A preliminary version of this paper appears in the Proceedings of the *18th ACM Conference on Computer and Communications Security (ACM CCS)*, 2011. This is the full version.

Ciphers that Securely Encipher their own Keys

Mihir Bellare[1] David Cash[2] Sriram Keelveedhi[3]

Abstract

In response to needs of disk encryption standardization bodies, we provide the first tweakable ciphers that are proven to securely encipher their own keys. We provide both a narrowblock design **StE** and a wideblock design **EtE**. Our proofs assume only standard PRP-CCA security of the underlying tweakable ciphers.

Keywords: Disk encryption, Ciphers, Key-dependent messages.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/mihir`. Supported in part by NSF grants CCF 0915675 and CNS 0904380.

[2] T.J. Watson Research Center, IBM, Email: `cdc@gatech.edu`. URL: `http://www.cse.ucsd.edu/~cdcash`. Part of this work was done while the author was at UCSD, supported by Mihir Bellare's NSF grant CCS 0915675, and at Ruhr-Universität Bochum, supported by Eike Kiltz's Sofja Kovalevskaja Award from the Alexander von Humboldt Foundation.

[3]Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `sriramkr@cs.ucsd.edu`. URL: `http://cseweb.ucsd.edu/users/skeelvee/`. Supported in part by NSF grants CCF 0915675 and CNS 0904380.

Contents

1 Introduction

In email archives of discussions of candidate enciphering schemes for disk encryption, conducted by the Security in Storage Working Group (IEEE P1619), one sees the intriguing comment "I have not received any meaningful response to the issue of grandma storing her keys on an encrypted drive. The WG must have considered it the first time it came up." The issue is whether it is safe to encipher the enciphering key itself. The group eventually came to the conclusion that key-enciphering security is important and desirable, for reasons that go beyond grandma. The presence of key-enciphering attacks on one candidate (LRW) then influenced rejecting it in favor of others on which attacks were not found.

Meanwhile, the group did value security proofs. EME [14], a wideblock cipher, is proven to achieve PRP-CCA security. (Disk encryption must be length preserving, so we are talking about ciphers, not randomized modes like CBC.) A variant EME2 [11] was standardized on the strength of these proofs, a host of efficiency properties, and the *presumed* security of enciphering the key.

If (as seems the consensus in this domain) key enciphering security is a requirement, and (as also seems the consensus) proofs are also important, a discrepancy becomes obvious, namely that, while there is a proof of PRP-CCA security, there is no proof of key enciphering security, confidence in the latter resting only on the absence of discovered attacks.

The question this raises is, can we fill the gap? We want efficient tweakable ciphers, both narrowblock (data is n bits long where n is the blocklength of the underlying blockcipher) and wideblock (data is mn bits long for $m \geq 2$). They should be not only proven PRP-CCA secure but also proven to securely encipher data that contains their own keys. (In the narrowblock case this means the message might equal the key, while in the wideblock case it means any block of the message might equal the key.) We do not want to use random oracles (ROs) but rather want to prove all this assuming only what was assumed for EME, namely the standard PRP-CCA security of the underlying blockcipher.

Is this possible? Proving security of encryption of key-dependent messages is notoriously hard. Standard reduction techniques fail completely. The first solutions [5] used ROs. Sophisticated, non-RO solutions based on novel techniques have emerged [6, 1, 7, 2] but they are far from practical.

In this paper we show nonetheless how to achieve the stated objective. We first provide **StE**, a simple and efficient transform that, applied to any tweakable PRP-CCA blockcipher, results in a tweakable narrowblock cipher that is not only proven PRP-CCA but also proven to securely encipher its own key, assuming only PRP-CCA security of the starting tweakable blockcipher. We then use **StE** as the basis for **EtE**, a transform that, applied to any tweakable PRP-CCA wideblock cipher, results in another tweakable wideblock cipher that, again, is not only proven PRP-CCA but also proven to securely encipher data containing its own key in any block, assuming only PRP-CCA security of the starting wideblock tweakable cipher and of the tweakable blockcipher underlying **StE**.

Motivated also by the above mentioned issues for disk encryption, Halevi and Krawczyk [12] had initiated the study of PRFs and PRPs secure for key dependent messages (KDMs). They were able to provide (1) standard model KDM secure PRFs, which are not invertible and thus cannot do disk encryption (2) KDM secure PRPs, but in the ideal cipher model. Ours are the *first* constructions of KDM secure PRPs with a proof of security in the standard (as opposed to random oracle or ideal cipher) model.

We instantiated **EtE** with AES as the base blockcipher, tweaked via XEX [17], and with EME as the base wideblock cipher. We implemented this with the AES-NI instruction set. We found that **EtE** is only 15% slower than plain EME.

Background. Security for key-dependent data was first considered for (randomized) IND-CPA encryption [5], so that solutions (which were provided in the RO model by [5]) are necessarily length increasing. For disk encryption, the encryption function must be length preserving. The desired primitive is a tweakable cipher [15] E: $\{0,1\}^n \times \mathcal{T} \times \{0,1\}^{mn} \to \{0,1\}^{mn}$ that deterministically maps an n-bit key K, a tweak $T \in \mathcal{T}$ and an m-block plaintext M to an m-block ciphertext $E(K, T, M)$. (So n is both the keylength and the blocklength.) It must be invertible, meaning $E(K, T, \cdot)$: $\{0,1\}^{mn} \to$

$\{0,1\}^{mn}$ is, for every K, T, a permutation whose inverse we denote by $E^{-1}(K, T, \cdot)$. This is a (tweakable) blockcipher, or a narrowblock design, if $m = 1$, and a wideblock design if $m > 1$. Both are of interest to the Security in Storage Working Group, the first under P1619 and the second under P1619.2. In disk encryption the tweak could be the sector number (wideblock) or the block index (narrowblock) and its use significantly increases security.

The standard security notion for a (tweakable) cipher is to be PRP-CCA secure [15]. (This is sometimes called strong PRP security following the terminology of [16].) We extend this to key dependent messages (KDM), defining what it means for E to be Φ-PRP-CCA secure where Φ is a class of functions that map n-bit keys to mn-bit inputs, following [5, 12]. The game picks a random challenge bit b, a random key K and a random permutation $\pi(T, \cdot)$: $\{0,1\}^{mn} \to \{0,1\}^{mn}$ for each T. It gives the adversary the standard oracles $\textsc{Fn}, \textsc{Fn}^{-1}$ from the PRP-CCA game and a new oracle $\textsc{KDFn}$. Oracle $\textsc{Fn}$, on input T, M, returns $E(K, T, M)$ if $b = 1$ and $\pi(T, M)$ otherwise; oracle $\textsc{Fn}^{-1}$, on input T, C, returns $E^{-1}(K, T, C)$ if $b = 1$ and $\pi^{-1}(T, C)$ otherwise; oracle $\textsc{KDFn}$, on input ϕ, T, where ϕ is required to be in Φ, returns $E(K, T, \phi(K))$ if $b = 1$ and $\pi(T, \phi(K))$ otherwise. To ensure non-triviality, the adversary is required to be legitimate, meaning does not query $\textsc{Fn}^{-1}(T, C)$ for a C previously received as a response to a $\textsc{KDFn}(T, \cdot)$ query. The adversary advantage is $2 \Pr[b = b'] - 1$ where b' is its output bit. Φ-PRP-CCA security obviously implies PRP-CCA security for all Φ, and coincides with it when $\Phi = \emptyset$.

What we mean by "encrypting the key" depends on whether the design is narrowblock or wideblock. In the narrowblock case, we are concerned with the obvious, namely that the message equals the key, captured formally by letting Φ consist of the identity function id. For wideblock designs, we seek security when *any block* of the message equals the key. (This reflects P1619.2 requirements.) We clarify that the key may occur in multiple blocks but may not overlap between blocks. Thus if $M = M[1] \ldots M[m]$ is the message then $M[i]$ may equal the key K for any i but we do not allow K to start somewhere in one block and end somewhere in the next block. In Section 4 we specify a Φ, that we denote $\mathcal{ID}_m$, that captures this formally.

The Security in Storage Working Group advocates security for encrypting the key because users or the system may store the key on the disk but also because, if it would create a vulnerability, an attacker could attempt to manipulate the system so that it transfers the key from memory to the disk. Overall, they considered it important enough to reject candidates without the property.

Attacks of Halevi and Krawczyk [12] show that, unlike for randomized encryption [5], there is no cipher that is Φ-PRP-CCA secure for the class Φ of all functions. To achieve security, we must restrict the class somehow. The restriction we make, namely to consider encrypting the key, results in a class capturing practical attacks for which we will show security to be achievable.

Motivated by the concerns and needs of the Security in Storage Working Group that we have explained, Halevi and Krawczyk [12] introduced the notions of KDM secure PRFs and PRPs and asked whether there exist Φ-PRFs or Φ-PRPs for non-trivial Φ. (Meaning, Φ contains some interesting non-constant function such as the identity.) They were able to give a standard model construct in the PRF case, but it was not invertible and thus could not be used for disk encryption. They also gave a PRP but the proof is in the ideal cipher model. Left open by their work is to provide a Φ-PRP for non-trivial Φ in the standard model. We resolve this, providing not only constructs, but efficient ones, with proofs not only in the standard model but assuming only standard PRP security of the underlying blockcipher, for non-trivial Φ of practical interest, with tweaks, and for both the narrow and wide block settings.

Approach. First we address the narrowblock case, providing a transform **StE** that turns a given PRP-CCA tweakable blockcipher into a {id}-PRP-CCA tweakable blockcipher, meaning the constructed cipher not only preserves the PRP-CCA security of the base one but, assuming only PRP-CCA security of the latter, is proven to securely encrypt its key. **StE** is of interest in its own right as the first (tweakable) blockcipher that can provably encipher its own key under standard and minimal assumptions.

