Tweakable Block Ciphers

Moses Liskov and Ronald L. Rivest

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
mliskov@theory.lcs.mit.edu, rivest@mit.edu

Abstract. We propose a new cryptographic primitive, the “tweakable block cipher.” Such a cipher has not only the usual inputs—message and cryptographic key—but also a third input, the “tweak.” The tweak serves much the same purpose that an initialization vector does for CBC mode or that a nonce does for OCB mode. Our proposal thus brings this feature down to the primitive block-cipher level, instead of incorporating it only at the higher modes-of-operation levels. We suggest that (1) tweakable block ciphers are easy to design, (2) the extra cost of making a block cipher “tweakable” is small, and (3) it is easier to design and prove modes of operation based on tweakable block ciphers. Moreover, we can prove tighter security bounds for certain modes of operation based on tweakable block ciphers than are known for the corresponding modes based on standard block ciphers (e.g. for OCB mode).

Keywords: block ciphers, tweakable block ciphers, initialization vector, modes of operation

1 Introduction

A conventional block cipher takes two inputs—a key $K \in \{0,1\}^k$ and a message (or plaintext) $M \in \{0,1\}^n$—and produces a single output—a ciphertext $C \in \{0,1\}^n$. The signature for a block cipher is thus (see Figure 1(a)):

$$E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n.$$  (1)

On the other hand, the corresponding operators for variable-length encryption have a different signature. These operators are usually defined as “modes of operation” for a block cipher, but they may also be viewed abstractly as another set of encryption operators. They take as input a key $K \in \{0,1\}^k$, an initialization vector (or nonce) $V \in \{0,1\}^\nu$, and a message $M \in \{0,1\}^*$ of arbitrary length, and produce as output a ciphertext $C \in \{0,1\}^*$. The signature for a typical encryption mode is thus:

$$E : \{0,1\}^k \times \{0,1\}^\nu \times \{0,1\}^* \to \{0,1\}^*.$$  

Block ciphers (pseudorandom permutations) are inherently deterministic: every encryption of a given message with a given key will be the same. Many modes
of operation have nonetheless a requirement for “essentially different” instances of the block cipher in order to prevent attacks that operate by, say, permuting blocks of the input.\(^1\) Attempts to resolve the conflict between keeping the same key for efficiency and yet achieving variability somehow often results in a design that uses a fixed key, but which attempts to achieve variability by manipulating the input before encryption, the output after encryption, or both. (See section 1.1 for examples.) Such designs seem inelegant—they are attempting to solve a problem with a primitive (a basic block cipher) that is not well suited for the problem at hand. Better to rethink what primitives are really wanted for such a problem.

This paper proposes to revise the signature of a block cipher so that it contains a notion of variability as well. The revised primitive operation, which we call a\(^\text{a tweakable block cipher}\), has the signature:

$$\tilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n .$$

(2)

For this operator, we call the new (second) input a “tweak” rather than a “nonce” or “initialization vector;” but the intent is similar. A tweakable block cipher thus takes three inputs—a \textit{key} \(K \in \{0,1\}^k\), a \textit{tweak} \(T \in \{0,1\}^t\), and a \textit{message} (or \textit{plaintext}) \(M \in \{0,1\}^n\)—and produces as output a \textit{ciphertext} \(C \in \{0,1\}^n\) (see Figure 1(b)).

![Diagram](image)

\textbf{Fig. 1.} (a) \textit{Standard block cipher} encrypts a message \(M\) under control of a key \(K\) to yield a ciphertext \(C\). (b) \textit{Tweakable block cipher} encrypts a message \(M\) under control of not only a key \(K\) but also a “tweak” \(T\) to yield a ciphertext \(C\). The “tweak” can be changed quickly, and can even be public. (c) Another way of representing a tweakable block cipher; here the key \(K\) shown inside the box.

