More On Key Wrapping

Rosario Gennaro Shai Halevi
IBM T.J. Watson Research Center

July 27, 2009

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 begin by observing that many standard encryption modes satisfy the key-warpping notion that mirrors CPA-security, even when used with a fixed IV (with the notable exception of CTR mode). To achieve the notion that mirrors authenticated encryption, we 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.

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

Contents

List of Figures

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 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

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

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

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?

We begin by observing that some standard encryption modes already meet our first notion (mirroring CPA security), even without IVs. Specifically, we show that ECB and CBC-with-fixed-IV meet this notion. On the other hand, it is clear the CTR-with-fixed-IV does not meet this notion.[1]

We then tackle the harder problem of providing authenticated key-wrap. Some simple solution includes using standard secure symmetric encryption, perhaps using generic composition techniques such as encrypt-then-authenticate [5, 14, 6] (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 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

[1]We remark that just like for regular encryption, the main use of CPA-security is as a building block for achieving stronger security notions. However, it may still be usable a-la-carte for applications needing only eavesdropping protection.

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.[2] These results are summarized in Table 1. We also discuss briefly how to extend these constructions to deal with keys whose length is not a multiple of the block length (e.g., using AES to wrap 192-bit keys) and how to authenticate "associated data" together with the wrapped key.

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., [7, 15, 1]). 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.

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 later. We formulate our definitions using "concrete security" notations. This style only formulates the notion of *the advantage of an attacker A* for the various setting (denoted $\mathbf{Adv}(A)$), and all the security statements explicitly state the advantage of attackers in terms of the resources available to them. (A "secure scheme" is implicitly defined as one where any "feasible" attacker has only an "insignificant" advantage, but this is never made formal.)

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

[2] 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.

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 Random-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

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

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

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

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

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

where \$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 [13, 5], 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).

2.2 Some extensions

The basic definitions above can be extended in standard ways along several axes. First, to deal with variable input length we let the attacker specify the length ℓ of the data key in each wrapping query. (Typically the length must be taken from some restricted set of "allowed lengths", e.g., only integral multiples of the block size.)

Second, to allow associated data (in the CCA game and the integrity game) we also let the attacker specify an arbitrary string of "associated data" A in its wrapping and unwrapping queries. In the CCA game, the forbidden queries to the unwrapping procedure are only those that contain ciphertexts that were returned from the wrapping procedure, and moreover have the same associated data as what was used to generate these ciphertexts. In the integrity game, the attacker is considered successful even if it "forges" an old ciphertext but with a new associated data.

Yet other standard variations are to allow stateful or "nonce-based" wrapping procedures [21], or to allow the procedures to depend on public parameters [5], or to require only "replayable CCA" (RCCA) [6] where the attacker may be able to generate new encryptions of the same data keys.

Another extension, which is a little more subtle, it to allow the attacker to see many wrappings of the same data keys. (This may be a realistic capability in some key-management systems.) One way to model this capability is by letting the attacker specify a "handle" in the wrapping queries: if the handle is new then the query is answered as before with new data keys chosen afresh. But if the handle is the same as in some previous query then the same data keys as in that previous query are used, and the attacker sees yet another wrapping of the same key.[3] We note that this capability is only meaningful for randomized (or stateful) wrapping procedures. For deterministic procedures, a repeated query will result in the same answer as the previous one.

Finally, we note that a plausible different syntactic choice would have been to allow the data-key to be an output of the wrapping procedure rather than an input to it. (This is how KEM schemes are defined in the public-key setting.) This choice may be quite useful in practice, since it will cover also key-derivation procedures (which are in many ways equivalent to key-wrapping procedures, see more discussion in Section 5). On the other hand, using this formulation would preclude the extension above where the same key is wrapped many times.

2.3 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 some of the plaintext is random and another part is chosen by the attacker. See discussion in 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

[3] One can imagine a more general capability, allowing an attacker that saw wrapped data keys to ask for a wrapping of a function of these keys (e.g., an XOR of two of them). It is not hard to see that allowing these "more general" queries is equivalent to giving the attacker full control over the plaintext data keys. Also, this would not be a realistic capability in any key-management architecture that we know of.

fact, some of our "provably secure" constructions from Section 4 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.

2.4 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*

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{enc.cpa}}_{\mathcal{C}}(enc=q) &\le \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathcal{KW}}(wrap=q) \;+\; q\cdot \mathbf{Adv}^{\mathsf{enc.cpa}}_{\mathcal{SE}}(enc=1),\\
\mathbf{Adv}^{\mathsf{enc.cca}}_{\mathcal{C}}(enc=q, dec=q') &\le \mathbf{Adv}^{\mathsf{kw.cca}}_{\mathcal{KW}}(wrap=q, unwrap=q') \;+\; q\cdot \mathbf{Adv}^{\mathsf{enc.cca}}_{\mathcal{SE}}(enc=1, dec=q'),\\
\mathbf{Adv}^{\mathsf{enc.int}}_{\mathcal{C}}(enc=q) &\le \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathcal{KW}}(wrap=q) + \mathbf{Adv}^{\mathsf{kw.int}}_{\mathcal{KW}}(wrap=q) \;+\; q\cdot \mathbf{Adv}^{\mathsf{enc.int}}_{\mathcal{SE}}(enc=1).
\end{aligned}$$

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 encryptions/wrappings and q' decryptions/unwrappings.