We would have liked to show that instantiating the base blockcipher of E of EME [14] with $F = \mathbf{StE}[E]$ results in a wideblock cipher that can provably encipher its own key, but this does not work. Instead, we show how to add a pre-processing step to EME, or any other PRP-CCA secure wideblock cipher, to get another wideblock cipher that now, when instantiated with F as the underlying blockcipher, is $\mathcal{ID}_m$-PRP-CCA secure, meaning not only PRP-CCA secure but able to securely encipher messages that contain the key in any block. Let us now look at these two contributions more closely.

Narrowblock design. Our starting point is a suggestion of [6] to securely encrypt keys with a *randomized* encryption scheme by exchanging the key with another point. To make this work for deterministic encryption we swap with a "hidden" point, the latter determined by encryption under the key of a constant. A crucial idea is to use tweaks, defining the hidden point via a tweak not used for anything else.

We take as given a tweakable blockcipher $E\colon \{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ assumed to have normal, meaning PRP-CCA, security, but not necessarily able to securely encrypt its key. We pick an arbitrary tweak $\gamma \in \mathcal{T}$ for E and also an arbitrary point $\alpha \in \{0,1\}^n$. Both α and γ are public and known to the adversary. Our **StE** (Swap-then-Encipher) transform now turns E into another tweakable blockcipher $F\colon \{0,1\}^n \times \mathcal{T} \setminus \{\gamma\} \times \{0,1\}^n \to \{0,1\}^n$ whose tweakspace $\mathcal{T} \setminus \{\gamma\}$ is that of E with γ removed, meaning γ is not allowed as a tweak for F. We define F via

```
$\underline{F(K, T, M)}$
$H \leftarrow E(K, \gamma, \alpha)$
If $M = K$ then $Y \leftarrow E(K, T, H)$
Else If $M = H$ then $Y \leftarrow E(K, T, K)$
Else $Y \leftarrow E(K, T, M)$
Return $Y$
```

In Section 3 we show that F is invertible, as required to be a cipher. A curious aspect of the design is that at the third line we actually encrypt the key K with the original blockcipher E. Given that the latter may not necessarily be able to securely encrypt its key, why would this work? The answer is that we only do this if $M = H$ and we ensure the latter is very unlikely. Thus H is our "hidden" point.

Making sure H is hidden takes some care. It is not so merely because its computation depends on K, for the adversary will have an oracle that (at least in the real game) allow it to compute F under K for any tweak of the adversary's choice and this might be used to extract information about H. The crucial point is that γ, the tweak used in computing H, was removed from the tweak space of F so the adversary cannot use it.

The proof that F indeed securely encrypts its key is done by reduction to the assumed PRP-CCA security of E and is delicate due to the cyclic nature of the construction.

It is crucial for disk encryption that F is invertible. It is to ensure this that we need the swap that effectively exchanges the roles of K and H.

At first glance it would appear that **StE** doubles the cost since it requires two invocations of E, one to get H and the other to get Y. This, however, is also true of XEX-AES [17]: presented as a fast tweakable blockcipher, XEX actually requires two invocations of the underlying blockcipher. In both cases, however, the answer is the same, namely that the ciphers will be used in a mode where the extra blockcipher computation is done once and its cost is thus amortized out so that the effective cost of the cipher is one blockcipher call. Specifically if F, like XEX-AES, is used as the base blockcipher in a wideblock design to encipher a sector consisting of m blocks (eg. $m = 32$), we can compute H just once across the m encipherings so **StE** effectively involves just a single blockcipher invocation.

The IEEE 1619 standard for narrowblock encryption is based on XEX-AES [17]. **StE** is easily applied here, yielding an alternative narrowblock candidate that is as efficient as the current one when the cost of computing H is amortized out, but also has provable security for enciphering the key.

Wideblock design. EME [14] is a wideblock tweakable cipher that is proven PRP-CCA secure assuming only PRP-CCA security of the underlying blockcipher. It makes two passes through the data and runs at 2 blockcipher invocations per message block. One of its attractions, and its advantage over the earlier CMC [13], is that it is parallelizable. No attack is known when one encrypts messages containing the key in any block, but nor is there any proof that such an attack is absent.

We wish to enhance designs like EME so that PRP-CCA security is preserved but provable key-enciphering security is also added, without assuming any more of the underlying blockcipher than PRP-CCA security. Our **EtE** transform does just this. It makes an ECB pass through the data using $F = \mathbf{StE}[E]$ under one tweak and then applies any wideblock PRP-CCA cipher (EME would be one possibility) using F with another tweak. We prove that the resulting cipher is $\mathcal{ID}_m$-PRP-CCA secure assuming only that tweakable blockcipher E is PRP-CCA secure. Thus **EtE** provides a generic way to upgrade any PRP-CCA wideblock cipher to also be able to securely encipher messages containing the key in any block, at the cost of one extra blockcipher operation per block. The design preserves parallelizability, so that when instantiated with a parallelizable wideblock cipher like EME, the resulting cipher is also parallelizable.

Instantiation and implementation. Getting the best out of **EtE** requires good choices for the underlying components and some optimizations. We use AES as the base blockcipher and XEX [17] to tweak it. We use EME as the wideblock cipher. Let J denote the resulting wideblock cipher as produced by **EtE**. We implemented J, as well as plain EME for comparison, on an Intel Xeon W3690 processor at 3.47 GHz with support for AES-NI running Linux version 2.6 with code compiled using `gcc -O3 -ftree-vectorize -msse2 -ffast-math` to enable vector instructions and perform other general optimizations. The base code was from Brian Gladman's EME2 implementation [9], which we modified to run as EME and use the hardware AES instructions. Our experiments show that J is only 15% slower than EME.

Recall that EME uses about two blockcipher calls per message block. Since J would thus use three blockcipher calls per message block, one might naively expect a 50% slowdown. However, EME computes various offsets that are added to the blockcipher inputs and outputs, and these computations are quite costly, so our simple ECB pass is less expensive compared to EME than one might imagine. We remark that AES-NI is now widely available in Intel and AMD processors on modern machines, and disk controllers might potentially have hardware AES support as well, so using AES-NI as a starting point is realistic.

2 Preliminaries

Notation. By $\mathsf{Maps}(D, R)$ we denote the set of all functions $f\colon D \to R$. We denote by id the identity function. Some of our definitions and proofs are expressed via code-based games [4]. Recall that such a game — see Figure 1 for an example — consists of an Initialize procedure, procedures to respond to adversary oracle queries, and a Finalize procedure. A game G is executed with an adversary A as follows. First, Initialize executes. Then A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to Finalize, and the output of the latter is the output of the game. We denote by "G^A" the event that this game output takes value true. Boolean flags are assumed initialized to false. The running time of an adversary by convention is the worst case time for the execution of the adversary with the game defining its security, so that the time of the called game procedures is included.

Tweakable blockciphers. A *tweakable blockcipher* [15] is a map $E\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^m \to \{0,1\}^m$ that takes input a k-bit key K, a *tweak* T drawn from the tweakspace $\mathcal{T}$ and a m-bit message M to return an m-bit output $E(K, T, M)$. The map $E(K, T, \cdot)\colon \{0,1\}^m \to \{0,1\}^m$ that on input $M \in \{0,1\}^m$ returns $E(K, T, M)$ is required to be a permutation and its inverse is denoted $E^{-1}(K, T, \cdot)$.

The standard notion of security for a tweakable blockcipher is to be a tweakable pseudorandom permutation [15]. This can be considered under either chosen-plaintext attack (usually called PRP security) or chosen-ciphertext attack (usually called strong PRP security) [15, 16], but we will use the terms PRP-CPA and PRP-CCA as more indicative of the models and more consistent with notation for other primitives. We will be working with PRP-CCA. To define it consider games Real_E and Rand_E of

```
proc INITIALIZE  // $\mathrm{Real}_E, \mathrm{Real}_{E,\Phi}$
$K \leftarrow\$ \{0,1\}^k$
proc FN$(T, M)$  // $\mathrm{Real}_E, \mathrm{Real}_{E,\Phi}$
Return $E(K, T, M)$
proc KDFN$(T, \phi)$  // $\mathrm{Real}_{E,\Phi}$
$M \leftarrow \phi(K)$
Return $E(K, T, M)$
proc FN$^{-1}(T, C)$  // $\mathrm{Real}_E, \mathrm{Real}_{E,\Phi}$
Return $E^{-1}(K, T, C)$
proc FINALIZE$(b')$  // $\mathrm{Real}_E, \mathrm{Real}_{E,\Phi}$
Return $(b' = 1)$
```

```
proc INITIALIZE  // $\mathrm{Rand}_E, \mathrm{Rand}_{E,\Phi}$
$K \leftarrow\$ \{0,1\}^k$ ; $\pi \leftarrow\$ \mathsf{TwPm}(\mathcal{T}, \{0,1\}^m)$
proc FN$(T, M)$  // $\mathrm{Rand}_E, \mathrm{Rand}_{E,\Phi}$
Return $\pi(T, M)$
proc KDFN$(T, \phi)$  // $\mathrm{Rand}_{E,\Phi}$
$M \leftarrow \phi(K)$
Return $\pi(T, M)$
proc FN$^{-1}(T, C)$  // $\mathrm{Rand}_E, \mathrm{Rand}_{E,\Phi}$
Return $\pi^{-1}(T, C)$
proc FINALIZE$(b')$  // $\mathrm{Real}_E, \mathrm{Real}_{E,\Phi}$
Return $(b' = 1)$
```

Figure 1: Games $\mathrm{Real}_E, \mathrm{Rand}_E$ to define PRP-CCA security, and games $\mathrm{Real}_{E,\Phi}, \mathrm{Rand}_{E,\Phi}$ to define KDM PRP-CCA security of tweakable blockcipher E: $\{0,1\}^k \times \mathcal{T} \times \{0,1\}^m \to \{0,1\}^m$.

