Tag Size *Does* Matter: Attacks and Proofs for the TLS Record Protocol

Kenneth G. Paterson[1], Thomas Ristenpart[2], and Thomas Shrimpton[3]

[1] Information Security Group, Royal Holloway, University of London, UK
[2] Dept. of Computer Sciences, University of Wisconsin–Madison, USA
[3] Dept. of Computer Science, Portland State University

Abstract. We analyze the security of the TLS Record Protocol, a MAC-then-Encode-then-Encrypt (MEE) scheme whose design targets confidentiality and integrity for application layer communications on the Internet. Our main results are twofold. First, we give a new distinguishing attack against TLS when variable length padding and short (truncated) MACs are used. This combination will arise when standardized TLS 1.2 extensions (RFC 6066) are implemented. Second, we show that when tags are longer, the TLS Record Protocol meets a new length-hiding authenticated encryption security notion that is stronger than IND-CCA.

1 Introduction

TLS is perhaps the Internet's most widely used security protocol. At its heart lies a sub-protocol for integrity-protecting and encrypting data, called the TLS Record Protocol. The current version of this protocol, TLS 1.2, is specified in [12], though earlier versions [10, 11] are still in widespread use. At a high level, the TLS Record Protocol makes use of a MAC-then-Encode-then-Encrypt (MEE) construction, where the "Encode" step takes care of any padding that might be needed prior to the encryption step. For reasons that will become clear, we focus on MEE when used with CBC mode.

In this case, TLS 1.2 works as follows to protect a message M whose bit-length $m = |M|$ must be a multiple of eight. Let n be the block size of the block cipher underlying CBC. Then, one chooses a fresh n-bit IV to use with CBC mode to encrypt the bit string $M \| T \| P \cdots_{p+1} P$. Here T is a τ-bit message authentication tag produced by running HMAC over M and some header information including a sequence number and $P \cdots_{p+1} P$ is the bit string formed by concatenating together $p+1$ copies of the string P. The value P is the byte-encoding of the number p, which indicates the number of padding bytes. It is required that $\ell = m+\tau+8(p+1)$ be a multiple of n. We refer to this scheme as $\mathsf{MEE\text{-}TLS\text{-}CBC}$. A common instantiation uses AES and HMAC-SHA1, making $n = 128$ and $\tau = 160$.

Implementations can choose p in different ways. One is to use minimal-length padding by letting $p \geq 0$ to be the smallest possible value that results in ℓ being a multiple of n. Another is to use larger values of p in order to generate extra padding. GnuTLS [14], for example, randomly selects p from the set of possible

	MAC	Encoding	Security target
BN00 [3]	SUF-CMA	Concatenation	IND-CPA + PTXT
K01 [15]	SUF-CMA	Concatenation, tag fills one block	IND-CPA + CUF-CPA
K01*	SUF-CMA	Concatenation, tag fills one block	IND-CPA + CTXT
MT10 [18]	SUF-CMA	Any function	Secure channel
This work	PRF	TLS's padding, $m + \tau > n - 8$	LHAE

Fig. 1. Summary of positive results known about MEE under various assumptions about the MAC. The restriction on padding of our result involves the message length m, tag length τ, and block length n. Our attack shows the necessity of this restriction for security.

values. As indicated in the TLS specification, the intent is to combat traffic analysis attacks that exploit plaintext message lengths [16, 21, 23–26].

This paper. We provide the first analysis of the security of MEE-TLS-CBC as an authenticated encryption (AE) scheme. We start by strengthening traditional AE notions [3, 20] to cover the goal of hiding plaintext lengths that motivates the use of extra padding. Using our new length-hiding AE (LHAE) notion, we provide complementary negative and positive results about MEE-TLS-CBC for general m, τ, and n. When $m + \tau \leq n - 8$ and extra padding is used, we give an attack that allows a man-in-the-middle to readily distinguish between messages of different lengths. A variant of this attack rules out proving traditional AE security as well. On the other hand, we show that when $m + \tau > n - 8$ one provably achieves LHAE security. This positive result holds for a generalization of TLS encoding; it may be applicable in other settings where MEE is used with CBC.

In the current TLS standard [12], the allowed primitives are such that $n \leq 128$ and $\tau \geq 160$. Here the attack does not apply and our positive results provide strong evidence of security. More worrisome is the use of truncated MACs, where $\tau = 80$ and the attack would apply. Truncated MACs are used widely in other protocols (e.g., IPSec [17]) and are standardized as a TLS extension in RFC 6066 [13].

Prior work on MEE. Before describing our results in a bit more detail, we briefly summarize the literature as it applies to MEE-TLS-CBC — see Figure 1. Bellare and Namprempre (BN00) [3] introduced two notions of integrity: integrity of plaintexts (PTXT) and of ciphertexts (CTXT). They showed that MEE with any invertible encoding step is IND-CPA and has integrity of plaintexts (PTXT) assuming the mac is strongly unforgeable (SUF-CMA), but argue that PTXT is insufficient for applications because one should target CTXT. Meeting both IND-CPA and CTXT is one of several equivalent formulations for AE security [20].

Krawczyk (K01) [15] analyzed a variant of MEE-TLS-CBC in which m must be a multiple of n, the tag length is $\tau = n$, and no padding is used. He showed that this variant —which does not arise in TLS— achieves a notion of integrity he calls CUF-CPA. This is weaker than CTXT, though a straightforward extension of

K01's techniques prove that this variant is both IND-CPA and CTXT secure; we list this as K01* in Figure 1. While we will build on the techniques underlying these results, the fact that they ignore padding makes them of limited direct relevance to TLS security. Indeed, as the attacks in [9, 22], discussed further below, indicate, the way padding is handled is crucial to the (in)security of MEE-TLS-CBC.

Maurer and Tackmann (MT10) [18] considered MEE with encoding steps being any function, thus restricting attention to minimum-length padding only. They provide a secure channel notion formalized within a new constructive cryptography framework, but the details of this framework (at the time of our writing) have not yet emerged, making comparison with our results for minimum-length padding TLS premature. Our approach uses a more traditional game-based treatment.

As it stands, none of the prior works analyze the AE security of the version of MEE-TLS-CBC used within the standard nor do they treat the length-hiding goal of extra padding.

Length-hiding encryption. Our technical results begin by generalizing encryption to consider the length-hiding goal targeted by TLS. The explicitly stated intent is that applications should be able to hide the length of plaintexts up to some granularity. As mentioned above, the GnuTLS client [14] attempts to obfuscate plaintext length patterns by selecting the amount of padding for each message randomly. This means that for a given message length, the application may vary the amount of padding used. Standard-compliant decryption implementations must support ciphertexts including such extra padding.

This choice was perhaps prescient: attacks taking advantage of leaked plaintext lengths allow inferring web browsing habits [16, 21, 26] and voice-over-IP conversations [23–25]. Note that even when only minimal-length padding is used, MEE-TLS-CBC nevertheless seemingly should hide lengths that are padded to the same multiple of n. Given [16, 21, 23–26], MEE-TLS-CBC seems to have a small security advantage over MEE using OTP — the latter always leaks precise plaintext lengths. Traditional security notions that explicitly allow message lengths to leak (e.g., IND-CPA, IND-CCA) are too weak to surface this distinction.