\(^1\) Actually this problem is more broad than in only modes of operation, which turn block ciphers into secure encryption schemes. Similar issues also come up in the design of hash functions, MACs, and other applications built on block ciphers.
In designing a tweakable block cipher, we have certain goals. First of all, obviously, we want any tweakable block ciphers we design to be as efficient as possible (just as with any scheme), both for encryption and for decryption.

In particular, a tweakable block cipher should have the property that changing the tweak should be much less costly than changing the key. Many block ciphers have the property that changing the encryption key is relatively expensive, since a “key setup” operation needs to be performed. In contrast, changing the tweak should be very cheap.

Some cryptographic modes of operation such as the Davies-Meyer hash function (see Menezes et al. [6, Section 9.40]) have fallen into disfavor because they have a feedback path into the key input of the block cipher. Since for many block ciphers it is relatively expensive to change the key, these modes of operation are relatively inefficient compared to similar modes that use the same key throughout. See, for example, the discussion by Rogaway et al. [7] explaining the design rationale for the OCB mode of operation, which uses the same cryptographic key throughout.

A tweakable block cipher should also be secure. In order to deal with the issue of security, we have to decide how much control of the tweak input we can give to the adversary. We would like that even if an adversary has control of the tweak input, the tweakable block cipher remains secure. We’ll define what this means more precisely later on. But intuitively, each fixed setting of the tweak gives rise to a different, apparently independent, family of standard block cipher encryption operators.

We wish to carefully distinguish between the function of the key, which is to provide uncertainty to the adversary, and the role of the tweak, which is to provide variability. The tweak is not intended to provide additional uncertainty to an adversary. Keeping the tweak secret need not provide any greater cryptographic strength.

The point of this paper is to suggest that by cleanly separating the roles of cryptographic key (which provides uncertainty to the adversary) from that of tweak (which provides independent variability) we may have just the right tool for many cryptographic purposes.

### 1.1 Related Work

One motivating example for this introduction of tweakable block ciphers is the DESX construction introduced by Rivest (unpublished). See Figure 2(a). The reason for introducing DESX was to cheaply provide additional key information for DES. The security of DESX has been analyzed by Kilian and Rogaway [5]; they show that DESX with n-bit inputs (and tweaks) and k-bit keys has an effective key-length of $k+n-1-\log m$ where the adversary is limited to \( m \) oracle calls.

Similarly, if one looks at the internals of the recently proposed “offset codebook mode” (OCB mode) of Rogaway et al. [7], one sees DESX-like modules that may also be viewed as instances of a tweakable block ciphers. That is, the pre-
and post-whitening operations are essentially there to provide distinct families of encryption operators, i.e. they are “tweaked.”

In a similar vein, Bihm and Biryukov [1] suggest strengthening DES against exhaustive search by (among other things) applying a DESX-like construction to each of DES’s S-boxes.

Even and Mansour [4] have also investigated a similar construction (see Figure 2(b)) where the inner encryption operator is fixed and public. They show (see also Daemen[3]) that the effective key length here is \( n - \lg l - \lg m \) where the adversary is allowed to make \( l \) calls to the encryption/decryption oracles and \( m \) calls to an \( F/F^{-1} \) oracle.

### 1.2 Outline of this paper

In Section 2 we then discuss and formalize the notion of security for tweakable block ciphers. In Section 3 we suggest several ways of constructing tweakable block ciphers from existing block ciphers, and prove that the existence of tweakable block ciphers is equivalent to the existence of block ciphers. We also suggest a “standard” way of tweaking a round-based block cipher. Then in Section 4 we suggest several new modes of operation utilizing tweakable block ciphers, and give simple proofs for some of them. Section 5 concludes with some discussion and open problems.
2 Definitions

The security of a block cipher $E$ (e.g., parameterized as in equation (1)) can be quantified as $\text{Sec}_E(q, t)$—the maximum advantage that an adversary can obtain when trying to distinguish $E(K, \cdot)$ (with a randomly chosen key $K$) from a random permutation $\Pi(\cdot)$, when allowed $q$ queries to an unknown oracle (which is either $E(K, \cdot)$ or $\Pi(\cdot)$) and when allowed computation time $t$. This advantage is defined as the difference between the probability the adversary outputs 1 when given oracle access to $E$ and the probability the same adversary outputs 1 when given oracle access to $\Pi$.