Figure 1. In Rand, we are denoting by $\mathsf{TwPm}(\mathcal{T}, \{0,1\}^m)$ the set of all π: $\mathcal{T} \times \{0,1\}^m \to \{0,1\}^m$ such that $\pi(T, \cdot)$ is a permutation on $\{0,1\}^m$ for each $T \in \mathcal{T}$. In this case, $\pi^{-1}(T, \cdot)$: $\{0,1\}^m \to \{0,1\}^m$ is the inverse of $\pi(T, \cdot)$. We define the prp-cca advantage of A as

$$\mathbf{Adv}_E^{\text{prp-cca}}(A) = \Pr[\mathrm{Real}_E^A] - \Pr[\mathrm{Rand}_E^A].$$

KDM PRP-CCA SECURITY. We extend PRP-CCA to allow for the encryption of key-dependent messages via games $\mathrm{Real}_{E,\Phi}$ and $\mathrm{Rand}_{E,\Phi}$ of Figure 1. Oracle KDFN takes input a tweak T and a function $\phi \in \mathsf{Maps}(\{0,1\}^k, \{0,1\}^m)$ and derives M as $\phi(K)$. If $\Phi \subseteq \mathsf{Maps}(\{0,1\}^k, \{0,1\}^m)$ we say that adversary A is Φ-restricted if the argument ϕ in its KDFN queries is always from Φ. We define the Φ-kdm-prp-cca advantage of such an A to be

$$\mathbf{Adv}_{E,\Phi}^{\text{prp-cca}}(A) = \Pr[\mathrm{Real}_{E,\Phi}^A] - \Pr[\mathrm{Rand}_{E,\Phi}^A].$$

We continue, thus, to denote the notion via prp-cca, the key-dependent messages indicated by the extra subscript Φ in the advantage function.

Consider the following strategy for A. It makes KDFN query T, id to get back a ciphertext C and then queries FN$^{-1}(T, C)$. The response will be $M = \mathrm{id}(K) = K$, and now A has the key and can easily win. (That is, get a high advantage, for example by returning 1 if $E(K, T, M') = \mathrm{FN}(T, M')$ for some T and some $M' \neq M$, and 0 otherwise.) To preclude this, we require that A is *legitimate*, meaning that, for all T, ϕ, it never makes a FN$^{-1}(T, C)$ query for C previously received in response to a KDFN(T, ϕ) query. The analogy is the definition of IND-CCA secure encryption where the adversary is not allowed to query to the decryption oracle a ciphertext it previously received as a challenge. The (necessary) assumption that adversaries are legitimate is made throughout and is implicit in all our results. We remark that we do not need to prohibit A from query FN$^{-1}(T, C)$ for C previously returned in response to query FN(T, M), because A already knows the message M it would get as response. So no restriction is present in the plain prp-cca notion. But when key-dependent messages are introduced, we must require legitimacy.

KDM-secure IND-CPA encryption was defined by [5]. Subsequently, KDM security of tweakable blockciphers was defined by Halevi and Krawczyk [12]. Because blockciphers, unlike encryption schemes, are deterministic, KDM security is not achievable when arbitrary functions ϕ are allowed in KDFN queries. This is analogous to what happens with related-key attacks (RKAs) and thus, as with the formalization of RKA security from [3], Halevi and Krawczyk [12] parameterize the advantage and definition of KDM security for tweakable blockciphers by a class Φ of functions. Our narrow-block design **StE** will achieve security when $\Phi = \{\mathrm{id}\}$, corresponding to encrypting the key.

Halevi and Krawczyk [12] do not explicitly state the legitimacy condition. They also allow the possibility of decryption of key-dependent ciphertexts, via an extra oracle. We have not considered this capability because, while plaintexts may be key dependent, we did not see why ciphertexts would be key dependent. Our schemes do not aim to achieve security in the presence of decryption of key-dependent ciphertexts.

3 Narrowblock Cipher

In this section we show how to construct a narrowblock tweakable blockcipher that can securely encipher its own key.

Construction. Let E: $\{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher whose key length and message length are the same value n. (For example, $n = 128$ for XEX-AES [17].) We fix an arbitrary tweak $\gamma \in \mathcal{T}$ as well as an arbitrary message $\alpha \in \{0,1\}^n$. Both γ and α are public parameters of the system known to the adversary. Our Swap-then-Encipher transform $\mathbf{StE}_{\gamma,\alpha}$ associates to E another tweakable blockcipher $F = \mathbf{StE}_{\gamma,\alpha}[E]$: $\{0,1\}^n \times \mathcal{T} \setminus \{\gamma\} \times \{0,1\}^n \to \{0,1\}^n$ whose tweak space $\mathcal{T} \setminus \{\gamma\}$ is that of E with the point γ removed. The function F is defined as follows:

```
$\underline{F(K,T,M)}$
01  $H \leftarrow E(K,\gamma,\alpha)$
02  If $M = K$ then $Y \leftarrow E(K,T,H)$
03  Else If $M = H$ then $Y \leftarrow E(K,T,K)$
04  Else $Y \leftarrow E(K,T,M)$
05  Return $Y$
```

Before considering security, We establish that F is invertible as required to be a tweakable blockcipher. Indeed, the inverse F^{-1} is given by

```
$\underline{F^{-1}(K,T,C)}$
01  $Y \leftarrow D(K,T,C)$
02  $H \leftarrow E(K,\gamma,\alpha)$
03  If $Y = K$ then $M \leftarrow H$
04  Else If $Y = H$ then $M \leftarrow K$
05  Return $M$
```

Efficiency. In the form it is described above, evaluating F requires two calls to the underlying cipher E. But in practice we are enciphering a message consisting of multiple blocks. In this case the value $E(K,\gamma,\alpha)$ can be computed just once and then cached, which means each evaluation of $F(K,T,M)$ requires one call to E with the same key and tweak value. Similarly, $E(K,\gamma,\alpha)$ can also be cached for inversion. Thus, the amortized cost of $\mathbf{StE}_{\gamma,\alpha}$ is the same as that of E. We remark that the situation here is analogous to that of XEX [17]. XEX too needs two applications of the underlying blockcipher, but one can usually be amortized out. **StE** is no worse.

Security. The following theorem says that $F = \mathbf{StE}_{\gamma,\alpha}[E]$ is a PRP-CCA that can securely encipher its own key, or more formally, that it is $\{\mathrm{id}\}$-kdm-prp-cca secure, assuming only that E meets the standard PRP-CCA notion of security.

Theorem 3.1 *Let E: $\{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. Let $\gamma \in \mathcal{T}$ and $\alpha \in \{0,1\}^n$. Let $F = \mathbf{StE}_{\gamma,\alpha}[E]$: $\{0,1\}^n \times \mathcal{T} \setminus \{\gamma\} \times \{0,1\}^n \to \{0,1\}^n$ be the tweakable blockcipher associated to E via the $\mathbf{StE}_{\gamma,\alpha}$ transform as defined above. Let $\Phi = \{\mathrm{id}\}$ consist of the identity function. Let A be an adversary making at most Q oracle queries. Then there is an adversary B such that*

$$\mathbf{Adv}^{\text{prp-cca}}_{F,\Phi}(A) \;<\; 2 \cdot \mathbf{Adv}^{\text{prp-cca}}_{E}(B) + \frac{3Q}{2^n - 1}\;.$$

```
proc Initialize // $G_0, G_1$
$K \leftarrow\!\!\$\ \{0,1\}^n$ ; $H \leftarrow E(K, \gamma, \alpha)$

proc Fn$(T, M)$ // $\boxed{G_0}$ , $G_1$
$X \leftarrow M$
If $M = K$ then
    bad $\leftarrow$ true ; $\boxed{X \leftarrow H}$
Else If $M = H$ then
    bad $\leftarrow$ true ; $\boxed{X \leftarrow K}$
Return $E(K, T, X)$

proc KDFn$(T, \phi)$ // $G_0, G_1$
Return $E(K, T, H)$

proc Fn$^{-1}(T, C)$ // $\boxed{G_0}$, $G_1$
$X \leftarrow E^{-1}(K, T, C)$ ; $M \leftarrow X$
If $X = K$ then
    bad $\leftarrow$ true ; $\boxed{M \leftarrow H}$
Else If $X = H$ then
    bad $\leftarrow$ true ; $\boxed{M \leftarrow K}$
Return $M$

proc Finalize$(b')$ // $G_0, G_1, G_2$
Return $(b' = 1)$

proc Finalize$(b')$ // $G_3, G_4$
Return $(\mathsf{bad}_1 \vee \mathsf{bad}_2)$
```

```
proc Initialize // $G_2$
$\pi \leftarrow\!\!\$\ \mathsf{TwPm}(\mathcal{T}, \{0,1\}^n)$
$H \leftarrow \pi(\gamma, \alpha)$

proc Fn$(T, M)$ // $G_2$
Return $\pi(T, M)$

proc KDFn$(T, \phi)$ // $G_2$
Return $\pi(T, H)$

proc Fn$^{-1}(T, C)$ // $G_2$
Return $\pi^{-1}(T, C)$

proc Initialize // $G_3$
$K \leftarrow\!\!\$\ \{0,1\}^n$
$H \leftarrow E(K, \gamma, \alpha)$
$H' \leftarrow E(K, \gamma, \alpha')$

proc Initialize // $G_4$
$\pi \leftarrow\!\!\$\ \mathsf{TwPm}(\mathcal{T}, \{0,1\}^n)$
$H \leftarrow \pi(\gamma, \alpha)$
$H' \leftarrow \pi(\gamma, \alpha')$
```

```
proc Fn$(T, M)$ // $G_3$
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow$ true
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow$ true
Return $E(K, T, M)$

proc KDFn$(T, \phi)$ // $G_3$
Return $E(K, T, H)$

proc Fn$^{-1}(T, C)$ // $G_3$
$M \leftarrow E^{-1}(K, T, C)$
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow$ true
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow$ true
Return $M$

proc Fn$(T, M)$ // $G_4$
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow$ true
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow$ true
Return $\pi(T, M)$

proc KDFn$(T, \phi)$ // $G_4$
Return $\pi(T, H)$

proc Fn$^{-1}(T, C)$ // $G_4$
$M \leftarrow \pi^{-1}(T, C)$
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow$ true
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow$ true
Return $M$
```

Figure 2: Games used in the proof of Theorem 3.1.

