The Landscape of Committing Authenticated Encryption

Mihir Bellare
University of California San Diego
mbellare@ucsd.edu

Viet Tung Hoang
Florida State University
tvhoang@cs.fsu.edu

Cong Wu
Florida State University
wu@cs.fsu.edu

1 Introduction

Authenticated encryption (AE) currently aims to provide privacy and authenticity of the communicated data. Recent attacks and applications [3, 4, 8, 15, 16, 21] motivate an additional security attribute, namely to be committing. This means a ciphertext should be a commitment to the key, and beyond that, possibly to all the inputs to the encryption algorithm.

In the proposed talk, we will start by surveying the landscape of committing AE, listing available schemes and comparing then along a set of metrics involving both security and cost. We will see that committing security can be achieved quite cheaply if one is content with 64-bit security, but we will argue that, due to offline attacks, standards should target 128-bit security. But current schemes for this all pay in increased ciphertext size compared to the underlying (non-committing) AE scheme. We present a new, general, and arguably surprising technique that succeeds in eliminating this overhead, providing 128 bits of committing security, with no increase in ciphertext size compared to the underlying AE scheme. We report on implementation-based performance assessments that show that our new schemes compare well in computation time to alternatives.

2 Background and definitions

AE SCHEMES. Recall that in a nonce-based symmetric encryption scheme SE, encryption takes key K, nonce N, associated data A and message M to deterministically return a ciphertext $C \leftarrow \mathsf{SE.Enc}(K,N,A,M)$, with decryption recovering via $M \leftarrow \mathsf{SE.Dec}(K,N,A,C)$ [24, 25]. AE security asks for privacy of the message and authenticity of both the message and the associated data. In its most basic form, called UNAE (Unique-Nonce AE security) this is under the assumption that nonces are unique, meaning never reused across encryptions [24, 25]. While there are stronger notions (such as misuse-resistant AE [26]), since those notions are not yet standardized, we will focus here on UNAE.

A central scheme is GCM [22]. It is a NIST standard [14] and is used in TLS [27]. Other standardized and widely-used schemes are XSalsa20/Poly1305 and ChaCha20/Poly1305 [9, 10, 11]. All these are UNAE-secure.

COMMITTING SECURITY. We recall notions of what it means for a ciphertext $C \leftarrow \mathsf{SE.Enc}(K,N,A,M)$ to be a commitment, following [6]. The first notion, CMT-1, asks that the commitment be to the key K. In the game formalizing this, the adversary returns a pair $((K_1,N_1,A_1,M_1),(K_2,N_2,A_2,M_2))$ satisfying $K_1 \neq K_2$, and is successful if $\mathsf{SE.Enc}(K_1,N_1,A_1,M_1) = \mathsf{SE.Enc}(K_2,N_2,A_2,M_2)$. Extending this, CMT-4 asks that the commitment be, not just to the key, but to K,N,A,M, meaning to *all* the inputs to SE.Enc. The game changes only in the requirement $K_1 \neq K_2$ being replaced by $(K_1,N_1,A_1,M_1) \neq (K_2,N_2,A_2,M_2)$.

As a mnemonic, think of the integer ℓ in the notation CMT-ℓ as the number of inputs of SE.Enc to which we commit. Note that CMT-4 is equivalent to saying that the function SE.Enc is collision resistant. We will explore the implications of this connection later.

Clearly CMT-4 $\rightarrow$ CMT-1, meaning any scheme that is CMT-4-secure is also CMT-1-secure. So why consider CMT-1 separately? The answer is that CMT-1 security can be achieved more cheaply than CMT-4. But many applications only need CMT-1, and can thus benefit from cheaper schemes dedicated to this goal.

WHY COMMIT? The canonical method for password-based encryption (PKCS#5 [18]) uses a symmetric encryption scheme SE, such as GCM, as a tool. In a surprising attack, LGR [21] show that absence of CMT-1 security in SE leads to a break of the overlying password-based encryption scheme. This attack is circumvented if SE is CMT-1-secure [8].

Broadly, we have seen protocols failing due to absence of CMT-1 security in an underlying encryption scheme and then fixed by its being added. ABN [3] illustrate this when the protocol is PEKS [12]; they also note that when encryption strives to be anonymous, CMT-1 security is necessary for unambiguous decryption. FOR [15] illustrate the issue for an encryption-using Oblivious Transfer protocol and note that encryption not being key-committing has lead to attacks on Private Set Intersection protocols [20]. ADGKLS [4] describe in detail three real-world security failures —the domains are key rotation, envelope encryption and subscribe-with-Google— arising from lack of CMT-1 security.

