H(X, Z)$, and we show how to use this property in an attack (since in ECB we use the same procedure to encrypt the nonce as we do the key blocks). Consider the following Hash-then-ECB scheme for wrapping a two-block data key: the hash function uses a block cipher E and depends on two secret cipher-keys, which we denote by h_1, h_2. Specifically, our hash function is defined as

$$H_{h_1,h_2}(Y, Z) = E_{h_1}^{-1}\left(E_{h_1}(Y) \oplus E_{h_2}(Z)\right)$$

It is not hard to see that this function H is second-preimage-resistant as per the definition from Eq. (4): if we replace the cipher with two random permutations

then we have $\mathbf{Adv}^{\mathsf{spr}}_{H} = 2^{-n}$. (In fact, H is nearly a pairwise-independent hash function in this case.) On the other hand, if $X = H_{h_1,h_2}(Y,Z)$ then

$$\begin{aligned} &E_{h_1}(H_{h_1,h_2}(X,Z)) \;= \\ &\quad = \; E_{h_1}(X) \oplus E_{h_2}(Z) \;=\; (E_{h_1}(Y) \oplus E_{h_2}(Z)) \oplus E_{h_2}(Z) \;=\; E_{h_1}(Y) \end{aligned}$$

and therefore also $H_{h_1,h_2}(X,Z) = Y$. An attacker on HtECB, after seeing a ciphertext $C = \langle C_0, C_1, C_2\rangle$ can therefore produce the valid forged ciphertext $C^* = \langle C_1, C_0, C_2\rangle$.

publicSPR-then-ECB/CBC is secure. When H is a public function, on the other hand, we show that second-preimage-resistance is sufficient, under a mild structural condition. Specifically we need to assume that for a random input data-key K, the nonce $N = H(K)$ is also (close to being) a random n-bit block. Below we call a function with that property *well-spread*.

Lemma 3. *Let H be a public well-spread hash function with input length ℓn and output length n, and consider the construction HtECB, using H for the hash function and with a truly random permutation for the block cipher. Then for any bound q on the number of wrapping queries, we have*

$$\begin{aligned} &\mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtECB}}(\mathit{wrap} = q),\ \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtCBC}}(\mathit{wrap} = q) \le \binom{q}{2}(\alpha_H + \ell^2/2^n) \\ &\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECB}}(\mathit{wrap} = q),\ \mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCBC}}(\mathit{wrap} = q) \le O(q\ell) \cdot \mathbf{Adv}^{\mathsf{spr}}_{H} \end{aligned}$$

The running-time bounds on the various attackers that are hidden in the expressions above are all about equal: they differ by at most the time that it takes to compute q wrappings.

Proof. (sketch): Below we only prove the bound on $\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECB}}$, the proof for $\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCBC}}$ is similar, and the RPA-bounds are straightforward. Denote the transcript of a q-query attack against HtECB by

$$\{(K_i, C_i) \;:\; K_i = \langle K_i[1], \ldots, K_i[\ell]\rangle,\ C_i = \langle C_i[0], C_i[1], \ldots, C_i[\ell]\rangle\}_{i\in[1,q]}$$

and let $C^* = \langle C^*[0], C^*[1], \ldots, C^*[\ell]\rangle$ be the attempted forged ciphertext. Also let $K^*[j] = E^{-1}(C^*[j])$ for $j = 0, 1, \cdots, \ell$ and $N^* = H(K^*[1], \ldots, K^*[\ell])$, so C^* is valid iff $N^* = H(K^*[0])$.

We have three types of ciphertext C^* to consider: either $C^*[0]$ is different from all the $C_i[j]$'s, or it is equal to one of the $C_i[0]$'s, or it is equal to one of the $C_i[j]$'s for $j > 0$. Denote the probability of C^* of the first type being valid by ε_1 and probability of C^* of the second type being valid by ε_2, and the probability of C^* of the third type being valid by ε_3. We show three collision-finders for H: one with success probability $\varepsilon_1 - O(q\ell/2^n)$, the second with with success probability ε_2/q, and the third with success probability $\varepsilon_3/q\ell$.

The first collision finder (that needs to work when $C^*[0] \neq C_i[j]$ for all i, j) gets a random input X and computes $N = H(X)$. (Recall that H is well spread

and public, so N is nearly uniform and we can compute it.) Now the collision finder plays the integrity-of-ciphertext game with the attacker, choosing at random values for the K_i's and for the permutation E and its inverse E^{-1} as needed. When the attacker outputs C^*, the collision finder sets $E^{-1}(C^*[0]) = N$, which is a valid assignment with probability $1 - q\ell/2^n$, and returns K^* as the second preimage. Note that K^* is different from X with overwhelming probability, and the ciphertext C^* is valid iff indeed $H(K^*) = N$.