Moreover, the running time of B is about equal to the running time of A.

The intuition is that, during the $\mathrm{Real}_{F,\{\mathrm{id}\}}$ and $\mathrm{Rand}_{F,\{\mathrm{id}\}}$ games, an adversary is unlikely to trigger the test in lines 02 and 03 in the computation of F, except by submitting $\phi = \mathrm{id}$ to its oracle. Once this established, we can give a reduction to the prp-cca security of E. For the first part, we show that K and H are hard to guess using the prp-cca security of E. If the adversary guesses K, then we can show how to break the prp-cca security of E, a contradiction. We can show the same if the adversary guesses H. This is where we use the fact that the tweak γ is never allowed elsewhere, effectively making H look random. The proof implements this approach and deals with other complications.

Proof: We begin with the games in Figure 2. Recall that here $\Phi = \{\mathrm{id}\}$, so it is assumed that $\phi = \mathrm{id}$ in any KDFn query. Game G_0 includes the boxed code and hence is the same as Real_F^A, and thus we have

$$\begin{aligned}
\mathbf{Adv}^{\text{prp-cca}}_{F,\Phi}(A) &= \Pr[G_0^A] - \Pr[G_2^A] \\
&= \Pr[G_1^A] - \Pr[G_2^A] + \Pr[G_0^A] - \Pr[G_1^A] \\
&\le \Pr[G_1^A] - \Pr[G_2^A] + \Pr[\mathrm{BAD}(G_1^A)] \ .
\end{aligned}$$

The inequality is by the Fundamental Lemma of Game Playing [4] since G_0, G_1 are identical until bad. We design adversary B_1 so that

$$\Pr[G_1^A] - \Pr[G_2^A] \le \mathbf{Adv}^{\text{prp-cca}}_{E}(B_1) \ . \tag{1}$$

The simulation is straightforward since G_1 does not include the boxed code. B_1 has oracles Fn, Fn^{-1}. It starts by letting $H \leftarrow \mathrm{Fn}(\gamma, \alpha)$. It then runs A. When A makes a query (T, M) to its Fn oracle,

B_1 queries its own $\textsc{Fn}$ oracle with (T, M) and forwards the response to A. Similarly when A makes a query (T, C) to its $\textsc{Fn}^{-1}$ oracle, B_1 queries its own $\textsc{Fn}^{-1}$ oracle with (T, C) and forwards the response to A. When A queries its $\text{KDF}\textsc{n}$ oracle with T, id, adversary B_1 responds with $\textsc{Fn}(T, H)$. When A halts with output b', so does B_1. We clearly have

$$\Pr[\text{Real}_E^{B_1}] = \Pr[\text{G}_1^A] \; .$$

Since the tweak γ is not used in any queries of A, the point H in G_2 is random and can be viewed as playing the role of K in game Rand_E. Thus we also have

$$\Pr[\text{Rand}_E^{B_1}] = \Pr[\text{G}_2^A] \; .$$

Thus we have Equation (1).

It remains to upper bound $\Pr[\textsc{Bad}(\text{G}_1^A)]$. Fix a point $\alpha' \in \{0,1\}^n \setminus \{\alpha\}$ and consider games G_3, G_4 of Figure 2. Game G_3 replaces the $M = K$ test from G_1 with the test $E(M, \gamma, \alpha') = H'$. The purpose is to avoid referring explicitly to K, thereby opening the way for a reduction to the assumed prp-cca security of E. This new test, however, will certainly return true if $M = K$ and hence

$$\begin{aligned} \Pr[\textsc{Bad}(\text{G}_1^A)] &\le \Pr[\text{G}_3^A] \\ &= \Pr[\text{G}_3^A] - \Pr[\text{G}_4^A] + \Pr[\text{G}_4^A] \; . \end{aligned}$$

We stress that in G_4, where we move to the random world by replacing $E(K, \cdot, \cdot)$ by $\pi(\cdot, \cdot)$, the test still uses E, invoking the blockcipher here explicitly on inputs M, γ, α'. (The test does not use π.) We design adversary B_2 so that

$$\Pr[\text{G}_3^A] - \Pr[\text{G}_4^A] \le \mathbf{Adv}_E^{\text{prp-cca}}(B_2) \; . \tag{2}$$

The simulation is again straightforward now that K is not referred to explicitly in either game. B_2 has oracles $\textsc{Fn}, \textsc{Fn}^{-1}$. It starts by letting $H \leftarrow \textsc{Fn}(\gamma, \alpha)$ and $H' \leftarrow \textsc{Fn}(\gamma, \alpha')$ and initializing boolean variables $\mathsf{bad}_1, \mathsf{bad}_2$ to false. It then runs A. When A makes a query (T, M) to its $\textsc{Fn}$ oracle, B_2 does the following:

```
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return $\textsc{Fn}(T, M)$ to $A$
```

Similarly when A makes a query (T, C) to its $\textsc{Fn}^{-1}$ oracle, B_2 does the following:

```
$M \leftarrow \textsc{Fn}^{-1}(T, C)$
If $E(M, \gamma, \alpha') = H'$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
Else If $M = H$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return $M$ to $A$
```

When A queries its $\text{KDF}\textsc{n}$ oracle with T, id, adversary B responds with $\textsc{Fn}(T, H)$. When A halts with output b', adversary B halts with output 1 if $(\mathsf{bad}_1 \vee \mathsf{bad}_2)$ has value true and 0 otherwise. We have

$$\Pr[\text{Real}_E^{B_2}] = \Pr[\text{G}_3^A] \text{ and } \Pr[\text{Rand}_E^{B_2}] = \Pr[\text{G}_4^A]$$

which implies Equation (2).

Let B be the adversary that picks c at random from $\{1, 2\}$ and runs B_c. Then at this point we have

$$\mathbf{Adv}_{F,\Phi}^{\text{prp-cca}}(A) \le 2 \cdot \mathbf{Adv}_E^{\text{prp-cca}}(B) + \Pr[\text{G}_4^A]$$

and proceed to upper bound the last term above, considering separately the probability of setting bad_1 and that of setting bad_2. Since γ is not in the tweak space of F, game G_5 of Figure 3 picks π from

```
proc INITIALIZE  // $G_5$
$\pi \leftarrow_\$ \mathsf{TwPm}(\mathcal{T} \setminus \{\gamma\}, \{0,1\}^n)$
$H \leftarrow_\$ \{0,1\}^n$ ; $S \leftarrow \emptyset$

proc FN$(T, M)$  // $G_5$
$S \leftarrow S \cup \{E(M, \gamma, \alpha')\}$
Return $\pi(T, M)$

proc KDFN$(T, \phi)$  // $G_5$
Return $\pi(T, H)$

proc FN$^{-1}(T, C)$  // $G_5$
$M \leftarrow \pi^{-1}(T, C)$
$S \leftarrow S \cup \{E(M, \gamma, \alpha')\}$
Return $M$

proc FINALIZE$(b')$  // $G_5$
$H' \leftarrow_\$ \{0,1\}^n \setminus \{H\}$
Return $(H' \in S)$
```

```
proc INITIALIZE  // $G_6, G_7$
For all $T \in \mathcal{T} \setminus \{\gamma\}$ do
    $Y_T \leftarrow_\$ \{0,1\}^n$ ; $R(T) \leftarrow \{Y_T\}$
$H \leftarrow_\$ \{0,1\}^n$

proc FN$(T, M)$  // $\boxed{G_6}$, $G_7$
If $\pi[T, M]$ then return $\pi[T, M]$
$C \leftarrow_\$ \{0,1\}^n \setminus R(T)$
If $M = H$ then
    bad $\leftarrow$ true ; $\boxed{C \leftarrow Y_T}$
$D(T) \leftarrow D(T) \cup \{M\}$
$R(T) \leftarrow R(T) \cup \{C\}$
$\pi[T, M] \leftarrow C$ ; $\pi^{-1}[T, C] \leftarrow M$
Return $\pi[T, M]$

proc KDFN$(T, \phi)$  // $G_6, G_7$
Return $Y_T$

proc FN$^{-1}(T, C)$  // $\boxed{G_6}$, $G_7$
If $\pi^{-1}[T, C]$ then return $\pi^{-1}[T, C]$
$M \leftarrow_\$ \{0,1\}^n \setminus D(T)$
If $C = Y_T$ then
    bad $\leftarrow$ true ; $\boxed{M \leftarrow H}$
Else If $M = H$ then
    bad $\leftarrow$ true
    $\boxed{M \leftarrow_\$ \{0,1\}^n \setminus (D(T) \cup \{H\})}$
$D(T) \leftarrow D(T) \cup \{M\}$
$R(T) \leftarrow R(T) \cup \{C\}$
$\pi[T, M] \leftarrow C$ ; $\pi^{-1}[T, C] \leftarrow M$
Return $M$

proc FINALIZE$(b')$  // $G_6, G_7$
Return bad
```

```
proc INITIALIZE  // $G_8$
For all $T \in \mathcal{T} \setminus \{\gamma\}$ do
    $Y_T \leftarrow_\$ \{0,1\}^n$ ; $R(T) \leftarrow \{Y_T\}$
$S \leftarrow \emptyset$

proc FN$(T, M)$  // $G_8$
If $\pi[T, M]$ then return $\pi[T, M]$
$C \leftarrow_\$ \{0,1\}^n \setminus R(T)$
$S \leftarrow S \cup \{M\}$
$D(T) \leftarrow D(T) \cup \{M\}$
$R(T) \leftarrow R(T) \cup \{C\}$
$\pi[T, M] \leftarrow C$ ; $\pi^{-1}[T, C] \leftarrow M$
Return $\pi[T, M]$

proc KDFN$(T, \phi)$  // $G_8$
Return $Y_T$

proc FN$^{-1}(T, C)$  // $G_8$
If $\pi^{-1}[T, C]$ then return $\pi^{-1}[T, C]$
$M \leftarrow_\$ \{0,1\}^n \setminus D(T)$
If $C = Y_T$ then bad $\leftarrow$ true
Else $S \leftarrow S \cup \{M\}$
$D(T) \leftarrow D(T) \cup \{M\}$
$R(T) \leftarrow R(T) \cup \{C\}$
$\pi[T, M] \leftarrow C$ ; $\pi^{-1}[T, C] \leftarrow M$
Return $M$

proc FINALIZE$(b')$  // $G_8$
$H \leftarrow_\$ \{0,1\}^n$
Return $(H \in S) \vee$ bad
```

Figure 3: More games used in the proof of Theorem 3.1.