Proof (sketch): The proof of the CCA statement is essentially the same as the KEM/DEM proof in the context of public key encryption [7, Thm.5]. The other proofs are similar (but simpler).

For the first two inequalities, consider a modification of the CPA and CCA games where the encryption procedure chooses two independent random data keys D, D', wraps D' to get $C_1 \leftarrow$

$\mathsf{Wrap}_\mathsf{W}(\mathsf{D}')$ but uses D to encrypt the message, setting $C_2 \leftarrow \mathsf{Enc}_\mathsf{D}(M)$, and records the pair D, D'. (Thereafter, if any decryption query specifies (C_1', C_2') such that $\mathsf{Unwrap}_\mathsf{W}(C_1') = \mathsf{D}'$ then it returns $\mathsf{Dec}_\mathsf{D}(C_2')$.) Denote this modified game by $\mathcal{C}'$. Then a successful attack against $\mathcal{C}'$ corresponds to an attack against the underlying encryption $\mathcal{SE}$, since $\mathcal{C}'$ never reveals any information about the keys that are used to encrypt the actual messages (and instead only wraps the dummy keys D'). On the other hand, distinguishing between $\mathcal{C}$ and $\mathcal{C}'$ amounts to a successful attack against the underlying key-wrapping $\mathcal{KW}$.

For the last inequality, consider further the event in which the forged ciphertext includes one of the wrapped keys that were returned by the encryption oracle. Then this event (in the modified game $\mathcal{C}'$) corresponds to an attack on the integrity of the encryption scheme $\mathcal{SE}$, and the complement (where the forged ciphertext includes a new wrapped key) corresponds to an attack on the integrity of the key wrapping scheme $\mathcal{KW}$. ■

3 RPA-Secure Key Wrapping

We now turn to constructions of secure key-wrapping schemes, and begin with RPA-security. We note that most standard modes of operation for block ciphers already provide this level of security, even when used with a fixed IV.[4] Below we consider the modes ECB and CBC, as well as "masked versions" of them where inputs and/or outputs are XOR-ed with some key-dependent values. All these modes accept plaintext whose length is an integral multiple of the block length of the underlying cipher (which we denote by n). On plaintext blocks $P_1, P_2, \ldots$ and cipher key K, ECB mode computes the ciphertext blocks as $C_i = E_K(P_i)$, and CBC computes $C_i = E_K(C_{i-1} \oplus P_i)$, where E is the underlying block cipher and C_0 is the IV for CBC. In the "masked" variants (which we call below ECB-X and CBC-X for lack of better names), a sequence of mask values $X_1, X_2, \ldots$ is derived from a different part of the key (which is independent of the cipher key K), and the ciphertext blocks are computes as $C_i = E_K(X_i \oplus P_i) \oplus X_i$ and $C_i = E_K(C_{i-1} \oplus P_i) \oplus X_i$, respectively. The masks are "XOR universal", which means that for any $i \neq j$, the value $X_i \oplus X_j$ is (pseudo) random and uniform over a random choice of the key.

Lemma 2 *ECB, ECB-X, and CBC, CBC-X with a fixed IV are all RPA-secure key wrapping schemes (upto the birthday bound). Specifically, replacing the block cipher with a truly random permutation, and considering attackers that ask wrapping queries totaling of at most q n-bit blocks, we get for all of these modes* $\mathbf{Adv}^{\mathsf{kw.rpa}}(\textit{blocks} = q) = O(q^2/2^n)$.

Proof (sketch:) For all of these modes, for any fixed key (and fixed IV if any) there is a length-preserving permutation between the plaintext to be encrypted and the sequence of blocks that are used as inputs to the underlying blocks cipher. It follows that when the data-key to be wrapped if chosen at random in $\{0,1\}^{\ell n}$ then also the inputs to the underlying cipher are a sequence of ℓ random n-bit blocks. It follows that for the entire set of $2\ell n$ blocks that represent all the pairs $\mathsf{D}_0, \mathsf{D}_1$ that the attacker sees in the attack, no two blocks are equal except with probability of $O(q^2/2^n)$. And if no such collisions occur, the view of the attacker (as determined by the choice of the random permutation) is identical, regardless of whether it is D_0 or D_1 that is wrapped. ■

[4]One notable exception is CTR mode. This demonstrates once again that the reliance of CTR mode on its IV is more fundamental than for many of the other modes. In CTR mode a repeated IV is disastrous no matter what the plaintext is, while other modes only need the IV's to guard against re-encryption of the same plaintext.

4 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, 14, 6], here too one gets RCCA-security from any MAC and authenticated key-wrap when the MAC is "strongly unforgeable".[5]

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."[6] 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.

4.1 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 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 XOR-ing 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 $K_1, K_2 \oplus C_1 \oplus C_2, K_3 \oplus C_1 \oplus C_2$.