The second collision finder (that needs to work when $C^*[0] = C_i[0]$ for some i) also begins by getting some random input X and computing $N = H(X)$. Again, the collision finder plays the integrity-of-ciphertext game with the attacker, but now it chooses at random a query i and uses X as the data-key K_i. Clearly If C^* is a valid forgery and $C^*[0] = C_i[0]$ (which happens with probability ε_2/q) then K^* is a second preimage of N.

The third collision finder (that needs to work when $C^*[0] = C_i[j]$ for $j > 0$) also begins by getting some random input X and computing $N = H(X)$. Again, the collision finder plays the integrity-of-ciphertext game with the attacker, but now it chooses at random a query i and a block h, and uses N as the data-key block $K_i[j]$. Again, if C^* is a valid forgery and $C^*[0] = C_i[j]$ (which happens with probability $\varepsilon_3/q\ell$) then K^* is a second preimage of N. □

XOR-then-ECB/CBC is not secure. It turns out that second preimage resistance is not a necessary condition when using ECB or CBC. Below we show that even a simple public linear function may be sufficient in this case. However, not every linear function works, and in particular just taking the XOR of the key blocks is *not secure*. Let $K[1], K[2]$ be a two-block data-key, and let $C[0], C[1], C[2]$ be the ciphertext corresponding to it using XOR-then-ECB key wrapping. Then one can check that the ciphertext $C[1], C[0], C[2]$ is a valid ciphertext, corresponding to the data key $K[1] \oplus K[2], K[2]$. The same attack works also for CBC.

Linear-then-ECB/CBC may be secure. Below we show, however, that the "permutation attack" from above is in some sense the only one that matters when using ECB or CBC with a public linear function. Specifically, we show that using a public linear function of the form $H(K[1], \ldots, K[\ell]) = \sum_j \alpha_j K[j]$ where the α_j's are linearly independent, is already enough to get some level of security. (For example, we can use $\alpha_j = \alpha^j$ where α is a primitive element in $GF(2^n)$.) However, the security level deteriorates quickly with ℓ: the advantage bound that we prove is only $(q(\ell+1))^{\ell+1}/2^n$. For the typical case $\ell = 2$ this means security level of $O(2^{n/3})$, which may be sufficient in many applications. But for longer keys this construction may not be secure enough to be used in practice.

Lemma 4. *Fix some $\ell < n$, and let $H(K[1], \ldots, K[\ell]) \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} \alpha_j K[j]$, where the α_j's are linear operations over $\{0,1\}^n$ such that the set $\{1, \alpha_1, \ldots, \alpha_\ell\}$ is linearly independent. (That is, there is no nontrivial 0-1 combination of the α's and 1 that sums up to zero.)*

Consider the HtECB and HtCBC constructions using H for the hash function, and with a truly random permutation for the block cipher. Then for any bound q on the number of wrapping queries, we have

$$\mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtECB}}(wrap = q),\ \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtCBC}}(wrap = q) \le O(q^2\ell^2/2^n)$$
$$\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECB}}(wrap = q),\ \mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCBC}}(wrap = q) \le O((q(\ell+1))^{\ell+1}/2^n)$$

In the long version we also include analysis for the constructions Hash-then-ECB-X/CBC-X and Hash-then-XEX. Some variations of our constructions for variable-input-length and associated-data are mentioned in Appendix B.

4 Conclusions

In this work we examined the practice of key-wrapping, and in particular the implementation template of Hash-then-Encrypt. We argued that this template may be attractive in practice, especially in cases where the key-wrapping is used to "glue" together existing incompatible systems. We considered a wide array of "hash functions" and encryption modes, showed how to break some combinations and proved security bounds for others. Although none of the combinations that we considered meets the notion of deterministic authenticated encryption due to Rogaway and Shrimpton [22], we argued that some of them are still secure enough for key-wrapping. To make this argument, we measured them against weaker notions of security, which are arguably sufficient for most applications of key-wrapping.

We would like to stress again that given the choice, one should prefer more robust implementations, such as standard authenticated encryption or the SIV mode of Rogaway and Shrimpton. But in cases where these options are not available, we believe that our results may provide guidance to what can or cannot be used safely.