To treat MEE-TLS-CBC in its full generality, then, we formalize *length-hiding encryption.* We extend the usual syntax of authenticated encryption scheme with associated data (AEAD) to allow the encryption algorithm to take an extra ciphertext-length parameter, in addition to the usual key, header, and message. This allows the user to indicate the desired length of ciphertext.

We correspondingly upgrade the traditional security notions, which do not capture length hiding, and introduce a *length-hiding authenticated encryption* (LHAE) security notion. Our all-in-one definition gives an attacker access to a left-or-right encryption oracle on pairs of chosen messages M_0, M_1 of arbitrary lengths and a *chosen* ciphertext-length. As usual, the attacker's job is to output its guess for a hidden bit b. The LHAE definition captures length hiding in settings where applications may adaptively vary padding per message (such as GnuTLS). Of course, a special case of our security notion is arrived at by restricting to same-length messages: this corresponds to a left-or-right indistinguishability variant of the all-in-one AE notion of Rogaway and Shrimpton [20]. Proving LHAE security therefore establishes AE security as a special case.

New attacks against MEE-TLS-CBC. Our work brings to light interesting new attacks against TLS. Consider MEE-TLS-CBC with when $m + \tau \leq n - 8$. This means that a complete message M (of m bits), a tag, and at least one padding byte can fit into a single CBC block of size n. Then an attacker, given an encryption C of a message M that is created using longer-then-minimum padding, is able to create another encryption C' of the same message M; we call this a *decryption collision*.[1] This immediately violates the ciphertext integrity (CTXT) of MEE-TLS-CBC, thereby ruling out AE or LHAE security, and can easily be extended to build an IND-CCA distinguisher as well.

It may seem that this deficiency is not dangerous. After all it just shows that an attacker can generate a new ciphertext that decrypts to an already legitimately encrypted message, and this does not threaten the security of TLS as a secure channel protocol. Indeed, some formulations of channel security [6–8], including that of [18], explicitly exclude decryption collisions from being considered as an insecurity. Nevertheless, it rules out meeting the AE security notion targeted, and met, by other designs.

What's more, decryption collisions prove obviously damaging in the length-hiding setting. We will show that they can be used to allow an attacker to distinguish between encryptions of messages of *different* lengths, for example "YES" and "NO". This defeats the TLS design intention of hiding plaintext lengths at this level of granularity. The distinguishing attack would be simple to mount in practice by a man-in-the-middle.

TLS 1.2 (and older versions) specifies $n \in \{64, 128\}$ (DES, AES) and $\tau \geq 160$ (HMAC-SHAx), so this attack does not affect the security of TLS as specified in version 1.2. However 80-bit truncated MACs are explicitly defined for use in extensions to TLS 1.2 [13]. Our attack would therefore apply to TLS using CBC-AES with these truncated MACs and extra padding. We are unaware of any current implementations that are vulnerable, but this will change if, for example, GnuTLS implemented the TLS 1.2 truncated MAC extension.

LHAE security of MEE-TLS-CBC. Now the good news. We complement our negative results by proving LHAE security for MEE-TLS-CBC exactly when the above attacks do not work: when $m+\tau > n-8$ or no extra padding is used. The analysis is involved, as one may expect given the sharp divide between security and insecurity. Let us look at it from a high-level.

The natural starting point for our analysis is the K01* result for concatenation encoding, $\tau = n$, block-aligned tags, and no padding. Here one splits the task of proving authenticated encryption security into two key steps (leveraging

[1] The terminology from [6] would call this a replay. We reserve replay for the more traditional security goal of not accepting the same message twice, even if derived from the same ciphertext. Achieving replay resistance requires stateful decryption.

techniques from [20]): showing separately IND-CPA and CTXT security. IND-CPA security is immediate from the IND-CPA security of CBC mode. A general result gives that many-query CTXT is implied by q_d times the advantage of single-query CTXT where q_d is the number of decryption queries. So what remains is showing single-query CTXT. The K01* analysis applies the security of the block cipher as a strong PRP to move to a setting in which the adversary learns nothing about MAC tags from encryption queries and, moreover, for its single decryption query submits a ciphertext consisting of blocks that were output during encryption. The proof concludes via a case analysis partitioned according to which ciphertext blocks are used and how they relate to where tags were located within the encryption queries. The alignment of tags with block boundaries eases this analysis, but it is still relatively involved.

Several new difficulties arise in applying this approach to MEE-TLS-CBC. Foremost of these is that the case analysis becomes significantly more complex, as tags may (for example) span multiple blocks and variable-length padding is allowed. Also the K01* approach only provides a loose bound, approximately $2^{n/3}$, because it proves single-query CTXT and then uses a general hybrid argument to conclude multi-query CTXT. Finally, none of the general results apply to length-hiding encryption. The last issue is the easiest to handle, and in the full version we show that length-hiding IND-CPA and CTXT together imply LHAE. The other issues prove more troublesome. We therefore first simplify our task by introducing a new security notion that will enable further modularity.

Collision-resistant decryption security. Recall that our attack above found decryption collisions: the adversary computed a new ciphertext that decrypts to a previously encrypted message. We formalize resistance to such attacks and call the resulting notion *collision-resistant decryption* (CRD). It turns out that CRD exactly characterizes the gap between CTXT and PTXT: we prove that a scheme is CTXT if and only if it is both PTXT and CRD.

With this new characterization of CTXT in hand, we proceed as follows. We show (in the full version) that MEE is length-hiding IND-CPA secure and PTXT secure. Both of these results follow straightforwardly from the techniques of [3]. Thus to show LHAE of MEE-TLS-CBC reduces to proving CRD security. Here we still have technical hurdles, including the fact that we must directly analyze multi-query CRD, deal with arbitrary tag locations and sizes, and account for variable length padding. What's more, we must observe precise requirements on tag and message lengths to avoid our attacks. To make this task slightly easier, we assume that the MAC is a secure PRF. While this is a stronger assumption than SUF-CMA, the MAC used by TLS is HMAC, which must be a good PRF in other parts of the TLS protocol.

Stateful LHAE. In fact the TLS record protocol uses both stateful encryption and stateful decryption, enabling replay resistance. We handle this, too. In the full version we formalize a stateful LHAE notion (generalizing a definition of [4]) and show that one can easily lift all our results to the stateful setting.

Prior versions of TLS. We have concentrated on the TLS 1.2 standard, though all our results apply to TLS 1.1 as well. TLS 1.0 differs in two key ways, changing the applicability of our results. First, standard-compliant implementations of TLS 1.0 allowed an attacker to distinguish between decoding failures (arising from incorrectly formatted padding) and authentication failures (arising from MAC verification failures). It was shown in [9, 22] how this difference could be exploited to decrypt ciphertexts in the OpenSSL implementation of TLS. Consequently, the TLS 1.1 and 1.2 specifications mandate that implementations prevent such attacks by enforcing uniform error reporting (both in terms of timing and the actual message returned). Our positive results are in this uniform error reporting model and don't necessarily apply when non-uniform error reporting is in effect.