A block cipher may be considered secure when $\text{Sec}_E(q, t)$ is sufficiently small.

We may measure the security of a tweakable block cipher $E$ (parameterized as in equation (2)) in a similar manner as the maximum advantage $\text{Sec}_E(q, t)$ an adversary can obtain when trying to distinguish $E(\cdot, \cdot)$ from a “tweakable random permutation” $\Pi(\cdot, \cdot)$ where $\Pi$ is just a family of independent random permutations parametrized by $T$. That is, for each $T$, we have that $\Pi(T, \cdot)$ is an independent randomly chosen permutation of the message space. Note that the adversary is allowed to choose both the message and tweak for each oracle call.

A tweakable block cipher $E$ may be considered secure when $\text{Sec}_E(q, t)$ is sufficiently small.

Note that the key and the tweak are not equivalent inputs. While fixing the tweak gives a secure block cipher, fixing the key to some public value and using the tweak as a key input may not—consider the construction $E_K(T, M) = E_K(T \oplus E_K(M))$ that we later prove to be a secure tweakable block cipher.

3 Constructions

In this section we show that the existence of block ciphers and the existence of tweakable block ciphers are equivalent. One direction is easy: if we let $E_K(M) = E_K(0^t, M)$, it is easy to see that if $E$ is a secure tweakable block cipher then $E$ must be a secure block cipher.

The other direction is more difficult. Some simple attempts to construct a tweakable block cipher from a block cipher fail.

For example, the DESX analogue:

$$E_K((T_1, T_2), M) = E_K(M \oplus T_1) \oplus T_2$$

fails because an adversary can notice that flipping the same bits in both $T_1$ and $M$ has no net effect.

Similarly, taking an ordinary block cipher and splitting its key into a key for the tweakable cipher and a tweak:

$$\bar{E}_K(T, M) = E_{K\parallel T}(M)$$

or xorring the tweak into the key:

$$E_K(T, M) = E_{K\oplus T}(M)$$
need not yield secure tweakable block ciphers, since a block cipher need not depend on every bit of its key. (Biham’s related-key attacks of Biham [2] would be relevant to this sort of design.)

The following theorem gives a construction that works.

**Theorem 1.** Let

$$\bar{E}_K(T, M) = E_K(T \oplus E_K(M)).$$

$\bar{E}$ is a secure tweakable block cipher. More precisely,

$$\text{Sec}_{\bar{E}}(q, t) < \text{Sec}_E(q, t) + \Theta(q^2/2^n).$$

**Proof.** We assume that $E$ has security function $\text{Sec}_E(q, t)$ and assume that an adversary $A^2$ exists that achieves an advantage $\text{Sec}_{\bar{E}}(q, t)$ when distinguishing $\bar{E}$ from a tweakable random permutation.

We have the following cases.

*Case i:* $A$ can distinguish between $\bar{E}_K$ and $H^1$, where $H^1(T, M) = \Pi(T \oplus \Pi(M))$.

If this is the case, we can use $A$ to distinguish $E$ from $\Pi$.

*Case ii:* $A$ can distinguish between $H^1$ and $H^2$ where $H^2(T, M) = R(T \oplus R(M))$, where $R$ is a random function. It is easy to see that the advantage in distinguishing a random function from a random permutation is $\Theta(q^2/2^n)$.

*Case iii:* $A$ can distinguish $H^2$ from $H^3$, where $H^3(T, M) = R_1(T \oplus R_1(M))$, where $R_1$ and $R_2$ are random functions. Suppose $(T_1, M_1), \ldots, (T_q, M_q)$ are all the queries $A$ makes to the oracle, and suppose no collisions of the following type happen: $T_i \oplus R(M_j) = M_j$. With no such collisions, $H^2$ cannot be distinguished from $H^3$ as the outer application of $R$ takes place on a set of inputs disjoint from the inputs to the inner application of $R$, and so the outer outputs are independently random, just as the outputs of $R_2$ would be.