4.2 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. We consider below encryption using CTR mode, the modes ECB and CBC and their masked versions ECB-X

[5]A MAC is "strongly unforgeable" if the attacker cannot even produce a new valid authentication tag for a previously-authenticated message. This distinction is of no practical importance, since all the MACs that we use in practice are likely to be "strongly unforgeable."

[6]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".

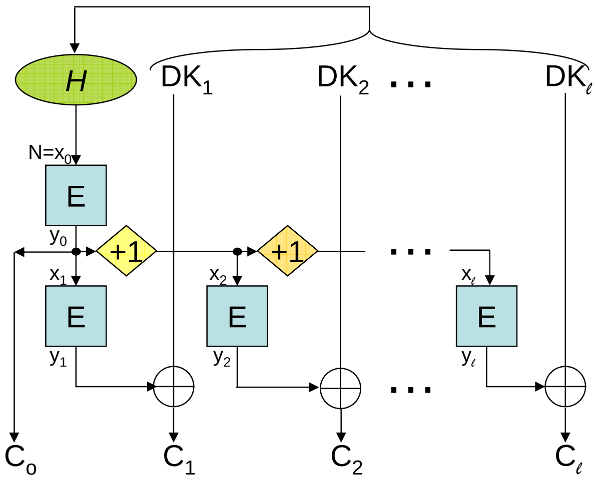

Figure 1: HtCTR

and CBC-X, and also "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.[7]

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).[8]

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.

4.3 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

[7] 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."

[8] Addition modulo 2^n is used for ease of presentation, in fact one can increase any subfield of the counter that has more than $\log \ell$ bits.

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 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

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

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 $\alpha_H \leq \mathbf{Adv}_H^{\mathsf{spr}}$, 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 3 *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*

$$\begin{aligned}
\mathbf{Adv}_{\mathrm{HtCTR}}^{\mathsf{kw.rpa}}(\mathit{wrap}=q) &\leq 2\binom{q}{2}\alpha_H + O(q^2\ell/2^n),\\
\mathbf{Adv}_{\mathrm{HtCTR}}^{\mathsf{kw.int}}(\mathit{wrap}=q) &\leq q\mathbf{Adv}_H^{\mathsf{spr}} + \binom{q}{2}\alpha_H + O(q^2\ell^2/2^n),\\
\mathbf{Adv}_{\mathrm{HtCTR}}^{\mathsf{kw.int}}(\mathit{wrap}=1) &\geq \mathbf{Adv}_H^{\mathsf{spr}}
\end{aligned}$$

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):
The if direction. We only prove here the inequality regarding $\mathbf{Adv}_{\mathrm{HtCTR}}^{\mathsf{kw.int}}$, the proof for $\mathbf{Adv}_{\mathrm{HtCTR}}^{\mathsf{kw.rpa}}$ is similar (but quite simpler). Consider an attacker A against HtCTR that sees q wrapped keys, $(K_i, C_i) = (\langle K_i[1], \ldots, K_i[\ell]\rangle, \langle C_i[0], C_i[1], \ldots, C_i[\ell]\rangle)$ for $i = 1, 2, \ldots, q$, and produces an attempted forgery $C^* = \langle C^*[0], C^*[1], \ldots, C^*[\ell]\rangle$. Let $HCOL$ denote the event where $C^*[0] = C_i[0]$ for some $i \in [1, q]$. We show that

$$\begin{aligned}
\Pr[\text{event } HCOL \text{ occurs and } C^* \text{ is valid}] &\stackrel{(a)}{\leq} q\mathbf{Adv}_H^{\mathsf{spr}} + \binom{q}{2}\alpha_H\\
\Pr[\text{event } HCOL \text{ does not occur and } C^* \text{ is valid}] &\stackrel{(b)}{\leq} O(q^2\ell^2/2^n)
\end{aligned}$$

Inequality (a) is straightforward: Since $C^*[0] = C^i[0]$ but $C^* \neq C_i$ then there is some block $j > 0$ such that $C^*[j] \neq C_i[j]$. Hence when decrypting we get $K^*[j] = C^*[j] \oplus E_w(C^*[0] + j - 1) = C^*[j] \oplus E_w(C_i[0] + j - 1) \neq C_i[j] \oplus E_w(C_i[0] + j - 1) = K_i[j]$. But for C^* to be valid it must hold that $H(K^*) = E^{-1}(C^*[0]) = E^{-1}(C_i[0]) = H(K_i)$. This means that the attacker A sees the random first preimage K_i and is able to come up with a different K^* such that $H(K^*) = H(K_i)$, violating the second-preimage resistance of H.

Formalizing this argument takes a little care in the case where H has a secret key, since a collision finder must simulate all the wrapping queries to A without knowing that secret key. However, since E is a random permutation then the simulator does not need to know the input to E in order to simulate the output, it can just choose distinct random values. The simulator needs to guess the index i (hence the q factor), and the simulation is only accurate as long as all the $C_i[0]$'s are distinct (hence the term $\binom{q}{2}\alpha_H$).