CMT-4 is a simple, optimally-strong goal: we commit to everything. This means all 4 of the inputs to the encryption algorithm: key, nonce, associated data and message. Some motivation comes from applications; for example, GLR [16] show that committing to header and message is needed for an AE scheme to provide message franking, a capability in messaging systems that allows a receiver to report the receipt of abusive content. But the larger benefit is to increase ease of use and decrease risk of error or misuse. An application designer is spared the burden of trying to understand to exactly which encryption inputs the application needs a commitment; with CMT-4, she is covered.

ATTACKS ON CURRENT SCHEMES. AE schemes that are currently standardized or in use were not designed to have committing security, but it is natural to ask if they happen to have it already. The answer is no. Attacks from [4, 16, 21] show that GCM, XSalsa20/Poly1305, ChaCha20/Poly1305 and OCB [25] are all CMT-1-insecure. MLGR [23] give attacks on CCM [29], EAX [7] and SIV [26]. MLGR [23] and CR [13] give attacks on OCB3 [19]. The conclusion is that we need new schemes.

3 Landscape of committing AE schemes

The research literature provides many committing AE schemes. Here we aim to provide a survey. We first discuss some metrics for the quality of committing AE schemes. Then we will survey how different schemes fare under these metrics.

METRICS. We will consider schemes that are UNAE-secure. Then we will additionally classify them as per the following metrics:

1. *Type of CMT security.* The type of CMT security achieved. There are two choices, namely CMT-1 and CMT-4 as defined above.

2. *Number of bits of CMT security.* Some schemes provide 64 bits of CMT security while others provide 128 or more. The former may suffice in some contexts and permits cheaper schemes, but CMT security (unlike AE security) is subject to offline attack, and so we suggest that a higher number of bits of security (128) is a better choice for a standard.

3. *Ciphertext overhead.* This is defined as the length of the ciphertext minus the length of the message. Now, the authenticity requirement of AE necessitates a tag, which already adds ciphertext overhead. (For GCM, this is 96–128 bits.) However, CMT security may add further ciphertext overhead, the amount depending on the number of bits of CMT security provided.

4. *Computational cost.* We assess the performance of encryption and decryption for CMT-secure schemes by implementation. We use GCM as a baseline, expressing costs as percentage overhead over GCM. The picture here is more complex than for the other dimensions discussed above. Our implementations show that the overhead depends on message length. Schemes show significant performance overheads for short messages, but for long messages and fixed-size AD, the gaps are more narrow.

SCHEMES COMPARED. Figure 1 compares schemes along the first three metrics discussed above. The last (computational cost) is depicted in Figures 2 and 3.

The Figure 1 schemes are based on AES, so $n = 128$. Some schemes allow an integer parameter $e \geq 128$ that determines the number of bits of committing security as shown, with a corresponding increase in ciphertext expansion as shown. We first discuss known schemes and then turn to ours, shown in the grey rows.

As a comparison point, we first show GCM, which has no committing security. The ciphertext expansion is the length of the tag. We have listed the maximum tag length of $n = 128$ bits, but GCM allows shorter tags.

Next we show schemes achieving CMT-1 security. CAU-C1 is a tiny modification of GCM from [6] that provides CMT-1 with no increase in ciphertext size over GCM (ciphertext expansion remains $n = 128$ as in GCM) and a very small increase in computational cost. The drawback is that it has only 64-bits of committing security. The Amazon and libsodium schemes can provide 128 bits of CMT-1 security by setting $e = 384$, but this incurs a 384-bit ciphertext expansion, 256 bits more than GCM or CAU-C1. Also, due to the use of cryptographic hash functions, these schemes are significantly slower than GCM, as shown in Figure 2. The padding fix of [5] pads the message before encrypting with GCM. It can provide at most 96-bits of CMT-1-security, achieved by setting $e = 384$. This is better than 64 but still short of the desired 128. It also has 384-bit ciphertext expansion in this mode.

Turning to CMT-4, recall that this goal is equivalent to asking that the encryption function SE.Enc be collision resistant. As pointed out in [6], since the ciphertext size is independent of the size of the associated data, this necessitates collision-resistant hashing, which has lead schemes to use cryptographic hash functions like SHA256. This increases cost, making CMT-4 schemes slower than CMT-1 schemes.