The second difference is that in TLS 1.0 CBC mode used chained IVs, meaning that the IV used to encrypt a message is set to the last ciphertext block from the previously sent ciphertext. As reported in [19], Rogaway and Dai found distinguishing attacks that exploit chained IVs, and so in TLS 1.1 and beyond, dedicated IVs are required. Our attacks and proof only apply when dedicated IVs are used as in TLS 1.1 and 1.2.

Recap and discussion. Putting together all our results, we see that the exact nature of encoding in MEE must be carefully considered when analysing protocols based upon it. Our attacks and positive results characterize the parameters under which MEE-TLS-CBC falls to (at least) distinguishing attacks and those under which we can have significantly better confidence in security via our proofs. To recap, tag size matters: too small and security fails, large enough and LHAE security can be proved.

We are in contact with those involved in TLS standardization, and hope that vulnerabilities in future versions can be avoided. There are several ways to protect TLS from these problems. For example, one could include the padding length in the MAC scope. Our attacks would no longer work and, in fact, one should be able to prove LHAE security. The best solution is to stop using using MEE-based encryption within TLS (and elsewhere). Instead, one could use Encrypt-then-MAC or one of the dedicated AE schemes. We note that our LHAE notion is interesting for these as well, allowing one to show, for example, that Encrypt-then-MAC achieves some degree of length hiding in the case where one uses CBC.

2 Notation, Syntax and Basic Security Notions

Notation. When $\mathcal{X}$ is a set, we write $X \leftarrow_{\$} \mathcal{X}$ to mean that a element (named X) is uniformly sampled from $\mathcal{X}$. We overload the notation for probabilistic or stateful algorithms, writing $X \leftarrow_{\$} M$ to mean that algorithm M runs and outputs value named X. The set $\{0,1\}^{\leq n}$ contains all bitstrings of length at most n bits, and as usual $\{0,1\}^*$ is the set of all finite length strings. When X and Y are strings, we write $X \| Y$ for their concatenation. When $X \in \{0,1\}^*$ we write $|X|$ for its length. For a tuple of strings $(X_1, X_2, \ldots, X_b)$ we define $|(X_1, X_2, \ldots, X_b)| = |X_1 \| X_2 \| \cdots \| X_b|$.

We often use the notation $M \Rightarrow x$ to denote the event (defined over some specified probability space) that at some algorithm M outputs value x.

An *adversary* A is a probabilistic algorithm that takes zero or more oracles, these denoted as superscripts.

Function Families, PRFs and SPRPs. Fix sets $\mathcal{D}, \mathcal{R}$ and non-empty set $\mathcal{K}$. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a mapping. For each $K \in \mathcal{K}$ we write $F_K(\cdot)$ for $F(K, \cdot)$ and thus think of F as a *function family* indexed by K. Let $\text{Func}(\mathcal{D}, \mathcal{R})$ denote the set of all functions from $\mathcal{D}$ to $\mathcal{R}$. Let A be an adversary. We define $\mathbf{Adv}_F^{\text{prf}}(A) = \Pr\left[\, K \leftarrow\!\!\$\, \mathcal{K} ;\, A^{F_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, f \leftarrow\!\!\$\, \text{Func}(\mathcal{D}, \mathcal{R}) \,:\, A^{f(\cdot)} \Rightarrow 1 \,\right]$ to be the *PRF*-advantage of A attacking F. We overload notation and write $\mathbf{Adv}_F^{\text{prf}}(t, q, \mu)$ to mean the maximum of $\mathbf{Adv}_F^{\text{prf}}(A)$ over all adversaries A that run in time t, ask q queries, these totalling μ bits in length.

Fix integers $k, n > 0$, and let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a function family. If for every $K \in \{0,1\}^k$ we have that $E_K(\cdot)$ is a permutation (bijective mapping), then E is a *blockcipher*, and we call n the *blocksize*. We write $\text{Perm}(n)$ for the set of all permutations over $\{0,1\}^n$. We define $\mathbf{Adv}_E^{\text{sprp}}(A) = \Pr\left[\, K \leftarrow\!\!\$\, \{0,1\}^k \,:\, A^{E_K(\cdot), E_K^{-1}(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \pi \leftarrow\!\!\$\, \text{Perm}(n) \,:\, A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1 \,\right]$ to be the *strong PRP*-advantage of A attacking F. Again, we overload our notation and write $\mathbf{Adv}_E^{\text{sprp}}(t, q_1, q_2)$ to mean the maximum of $\mathbf{Adv}_E^{\text{sprp}}(A)$ over all adversaries A that run in time t, asking a total of q queries to its oracles.

Encryption Schemes and MACs. An *encryption scheme* $\mathsf{SE} = (\mathsf{K}_{\text{se}}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of algorithms. The probabilistic algorithm K_{se} samples from a finite and non-empty set $\mathcal{K}_{\text{se}}$. The *encryption algorithm* Enc and *decryption algorithm* Dec take an input $(K, \ell, H, M) \in \mathcal{K}_{\text{se}} \times \mathbb{N} \times \{0,1\}^* \times \{0,1\}^*$ (the key, output length, associated data, and message or ciphertext) and outputs either a string or the distinguished output $\perp$. The encryption algorithm can be probabilistic while decryption is always deterministic. We assume there are sets $\mathcal{H} \subseteq \{0,1\}^*$ (the header space), $\mathcal{L} \subseteq \mathbb{N}$ (the requested length space), $\mathcal{M} \subseteq \{0,1\}^*$ (the message space) such that for all $K \in \mathcal{K}_{\text{se}}$ it holds that $\Pr[\mathsf{Enc}_K(\ell, H, M) \in \{0,1\}^*] = 1$ if $(\ell, H, M) \in \mathcal{L} \times \mathcal{H} \times \mathcal{M}$ and $\Pr[\mathsf{Enc}_K(\ell, H, M) = \perp] = 1$ if $(\ell, H, M) \notin \mathcal{L} \times \mathcal{H} \times \mathcal{M}$. For correctness we require that for all $(K, \ell, H, M) \in \mathcal{K}_{\text{se}} \times \mathcal{L}, \mathcal{H}, \mathcal{M})$ it holds that $\Pr[\mathsf{Dec}_K(H, \mathsf{Enc}_K(\ell, H, M)) = M] = 1$.

We further make a restriction that whether or not Enc returns $\perp$ does not vary with the message length (all other inputs kept equal). Formally, for all keys $(K, \ell, H) \in \mathcal{K}_{\text{se}} \times \mathcal{L} \times \mathcal{H}$ and for all $M, M' \in \mathcal{M} \times \mathcal{M}$ such that $|M| = |M'|$ it holds that for all coins $\mathsf{Enc}_K(\ell, H, M) = \perp$ iff $\mathsf{Enc}_K(\ell, H, M') = \perp$.

Let us make a few comments on what this syntax captures. First, because ℓ is a parameter of encryption, the syntax supports encryption schemes that return variable-length ciphertexts of the same plaintext M. Second, for any fixed plaintext length m, either all $M \in \{0,1\}^m$ encrypt to valid ciphertexts, or none of them do. Third, if ℓ and M are such that encryption would return $\perp$ (e.g. because $\ell < |M|$, or the encryption algorithm does not support ciphertexts of length ℓ), then it does so always. Finally, since decryption does not take the length parameter ℓ, our correctness requirement implicitly demands that the length of the underlying plaintext can be inferred given (K, H, C) where $C = \mathsf{Enc}_K(\ell, H, M)$.

Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Then the encryption scheme CBC[E] has message space $\{0,1\}^{n+}$ (all strings that are a multiple of n bits). Key generation K_{se} outputs a random $K \leftarrow_{\$} \{0,1\}^k$. On input including a message $M = M_1 \,\|\, \cdots \,\|\, M_m \in \{0,1\}^{nm}$, encryption ignores any requested length or header inputs and returns the ciphertext $C_0 \,\|\, \ldots \,\|\, C_{m+1}$ where $C_0 \leftarrow_{\$} \{0,1\}^n$ and $C_i \leftarrow E_K(C_{i-1} \oplus M_i)$ for $1 \le i \le m$.

Fix an integer $\tau > 0$. A *message authentication code* (MAC) is a function family $F\colon \mathcal{K}_{\mathrm{ma}} \times \mathcal{D} \to \{0,1\}^\tau$, where τ is the *tag length* of the MAC.

Conventions. The running time of algorithms (e.g. adversaries) is relative to some implicit underlying RAM model of computation. The running time of an adversary is assumed to include the time to execute the entire experiment in which it executes, including (for example) the time for its oracles to execute. Throughout we fix the convention that adversaries do not ask *pointless* queries: they do not query an oracle on a value outside of its domain, nor on values that are defined to cause a $\perp$ return value. Also, adversaries are assumed not to repeat queries to deterministic oracles. This convention is made without loss of generality.

3 MAC-Encode-Encrypt and the TLS Record Protocol

The TLS Record Protocol uses the MAC-then-encode-then-encrypt paradigm. The algorithm first applies a message authentication scheme to the message and header to derive a tag. The message and tag are then encoded into a bit string according to some encoding rules. Finally an encryption scheme is used to encrypt the result.

Encoding schemes. An *encoding scheme* $\mathsf{CODE} = (\mathsf{encode}, \mathsf{decode})$ is a pair of deterministic algorithms. The *encoding algorithm* encode takes an input $(\ell, M, T) \in \mathbb{N} \times \{0,1\}^* \times \{0,1\}^*$ (the output length, message, and tag) and returns a string of length ℓ or the distinguished symbol $\perp$. An encoding scheme is assume to have a fixed maximum allowable output length $\ell_{\max}$. If $\ell < |M| + |T|$ or $\ell > \ell_{\max}$ then encode returns $\perp$. The *decoding algorithm* decode takes an input in $\{0,1\}^*$ and returns an element of $\{0,1\}^* \times \{0,1\}^*$ or $(\perp, \perp)$. If either algorithm is called on an input outside of its specified domain, it returns an appropriate failure symbol. For correctness we require that, for all ℓ, M, and T such that $\mathsf{encode}(\ell, M, T) \neq \perp$ we have $\mathsf{decode}(\mathsf{encode}(\ell, M, T)) = (M, T)$.

The MEE AEAD scheme. We define the MEE scheme that forms the basis for encryption in TLS, some modes of IPSec, and elsewhere. Fix some block size n. Let $\mathsf{SE} = (\overline{\mathsf{K}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be an encryption scheme with a message space $\{0,1\}^{n+}$ (all strings of length a multiple of n). We assume that, given inputs of an appropriate length, the algorithms $\overline{\mathsf{Enc}}$, $\overline{\mathsf{Dec}}$ are failure-free. Let $F\colon \mathcal{K}_{\mathrm{ma}} \times \{0,1\}^* \to \{0,1\}^\tau$ be a function. Let $\mathsf{CODE} = (\mathsf{encode}, \mathsf{decode})$ be an encoding

```
alg. $\mathsf{K}$:
    $K_{\mathrm{se}} \leftarrow\!\!\$\ \overline{\mathsf{K}}$
    $K_{\mathrm{ma}} \leftarrow\!\!\$\ \mathcal{K}_{\mathrm{ma}}$
    Ret $(K_{\mathrm{ma}}, K_{\mathrm{se}})$

alg. $\mathsf{Enc}_K(\ell, H, M)$:
    $(K_{\mathrm{ma}}, K_{\mathrm{se}}) \leftarrow K$
    $T \leftarrow\!\!\$\ F_{K_{\mathrm{ma}}}(H, M)$
    $X \leftarrow \mathsf{encode}(\ell - n, M, T)$
    If $X = \perp$ then Ret $\perp$
    Ret $Y \leftarrow\!\!\$\ \overline{\mathsf{Enc}}_{K_{\mathrm{se}}}(X)$

alg. $\mathsf{Dec}_K(H, C)$:
    $(K_{\mathrm{ma}}, K_{\mathrm{se}}) \leftarrow K$
    $X \leftarrow \overline{\mathsf{Dec}}_{K_{\mathrm{se}}}(C)$
    $(M, T) \leftarrow \mathsf{decode}(X)$
    If $(M, T) = (\perp, \perp)$ then Ret $\perp$
    If $F_{K_{\mathrm{ma}}}(H, M) \neq T$ then Ret $\perp$
    Ret $M$
```

Fig. 2. Algorithms for the MEE generic composition

scheme for which the outputs of encode all have bit lengths a positive multiple of n. Then $\mathsf{MEE}[F, \mathsf{CODE}, \mathsf{SE}] = (\mathsf{K}, \mathsf{Enc}, \mathsf{Dec})$ is defined as shown in Figure 2.

Notice that Enc takes as input a requested ciphertext length ℓ, as well as associated data H and message M. The inclusion of ℓ allows for variable length padding to be used, while the inclusion of H allows us to incorporate additional fields in the MAC scope, for example, TLS's sequence numbers and compression type and version fields. Notice that Dec can fail either because of a failure to properly decode the message X, or because of a failure to verify the MAC tag T. However, in our specification of the MEE scheme, these error events are *not* distinguishable. This prevents the attacks of [9, 22] and is in-line with the TLS specification [12]. In TLS, any such errors are *fatal*, leading to the destruction of the TLS connection and the disposal of the keys, meaning that an attacker can no longer interact with the protocol. In our description of MEE, these errors are non-fatal, allowing an attacker to continue to interact with the MEE scheme after an error has arisen. It is easy to see that security with non-fatal errors immediately implies security with fatal errors, since any adversary in the former case is more powerful than in the latter case. Thus any security results we prove about MEE will imply security for the more realistic version of MEE in which errors are fatal.

TLS ENCODING. Let $\mathsf{TLScode} = (\mathsf{TLSencode}, \mathsf{TLSdecode})$ be the encoding scheme defined in Figure 3. This scheme is parameterized by the integers ψ, n, and τ, representing the maximal padding length, a block length, and a tag length. Recall that we work with bits in our algorithmic descriptions and cryptographic analysis, rather than with bytes as in the TLS specification [12].

For TLS, ψ can be as large as 2048, since the longest padding pattern that is permitted consists of 256 copies of the byte value FF_x. However, an implementation may select a smaller value of ψ. Note that this scheme has a decoding algorithm permitting variable length padding of any length (not limited by ψ). This decoding algorithm checks every byte of padding to ensure that it is correct. It also allows the final message M (obtained after removing padding and parsing the resulting string into message M and MAC tag T) to be of zero length. Again, these choices are in accordance with the TLS specification [12].