Furthermore, the probability of any such collisions occurring is $\Theta(q^2/2^n)$. What is the probability that $(T_i, M_i)$ collides with any previous pair? If $M_i$ is a new value then it is easy to see that the probability is at most $(i - 1)/2^n$. What if $M_i$ is not new? In this case, either $(T_i, M_i)$ will collide or it won’t, since all the random decisions have been made. However it is important to note that if no collisions have happened before, then every oracle response the adversary gets is just a new random value. Thus, the values the adversary gets are independent from the $T$’s the adversary produces. Conversely, $T_i$ must be independent from the distribution of $R$. Thus, even though $T_i$ is not necessarily chosen randomly, no matter how the adversary picks $T_i$, it has a probability of at most $(i - 1)/2^n$ of being one that causes a collision. Adding all these probabilities up, we see that the probability that any collision occurs is $\Theta(q^2/2^n)$.

*Case iv:* $A$ can distinguish $H^3$ from $H^4$, where $H^4(T, M) = \Pi_2(T \oplus \Pi_1(M))$, where $\Pi_1$ and $\Pi_2$ are random permutations. Again, if this were the case then $A$ could allow us to distinguish a random function from a random permutation.

*Case v:* $A$ can distinguish between $H^4$ and $\Pi$. We prove that this is impossible directly. Suppose $(T_1, M_1), \ldots, (T_q, M_q)$ are all the (distinct) queries the adversary makes to the oracle. Suppose that there is no $i \neq j$ such that $\Pi_1(M_i) \oplus T_i$ =
$\Pi_i(M_j) \oplus T_j$. Then, the values the adversary receives are totally indistinguishable from those that $\Pi$ would return – for any given $T_i$, the output will be a random value that was never returned for that $T_i$ before. However, we can limit the probability that this happens. Note that if $M_i = M_j$ or $T_i = T_j$ then this cannot happen (as we assumed that $(T_i, M_i)$ is distinct from $(T_j, M_j)$).

The probability that $(T_i, M_i)$ “collides” with any previous pair is at most $(i - 1)/(2^n - i + 1)$. This is because the highest possible numerator occurs when no previous pair was such that $M_j = M_i$. The output of $\Pi_1$ is random, and at worst there will be one outcome for every $j < i$ such that this pair collides with pair $j$. Similarly, the lowest possible denominator occurs when there is at least one pair that $(T_i, M_i)$ could collide with, but every pair so far has used a different $M_i$. Now, $\Pi_1$ is restricted to only $2^n - i + 1$ outcomes. Summing this expression for $i = 1$ to $q$ yields $\Theta(q^2/2^n)$. The adversary gains another possible $\Theta(q^2/2^n)$ advantage in this case.

Thus, we see that this construction only “degrades” $\text{Sec}_{2q,t}$ by $\Theta(q^2/2^n)$ to obtain $\text{Sec}^\epsilon_{2q,t}$.

Note that this construction has the nice property that changing the tweak is easy (no “key setup” required). Furthermore, we do not require a longer key than the block cipher did for the same level of security. However, the construction has an overall cost (running time) that is twice that of the underlying block cipher.

This completes our proof that the existence of (secure) tweakable block ciphers is equivalent to the existence of (secure) block ciphers. We leave as open problems devising constructions that are more efficient, or that have tighter bounds, than the construction of Theorem 1.

### 3.1 A more practical construction

It would be tempting at this point to declare the problem of constructing tweakable block ciphers to be “solved” and go on to other issues. However, we consider it an interesting challenge to construct secure tweakable block ciphers directly, rather than by building them out of existing block ciphers. Of course, such constructions must be rather heuristic in nature, just as constructing block ciphers de novo is heuristic.