$\mathsf{TwPm}(\mathcal{T} \setminus \{\gamma\}, \{0,1\}^n)$ rather than $\mathsf{TwPm}(\mathcal{T}, \{0,1\}^n)$, and picks H, H' as random, distinct points. The game can move the setting of bad_1 and the choice of H' to FINALIZE without impacting what is returned to the adversary. At the end of the execution, the set S can have size at most Q so

$$\Pr[G_4^A \text{ sets } \mathsf{bad}_1] = \Pr[G_5^A] \leq \frac{Q}{2^n - 1} .$$

Bounding the probability that bad_2 is set in G_4 is more difficult because information about H does reach the adversary via the KDFN query. Our intent is to move to a game where H is not referred to in replying to adversary oracle queries. We begin with game G_6 of Figure 3 which samples π lazily. The arrays $\pi[\cdot,\cdot]$ and $\pi^{-1}[\cdot,\cdot]$ are assumed initially everywhere undefined, and get filled in as the game progresses. A test "If $\pi[T, M]$" returns true if $\pi[T, M]$ is defined, and false otherwise, and similarly for "If $\pi^{-1}[T, C]$". The game begins by picking a random Y_T for each T that is intended to stand for $\pi[T, H]$ but not assigned to the latter so as to avoid using H. Instead, whenever $\pi[T, H]$ or $\pi^{-1}[T, Y_T]$ are called for, the game sets bad and corrects via the boxed code, which is included in G_6. A variant of the Fundamental Lemma of Game Playing from [4] says that identical until bad games have the same probability of setting bad and hence

$$\Pr[G_4^A \text{ sets } \mathsf{bad}_2] = \Pr[G_6^A] = \Pr[G_7^A] \; .$$

But game G_7, which excludes the boxed code, does not refer to H in replying to adversary oracle queries, and hence

$$\Pr[G_7^A] = \Pr[G_8^A] \; .$$

Since A is legitimate, a $\mathrm{FN}^{-1}(T, C)$ query with $C = Y_T$ must occur before it receives Y_T from KDFN and hence is made with no knowledge of Y_T. Thus the probability that a query of A sets bad is at most 2^{-n}. (We remark that this is the place we use the assumption that A is legitimate, without which bad could be set with probability one.) On the other hand the set S has size at most Q at the end of the execution of A with the game. So $\Pr[G_8^A] \leq 2Q \cdot 2^{-n}$. Putting this together with the above concludes the proof. ▌

The IEEE 1619 standard for narrowblock encryption is based on XEX-AES [17]. **StE** is an alternative narrowblock candidate that is essentially as efficient as the current one but also has provable security for enciphering the key.

4 Wideblock cipher

We provide a simple and general way to enhance a given PRP-CCA wideblock cipher to be able to encrypt messages that might, in any block, contain the key. Our construction simply puts an ECB layer in front of the given cipher and then uses **StE** as the base tweakable blockcipher for both the ECB pass and the application of the wideblock cipher. This works for any given wideblock cipher that is a mode of operation of a blockcipher, meaning it uses the key only to key an underlying blockcipher, as is the case with EME and other standard designs. **StE** is used with one tweak for the ECB pass and then, by fixing a different tweak, yields a blockcipher to instantiate the wideblock mode of operation.

We begin, below, by defining what it means for a wideblock design to be a mode of operation and what it means to assume it is PRP-CCA secure. Then we provide our construction and prove it secure.

MODES. A wideblock mode of operation $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ is a pair of oracle algorithms. Given oracle access to a permutation $g\colon \{0,1\}^n \to \{0,1\}^n$, algorithm WB takes input a tweak $T \in \mathcal{T}$ and an m-block message $X \in \{0,1\}^{mn}$ to return an m-block ciphertext denoted $\mathsf{WB}(T, X : g)$. Given oracle access to g^{-1}, algorithm WB^{-1} takes input a tweak $T \in \mathcal{T}$ and an m-block ciphertext $Y \in \{0,1\}^{mn}$ to return an m-block message denoted $\mathsf{WB}^{-1}(T, Y : g^{-1})$. It is required that $\mathsf{WB}^{-1}(T, \mathsf{WB}(T, X : g) : g^{-1}) = X$ for all choices of the inputs and oracles. Here $\mathcal{T}, n, m$ are the tweak space, blocklength and number of blocks associated to $\mathbf{WB}$.

If $N\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a blockcipher then $\mathbf{WB}$ associates to it a tweakable blockcipher $W = \mathbf{WB}[N]$ where $W\colon \{0,1\}^n \times \mathcal{T} \times \{0,1\}^{mn} \to \{0,1\}^{mn}$ is defined by $W(K, T, X) = \mathsf{WB}(T, X : E(K, \cdot))$ and $W^{-1}(K, T, Y) = \mathsf{WB}^{-1}(T, Y : E^{-1}(K, \cdot))$. EME is an example of a wideblock mode of operation that, in this way, transforms a given blockcipher into a wideblock tweakable cipher.

Let $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ be a wideblock mode of operation. Consider the games of Figure 4 and let

$$\mathbf{Adv}^{\text{prp-cca}}_{\mathbf{WB}}(B) = \Pr[\mathrm{RO}^B_{\mathbf{WB}}] - \Pr[\mathrm{Rand}^B_{mn}] \; .$$

Game $\mathrm{RO}_{\mathbf{WB}}$ instantiates the oracles of WB and WB^{-1} with a random narrow block permutation and its inverse, respectively, while game Rand_{mn} responds to oracle queries via a random wideblock permutation. Now let

$$\mathbf{Adv}^{\text{prp-cca}}_{\mathbf{WB}}(Q, n, m, t) = \max_B \mathbf{Adv}^{\text{prp-cca}}_{\mathbf{WB}}(B)$$

where the maximum is over all adversaries B making at most Q oracle queries with the tweak argument in each query having length at most nt. Proving security of a mode of operation ubiquitously proceeds

```
proc Initialize // $\mathrm{RO}_{\mathbf{WB}}$
$\pi \leftarrow_\$ \mathsf{Pm}(\{0,1\}^m)$
proc Fn$(T,M)$ // $\mathrm{RO}_{\mathbf{WB}}$
Return $\mathsf{WB}(T, M : \pi)$
proc Fn$^{-1}(T,C)$ // $\mathrm{RO}_{\mathbf{WB}}$
Return $\mathsf{WB}^{-1}(T, C : \pi^{-1})$
proc Finalize$(b')$ // $\mathrm{RO}_{\mathbf{WB}}$
Return $(b' = 1)$
```

```
proc Initialize // $\mathrm{Rand}_{mn}$
$\pi \leftarrow_\$ \mathsf{TwPm}(\mathcal{T}, \{0,1\}^{mn})$
proc Fn$(T,M)$ // $\mathrm{Rand}_{mn}$
Return $\pi(T,M)$
proc Fn$^{-1}(T,C)$ // $\mathrm{Rand}_{mn}$
Return $\pi^{-1}(T,C)$
proc Finalize$(b')$ // $\mathrm{Rand}_{mn}$
Return $(b' = 1)$
```

```
proc Initialize // J
$\pi \leftarrow_\$ \mathsf{TwPm}(\mathcal{T}, \{0,1\}^n)$
$K \leftarrow_\$ \{0,1\}^n$
proc Fn$(T,M)$ // J
Return $\pi(T,M)$
proc KDFn$(T, \mathrm{id}_M)$ // J
For $i = 1, \ldots, m$ do
    If $M[i] = \perp$ then $M[i] \leftarrow K$
Return $\pi(T,M)$
proc Fn$^{-1}(T,C)$ // J
Return $\pi^{-1}(T,C)$
proc Finalize$(b')$ // J
Return $(b' = 1)$
```

Figure 4: On the left are games $\mathrm{RO}_{\mathbf{WB}}, \mathrm{Rand}_{mn}$ to define security of wideblock mode of operation $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$. On the right is the final game J for the proof of Theorem 4.1.

by upper bounding this advantage as a function of Q, n, m, t. This is a purely information theoretic setting, and absolute bounds are provided. For example, for EME, it is shown in [14] that this advantage is at most $7(Qt + qn + 1)^2/2^n$. (EME requires $m \le n$, assumed in this bound.) PRP-CCA security of $\mathbf{WB}[N]$ follows from the assumption that N is a PRP-CCA secure blockcipher by a standard reduction argument.