Next we prove inequality (b). First we observe that with high probability over the K_i's and the C_i's, the only points x for which both $E^{-1}(x)$ is defined and $E(x + j - 1)$ is defined *for any* $j \in [1, \ell]$ (from the attacker's perspective) are the points $x = C_i[0]$. More precisely, consider the usual process of "lazy evaluation" of E for generating the view of the attacker (K_i, C_i), $i = 1, \ldots, q$, and let $ECOL$ be the following event:

$$ECOL \stackrel{\text{def}}{=} \left\{ \exists x \notin \{C_1[0], \ldots, C_q[0]\},\ \exists j \in [1, \ell]\ s.t.\ \begin{array}{l} E^{-1}(x) \text{ is defined, and} \\ E(x + j - 1) \text{ is defined} \end{array} \right\}$$

Note that E is defined at the points $H(K_i)$ and $C_i[0] + j - 1$, and E^{-1} is defined at the points $C_i[0]$ and $C_i[j] \oplus K_i[j]$. The event $ECOL$ is when the intersection between these sets contains more than the $C_i[0]$'s, which happens with probability $O(q^2\ell^2/2^n)$. If neither $HCOL$ nor $ECOL$ occur, then we know that either $E^{-1}(C^*[0])$ is undefined or else all of the values $E(C^*[0] + j - 1)$ are undefined for $j = 1, \ldots, \ell$.

If $E^{-1}(C^*[0])$ is undefined then we can first extend E so that it is defined at all the points $E(C^*[0] + j - 1)$ for $j = 1, \ldots, \ell$, which determines the blocks $K^*[j] = C^*[j] \oplus E(C^*[0] + j - 1)$. This still leaves $E^{-1}(C^*[0])$ undefined (except with probability $\approx \ell/2^n$), but now $N^* = H(Z^*)$ is already fixed. Hence the probability that C^* is valid (namely $E^{-1}(C^*[0]) = N^*$) is $O(\ell/2^{-n})$.

If $E^{-1}(C^*[0])$ is defined then it must be of the form $C_i[0] + j - 1$ for some $i \in [1, q]$, $j \in [1, \ell]$. (This is because the only other points where E^{-1} is defined are $C_i[0]$, and $C^*[0]$ is not one of them because the event $HCOL$ does no occur.) Hence $E^{-1}(C^*[0])$ is a random value (over the choice of the permutation E) which is part of the view of the attacker. On the other hand, the values of $E(C^*[0] + j - 1)$ are all undefined, and therefore the blocks $K^*[j] = C^*[j] \oplus E(C^*[0] + j - 1)$ are all nearly uniform and independent of the view of A (and in particular of $E^{-1}(C^*[0])$).[9] Again, this means that the probability of getting $E^{-1}(C^*[0]) = H(K^*)$ is at most $O(q\ell/2^n)$.

Concluding the two sub-cases, we find that the probability that $HCOL$ does not occur and yet C^* is valid, is bounded by $O(q^2\ell^2/2^n) + O(\ell^2/2^n) + O(q\ell/2^n) = O(q^2\ell^2/2^n)$.

The only-if direction: It is easy to use a collision-finder CF for the function H for an attack on HtCTR: Given a single wrapped key (K, C), provide the collision finder with the first preimage K. If it returns a second preimage K' then output as the forged ciphertext $C' = C[0], C'[1], \ldots, C'[\ell]$ where $C'[j] = C[j] \oplus K[j] \oplus K[j]$. It is easy to see that C' is valid if and only if $H(K) = H(K')$. ■

[9] By "nearly uniform and independent" we mean that the distribution on $(E^{-1}(C^*[0]), K^*)$ is close upto $O(q\ell/2^n)$ to the distribution where they are uniformly random and independent. The term $O(q\ell/2^n)$ is the probability that choosing at random any of the values for $E(C^*[0] + j - 1)$ happens to collide with something that was already defined.

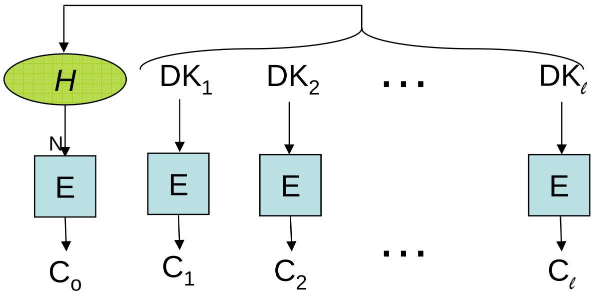

Figure 2: HtECB

Some constructions that are likely to meet the second-preimage resistant condition that is needed in Lemma 3 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.