In this section we suggest a basic method of introducing tweaks into any round-based block cipher, such as DES, AES, or RC6. Let us suppose that the block size is $n$ bits. The tweakable version of the block cipher will accept as an additional input a tweak $T$ of size $n$ bits. Suppose also that the block cipher has $R$ rounds, and that $R$ is even.

Our proposal is (see Figure 3):

1. Modify the block cipher so that it also XOR’s the tweak into the current state:
   - just after the first round,
   - after round $R/2$ (that is, at the mid-point), and
   - just before the last round.
We call this strategy for adding tweaks to a block cipher that operates by rounds the “standard tweak”.

We can see that the tweakable block cipher is only slightly more expensive (by three XOR’s) than the underlying block cipher. Furthermore, this construction seems well motivated by the construction of the previous section (we tweak in the middle). The use of the tweak near near the ends provides additional mixing of the tweak into the computation.

![Diagram](image)

**Fig. 3.** The standard way of tweaking a round-based block cipher. The tweak $T$ is xor-ed in after the first round, at the mid-point, and before the last round. This figure illustrates tweaking a 10-round block cipher, such as AES with a 128-bit key.

4 **Tweakable Modes of Operation**

The new “tweak” input of a tweakable block ciphers enables a multitude of new modes of operation. Indeed, these new modes may really be the “payoff” for introducing tweakable block ciphers. In this section we sketch three such possible modes, and leave the remainder to your imagination. We just describe the first two, and prove secure the third, which is perhaps the most interesting of the three (it is an analogue to OCB mode for authenticated encryption).
4.1 Tweak Block Chaining (TBC)

Tweak block chaining (TBC) is similar to cipher block chaining (CBC). An initial tweak $T_0$ plays the role of the initialization vector (IV) for CBC. Each successive message block $M_i$ is encrypted under control of the encryption key $K$ and a tweak $T_{i-1}$, where $T_i = C_i$ for $i > 0$. See Figure 4.

Fig. 4. Tweak block chaining: a chaining mode for a tweakable block cipher. Each ciphertext becomes the tweak for the next encryption.

To handle messages whose length is greater than $n$ but not a multiple of $n$, a variant of ciphertext-stealing [8] can be used; see Figure 5.

Fig. 5. Ciphertext stealing for tweak block chaining handles messages whose length is at least $n$ bits long but not a multiple of $n$. Let $r$ denote the length of the last (short) block $M_m$ of the message. Then $|C_m| = |M_m| = r$ and $|C'| = n - r$. Here $X$ denotes the rightmost $n - r$ bits of $C_{m-2}$ (or of $T_0$ if $m = 2$).
One can also adapt the TBC construction to make a TBC-MAC in the same manner that one can use the CBC construction to make a CBC-MAC. Both TBC and TBC-MAC modes still need a careful security analysis.

4.2 Tweak Chain Hash (TCH)

To make a hash function, one can adapt the Matyas-Meyer-Oseas construction (see Menezes et al. [6, Section 9.40]). Figure 6 illustrates this construction, which uses a fixed public key $K$ in the tweakable block cipher, and chains through the tweak input.

![TCH Diagram]

Fig. 6. The tweak chain hash (TCH). Here $T_0$ is a fixed initialization vector. A fixed public key $K$ is used in the tweakable block cipher. The message $M$ is padded in some fixed reversible manner, such as by appending a 1 and then enough 0's to make the length a multiple of $n$. The value $H$ is the output of the hash function.

We don’t know if this construction is secure. With a strong additional property on the tweakable block cipher, namely that for a fixed known key and fixed unknown tweak, we still get a pseudorandom permutation, we could adapt the proof of the Davies-Meyer hash function. However, as we noted in section 2, this is not the case for all tweakable block ciphers.2

4.3 Tweakable Authenticated Encryption (TAE)

In this section we suggest an authenticated mode of encryption (TAE) based on the use of a tweakable block cipher. This mode can be viewed as a paraphrase or restatement of the architecture of the OCB (offset codebook) mode proposed by Rogaway et al. [7] to utilize tweakable block ciphers rather than DESX-like modules. The result is shown in Figure 7. (The reader may need to consult the OCB paper to follow the rather terse description given here.)