GENERALIZING TLS ENCODING. For the purposes of our positive results, we will analyze a generalization of TLS encoding. An encoding scheme $\mathsf{CODE} =$

```
alg. TLSencode($\ell, M, T$):
If $\ell \bmod n \neq 0$ then Ret $\perp$
$p \leftarrow \ell - (|M| + |T|)$
If $p < 8$ then Ret $\perp$
If $p > \psi$ then Ret $\perp$
If $p \bmod 8 \neq 0$ then Ret $\perp$
$P \leftarrow \text{int2byte}(p/8) - 1$
$X \leftarrow M \parallel T \parallel P \cdots_{P+1} P$
Ret $X$
```

```
alg. TLSdecode($X$):
If $|X| \bmod n \neq 0$ then Ret $(\perp, \perp)$
$(X, P) \leftarrow \text{split}_{|X|-8,8}(X)$
$b \leftarrow \text{byte2int}(P)$
$p \leftarrow 8 \cdot b$
If $|X| - p - \tau < 0$ then Ret $(\perp, \perp)$
For $i = 1$ to $b$ do
    $(X, P') \leftarrow \text{split}_{|X|-8,8}(X)$
    If $P \neq P'$ then Ret $(\perp, \perp)$
$(M, T) \leftarrow \text{split}_{|X|-\tau,\tau}(X)$
Ret $(M, T)$
```

Fig. 3. Algorithms for the TLS encoding scheme

(encode, decode) is *MEE sufficient* if it is parameterized by a block length n and tag length τ and has the following properties:

(1) The output encode(ℓ, M, T) consists of a string $M \parallel T \parallel P \in \{0,1\}^{in}$ for some $i \geq 1$ and where $|P| = \ell - |M| - |T|$. The particular padding P is uniquely determined by $|P|$.

(2) Algorithm decode(X) for $|X| = \ell$ returns (M, T) only if encode(ℓ, M, T) outputs $M \parallel T \parallel P$.

(3) CODE yields *prefix-free padding*, which means that for any M, M' such that $|M| = |M'|$, for any T, T', the padding P returned by encode(ℓ, M, T) is not a prefix of the padding returned by encode(ℓ', M', T') for any $\ell \neq \ell'$.

One may be able to relax property (1) in various ways and still prove security, but we focus on this case for greatest simplicity (while still covering TLS encoding). Property (1) and the invertibility of encoding indicate that for any strings M, T and number ℓ for which encode(ℓ, M, T) does not output $\perp$, there is a single string P such that encode(ℓ, M, T) outputs $M \parallel T \parallel P$.

In the proof of our main technical result, Theorem 2, it will be useful to assume that one can extract from encode a routine called Pad that, on input $(|M|, \ell)$, simply returns the padding P from $M \parallel T \parallel P$. Similarly, it will be useful to assume that one can extract from decode: (1) a routine called Parse that, on input X, returns the appropriate triple M, T, P; and (2) a routine called PadCheck that, on input $(|M|, P, |X|)$, returns 1 if P is the correct padding, and 0 otherwise. It is easy to see that such routines can be extracted from TLSencode and TLSdecode.

For notational clarity and letting F be some function family that will be clear from context, we let MEE-GEN-CBC = MEE$[F, \text{CODE}, \text{CBC}]$ be a mnemonic defining the scheme that uses a MEE-sufficient encoding scheme CODE with CBC. In particular, we let MEE-TLS-CBC = MEE$[F, \text{TLScode}, \text{CBC}]$. When we need to be explicit, we write CBC$[E]$ to mean that CBC encryption is done over a function family E.

```
main $\mathrm{LHAE}_{\mathsf{SE}}$:
    $K \leftarrow_{\$} \mathcal{K}_{\mathrm{se}}$
    $b \leftarrow_{\$} \{0,1\}$
    $b' \leftarrow A^{\mathbf{Enc},\mathbf{Dec}}$
    Ret $(b' = b)$

procedure $\mathbf{Enc}(\ell, H, M_0, M_1)$:
    $C_0 \leftarrow_{\$} \mathsf{Enc}_K(\ell, H, M_0)$
    $C_1 \leftarrow_{\$} \mathsf{Enc}_K(\ell, H, M_1)$
    If $C_0 = \perp$ or $C_1 = \perp$ then
        Ret $\perp$
    $\mathcal{C} \overset{\cup}{\leftarrow} C_b$ ; Ret $C_b$

procedure $\mathbf{Dec}(H, C)$:
    If $b = 1 \wedge C \notin \mathcal{C}$ then
        Ret $\mathsf{Dec}_K(H, C)$
    Ret $\perp$
```

Fig. 4. Length-hiding AEAD security game

4 Length-Hiding Authenticated Encryption

Here we formalize security goals for the TLS Record Protocol, and establish some basic results about these goals. We target authenticated encryption security, which requires (informally) that an adversary cannot generate new, valid ciphertexts itself, nor learn partial information about encrypted plaintexts. Note that this implies traditional chosen-ciphertext attack security. One security aspect traditional AE security goals do not treat, however, is length hiding. As we saw in the previous section, the TLS standard includes the option for variable-length-padding so that applications can choose to hide exact message lengths. Even in the minimal-length-padding case some amount of length hiding could exist since one must pad to the next block boundary. Classical security goals, such as semantic security and the stronger AE notion mentioned above, explicitly leak message lengths. Thus one cannot use these to reason about the length-hiding capabilities of MEE-TLS-CBC. We therefore give a new security notion to capture length hiding under chosen-length attacks. It generalizes the randomized AEAD security notion given in [20].

Length-hiding AEAD security. Let $\mathsf{SE} = (\mathsf{K}_{\mathrm{se}}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and let A be an adversary. Figure 4 details a security length-hiding authenticated-encryption game. We define the LHAE-advantage (of A) to be $\mathbf{Adv}^{\mathrm{lh\text{-}ae}}_{\mathsf{SE}}(A) = 2 \cdot \Pr\left[\,\mathrm{LHAE}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right] - 1$. Let LHAE1 (resp. LHAE0) be the LHAE game except with b set to one (resp. zero). Then a standard argument gives that $\mathbf{Adv}^{\mathrm{lh\text{-}ae}}_{\mathsf{SE}}(A) = \Pr\left[\,\mathrm{LHAE1}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{LHAE0}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{false}\,\right]$. We write $\mathbf{Adv}^{\mathrm{lh\text{-}ae}}_{\mathsf{SE}}(q_e, \mu_e, q_d, \mu_d)$ to mean the maximum of $\mathbf{Adv}^{\mathrm{lh\text{-}ae}}_{\mathsf{SE}}(A)$ taken over all adversaries A that run for t computational steps, asking at most q_e queries to its left oracle that result in ciphertexts of total length μ_e bits, and q_d queries to its right oracle that total μ_d bits in length. Restricting attention to adversaries A for which $q_d = \mu_d = 0$ yields a length-hiding version of the IND-CPA notion, which we denote by LH-IND-CPA. We let $\mathbf{Adv}^{\mathrm{lh\text{-}ind\text{-}cpa}}_{\mathsf{SE}}(A) = \mathbf{Adv}^{\mathrm{lh\text{-}ae}}_{\mathsf{SE}}(A)$ for A that make no decryption queries.