4.4 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 5 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 mode, 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

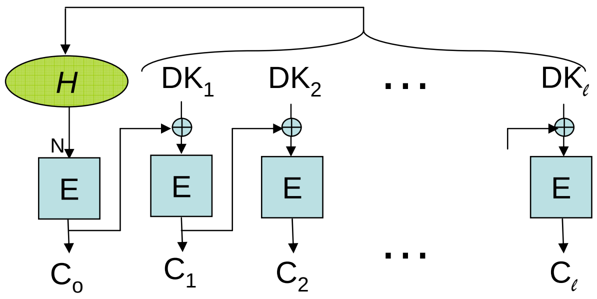

Figure 3: HtCBC

(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 $\mathbf{Adv}_H^{\mathsf{spr}} = 2^{-n}$. (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

$$E_{h_1}(H_{h_1,h_2}(X,Z)) \;=\; E_{h_1}(X) \oplus E_{h_2}(Z) \;=\; (E_{h_1}(Y) \oplus E_{h_2}(Z)) \oplus E_{h_2}(Z) \;=\; E_{h_1}(Y)$$

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 4 *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*

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

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. (See discussion in Appendix 5.)

Lemma 5 *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

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

Proof Again we only prove the bound on $\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECB}}$. The proof uses the following well-known simple fact:

Fact 4.1 *Let $\mathcal{S} = \{R_1, R_2, \ldots, R_q, R^*\}$ be a system of linear relations over an N-element field. If all the relations in $\mathcal{S}$ are linearly independent then when choosing all the variables uniformly at random in the field, the probability that R^* is satisfied is $1/N$, even conditioned on all the R_i's being satisfied.* ■

We only prove here the inequality regarding $\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECB}}$. Fix the parameters ℓ, n, q and denote the transcript of a q-query attack against HtECB by

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

Below we also use the following notations:

- For each i we denote $K_i[0] \stackrel{\text{def}}{=} \sum_j \alpha_j K_i[j]$ (so we have $C_i[j] = E(K_i[j])$ for $j = 0, 1, \ldots, \ell$).
- The attempted forgery that the attacker outputs is denoted $C^* = \langle C^*[0], C^*[1], \ldots, C^*[\ell]\rangle$, and denote $K^*[j] = E^{-1}(C^*[j])$.
- Denote $\alpha_0 \stackrel{\text{def}}{=} 1$, so for all j we have the relation $R_j : \sum_{j=0}^{\ell} \alpha_j K_i[j] = 0$. Similarly, the attempted forgery is successful if it satisfies the relation $R^* : \sum_{j=0}^{\ell} \alpha_j K^*[j] = 0$, and yet C^* is different than all the C_i's.

The rest of the proof goes roughly by showing that R^* is linearly independent of all the R_i's. First, we note that if any of the clocks $C^*[r]$ is different from all the the ciphertext blocks $C_i[j]$, $i \in [1, q]$, $j \in [0, \ell]$, then $E^{-1}(C^*[r])$ is undefined from the attacker's perspective, and can assume any of at least $2^n - q(\ell+1)$ values with equal probability. Call the event that such a ciphertext block exists in C^* "event U" (for Undefined), and the argument above implies that

$$\Pr[\text{event } U \text{ occurs and } C^* \text{ is valid}] \leq \frac{1}{2^n - q(\ell+1)}$$

(Note that we rely here on the condition that the α_j's are linearly independent: the same block $C^*[r]$ can appear several times in C^* and we need the coefficient of $E^{-1}(C^*[r])$ in the relation R^* to be non-zero.) Next we prove a bound $\Pr[\text{event } U \text{ does not occur and } C^* \text{ is valid}] \leq O((q(\ell+1))^{\ell+1})/2^n$, which would complete the proof.

Note that there are at most $(q(\ell+1))^{\ell+1}$ different ciphertexts C^* in which all the blocks $C^*[j]$ are taken from the other ciphertexts C_i (because each of the $\ell+1$ blocks must be equal to one of the $q(\ell+1)$ ciphertext blocks in the transcript). Each of these choices implies a relation R^*, and we now show that each one of these relations is linearly independent of the R_i's. Since the $K_i[j]$'s were all chosen at random subject to the relations R_i, Fact 4.1 tels us that each of these potential relations R^* holds only with probability 2^{-n}. And using the union bound we get that the probability that any of the $(q(\ell+1))^{\ell+1}$ potential ciphertexts is valid, is bounded by $(q(\ell+1))^{\ell+1}/2^n$.

It is left to check that all these potential relations R^*'s are linearly independent of the R_i's, so we fix one of these potential ciphertext C^*, and consider the corresponding relation R^*. We first observe that if C^* is anything other than a permutation of the blocks from one of the ciphertext C_i, then it means that each C_i has a block that does not appear in C^*. This means that each relation R_i depends on some variable that is not in R^* (and also not in the other $R_{i'}$'s), so all of the relations are linearly independent. It is left to consider only permutations of one of the C_i's, and we need to show that the resulting R^* is linearly independent of the relation R_i. (The other $R_{i'}$'s are irrelevant, since they are defined over a disjoint set of variables.)