2 One tweakable block cipher construction that does have this property is $\tilde{E}_K(T, M) = E_K(E_T(E_K(M)))$, but this is not as desirable a construction as it is not easy to change the tweak.
The tweak $Z_i$ for $i > 0$ is defined as the concatenation of the nonce $N$, an $n/2$-1-bit representation of the integer $i$, and a zero bit 0: $Z_i = N||i||0$. The tweak $Z_0$ is defined as the concatenation of the nonce $N$, an $n/2$-1-bit representation of the integer $b$, where $b$ is the bit-length of the message $M$, and a one bit 1: $Z_0 = N||b||1$. The message $M$ is divided into $m-1$ blocks $M_1, \ldots, M_{m-1}$ of length $n$ and one last block $M_m$ of length $r$ for $0 < r \leq n$ (except that if $|M| = 0$ then the last (and only) block has length 0). Each ciphertext block $C_i$ has same length as $M_i$. The function $\text{len}(M_m)$ produces an $n$-bit binary representation of the length $r$ of the last message block. The last message block $M_m$ is padded with zeros if necessary to make it length $n$ before xoring. The checksum is $(M_1 \oplus \cdots \oplus M_{m-1} \oplus (M_m||0^r))$. The parameter $\tau$, $0 \leq \tau \leq n$ specifies the desired length of the authentication tag.
The OCB paper goes to considerable effort to analyze the probability that various encryption blocks all have distinct inputs. We show that an authenticated encryption mode such as TAE based on a tweakable block cipher is much simpler to analyze, since the use of tweaks obviates this concern.

We will in fact give a fairly easy proof that a tweakable block cipher used in TAE mode gives all the security properties claimed for OCB mode. Rogaway et al. claim that OCB mode is:

- **Unforgeable.** Any nonce-respecting\(^3\) adversary can forge a new valid encryption with probability at worst negligibly greater than \(2^{-\tau}\) (where \(\tau\) is the bit length of the authentication tag produced).
- **Pseudorandom.** To any nonce-respecting adversary, the output of OCB mode is pseudorandom. In other words, no adversary can distinguish between an OCB mode oracle and a random function oracle. [7]

We now prove that TAE mode satisfies these properties.

**Theorem 2.** If \(\overline{E}\) is a secure tweakable block cipher, then \(\overline{E}\) used in TAE mode will be pseudorandom and unforgeable. It is pseudorandom in the sense that an adversary will have no more advantage in distinguishing TAE mode output from a random function oracle than it has in distinguishing \(E\) from \(\Pi\), and it is unforgeable in the sense that an adversary will have chance at most \(2^{-\tau} + O(2^{-2n})\) of forging a correct tag on any message for which it was not previously given the correct tag.

**Proof.** To prove that TAE mode is pseudorandom, we note that no tweak is ever repeated when the adversary is nonce-respecting. Now, if an adversary \(A\) were able to distinguish between a random function oracle and a TAE mode oracle, then we could distinguish the tweakable block cipher \(E\) from \(\Pi\) as follows. Given an oracle \(O\), we simply run \(A\), and answer \(A\)'s oracle queries by simulating TAE mode with \(O\) instead of \(E\). Now, if \(O = E\) then we are in fact providing \(A\) with a TAE mode oracle. However, if \(O = \Pi\) we are providing a random oracle. To see this, note that since no tweak is ever repeated, every part of every output is an independent random value. Thus, if we just give the answer \(A\) gives, we are correct whenever \(A\) is correct, and thus we defeat the security of \(E\).