Construction. Let ECB denote the oracle algorithm that given oracle access to a permutation g: $\{0,1\}^n \to \{0,1\}^n$, and given input $M = M[1] \ldots M[m] \in \{0,1\}^{mn}$, returns the m-bit string $C = \mathsf{ECB}(M : g)$ defined by $C[i] = g(M[i])$ for $1 \le i \le m$. Let $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ be a wideblock mode of operation with tweakspace $\mathcal{T}$, blocklength n and number of blocks m. Let F: $\{0,1\}^n \times \{\gamma_1, \gamma_2\} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher with a tweakspace of size two. (Of course, any tweakable blockcipher with an even larger tweakspace will do. Just drop all but two possible tweaks.) Our $\mathbf{EtE}$ (ECB-then-encipher) transform associates to $\mathbf{WB}$ and F a tweakable wideblock cipher $J = \mathbf{EtE}[\mathbf{WB}, F]$ where J: $\{0,1\}^n \times \mathcal{T} \times \{0,1\}^{mn} \to \{0,1\}^{mn}$ is defined as follows:

```
$J(K,T,M)$
01  $X \leftarrow \mathsf{ECB}(M : F(K, \gamma_1, \cdot))$
02  $Y \leftarrow \mathsf{WB}(T, X : F(K, \gamma_2, \cdot))$
03  Return $Y$
```

The inverse is defined by

```
$J^{-1}(K,T,Y)$
01  $X \leftarrow \mathsf{WB}^{-1}(T, Y : F^{-1}(K, \gamma_2, \cdot))$
02  $M \leftarrow \mathsf{ECB}(X : F^{-1}(K, \gamma_1, \cdot))$
03  Return $X$
```

We take advantage here of the fact that $\mathsf{ECB}(\cdot : g^{-1})$ is the inverse of $\mathsf{ECB}(\cdot : g)$.

When instantiated with EME in the role of $\mathbf{WB}$ this uses three blockcipher invocations per block, but retains the parallelizability of EME. F would be obtained by applying $\mathbf{StE}$ to some tweakable blockcipher with a tweakspace of size three.

The class $\mathcal{ID}_m$. We formally define the class Φ capturing occurrence of the key in any block of the message. Associate to any m-vector M over $\{0,1\}^n \cup \{\perp\}$ the function id_M: $\{0,1\}^n \to \{0,1\}^{mn}$ that on input a key $K \in \{0,1\}^n$ returns the message $M' = M'[1] \ldots M'[m]$ defined by $M'[i] = M[i]$ if $M[i] \neq \perp$ and $M'[i] = K$ if $M[i] = \perp$ for all $1 \leq i \leq m$. Then $\mathcal{ID}_m$ is the class of all id_M as M ranges over all m-vectors over $\{0,1\}^n \cup \{\perp\}$. This is the class Φ we will consider.

SECURITY. Assume $\mathbf{WB}$ is PRP-CCA secure and assume F is {id}-PRP-CCA secure, meaning can safely encrypt its own key. We claim that J is $\mathcal{ID}_m$-PRP-CCA secure, meaning can safely encrypt messages that contain the key in any block.

Theorem 4.1 *Let* $\mathbf{WB} = (\mathsf{WB}, \mathsf{WB}^{-1})$ *be a wideblock mode of operation with tweakspace* $\mathcal{T}$*, blocklength* n *and number of blocks* m*. Let* $F\colon \{0,1\}^n \times \{\gamma_1, \gamma_2\} \times \{0,1\}^n \to \{0,1\}^n$ *be a tweakable blockcipher. Let* $J = \mathbf{EtE}[\mathbf{WB}, F]$ *be the wideblock tweakable cipher associated to* $\mathbf{WB}$ *and* F *via the* $\mathbf{EtE}$ *transform as defined above. Let* $\Phi = \mathcal{ID}_m$*. Let* A *be an adversary making at most* Q *oracle queries with the tweak argument in each query having length at most* nt*. Then there is an adversary* B *such that*

$$\mathbf{Adv}^{\text{prp-cca}}_{J,\Phi}(A) \leq 2 \cdot \mathbf{Adv}^{\text{prp-cca}}_{F,\{\text{id}\}}(B) + 2\delta(\mathbf{WB}) + \frac{2Q^2(m^2+2)}{2^{n+2}} ,$$

where $\delta(\mathbf{WB}) = \mathbf{Adv}^{\text{prp-cca}}_{\mathbf{WB}}(Q,n,m,t)$*. Moreover, the running time of* B *is about equal to the running time of* A*.*

The proof would seem at first to be a quite straightforward simulation in which KDFN queries of A can be answered via KDFN queries of B. The subtle issue is that B needs to be legitimate and this means it cannot answer all FN^{-1} queries of A. Ensuring B is legitimate makes the proof more involved.

Proof: We begin with the games in Figure 5. We have

$$\begin{aligned}
\Pr[\text{Real}^A_{J,\mathcal{ID}_m}] &= \Pr[\text{G}^A_0] \\
&= \Pr[\text{G}^A_0] - \Pr[\text{G}^A_1] + \Pr[\text{G}^A_1] \\
&\leq \Pr[\text{G}^A_1] + \Pr[\text{BAD}(\text{G}^A_1)] , \qquad (3)
\end{aligned}$$

the inequality by the Fundamental Lemma of Game Playing [4]. We will design (legitimate!) B_1, B_2 so that

$$\Pr[\text{G}^A_1] - \Pr[\text{H}^A] \leq \mathbf{Adv}^{\text{prp-cca}}_{F,\{\text{id}\}}(B_1) \qquad (4)$$

$$\Pr[\text{BAD}(\text{G}^A_1)] - \Pr[\text{BAD}(\text{H}^A)] \leq \mathbf{Adv}^{\text{prp-cca}}_{F,\{\text{id}\}}(B_2) . \qquad (5)$$

Adversaries B_1, B_2 are almost the same, differing only in how they take their final decision, and accordingly we unify their descriptions. For $i \in \{1,2\}$, adversary B_i has access to oracles $\text{FN}, \text{KDFN}, \text{FN}^{-1}$ and simulates oracles of the same name for A. It starts by setting $S \leftarrow \emptyset$ and $\mathsf{bad} \leftarrow \mathsf{false}$ and then it runs A, answering its oracle queries as follows. When A queries $\text{FN}(T, M)$, adversary B_i does the following:

```
For $i = 1, \ldots, m$ do $X[i] \leftarrow \text{FN}(\gamma_1, M[i])$
$Y \leftarrow \mathsf{WB}(T, X : \text{FN}(\gamma_2, \cdot)))$
Return $Y$ to $A$.
```

The FN calls made here by B_i are to its own FN oracle. When A queries $\text{KDFN}(T, \text{id}_M)$, adversary B_i does the following:

```
For $i = 1, \ldots, m$ do
    If $M[i] = \perp$ then $X[i] \leftarrow \text{KDFN}(\gamma_1, \text{id})$
    Else $X[i] \leftarrow \text{FN}(\gamma_1, M[i])$
$S \leftarrow S \cup \{X[i] : 1 \leq i \leq m\}$
$Y \leftarrow \mathsf{WB}(T, X : \text{FN}(\gamma_2, \cdot)))$
Return $Y$ to $A$.
```

```
proc INITIALIZE // $G_0, G_1$
$K \leftarrow_{\$} \{0,1\}^n$ ; $S \leftarrow \emptyset$

proc FN$(T, M)$ // $G_0, G_1$
$X \leftarrow \mathsf{ECB}(M : F(K, \gamma_1, \cdot))$
$Y \leftarrow \mathsf{WB}(T, X : F(K, \gamma_2, \cdot))$
Return $Y$

proc KDFN$(T, \mathrm{id}_M)$ // $G_0, G_1$
For $i = 1, \ldots, m$ do
    If $M[i] = \bot$ then $M[i] \leftarrow K$
$X \leftarrow \mathsf{ECB}(M : F(K, \gamma_1, \cdot))$
$S \leftarrow S \cup \{ X[i] \ : \ 1 \le i \le m \}$
$Y \leftarrow \mathsf{WB}(T, X : F(K, \gamma_2, \cdot))$
Return $Y$

proc FN$^{-1}(T, C)$ // $\boxed{G_0}$, $G_1$
$X \leftarrow \mathsf{WB}^{-1}(T, C : F^{-1}(K, \gamma_2, \cdot))$
$S' \leftarrow \{ X[i] \ : \ 1 \le i \le m \}$
$M \leftarrow \bot$
If $S \cap S' \neq \emptyset$ then bad $\leftarrow$ true
    $\boxed{M \leftarrow \mathsf{ECB}(X : F^{-1}(K, \gamma_1, \cdot))}$
Else $M \leftarrow \mathsf{ECB}(X : F^{-1}(K, \gamma_1, \cdot))$
Return $M$

proc FINALIZE$(b')$ // $G_0, G_1$
Return $(b' = 1)$
```

```
proc INITIALIZE // H
$\pi \leftarrow_{\$} \mathsf{TwPm}(\{\gamma_1, \gamma_2\}, \{0,1\}^n)$
$K \leftarrow_{\$} \{0,1\}^n$ ; $S \leftarrow \emptyset$

proc FN$(T, M)$ // H
$X \leftarrow \mathsf{ECB}(M : \pi(\gamma_1, \cdot))$
$Y \leftarrow \mathsf{WB}(T, X : \pi(\gamma_2, \cdot))$
Return $Y$

proc KDFN$(T, \mathrm{id}_M)$ // H
For $i = 1, \ldots, m$ do
    If $M[i] = \bot$ then $M[i] \leftarrow K$
$X \leftarrow \mathsf{ECB}(M : \pi(\gamma_1, \cdot))$
$S \leftarrow S \cup \{ X[i] \ : \ 1 \le i \le m \}$
$Y \leftarrow \mathsf{WB}(T, X : \pi(\gamma_2, \cdot))$
Return $Y$

proc FN$^{-1}(T, C)$ // H
$X \leftarrow \mathsf{WB}^{-1}(T, C : \pi^{-1}(\gamma_2, \cdot))$
$S' \leftarrow \{ X[i] \ : \ 1 \le i \le m \}$
$M \leftarrow \bot$
If $S \cap S' \neq \emptyset$ then bad $\leftarrow$ true
Else $M \leftarrow \mathsf{ECB}(X : \pi^{-1}(\gamma_1, \cdot))$
Return $M$

proc FINALIZE$(b')$ // H
Return $(b' = 1)$
```

```
proc INITIALIZE // $I_0, I_1$
$\pi_1 \leftarrow_{\$} \mathsf{TwPm}(\{\gamma_1\}, \{0,1\}^n)$
$\pi_2 \leftarrow_{\$} \mathsf{TwPm}(\mathcal{T}, \{0,1\}^n)$
$K \leftarrow_{\$} \{0,1\}^n$ ; $S \leftarrow \emptyset$

proc FN$(T, M)$ // $I_0, I_1$
$X \leftarrow \mathsf{ECB}(M : \pi_1(\gamma_1, \cdot))$
$Y \leftarrow \pi_2(T, X)$
Return $Y$

proc KDFN$(T, \mathrm{id}_M)$ // $I_0, I_1$
For $i = 1, \ldots, m$ do
    If $M[i] = \bot$ then $M[i] \leftarrow K$
$X \leftarrow \mathsf{ECB}(M : \pi_1(\gamma_1, \cdot))$
$S \leftarrow S \cup \{ X[i] \ : \ 1 \le i \le m \}$
$Y \leftarrow \pi_2(T, X)$
Return $Y$

proc FN$^{-1}(T, C)$ // $\boxed{I_0}$, $I_1$
$X \leftarrow \pi_2^{-1}(T, C)$
$S' \leftarrow \{ X[i] \ : \ 1 \le i \le m \}$
$M \leftarrow \bot$
If $S \cap S' \neq \emptyset$ then bad $\leftarrow$ true
    $\boxed{M \leftarrow \mathsf{ECB}(X : \pi_1^{-1}(\gamma_1, \cdot))}$
Else $M \leftarrow \mathsf{ECB}(X : \pi_1^{-1}(\gamma_1, \cdot))$
Return $M$

proc FINALIZE$(b')$ // $I_0, I_1$
Return $(b' = 1)$
```