So fix one of the ciphertexts C_i, and assume that C^* is a permutation of the blocks in C_i, call this permutation π. (We know that π is not the identity, since $C^* \neq C_i$.) We need to check that the two relations

$$R_i: \sum_{j=0}^{\ell} \alpha_j K_i[j] = 0, \qquad R^*: \sum_{j=0}^{\ell} \alpha_j K_i[\pi(j)] = 0$$

are linearly independent. We can re-write the relation R^* as $\sum_j \alpha_{\rho(j)} K_i[j] = 0$, where ρ is the inverse permutation of π, so showing that these relations are independent amounts to showing that there is no linear operation β such that $\alpha_{\rho(j)} = \beta\alpha_j$ for all $j = 0, 1, \ldots, \ell$. First, observe that for any $\beta \neq 1$ it holds that:

$$\sum_{j=0}^{\ell}(\alpha_{\rho(j)} \oplus \beta\alpha_j) \;=\; \sum_{j=0}^{\ell}\alpha_{\rho(j)} \;\oplus\; \beta\sum_{j=0}^{\ell}\alpha_j \;\overset{(*)}{=}\; \sum_{j=0}^{\ell}\alpha_j \;\oplus\; \beta\sum_{j=0}^{\ell}\alpha_j \;=\; (\beta \oplus 1)\sum_{j=0}^{\ell}\alpha_j \;\neq\; 0,$$

where Equality $(*)$ holds since ρ is a permutation, and the last term is non-zero since the α_j's are linearly independent and $\beta \neq 1$. On the other hand, since ρ is not the identity then there exists some value of j such that $\rho(j) \neq j$, which means that also $\alpha_{\rho(j)} \neq 1 \cdot \alpha_j$ (again since the α_j's are linearly independent). This completes the proof. We remark that the proof for $\mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCBC}}$ is nearly identical, except that the relations R_i are now defined as $K_i[0] \oplus \sum_{j=1}^{\ell} \alpha_j(K_i[j] \oplus C_i[j-1]) = 0$, and similarly for R^*. One can verify that the arguments from above apply also to these relations (with some easy modifications for the argument regarding the permuted ciphertext). ■

4.5 Hash-then-ECB-X/CBC-X

When using the masked versions of ECB and CBC, then even the XOR of the data-key blocks is already sufficient to get the same level of security as above. Roughly, this is because the permutation attacks on ECB and CBC are thwarted by the additional masking of the outputs.

Recall that in ECB-X and CBC-X, we derive an "XOR universal" sequence of masks $X[j]$ from the key, such that (i) for a random wrapping key and any $j \neq j'$, $X[j] \oplus X[j']$ is a (pseudo) random n-bit block, and (ii) the $X[j]$'s are independent of the cipher key that is used for ECB of CBC. Typically, this is done by having an additional n-bit element X as part of wrapping key, and setting

$X[j] \leftarrow j \cdot X$ (or sometime $X[j] \leftarrow \alpha^j \cdot X$ for some high-order element α in $GF(2^n)$). Then we XOR the mask value $X[j]$ to the j'th output block from ECB or CBC. (In ECB-X we also XOR $X[j]$ to the j'th input block before applying the cipher to it.[10])

Lemma 6 *Fix some ℓ, n, and let $H(K[1], \ldots, K[\ell]) \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} \alpha_j K[j]$, where all the α_j's are non-zero. Consider the HtECB-X or HtCBC-X 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*

$$\begin{array}{rcl} \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtECBX}}(\mathit{wrap}=q),\ \mathbf{Adv}^{\mathsf{kw.rpa}}_{\mathrm{HtCBCX}}(\mathit{wrap}=q) & \leq & O(q^2\ell^2/2^n) \\ \mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtECBX}}(\mathit{wrap}=q),\ \mathbf{Adv}^{\mathsf{kw.int}}_{\mathrm{HtCBCX}}(\mathit{wrap}=q) & \leq & O(q^{\ell+1}/2^n) \end{array}$$

Proof (sketch): We concentrate on ECB-X, and denote a q-query transcript by

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

Below we also use the following notations:

- For each i we denote $K_i[0] \stackrel{\text{def}}{=} \sum_j K_i[j]$, $P_i[j] = K_i[j] \oplus X_i$, $Q_i[j] = C_i[j] \oplus X_i$ (so we have $Q_i[j] = E(P_i[j])$ for $j = 0, 1, \ldots, \ell$).
- The attempted forgery that the attacker outputs is denoted $C^* = \langle C^*[0], C^*[1], \ldots C^*[\ell]\rangle$, and we denote $Q^*[j] = C^*[j] \oplus X_i$, $P^*[j] = E^{-1}(Q^*[j])$, and $K^*[j] = P^*[j] \oplus X[j]$.

Then for all j we have the relation $R_j : \sum_{j=0}^{\ell} P_i[j] \oplus \sum_{j=0}^{\ell} X_j = 0$. Similarly, the attempted forgery is successful if it satisfies the relation $R^* : \sum_{j=0}^{\ell} P^*[j] \oplus \sum_{j=0}^{\ell} X_j = 0$, and yet C^* is different than all the C_i's.

We note that the distribution over the transcript is close upto $O(q^2\ell^2/2^n)$ to being uniformly random and independent of the $X[j]$'s. (The distance is due to possible collisions of the form $P_i[j] = P_{i'}[j']$ for $j' \neq j$, and due to the fact that E is a random permutation and not a random function, so the $Q_i[j]$'s are not quite uniformly random in $\{0,1\}^n$.) Below we analyze the case of uniform and independent transcript, which introduces an error term of $O(q^2\ell^2/2^n)$.