CAU-C4 [6] hashes the key, nonce and associated data to derive a sub-key, under which the message is CAU-C1-encrypted

Scheme	From	Type of CMT security	Bits of CMT security	Ciphertext expansion
GCM	[14, 22]	–	0	n
CAU-C1	BH1 [6]	CMT-1	$n/2$	n
Amazon	ADG+ [5]	CMT-1	$e/2-n/2$	$e \geq n$
libsodium	libsodium [2]	CMT-1	$e/2-n/2$	$e \geq n$
Padding fix	ADG+ [5]	CMT-1	$e/2-3n/4$	$3n \geq e \geq 2n$
CAU-C1+	This paper	CMT-1	$e/2$	$2n \geq e \geq n$
NC1	This paper	CMT-1	$n-8-\lceil \log_2(n) \rceil$	n
CAU-C4	BH1 [6]	CMT-4	$n/2$	n
CTX	CR [13]	CMT-4	$e/2$	$e \geq n$
CTX+	This paper	CMT-4	$e/2$	$e \geq n$
NC4	This paper	CMT-4	$n-8-\lceil \log_2(n) \rceil$	n

Figure 1: **Comparison of committing AE schemes.** Here n is the block and key length of the underlying block cipher, and e is a parameter assumed to be at least n. Typically AES is the underlying block cipher, so $e \geq n = 128$. Grey rows indicate the new schemes from this work.

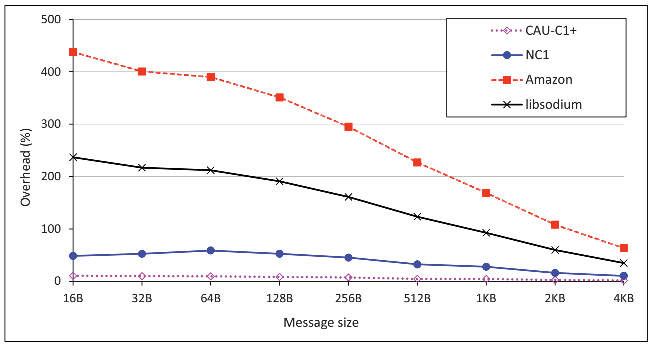

Figure 2: Relative overhead of CMT-1 schemes with respect to GCM. Smaller is better.

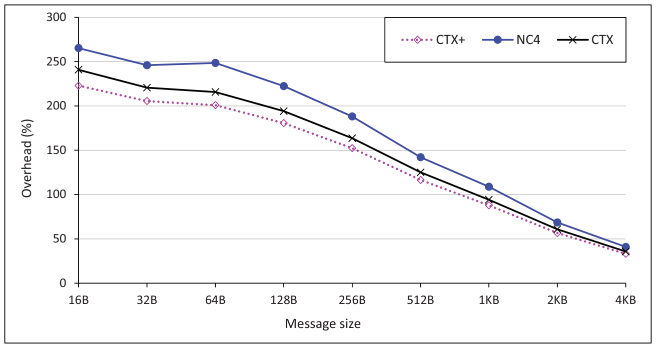

Figure 3: Relative overhead of CMT-4 schemes with respect to GCM. Smaller is better.

with the nonce and empty associated data. Ciphertext expansion remains $n = 128$ as in GCM. The scheme provides CMT-4 security, but, as with CAU-C1, only at the 64-bit level. CTX [13] replaces the GCM tag T with a hash of the key, T, and associated data. For 128-bit security, the hash function needs to have a 256-bit output, so the ciphertext expansion is 256-bits. Due to the use of cryptographic hash functions, CTX has appreciable computational overhead over GCM, as shown in Figure 3.

4 The challenge and our schemes

The prevailing understanding is that CMT-security is subject to birthday attacks, and thus 128-bit security requires a 256-bit ciphertext expansion. We break this barrier to give schemes that achieve 128-bit committing security with 128-bit ciphertext expansion. Let us start with a high-level explanation of why this is possible.

COMMITTING WITH OPTIMAL EXPANSION. For CMT-1, the birthday attack does not apply in general. It does, however, for current schemes, leading to 256-bit expansion for 128-bit committing security. Our new schemes will cut the expansion to 128 bits while preserving the committing security.

Recall that CMT-4 is asking that the function SE.Enc be collision resistant. Moreover, the length of the ciphertext does not depend on the length of the associated data. So, as noted in [6], a birthday attack is always possible. This has lead people to conclude that 128-bit security requires 256-bit expansion. We start with the observation that the birthday attack does not actually necessitate a 256-bit expansion; what it necessitates is a ciphertext length of at least 256 bits. Now, if the message M is at least 128 bits, it is possible in principle to have a ciphertext expansion of only 128 bits. (The expansion would, in general, be $\max(128, 256 - |M|)$ bits.) This makes it possible to at least hope for only 128-bit expansion.