References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. Amanatidis, G., Boldyreva, A., O'Neill, A.: Provably-secure schemes for basic query support in outsourced databases. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 14–30. Springer, Heidelberg (2007)
3. An, J.H., Bellare, M.: Does encryption with redundancy provide authenticity? In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 512–528. Springer, Heidelberg (2001)
4. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)

5. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
6. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
7. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003); Preliminary version in CRYPTO 1998
8. Dodis, Y., Smith, A.: Entropic security and the encryption of high entropy messages. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 556–577. Springer, Heidelberg (2005)
9. Dworkin, M.: Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)
10. Gligor, V.D., Donescu, P.: Fast encryption and authentication: XCBC encryption and XECB authentication modes. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 92–108. Springer, Heidelberg (2002)
11. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
12. Jutla, C.S.: Encryption modes with almost free message integrity. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 529–544. Springer, Heidelberg (2001)
13. Katz, J., Yung, M.: Characterization of security notions for probabilistic private-key encryption. Journal of Cryptology 19(1), 67–95 (2006); Earlier version in STOC 2000, pp. 245–254
14. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
15. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
16. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
17. Meyr, C.H., Matyas, S.M.: Cryptography: A New Dimension in Computer Data Security. John Wiley & Sons, Chichester (1982)
18. Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and KDCs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 327–346. Springer, Heidelberg (1999)
19. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM Conference on Computer and Communications Security - ACM-CCS 2002, pp. 98–107. ACM, New York (2002)
20. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
21. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer, Heidelberg (2004)
22. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)

23. Russell, A., Wang, H.: How to fool an unbounded adversary with a short key. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 133–148. Springer, Heidelberg (2002)
24. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)

A The Rogaway-Shrimpton KIAE Notion

In an appendix to the long version of their paper [22], Rogaway and Shrimpton describe a notion of *key insertion authenticated encryption* that bares some similarities to our notion of authenticated key-wrapping: Specifically, they describe a setting where one applies a randomized encoding procedure to the adversarially-controlled message before encryption, and the attacker is given the corresponding ciphertext together with the randomness that was used in the encoding. The KIAE notion roughly requires that such an attacker cannot distinguish these (ciphertext, randomness) pairs from just random bits.

The crucial difference between our notions and KIAE is that we insist that the plaintext to be encrypted is completely random and outside of the attacker's control, whereas KIAE still allows the attacker to control parts of the plaintext.[5] Namely, our authenticated key-wrapping is a degenerate case of KIAE where the message and authenticated data are both empty. It thus follows that KIAE is still strictly stronger than all the security notions that we consider in this work.

In terms of usage, KIAE seems rather far removed from the way that key wrapping is typically used in practice: A typical applications would apply key-wrapping to a random data key, and then use that data key as a cryptographic key in some other scheme (say, to encrypt "real data" or a lower-level key in the hierarchy). The KIAE case seems to be targeted at applications where the data key (which is used as a cryptographic key elsewhere) is wrapped together with some "real data" in the same ciphertext. Hence in that notion the attacker gets to choose this "real data", while at the same time the data key is assumed to be random.

B Variations and Extensions

Variable input length. All the constructions and proofs in this section extend also to the case of variable input-length. (Although for key-wrap this case may not be very interesting, since symmetric keys are typically all of the same length.) When using second-preimage resistant hash function, we need to assume that it is second-preimage resistant even for variable input-length. The proofs for the linear hash functions need to be extended by considering the cases where the attempted forged ciphertext is an extension of the previous ciphertexts that were

[5] A smaller difference is that Rogaway-Shrimpton consider associated data as an integral part of their notion, whereas we view it as an optional extension. There are also some syntactic differences between these notions, but these are of no consequence.

obtained from the encryption oracle. For the "somewhat secure" constructions, the security bound for ECB and CBC will then be roughly $O((q\ell_{\max})^{\ell_{\max}})$ where ℓ_{max} is the largest allowable length of any data-key (expressed in 128-bit blocks). For the masked version ECB-X and CBC-X, we get $O(q^{\ell_{\max}})$, where in this case ℓ_{max} is the longest data-key that was returned by the wrapping oracle.

Input lengths that are not a multiple of the block length are handled by CTR without a problem. When using other modes, this can be handled by just padding the key. If one must preserve the length then ciphertext-stealing will also work.

Associated data. To handle associated data A, one must "hash" it together with the data-key, computing the nonce as $N \leftarrow H(\mathsf{D}, \mathsf{A})$. But as opposed to the data key, the associated data A is not random, so it should be modeled as controlled by the attacker.

For the constructions using second-preimage hashing, we must now make a stronger assumption on the hash function. Specifically, the function $H(\mathsf{D}, \mathsf{A})$ must meet the condition of "enhanced TCR" as described in [11], when D is viewed as the hashing seed: Namely an attacker that chooses A and gets a random D, should not be able to find different A', D' such that $H(\mathsf{D}, \mathsf{A}) = H(\mathsf{D}', \mathsf{A}')$. Still we note that since H can depend on a secret key then constructing such functions is easier, and in particular universal hashing satisfy even this stronger requirement.

For the constructions based on linear hashing, one way to incorporate associated data is by applying a PRF to it and then computing the nonce as $N = H(PRF(\mathsf{A})|\mathsf{D})$, where H is the same linear function from above. Practically speaking, however, if we are already using a PRF then we might as well compute $N = PRF(\mathsf{A}|\mathsf{D})$ (in which case we get back the SIV construction).