To prove that TAE mode is unforgeable, we do the same thing. Suppose some adversary \(A\) can forge encryptions in TAE mode. We will break \(E\) as follows. Given an oracle \(O\) we just run \(A\) and answer \(A\)'s oracle queries by simulating TAE mode with \(O\). When \(A\) gives an answer, we check to see if the answer is a successful forgery. If it is, we guess that \(O = E\) and if not, we guess that \(O = \Pi\). Since \(A\) is a successful adversary, if \(O = E\), it forges successfully with

---

\(^3\) By "nonce-respecting," it is meant that while the adversary has oracle access and control of the nonce, the adversary may never ask that a nonce be used more than once; the idea is that any oracle a real adversary would have access to would still not repeat nonces, even if manipulated in order to accept nonces from elsewhere.
probability nonnegligibly greater than $2^{-7}$. We will now show that if $O = \bar{\Pi}$ then $A$ forges with probability at most $2^{-7}$. Once we prove this we'll be done, since this reduction will be correct non-negligibly more often than it is incorrect.

Suppose now that $O = \bar{\Pi}$. First we note that if $A$ returns an answer with a new nonce, then $Z_0$ for the answer will be new and thus the correct answer will be a totally random $\tau$-bit string. In other words, $A$ will be correct with probability exactly $2^{-\tau}$.

Secondly, if $A$ returns an answer with an old nonce, then there are three cases. In the first case, all the ciphertext blocks are the same ciphertext blocks that were returned when that nonce was used previously. In this case, the forgery cannot possibly be correct since either it will be wrong or it will not be new. In the second case, the message is a different length than the message this nonce was queried with before. In this case, the forgery is correct with probability $2^{-\tau}$ since $Z_0$ will be different than before. In the third and final case, the message is the same length but there is at least one new ciphertext block. Now, the preimage of every block that is different is a random new value; the only constraint is that it is different than the previous value. Thus, the checksum is a random new value. If only one ciphertext block was changed, then the checksum must be different than the previous value. The adversary maximizes his chance of forgery by changing only one ciphertext block and guessing a tag that is different than the tag he obtained on his query. Thus, the forgery is correct with probability exactly

$$2^{n-\tau}/(2^n - 1) = 2^{-\tau} + 1/(2^n(2^n - 1)) = 2^{-\tau} + O(2^{-2n}) .$$

Thus, the probability that the forgery is correct is at most $2^{-\tau}$, which concludes the proof.

It is interesting to note that the construction loses almost nothing in terms of its advantage compared to the advantage of the tweakable block cipher! This is somewhat remarkable, and helps to emphasize our main point that tweakable block ciphers may be the most natural and useful construct for designing higher-level modes of operation.

By introducing tweakable block ciphers, we have “re-partitioned” the design problem into two (new) parts: designing good tweakable block ciphers, and designing good modes of operation based on tweakable block ciphers. We feel that this re-partitioning is likely to be more useful and fruitful than the usual structure, since certain issues (e.g. having to do with collisions, say) can be handled once and for all at the lower level, and can then be ignored at the higher levels, instead of having to be dealt with repeatedly at the higher levels.

## 5 Conclusions and Open Problems

We feel that the notions of a tweakable block cipher and tweakable modes of operation (that is, modes of operation based on tweakable block ciphers) are interesting and worthy of further study.
One advantage of this framework is the new division of issues between design and analysis of the underlying primitive and the design and analysis of the higher-level modes of operation. We feel that the new primitive may result in a more fruitful partition.

Some interesting open problems are:

- What is the security of “tweaked AES” (AES with our proposed “standard tweak”)?
- Improve the construction of Theorem 1 to achieve greater efficiency or a tighter security bound.
- Analyze the security of the tweak-block-chaining mode of encryption.
- Analyze the security of the tweak chain hash.
- Devise and analyze the security of other modes of operation based on tweakable block ciphers.
- Devise and analyze the security of tweakable stream ciphers.

Acknowledgments

We would like to thank Rogaway, Bellare, Black, and Krovetz for inspiring this line of research with their proposed OCB mode and Burt Kaliski and Phil Rogaway for very helpful discussions and suggestions.

References