The LHAE notion captures chosen-length attacks along two dimensions. First, we allow $|M_0| \neq |M_1|$ unlike in previous formulations of encryption security. This captures that an attacker cannot distinguish between the encryptions of two chosen messages of arbitrary lengths. We only require that queried messages both encrypt to a ciphertext (not $\perp$). This restriction is necessary to avoid trivial wins in which an attacker abuses two tuples (ℓ, H, M_0) and (ℓ, H, M_1) for which only one is handled by encryption. Second, we allow the adversary to adaptively pick ℓ for each query. A weaker notion restricts attention to a specific ℓ for the entire experiment. Indeed, this fixed-ciphertext-length notion may be sufficient for some applications. Our attacks (in the next section) show insecurity against this weaker notion, and so, by extension, the LHAE notion. On the other hand, our proofs target the stronger notion, meaning when the proofs are applicable, length-hiding security is achieved even if applications dynamically change ciphertext lengths for a single key as done by GnuTLS [14] or if one implemented traffic morphing [27] using MEE.

5 Attacking TLS for Short Messages and Tags

Next we sketch attacks against the MEE scheme as used in TLS and as described in Section 3. In this section, for convenience, we work bytewise.

We give an attack that causes a decryption collision (recall: two valid ciphertexts that decrypt to the same plaintext). For concreteness, let $n = 128$ and $\tau = 80$. This would be the case for truncated MACs [13]. Now suppose the attacker can obtain a ciphertext $C = C_0||C_1||C_2$ for a message M with $|M| = 40$. Then the attacker computes a new ciphertext $C' = C_0' \parallel C_1$ where $C_0' = C_0 \oplus \mathsf{0x00} \cdots_{14} \mathsf{0x00}\ \mathsf{0x00}\ \mathsf{0x10}$, where $\mathsf{0xab} \ldots_k \mathsf{0xab}$ signifies a total of k copies of the byte value $\mathsf{0xab}$. The plaintext underlying the CBC mode ciphertext C is $M \parallel T \parallel \mathsf{0x11} \cdots_{20} \mathsf{0x11}$. It is easy to verify that the plaintext underlying C' is $M \parallel T \parallel \mathsf{0x01}$, which is correctly formatted and, since it has the same message and tag as in C, will verify.

This attack can be extended to break $\mathsf{MEE\text{-}TLS\text{-}CBC}$ in the traditional IND-CCA sense. With parameters as before, suppose the attacker receives from its encryption oracle a 3-block encryption C of M_b, one of two 5-byte messages M_0, M_1. (The messages are the same length.) Then the attacker can modify C by truncation and bit flipping in the IV to produce a fresh ciphertext C' which is a valid encryption of M_b. At this point C' may be submitted to the decryption oracle and the returned plaintext will be M_b, allowing the attacker to win the IND-CCA game with probability 1. While this attack rules out MEE meeting IND-CCA security (for short messages and MACs), notice that it does not seem to translate into a mountable attack on TLS. This is because an attacker that intercepts C and sends C' instead will not see any difference in the behaviour of the TLS connection as compared to having just sent C. One may conclude from this that CTXT security, which is violated here, is overly strong and the abilty to find decryption collisions does not endanger security.

This intuition is wrong, and in fact what we'll see is that IND-CCA is in fact *too weak* to capture the problem that decryption collisions give rise to. Consider a client sending a short message, either "YES" or "NO" encoded as a 3-byte string or 2-byte string. Note these are of two *different* lengths, and so the IND-CCA security definition excludes such a pair from consideration. Let $M \in \{\mathsf{YES}, \mathsf{NO}\}$ denote the message the client encrypts, which is not known to the attacker. Assume the client uses extra padding (such as done by GnuTLS) to mask lengths; say the chosen extra padding during encryption was enough to fill up one extra block. The attacker intercepts the ciphertext $C = C_0 \parallel C_1 \parallel C_2$ generated by the client. It then generates a new ciphertext $C' = C'_0 \parallel C_1$ where

$$C'_0 = C_0 \oplus \mathsf{0x00} \cdots_{12} \mathsf{0x00\ 0x10\ 0x10\ 0x10\ 0x10}\ .$$

The attacker then forwards C' in place of C to the server and observes whether decryption succeeds (say, by seeing if the session is torn down). If decryption succeeds, the attacker knows that $M = \mathsf{NO}$ and otherwise that $M = \mathsf{YES}$. Why does this work? The plaintext for CBC underlying C is either $\mathsf{NO} \parallel T_{no} \parallel \mathsf{0x14} \cdots_{20} \mathsf{0x14}$ or $\mathsf{YES} \parallel T_{yes} \parallel \mathsf{0x13} \cdots_{19} \mathsf{0x13}$. If the former, then decrypting C' succeeds since the padding underlying C' is exactly $\mathsf{0x04} \cdots_{4} \mathsf{0x04}$. But in the latter case, the CBC decryption step applied to C' yields $\mathsf{YES} \parallel T'_{yes} \parallel \mathsf{0x03\ 0x03\ 0x03}$ where $T'_{yes} = T_{yes} \oplus \mathsf{0x00} \cdots_{9} \mathsf{0x00\ 0x10}$. Since the MAC tag is deterministic, it cannot be that this MAC verifies and so decryption fails.

This attack extends immediately to handle TLS's sequence numbers and associated data. It also extends to give LHAE attacks for a variety of pairs of message lengths, including combinations where one message is short (a few bytes) and the other is long (even up to 15 blocks in size). The example can be generalised to a variety of MAC sizes. Indeed, the attack still works in the extreme case where the MAC size is just 8 bits less than the block size[2], in which case one of the messages in the attack is of zero length, a length permitted in the TLS specification [12].

This distinguishing attack can be mounted in practice against TLS if an implementation uses sufficiently short MAC tags, such as those arising from the widespread use of truncated MACs (as done in IPsec and SSH). Fortunately TLS 1.2 does not support short enough MACs, but 80-bit truncated MACs are explicitly defined for use in extensions to TLS 1.2 [13]. In these extensions, then, we have a vulnerability: a man-in-the-middle attacker can violate TLS's confidentiality design goal.

6 The CRD Security Notion

We saw in the last section that MEE with TLS paddding is always LHAE insecure when $\tau + |M| \leq n - 8$ (where n is the underlying blockcipher length). Our goal in the rest of the paper is therefore to prove that when $\tau + |M| > n - 8$ the MEE scheme is LHAE secure for the generalized TLS encoding scheme described in Section 3. This will yield as a special case the first proof that the full TLS Record Protocol is secure for standard chosen-ciphertext attack models.

Consider first the non-length-hiding case. Then a natural approach is to target the two properties IND-CPA and ciphertext integrity (CTXT). Recall that CTXT [3] rules out the ability of an attacker to produce a valid ciphertext not before returned by an encryption oracle. A result of Rogaway and Shrimpton [20] states that satisfying both IND-CPA and CTXT is equivalent to AE security.

[2] This case is extreme because TLS is a byte-oriented protocol.

In the full versionwe state and prove a generalization of this equivalence for the length hiding setting and also show that MEE is length-hiding IND-CPA (LH-IND-CPA). The proofs are easy extensions of the proofs in the non-length-hiding setting.