Figure 5: Games used in the proof of Theorem 4.1.

Again the calls made by B_i in the code above are to its own FN, KDFN oracles. When A queries FN$^{-1}(T, C)$, adversary B_i does the following:

```
$X \leftarrow \mathsf{WB}^{-1}(T, X : \text{FN}^{-1}(\gamma_2, \cdot)))$
$S' \leftarrow \{X[i] : 1 \le i \le m\}$
$M \leftarrow \bot$
If $S \cap S' = \emptyset$ then
    For $i = 1, \ldots, m$ do $M[i] \leftarrow \text{FN}^{-1}(\gamma_1, X[i])$
Else bad $\leftarrow$ true
Return $M$ to $A$.
```

As before, the calls made by B_i here are to its own FN^{-1} oracle. So far there has been no difference between B_1, B_2 but now, when A halts, they compute their outputs differently, B_1 returning the same output as A, but B_2 returning 1 if bad = true and 0 otherwise.

We claim that B_i is legitimate, meaning that it never queries FN^{-1} at a point (T, C) which was output by a call to KDFN with tweak T. The FN^{-1} queries issued by B_1 occur at two points, both when processing FN^{-1} queries issued by A. Note that B_i only queries KDFN with tweak γ_1, and since the FN^{-1} queries used in the computation of WB^{-1} are all with the distinct tweak γ_2, these queries will not violate legitimacy. The other queries made by B_i to FN^{-1}, which *are* under γ_1, are only made if $S \cap S' = \emptyset$, and hence do not violate legitimacy. This explains why bad is set this way in the games.

Now we have

$$\begin{aligned}
\Pr[\mathrm{Real}_{F,\{\mathrm{id}\}}^{B_1}] &= \Pr[\mathrm{G}_0^A] \\
\Pr[\mathrm{Rand}_{F,\{\mathrm{id}\}}^{B_1}] &= \Pr[\mathrm{H}^A] \\
\Pr[\mathrm{Real}_{F,\{\mathrm{id}\}}^{B_2}] &= \Pr[\textsc{Bad}(\mathrm{G}_0^A)] \\
\Pr[\mathrm{Rand}_{F,\{\mathrm{id}\}}^{B_2}] &= \Pr[\textsc{Bad}(\mathrm{H}^A)]
\end{aligned}$$

which yields Equations (4) and (5).

Let B pick $i \in \{1,2\}$ at random and run B_i, and let $\delta(F) = \mathbf{Adv}_{F,\{\mathrm{id}\}}^{\text{prp-cca}}(B)$. Then from Equations (3), (4), (5) we have

$$\Pr[\mathrm{G}_0^A] \leq 2\delta(F) + \Pr[\mathrm{H}^A] + \Pr[\textsc{Bad}(\mathrm{H}^A)] \ . \tag{6}$$

We will design B_3, B_4 so that

$$\begin{aligned}
\Pr[\mathrm{H}^A] - \Pr[\mathrm{I}_1^A] &\leq \mathbf{Adv}_{\mathbf{WB}}^{\text{prp-cca}}(B_3) & (7)\\
\Pr[\textsc{Bad}(\mathrm{H}^A)] - \Pr[\textsc{Bad}(\mathrm{I}_1^A)] &\leq \mathbf{Adv}_{\mathbf{WB}}^{\text{prp-cca}}(B_4) \ . & (8)
\end{aligned}$$

Adversaries B_3, B_4 are almost the same, differing only in how they take their final decision, and accordingly we unify their descriptions. For $i \in \{3,4\}$, adversary B_i has access to oracles $\textsc{Fn}, \textsc{Fn}^{-1}$ and provides oracles $\textsc{Fn}, \textsc{KDFn}, \textsc{Fn}^{-1}$ to A. It starts by selecting K at random from $\{0,1\}^n$. It will then pick π at random from $\mathsf{TwPm}(\{\gamma_1\}, \{0,1\}^n)$. (This is a conceptual simplification. Adversary B_i can't really pick π in advance like this and remain efficient. Instead it will build π on the fly via lazy sampling.) It sets $S \leftarrow \emptyset$ and $\mathsf{bad} \leftarrow \mathsf{false}$ and then runs A. When A queries $\textsc{Fn}(T, M)$, adversary B_i does the following:

```
$X \leftarrow \mathsf{ECB}(M : \pi(\gamma_1, \cdot))$
$Y \leftarrow \textsc{Fn}(T, X)$
Return $Y$ to $A$.
```

When A queries $\textsc{KDFn}(T, \mathrm{id}_M)$, adversary B_i does the following:

```
For $i = 1, \ldots, m$ do
    If $M[i] = \perp$ then $M[i] \leftarrow K$
$X \leftarrow \mathsf{ECB}(M : \pi(\gamma_1, \cdot))$
$S \leftarrow S \cup \{X[i] : 1 \leq i \leq m\}$
$Y \leftarrow \textsc{Fn}(T, X)$
Return $Y$ to $A$.
```

When A queries $\textsc{Fn}^{-1}(T, C)$, adversary B_i does the following:

```
$X \leftarrow \textsc{Fn}^{-1}(T, C)$
$S' \leftarrow \{X[i] : 1 \leq i \leq m\}$
$M \leftarrow \perp$
If $S \cap S' = \emptyset$ then $X \leftarrow \mathsf{ECB}(M : \pi(\gamma_1, \cdot))$
Else $\mathsf{bad} \leftarrow \mathsf{true}$
Return $M$ to $A$.
```

So far there has been no difference between B_3, B_4 but now, when A halts, they compute their outputs differently, B_3 returning the same output as A, but B_4 returning 1 if $\mathsf{bad} = \mathsf{true}$ and 0 otherwise. We have

$$\begin{aligned}
\Pr[\mathrm{RO}^{B_3}_{\mathbf{WB}}] &= \Pr[\mathrm{H}^A] \\
\Pr[\mathrm{Rand}^{B_3}_{mn}] &= \Pr[\mathrm{I}^A_1] \\
\Pr[\mathrm{RO}^{B_3}_{\mathbf{WB}}] &= \Pr[\textsc{Bad}(\mathrm{H}^A)] \\
\Pr[\mathrm{Rand}^{B_3}_{mn}] &= \Pr[\textsc{Bad}(\mathrm{I}^A_1)]
\end{aligned}$$

which yields Equations (7) and (8).

Recall that $\delta(\mathbf{WB}) = \mathbf{Adv}^{\text{prp-cca}}_{\mathbf{WB}}(Q, n, m, t)$. Then from Equations (6), (7), (8) we have

$$\Pr[\mathrm{G}^A_0] - \Pr[\mathrm{I}^A_1] \leq 2\delta(F) + 2\delta(\mathbf{WB}) + \Pr[\textsc{Bad}(\mathrm{I}^A_1)] \ . \tag{9}$$

Consider game J of Figure 4. Composing a random permutation with any independently chosen permutation yields a random permutation, regardless of the distribution of the second permutation, so

$$\begin{aligned}
\mathbf{Adv}^{\text{prp-cca}}_{J,\mathcal{ID}_m}(A) &= \Pr[\mathrm{Real}^A_{J,\mathcal{ID}_m}] - \Pr[\mathrm{Rand}^A_{J,\mathcal{ID}_m}] \\
&= \Pr[\mathrm{G}^A_0] - \Pr[\mathrm{J}^A] \\
&= \Pr[\mathrm{G}^A_0] - \Pr[\mathrm{I}^A_0] \\
&= (\Pr[\mathrm{G}^A_0] - \Pr[\mathrm{I}^A_1]) + (\Pr[\mathrm{I}^A_1] - \Pr[\mathrm{I}^A_0]) \\
&\leq (\Pr[\mathrm{G}^A_0] - \Pr[\mathrm{I}^A_1]) + \Pr[\textsc{Bad}(\mathrm{I}^A_1)] \ .
\end{aligned}$$

Putting this together with Equation (9) we have

$$\mathbf{Adv}^{\text{prp-cca}}_{J,\mathcal{ID}_m}(A) \leq 2\delta(F) + 2\delta(\mathbf{WB}) + 2 \cdot \Pr[\textsc{Bad}(\mathrm{I}^A_1)] \ . \tag{10}$$