But are there schemes that can actually achieve this? We show that there are. Towards this we introduce and build a primitive called a committing concealer. Let us first overview our schemes.

OUR NEW SCHEMES. As a starting point, our first new CMT-1 scheme is CAU-C1+. It generalizes CAU-C1 by allowing a parameter e determining the CMT-1 security. Setting $e = 256$ yields the desired 128-bits of CMT-1 security but with ciphertext overhead 256 bits, which is already better than the Amazon and libsodium schemes. Moreover, CAU-C1+ reduces computational cost; as shown in Figure 2, CAU-C1+ has about the same speed as GCM itself. For us the main value is that it will be a tool for the next scheme.

Our second and main new CMT-1-scheme is NC1. It provides almost 128 bits of CMT-1 security but, remarkably, with ciphertext overhead only 128 bits. (More precisely, NC1 has about 112 bits of CMT-1 security, and its ciphertext overhead is $\max(128, 248 - |M|)$ bits if the message is M.) Figure 2 shows that, while there is some slowdown compared to GCM and CAU-C1+, our NC1 scheme is still considerably faster than the Amazon and libsodium schemes. Also the slowdown becomes negligible for long messages. NC1 is obtained by applying our general transform EwC to CAU-C1+.

For CMT-4 also we present two schemes. First, we observe an optimization of CTX. The latter processes the associated data twice. We observe that the first encryption pass with GCM can use the empty associated data. We call this scheme CTX+. The number of bits of CMT-4 security and ciphertext expansion remain the same as in CTX, but the computation time is reduced. As per Figure 3, CTX+ is about 5% faster than CTX for 5-byte associated data. With longer associated data, CTX+ will provide even greater speed compared to CTX.

Our main new scheme for CMT-4 security is NC4, obtained by applying our general transform EwC to CTX+. It provides almost 128 bits of CMT-4 security but with ciphertext overhead only 128 bits. (Again, NC4 has 112 bits of CMT-4 security, and its ciphertext overhead is $\max(128, 248 - |M|)$ bits if the message is M.) Figure 3 shows small slowdown compared to CTX and CTX+.

A STEPPING STONE: COMMITTING CONCEALER. The core of our EwC transform is a new primitive that we call a *committing concealer*. In a nutshell, a committing concealer is a committing AE scheme with no nonce or associated data. Also, unlike conventional AE schemes, a committing concealer only targets very short messages (say at most 15 bytes), and its ciphertext length is constant (say 31 bytes), meaning independent of the length of the message.

Our construction of a committing concealer first hashes (K, M) to derive a 15-byte tag T, and then hash (K, T) to mask the 16-byte padded M. To achieve committing security, the hash is based on the Davies-Meyer construction on AES.

At the first glance, given that we use a hash function with 128-bit output, it seems that we can at best achieve 64-bit security. However, modeling AES as an ideal cipher, we show that our construction delivers 112 bits of committing security. The root of the strong security lies in the circularity of first deriving T from hashing M, and then hashing T to mask M.

THE EwC TRANSFORM. Our EwC transform takes as input an AE scheme SE of 256-bit ciphertext overhead, such as CTX+ or CAU-C1+, and produces another scheme $\overline{\mathsf{SE}}$ of just $\max(128, 248 - |M|)$ bits of ciphertext overhead. Moreover, EwC preserves both CMT-1 and CMT-4 security.

Specifically, to encrypt a message M, we first parse it as $S \| P$, where $|P| = \min\{120, |M|\}$. (This means if $|M| \le 120$ then S is the empty string.) We then use SE to encrypt S (with the given key, nonce, and associated data) to derive a ciphertext core C^* and a 256-bit tag T. We then run the committing concealer to encrypt P with key T to produce a 248-bit T^*, and output $C^* \| T^*$. The overhead of the EwC transform is the cost of the committing concealer, and recall that this new primitive can be realized using just two sequential AES-256 calls.

5 Experiments

In this section, we empirically compare the performance of our schemes and the prior GCM-based AE schemes that offer (nearly) 128-bit committing security and constant overhead. As a result, we do not consider the padding fix, CAU-C1, or CAU-C4.

- For CMT-1 security, we compare our CAU-C1+ and NC1 the schemes in Amazon Cloud [1, 4] and the libsodium library [2].
- For CMT-4 security, we compare CTX, CTX+, and NC4.