We analyze two cases: either C^* has a "new block", or it does not. In this proof, a "new block" is a block $C^*[r]$ that is different from $C_i[r]$ for all $i \in [1,q]$. If C^* has such a block, then with all but probability $O(q\ell/2^n)$ over the choice of the $X[j]$'s, it holds that (i) E^{-1} is still undefined at $Q^*[r] = C^*[r] \oplus X[r]$, and (ii) $Q^*[r] \neq Q^*[r']$ for all $r' \neq r$. Then $P^*[r]$ (and thus also $K^*[r]$) is almost uniform and independent of the other blocks $K^*[r']$, and the $X[j]$'s, so the probability that R^* holds is close to 2^{-n}.

Next we analyze the case that every block in C^* already appeared in one of the C_i's in the same position. (Namely, for any $j \in [0,\ell]$ there is $i_j \in [1,q]$ such that $C^*[j] = C_{i_j}[j]$.) We note that there are only $q(\ell+1)$ such ciphertexts, and we now will show that for each of them we get a linearly independent relation R^*. Note that not all the blocks are taken from the same ciphertext C_i, or else we would have $C^* = C_i$. Hence as before for each $i \in [1,q]$ there must be a block j_i such that $C_i[j_i] \neq C^*[j_i]$. Hence the relation R_i depends on $K_i[j_i]$ but R^* does not (and neither are all the other $R_{i'}$'s) so all these relations are independent. ■

We also remark that it is easy to show that any second-preimage resistant hash function works for Hash-then-ECB-X/CBC-X: the proof is actually a simplified version of the "if direction" in the proof of Lemma 3.

[10]This makes no difference in our application where the input is chosen at random, but is important for standard encryption.

4.6 Hash-then-Tweak

We conclude the technical part by considering the Hash-then-Encrypt template using a "tweakable encryption mode" [16]. Namely, we compute as before $N = H(K)$, and then encrypt the i'th block of the data-key K using a tweak-value (N, i). We propose in particular to use Rogaway's XEX that natively uses tweaks of the form (N, i) [20], which gives us the construction HtXEX from above. It turns out that in this case, even the XOR function is already sufficient to get security upto the birthday bound.

Lemma 7 *Let $H(K[1], \ldots, K[\ell]) \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} \alpha_j K[j]$, where all the α_j's are non-zero. Consider the HtXEX ECB-X 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*

$$\mathbf{Adv}^{\mathsf{kw.rpa}}_{\text{HtXEX}}(\mathit{wrap} = q),\ \mathbf{Adv}^{\mathsf{kw.int}}_{\text{HtXEX}}(\mathit{wrap} = q) \quad \leq \quad O(q^2\ell^2/2^n)$$

Proof (sketch): Since XEX is a tweakable encryption mode, then we know that it is indistinguishable upto $O(q^2\ell^2/2^n)$ from using a different independent random permutation for each value of the tweak. So below we analyze this "idealized" version of XEX.

With the data keys K_i chosen at random, the probability of getting the same nonce in two wrapping queries $N_i = N_{i'}$ is $\binom{q}{2}/2^n$. Hence except with this probability, all the ciphertexts that are returned from wrapping queries are uniformly random and independent.

Consider now the attempted forgery $C^* = \langle C^*[0], C^*[1], \ldots, C^*[\ell] \rangle$. If $C^*[0] \neq C_i[0]$ for all i then by the above argument we can consider using independent permutations for the decryption of C^*, in which case K^* is random the the probability that C^* is valid is 2^{-n}. If $C^*[0] = C_i[0]$ but $C^*[j] \neq C_i[j]$ for some $j \geq 1$ then $X^*[j]$ would be chosen at random subject to $K^*[j] \neq K_i[j]$, and the probability that C^* is valid would be $1/(2^n - 1)$. ■

Again, it is easy to prove that any second-preimage resistant hash function works for Hash-then-Tweak.

4.7 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 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).

5 Further Discussion and 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. Before concluding we further discuss some practical points that we believe are relevant to the practice of key-wrapping.

Key-wrap vs. key-derivation. It is important to note that the objectives of key-wrapping can often be achieved also with key-derivation. In a key-derivation scheme, the data-key D is computed from the "master key" W by setting $\mathsf{D} = PRF_{\mathsf{W}}(N)$ where N is some nonce (or maybe context information).[11] The nonce N can be recorded (in the same manner as the ciphertext in a key-wrap scheme) and used later in conjunction with the master key W to recover the data key. Similarly to key-wrapping, security notions for key-derivation capture secrecy of the data-key and/or authentication (where the latter is sometimes implicit).

The choice between using key-wrapping or key-derivation may be rather arbitrary, there does not seem to be any difference between them in terms of their security. Key-wrapping is slightly more flexible, in that the same data-key may be wrapped by different master keys (or sometimes one can have the same data-key being wrapped several times by one master key, perhaps with different associated data).