The complexity of the analysis lies in showing CTXT. Consider the analysis by Krawczyk [15] for a restricted version of MEE with CBC that, unfortunately, does not cover any usage case of TLS. His proof shows that MEE is single-query CTXT in the case that $\tau = n$, and encoding is both injective and ensures that the tag fills exactly one plaintext block for the underlying encryption. These restrictions make a proof more manageable, in particular leading to a simpler final case analysis. In our setting, a direct CTXT analysis would require many more cases, these induced by the relaxation to variable length padding and the fact that tags may span multiple plaintext blocks. To ameliorate this complexity, we takemore modular approach to proving CTXT.

CRD SECURITY. We introduce a new notion of security for encryption schemes called *collision-resistant decryption* (CRD). This enables proofs of CTXT to be split into two self-contained parts and helps modularize our analysis further. Recall that plaintext integrity (PTXT) requires that an adversary not be able to construct a ciphertext that decrypts to a valid message that was not before queried to the encryption oracle. As mentioned above, CTXT rules out constructing any new ciphertext. As shown by Bellare and Namprempre [3], PTXT is a strictly weaker property than CTXT. We show that CRD is exactly the "gap" between the two properties. Informally, CRD security requires that an attacker cannot produce a new ciphertext that decrypts to a message previously queried to the encryption oracle. One can see, in fact, that the attacks of the previous section are, at their core, breaking MEE in the sense of CRD.

Let $\mathsf{SE} = (\mathsf{K}_{\rm se}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme, and let A be an adversary. We define the collision-resistant decryption advantage of A as $\mathbf{Adv}^{\rm crd}_{\mathsf{SE}}(A) = \Pr\left[\,\mathrm{CRD}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right]$ where the game $\mathrm{CRD}_{\mathsf{SE}}$ is defined in Figure 5. In the usual way, we write $\mathbf{Adv}^{\rm crd}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d)$ to mean the maximum of $\mathbf{Adv}^{\rm crd}_{\mathsf{SE}}(A)$ over all adversaries A that run for t computational steps, asking at most q_e queries to its encryption oracle that total at most μ_e bits in length and asking at most q_d queries to its test oracle that total at most μ_d bits in length.

Figure 5 also specifies the games $\mathrm{CTXT}_{\mathsf{SE}}$ and $\mathrm{PTXT}_{\mathsf{SE}}$. We similarly define $\mathbf{Adv}^{\rm ctxt}_{\mathsf{SE}}(A) = \Pr\left[\,\mathrm{CTXT}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right]$ and $\mathbf{Adv}^{\rm ptxt}_{\mathsf{SE}}(A) = \Pr\left[\,\mathrm{PTXT}^{A}_{\mathsf{SE}} \Rightarrow \mathsf{true}\,\right]$. We also define $\mathbf{Adv}^{\rm ctxt}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d)$ and $\mathbf{Adv}^{\rm ptxt}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d)$ analogously.

The following theorem shows that the combination of PTXT and CRD security yields CTXT security. We omit the straightforward proof.

Theorem 1. (PTXT + CRD $\Rightarrow$ CTXT) *Let* $\mathsf{SE} = (\mathsf{K}_{\rm se}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme. Then* $\mathbf{Adv}^{\rm int\text{-}ctxt}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d) \leq \mathbf{Adv}^{\rm int\text{-}ptxt}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d) + \mathbf{Adv}^{\rm crd}_{\mathsf{SE}}(t, q_e, \mu_e, q_d, \mu_d)$. ☐

Given Theorem 1 and our earlier remarks about LHAE being implied by LH-IND-CPA and CTXT, analyzing the LHAE security of any scheme can be separated into showing that LH-IND-CPA, PTXT and CRD are achieved. This

```
main CTXT_SE:
$K \leftarrow\!\!\$\ \mathcal{K}_{se};\ \mathcal{S} \leftarrow \emptyset;\ i \leftarrow 0$
$\mathsf{win} \leftarrow \mathsf{false}$
$(H^*, C^*) \leftarrow\!\!\$\ A^{\mathbf{Enc},\mathbf{Test}}$
Ret $\mathsf{win}$

procedure Enc$(\ell, H, M)$:
$i \leftarrow i + 1$
$H_i \leftarrow H\ ;\ M_i \leftarrow M$
$C_i \leftarrow\!\!\$\ \mathsf{Enc}_K(\ell, H_i, M_i)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(H_i, C_i)\}$
Ret $C_i$

procedure Test$(H^*, C^*)$:
$M^* \leftarrow \mathsf{Dec}_K(H^*, C^*)$
If $M^* \neq \perp \wedge (H^*, C^*) \notin \mathcal{S}$
    then $\mathsf{win} \leftarrow \mathsf{true}$
Ret $(M^* \neq \perp)$
```

```
main PTXT_SE:
$K \leftarrow\!\!\$\ \mathcal{K}_{se};\ \mathcal{S} \leftarrow \emptyset;\ i \leftarrow 0$
$\mathsf{win} \leftarrow \mathsf{false}$
$(H^*, C^*) \leftarrow\!\!\$\ A^{\mathbf{Enc},\mathbf{Test}}$
Ret $\mathsf{win}$

procedure Enc$(\ell, H, M)$:
$i \leftarrow i + 1$
$H_i \leftarrow H\ ;\ M_i \leftarrow M$
$C_i \leftarrow\!\!\$\ \mathsf{Enc}_K(\ell, H_i, M_i)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(H_i, M_i)\}$
Ret $C_i$

procedure Test$(H^*, C^*)$:
$M^* \leftarrow \mathsf{Dec}_K(H^*, C^*)$
If $M^* \neq \perp \wedge (H^*, M^*) \notin \mathcal{S}$
    then $\mathsf{win} \leftarrow \mathsf{true}$
Ret $(M^* \neq \perp)$
```

```
main CRD_SE:
$K \leftarrow\!\!\$\ \mathcal{K}_{se};\ \mathcal{S} \leftarrow \emptyset;\ i \leftarrow 0$
$\mathsf{win} \leftarrow \mathsf{false}$
$(H^*, C^*) \leftarrow\!\!\$\ A^{\mathbf{Enc},\mathbf{Test}}$
Ret $\mathsf{win}$

procedure Enc$(\ell, H, M)$:
$i \leftarrow i + 1$
$H_i \leftarrow H\ ;\ M_i \leftarrow M$
$C_i \leftarrow\!\!\$\ \mathsf{Enc}_K(\ell, H_i, M_i)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(H_i, C_i)\}$
Ret $C_i$

procedure Test$(H^*, C^*)$:
$M^* \leftarrow \mathsf{Dec}_K(H^*, C^*)$
If $M^* \neq \perp \wedge (H^*, C^*) \notin \mathcal{S}$
  $\wedge \exists i : (H^*, M^*) = (H_i, M_i)$
    then $\mathsf{win} \leftarrow \mathsf{true}$; Ret 1
Ret 0
```

Fig. 5. The CTXT, PTXT, and CRD experiments. The set $\mathcal{S}$ and the counter i are global variables in each game.

modularity is particularly beneficial for the MEE construction, where showing LH-IND-CPA and PTXT is straightforward. We defer discussion of these results to the full version. Instead, we focus next on the most involved task: showing CRD security of MEE using CBC and TLS padding.