We implement the schemes for 128-bit key, using the OpenSSL library (version 1.1.1n). Compilation is done with `-O3` optimization, and `-march=native` flag. We use SHA-512 to instantiate the cryptographic hash function in the schemes.

EXPERIMENT SETUP. We run experiments on a server with two Intel Xeon Gold 6240 processors. Each processor has 18 cores and 2.60 GHz base frequency. The server runs CentOS 7.8 with Linux 3.10.0-1160 kernel and has 192GB of DDR4 RAM.

HOW WE BENCHMARK. We evaluate the encryption speed for each scheme on messages from 16B to 4KB; the AD size is set to 5B, the situation in the TLS protocol. For each experiment, we run 10 iterations. In each iteration, we first run the operation for 50,000 times as a warm-up, and then measure the average latency for the next 2,000,000 times. We then compute the median of the 10 average timings; their standard deviation is within 3% of the median. We report the relative overhead compared to GCM; for example, an overhead of 35% means that this scheme is 1.35 time slower than GCM.

CMT-1 SCHEMES. The performance of CMT-1 schemes is shown in Fig. 2. Theoretically, the added cost is constant for every scheme. In reality, the overhead of the industry schemes is expensive, even for moderate data like 1KB, because they use SHA-512 for committing the key. In contrast, CAU-C1+ keeps the commitment cost modest for all data size. The scheme NC1 comes middle of the pack: using a committing concealer adds noticeable overhead to CAU-C1+, but it is still much cheaper than the industry schemes.

CMT-4 SCHEMES. The performance of CMT-4 schemes is shown in Fig. 3. The cost for having CMT-4 is constant in theory, but the actual overhead is expensive for small data, as one has to use a cryptographic hash function like SHA-512. As expected, CTX+ is the fastest, because theoretically it has the optimal performance. Moreover, even with 5-byte AD, there is an appreciable 5% difference between the running time of CTX and that of CTX+. On the other hand, NC4 is only about 15% slower than CTX+, as calling a committing concealer is much cheaper than evaluating a cryptographic hash function.

References

[1] AWS Encryption SDK 2.0. https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html, 2020.

[2] The Sodium cryptography library (Libsodium). https://libsodium.gitbook.io/doc, 2021.

[3] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, February 2010.

[4] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In *31st USENIX Security Symposium*, 2022.

[5] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 3291–3308. USENIX Association, August 2022.

[6] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In *EUROCRYPT 2022*, 2022.

[7] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer, Heidelberg, February 2004.

[8] Mihir Bellare and Laura Shea. Flexible password-based encryption: Securing cloud storage and provably resisting partitioning-oracle attacks. Cryptology ePrint Archive, Report 2023/197, 2023. https://eprint.iacr.org/2023/197.

[9] D Bernstein. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5, 2008.

[10] D Bernstein. The salsa20 family of stream ciphers. In *New stream cipher designs: The eSTREAM finalists, Lecture Notes in Computer Science*, volume 4986. Springer, 2008.

[11] Daniel J. Bernstein. The poly1305-AES message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 32–49. Springer, Heidelberg, February 2005.

[12] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

[13] John Chan and Phillip Rogaway. On committing authenticated-encryption. In *27th European Symposium on Research in Computer Security (ESORICS)*, 2022.

[14] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, November 2007.

[15] Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symm. Cryptol.*, 2017(1):449–473, 2017.

[16] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, August 2017.

[17] Shay Gueron and Yehuda Lindell. Better bounds for block cipher modes of operation via nonce-based key derivation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1019–1036. ACM Press, October / November 2017.

[18] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, Sep. 2000. https://datatracker.ietf.org/doc/html/rfc2898.

[19] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 306–327. Springer, Heidelberg, February 2011.

[20] Mikkel Lambæk. Breaking and fixing private set intersection protocols. Cryptology ePrint Archive, Report 2016/665, 2016. https://eprint.iacr.org/2016/665.

[21] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium*. USENIX Association, 2021.

[22] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, December 2004.

[23] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 379–407. Springer, Heidelberg, April 2023.

[24] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002.

[25] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 196–205. ACM Press, November 2001.

[26] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

[27] J. Salowey, A. Choudhury, and D. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288, Aug. 2008. https://datatracker.ietf.org/doc/html/rfc5288.

[28] Joseph Salowey, Abhijit Choudury, and David A. McGrew. AES Galois Counter Mode (GCM) cipher suites for TLS. RFC 5288, August 2008.

[29] D. Whiting, R. Housely, and N. Ferguson. Counter with CBC-MAC (CCM). IETF Network Working Group, RFC 3610, September 2003.