To complete the proof we bound the last term above. The assumption that A is legitimate (this is where we use it) implies that if it makes query $\textsc{Fn}^{-1}(T, C)$ then it made no previous $\textsc{KDFn}(T, \cdot)$ query that returned C. We can wlog also assume that no previous $\textsc{Fn}(T, \cdot)$ query returned C. Taking a hit of $Q^2 \cdot 2^{-mn-1}$ in the bound, we can view $\pi(T, \cdot)$ as a random function rather than a random permutation, so the response to a new $\textsc{Fn}^{-1}(T, C)$ query is a new, random string, meaning each block is uniformly distributed in $\{0,1\}^n$. If A made q_1 queries to $\textsc{KDFn}$ then the set S has size at most $q_1 m$ and each $\textsc{Fn}^{-1}$ inverse query has chance at most $q_1 m^2 2^{-n}$ of setting bad. If it made q_2 queries to $\textsc{Fn}^{-1}$, the overall chance of setting bad is at most $q_1 q_2 m^2 2^{-n}$. But $q_1 + q_2 \leq Q$ so $q_1 q_2 \leq Q^2/4$ and thus

$$\Pr[\textsc{Bad}(\mathrm{I}^A_1)] \leq \frac{Q^2}{2^{mn+1}} + \frac{Q^2 m^2}{2^{n+2}} \leq \frac{Q^2(m^2+2)}{2^{n+2}} \ .$$

Combining this with Equation (10) completes the proof. ▌

5 Implementation

We describe a fast AES-based instantiation of **EtE**. As a starting point, we need an AES-based tweakable blockcipher. LRW [15] and XEX [14] are possible choices. Both of them require two calls to the base AES blockcipher, but with XEX, one of the calls can be can be made just once and the value cached as the cipher is used to encipher many message blocks. We thus choose XEX. Applying **StE** to XEX yields a tweakable blockcipher F. Now we need to pick a wideblock cipher to play the role of **WB**. We pick EME which uses about two blockcipher calls per message block together with overhead for offset computations. Let J be the wideblock tweakable cipher produced by applying **EtE** to **WB** and F. We have optimized J and then implemented it and compared its speed to that of EME.

XEX. XEX over a blockcipher E: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ has tweakspace $\{0,1\}^n \times \mathbb{I}$, where $\mathbb{I}$ is the set of integers $[1, \ldots, 2^n - 2]$. Arithmetic operations in XEX are done in the field GF(2^n),

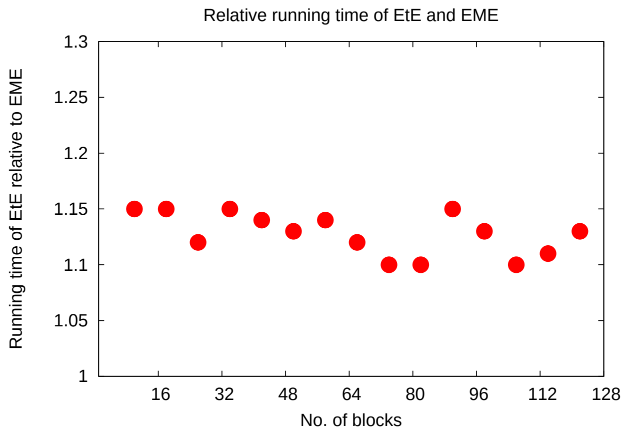

Figure 6: Running time of J on a scale where the running time of EME has been normalized to 1 and AES implemented via AES-NI.

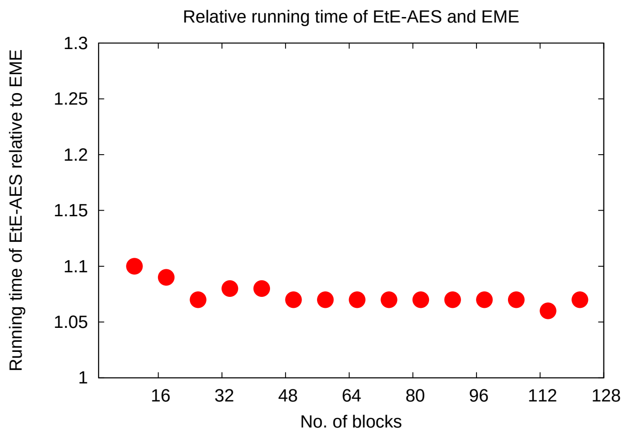

Figure 7: Running time **EtE** with plain AES on a scale where the running time of EME has been normalized to one. This shows that the extra AES computations from our ECB layer add little to the running time.

with elements of GF(2^n) viewed interchangeably as n-bit strings, integers in $[0, 2^n - 1]$ and as formal polynomials of degree $n - 1$ with binary coefficients. Addition is bitwise XOR and multiplication of two points is performed by multiplying them as formal polynomials modulo the irreducible polynomial $p_{128}(x) = x^{127} + x^7 + x^2 + x + 1$. XEX is defined by $\tilde{E}_K^{(N,i)}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where $\Delta = 2^i\mathsf{N}$ and $\mathsf{N} = E_K(N)$. For our purposes, we require a much smaller tweak space of two elements. We pick $t_0 = (0^n, 1)$ and $t_1 = (10^{n-1}, 1)$. Now, Δ can assume only two values (Δ_0, Δ_1) which can be precomputed and cached between successive calls, avoiding running E for the second time to calculate N. The XOR operations still remain, but they are parallelizable, using vector instructions [8].

Experiments. We compare the speeds of **EtE** and EME on a processor that has the AES-NI support for executing AES in hardware [10]. Given that **EtE** is intended for use in disk encryption at the sector level, which happens in hardware, the use of AES-NI is realistic.

Our results shown in Figure 6 show that encrypting with **EtE** is slower than EME by a modest 15% or less. We ran the tests described below on a Intel Xeon W3690 processor at 3.47 GHz with support for AES-NI running Linux version 2.6 with code compiled using `gcc -O3 -ftree-vectorize -msse2 -ffast-math` to enable vector instructions and perform other general optimizations. The base code was from Brian Gladman's EME2 implementation [9], which we modified to run as EME and use the hardware AES instructions.

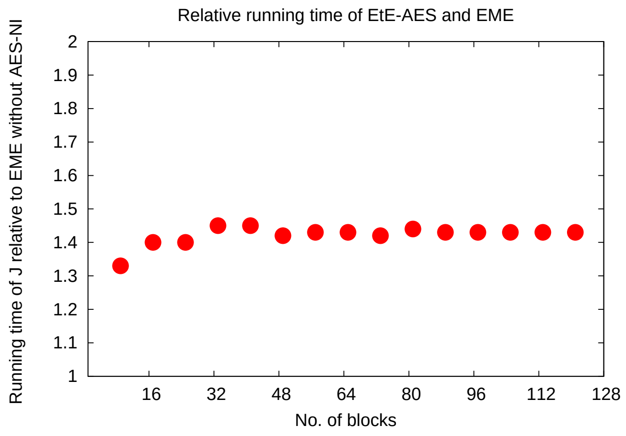

Figure 8: Running time of J on a scale where the running time of EME has been normalized to 1 and with AES implemented in software.

Discussion. Using XEX (on top of AES) and the extra ECB layer account for **EtE**'s slowdown relative to EME-AES. To understand how these modifications affect the speed individually, we timed **EtE** with AES instead of XEX. This cipher is only PRP-CCA secure, but we were interested in its running time. We see in Figure 7 that adding the extra ECB step makes up only 7% of the overhead and running XEX on top of AES makes up the remaining 8%. Since EME is a two pass scheme, we would expect that adding another ECB pass would amount to a close to 50% slowdown. Indeed, this can be observed when AES is implemented in software, as shown in Figure 8, where we see a 45% slowdown. But with AES in hardware, most of the time in EME-AES is spent outside the AES calls. This may be surprising at first, but we should keep in mind that in hardware, AES runs very fast. Thus, adding an extra AES operation per block is not too expensive, justifying our choice of adding an ECB layer.

References

[1] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Aug. 2009. (Cited on page 1.)

[2] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, May 2010. (Cited on page 1.)

[3] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, May 2003. (Cited on page 5.)

[4] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 4, 7, 9, 12.)

[5] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Aug. 2003. (Cited on page 1, 2, 5.)

[6] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision diffie-hellman. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Aug. 2008. (Cited on page 1, 3.)

[7] Z. Brakerski, S. Goldwasser, and Y. T. Kalai. Black-box circular-secure encryption beyond affine functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 201–218. Springer, Mar. 2011. (Cited on page 1.)

[8] N. Firasta, M. Buxton, P. Jinbo, K. Nasri, and S. Kuo. Intel AVX: New frontiers in performance improvements and energy efficiency. *Intel Software Network*, 2009. (Cited on page 16.)

[9] B. Gladman. Brian gladman's aes implementation. `http://gladman.plushost.co.uk/oldsite/AES/index.php`. (Cited on page 4, 16.)

[10] S. Gueron. Advanced encryption standard (AES) instructions set. Intel, `http://softwarecommunity.intel.com/articles/eng/3788.htm`, 2010. (Cited on page 16.)

[11] S. Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 315–327. Springer, Dec. 2004. (Cited on page 1.)

[12] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 466–475. ACM Press, Oct. 2007. (Cited on page 1, 2, 5, 6.)

[13] S. Halevi and P. Rogaway. A tweakable enciphering mode. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 482–499. Springer, Aug. 2003. (Cited on page 4.)

[14] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 292–304. Springer, Feb. 2004. (Cited on page 1, 2, 3, 11, 15.)

[15] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Aug. 2002. (Cited on page 1, 2, 4, 15.)

[16] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), 1988. (Cited on page 2, 4.)

[17] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Dec. 2004. (Cited on page 1, 3, 4, 6, 10.)