7 The CRD Security of MEE-GEN-CBC

In this section we give a formal security bound for MEE-GEN-CBC. In the following theorem we consider the case that $\tau \leq n$, where n is the blocksize of the blockcipher underlying CBC. In fact the bounds hold when $\tau > n$, too. Say that $\tau = n + n'$ for some $n' > 0$. Then we can reduce to the case considered by Theorem 2 by assuming that the adversary actually controls the first n' bits of T; essentially, they are treated as adversarially controlled message bits. Thus we can restrict our attention to the case that $\tau \leq n$, which simplifies our proof. Note that this does not significantly weaken our bound, since the dominating term is a function of n when $\tau > n$. We emphasize that, unlike prior proofs, we make no assumption about the position of the tag.

In what follows, let the *total plaintext length* of an encryption query (ℓ, H, M) in the CRD experiment be the total number of blocks that are consequently encrypted, i.e. the total number of blocks in $M \,\|\, T \,\|\, P$ where T is the tag and P is the padding.

Theorem 2. *Fix $n > 0$ and let $E\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let $\mathsf{CODE} = (\mathsf{encode}, \mathsf{decode})$ be MEE sufficient with blocklength n and taglength $\tau \leq n$. Let $F\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^\tau$ be a function family. Let $\mathsf{SE} = \mathsf{MEE\text{-}GEN\text{-}CBC}$, where CBC is over blockcipher E. Let A be a CRD-adversary that runs in time t; asks q_e encryption queries, the sum of whose total plaintext lengths is σ_e; and asks q_d Test queries, the sum of whose lengths is σ_d blocks. Let $\sigma = \sigma_e + \sigma_d$. Let $b_{\min}$ be the length (in bits) of the shortest message that A queries to its encryption oracle. Then, if $\tau + b_{\min} \geq n$, there exist adversaries B_1, B_2 such that*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{crd}}_{\mathsf{SE}}(A) \leq &\mathbf{Adv}^{\mathrm{prf}}_{F}(B_1) + \mathbf{Adv}^{\mathrm{sprp}}_{E}(B_2) \\ &+ \frac{.5\sigma^2 + \sigma_e^2 + 2\sigma_d\alpha(\alpha+1)q_e + q_e q_d}{2^n} + \frac{q_e q_d}{2^\tau}\end{aligned}$$

where where α is the number of distinct padding patterns. Here B_1 runs in time $t + \sigma\mathsf{Time}_E$ and asks at most $q+1$ queries, and B_2 runs in time $t + O(\sigma)$ and asks at most σ queries. □

The proof can be found in the full version. We note that for TLS with full variable-length padding the parameter α is equal to 256.

Similarly, we can consider the case that minimal length padding is enforced by the encoding scheme. Equivalently, we can restrict to CRD adversaries that query ciphertext lengths ℓ that result in padding only to the closest blocklength. Let us call such adversaries *minimal-length padding respecting.* This case results in exactly the same bound. However for TLS with minimum-length padding the value of α changes to 16.

Corollary 1. *Let all quantities and objects be as in Theorem 2, except that A is a minimal-length padding respecting CRD-adversary. Then, if $\tau + b_{\min} \geq n$, there exist adversaries B_1, B_2*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{crd}}_{\mathsf{SE}}(A) \leq &\mathbf{Adv}^{\mathrm{prf}}_{F}(B_1) + \mathbf{Adv}^{\mathrm{sprp}}_{E}(B_2) \\ &+ \frac{.5\sigma^2 + \sigma_e^2 + 2\sigma_d\alpha(\alpha+1)q_e + q_e q_d}{2^n} + \frac{q_e q_d}{2^\tau}\end{aligned}$$

where where α is the number of distinct padding patterns. Here B_1 runs in time $t + \sigma\mathsf{Time}_E$ and asks at most $q+1$ queries, and B_2 runs in time $t + O(\sigma)$ and asks at most σ queries. □

Acknowledgments. Kenny Paterson was supported by EPSRC Leadership Fellowship EP/H005455/1. Thomas Shrimpton was supported by NSF grants CNS 0627752 and CNS 0845610. Much of this work occurred while Ristenpart and Shrimpton visited RHUL, and they thank RHUL for their kind support.

References

1. Abadi, M., Rogaway, P.: Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). J. Cryptology 20(3), 395 (2007)

2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 1997), pp. 394–403. IEEE (1997)
3. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
4. Bellare, M., Kohno, T., Namprempre, C.: Authenticated encrytion in SSH: Provably fixing the SSH binary packet protocol. In: ACM Conference on Computer and Communications Security, pp. 1–11 (2002)
5. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
6. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing Chosen Ciphertext Security of Encryption Schemes. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
7. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Foundations of Computer Science – FOCS (2001)
8. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
9. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS Channel. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599. Springer, Heidelberg (2003)
10. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246 (January 1999), `http://www.ietf.org/rfc/rfc2246.txt`
11. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (April 2006), `http://www.ietf.org/rfc/rfc4346.txt`
12. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (August 2008), `http://www.ietf.org/rfc/rfc5246.txt`
13. Eastlake III, D.: Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (January 2011), `http://www.ietf.org/rfc/rfc6066.txt`
14. GnuTLS Documentation (2011), `http://www.gnu.org/software/gnutls/documentat.html`
15. Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
16. Liberatore, M., Levine, B.: Inferring the source of encrypted HTTP connections. In: ACM Conference on Computer and Communications Security, pp. 255–263 (2006)
17. Manral, V.: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4835 (April 2007), `http://www.ietf.org/rfc/rfc4835.txt`
18. Maurer, U., Tackmann, B.: On the Soundness of Authenticate-then-Encrypt: Formalizing the Malleability of Symmetric Encryption. In: Proc. 2010 ACM Conference on Computer and Communications Security (CCS 2010), pp. 505–515. ACM (2010)
19. Möller, B.: Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures, `http://www.openssl.org/~bodo/tls-cbc.txt`
20. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)

21. Sun, Q., Simon, D., Wang, Y., Russell, W., Padmanabhan, V., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: IEEE Symposium on Security and Privacy, pp. 19–30 (2002)
22. Vaudenay, S.: Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–546. Springer, Heidelberg (2002)
23. White, A., Matthews, A., Snow, K., Monrose, F.: Phonotactic Reconstruction of Encrypted VoIP conversations: Hookt on fon-iks. In: IEEE Symposium on Security and Privacy (2011)
24. Wright, C., Ballard, L., Coull, S., Monrose, F., Masson, G.: Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In: IEEE Symposium on Security and Privacy, pp. 35–49 (2008)
25. Wright, C., Ballard, L., Coull, S., Monrose, F., Masson, G.: Uncovering Spoken Phrases in Encrypted Voice over IP Conversations. ACM Trans. Inf. Syst. Secur. 13(4) (2010)
26. Wright, C., Monrose, F., Masson, G.: On Inferring Application Protocol Behaviors in Encrypted Network Traffic. Journal of Machine Learning Research 6, 2745–2769 (2006)
27. Wright, C., Coull, S., Monrose, F.: Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In: Network and Distributed Security Symposium – NDSS (2009)