On the practical security of the "somewhat secure" schemes. Recall that for some of the constructions in this paper we were only able to prove a security level of roughly $2^{n/(\ell+1)}$. We

[11] The notion of "key-derivation function" as we use it here assumes that the "master key" is already a uniformly random cryptographic key. The term key-derivation is sometimes used to describe the case where W is a "high entropy" value but not uniform. This concern is orthogonal the topic addressed by our work, so we ignore it here.

point out that within the attack model as we described in Section 2, this bound is likely to be tight. Essentially, an attacker sees all the random K_i's and C_i's, and in order to forge a new valid ciphertext it needs to find a small subset of them that satisfies a known linear relation. Our security bound said that the probability that *any solution exists* is bounded by something like $O(q^{\ell+1}/2^n)$, but an attacker must actually find that subset. We point out that for the simple linear relation $\sum_{i=i}^{\ell} X_i = 0$, one could use Wagner's work [24] to efficiently find a solution if one exists, and this attack can be extended (at least in some cases) also to arbitrary known linear relations.

On the other hand, in most applications of key-wrap the attacker does not get to see the data-keys K_i. For example, in a data-encryption application the attacker would only see encryption of data with these data keys, so we do not expect to find a matching attack. However, even for such applications we cannot prove security beyond $2^{n/(\ell+1)}$, except perhaps by making "strange and wonderful" assumptions on these data encryption scheme (e.g., Shannon's perfect cipher model).

On security beyond the birthday bound. A requirement that people sometimes make of key-wrap schemes is that they provide "security beyond the birthday bound", or more generally that they provide higher levels of security than what is expected from the data encryption schemes. The rationale for such requirements appear to be that wrapped keys are high value targets and should therefore be protected better than "mere data". Proponents of "high security" key wrap would argue that there is no point in using 256-bit keys if we wrap them using a scheme that only offers security upto $2^{128/2}$. We believe that such "high security" requirements are often misguided, for several reasons.

First, we note that there is no basis for equating data complexity of an attack with off-line key-search complexity. The latter measures a resource under the complete control of an attacker, while the former measure a resource which is controlled by the legitimate users of the system. In our opinion, an attack with data complexity 2^{60} is infinitely harder to mount than an exhaustive search for a 100-bit key: the former can never be mounted (because there is no system that actually wraps so many keys under one master key), while the latter may become feasible in some distant future.

Second, we note that the key-wrap application itself provides the attacker significantly fewer opportunities for attack: the keys are typically very short, and there aren't very many of them, so the attacker has much less data to work with than for a data-encryption application. In the vast majority of cases there will never be more than (say) one million keys that are wrapped under the same master key. Moreover, these keys are chosen at random and have no semantics, so chosen- and known-plaintext attacks are all but ruled out.

Third, in practice leaking information about keys is often *less harmful* than leaking information about data. Although the disclosure of a cryptographic key may be devastating, the disclosure of a few bits of a key is often quite inconsequential. Leaking the least significant bit of a message may be a serious breach (e.g., if the attacker knows that the message was either "buy" or "sell"), but leaking the least 64 bits of an AES-256 key is unlikely to have any practical consequences.

Our opinion therefore is that key-wrap schemes with security $q^2/2^{128}$ are secure enough for most (if not all) practical purposes, and even schemes with security $q^3/2^{128}$ are probably sufficient for most cases.

Acknowledgments

Te second author would like to thank Glen Jaquette for suggesting the problem and for several discussions.

References

[1] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2005.

[2] G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. In *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, volume 4602 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2007.

[3] J. H. An and M. Bellare. Does encryption with redundancy provide authenticity? In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 512–528. Springer, 2001.

[4] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO'07*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2007.

[5] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT'00*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[6] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology - CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[7] R. Cramer and V. Shoup. 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'98.

[8] Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. In *Theory of Cryptography - TCC'05*, volume 3378 of *Lecture Notes in Computer Science*, pages 556–577. Springer, 2005.

[9] M. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007.

[10] V. D. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Fast Software Encryption - FSE'01*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2001.

[11] S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. In *Advances in Cryptology - CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2006.

[12] C. S. Jutla. Encryption modes with almost free message integrity. In *Advances in Cryptology - EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.

[13] J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology*, 19(1):67–95, January 2006. Earlier version in STOC'00, pages 245-254.

[14] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology - CRYPTO'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[15] K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.

[16] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.

[17] C. H. Meyr and S. M. Matyas. *Cryptography: A New Dimension in Computer Data Security.* John Wiley & Sons, 1982.

[18] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer-Verlag, 1999.

[19] P. Rogaway. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security - ACM-CCS'02*, pages 98–107. ACM, 2002.

[20] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology - ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[21] P. Rogaway. Nonce-based symmetric encryption. In *Fast Software Encryption - FSE'04*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2004.

[22] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *Advances in Cryptology - EUROCRYPT'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

[23] A. Russell and H. Wang. How to fool an unbounded adversary with a short key. In *Advances in Cryptology - EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2002.

[24] D. Wagner. A generalized birthday problem. In *Advances in Cryptology – CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 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.[12] 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.

[12] 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.